Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th October 2025. Vol. 5. No. 1; e-ISSN: 2808-4519

Implementation of The Isolation Forest Algorithm for Mysql Query Performance Anomaly Detection Based on Data Performance Schema

Tengku Didi Ferdillah[1*], Relita Buaton[2], Siswan Syahputra [3]

[1,2,3]*STMIK Kaputama*

tengkudidiferdillah23@gmail.com[1*], bbcbuaton@gmail.com[2], siswansyahputra90@gmail.com[3]

Abstract

Monitoring query performance in database systems is often a manual and reactive process, proving inefficient for the early detection of issues that can impact application stability. This research aims to design and implement a system for automated and proactive query performance anomaly detection. This system utilizes data from MySQL's Performance Schema and applies an unsupervised machine learning algorithm, namely Isolation Forest, to identify queries with unusual behavior based on eight researcher-selected performance metrics. The detection process is implemented to run periodically in the background and send early notifications via email. Experiments were conducted by varying the contamination parameter, with the model's performance evaluated using Precision, Recall, and F1-Score metrics. The experimental results indicate that the configuration with contamination=0.1 yielded the most optimal performance, achieving an F1-Score of 0.39 and a Recall of 100% for the anomaly class. The developed system successfully demonstrated its ability to detect various types of anomalies, including the N+1 query problem, and offers an efficient solution to proactively improve database system performance.

Keywords: *Anomaly Detection, Isolation Forest, Query Performance, Machine Learning*

1. Introduction

To ensure reliable technology availability in this modern era, monitoring is necessary to ensure availability and standards that meet industry needs. Especially when it comes to data storage in databases, database issues often significantly impact application performance, particularly SQL query performance [1]. It was also noted that several companies experienced losses due to a decline in platform performance which had a significant impact on their users [2].

Therefore, it's important for application developers to minimize abnormal SQL queries and avoid manual detection for efficiency reasons. Query performance tends to deteriorate as data increases [3]. Which will automatically cause problems when the application gets bigger and has a lot of data.

Based on these problems, researchers chose the isolation forest algorithm to detect SQL performance, through data in the performance schema in columns that correlate with performance such as time, number, error, etc.

The primary focus of this research is on the application and evaluation of the Isolation Forest algorithm as the core method for analyzing query performance data. Unlike traditional approaches that might only monitor a single metric like execution time, this study tests the algorithm's ability to understand the multi-dimensional relationships between eight different performance features. Thus, the system not only looks for slow queries but also identifies more complex anomaly patterns, such as queries that are executed too frequently or scan an unusual number of rows, which are often the hidden root causes of performance issues.

To make the algorithm's analytical results functional and impactful, the model's output is presented in an interactive web dashboard. This dashboard provides rich data visualizations and anomaly details to facilitate in-depth analysis by administrators. Therefore, the combination of the algorithm's intelligent analysis with this visual and structured data presentation forms a comprehensive decision support tool that assists in efficiently overcoming the challenges of finding the root causes of query performance issues.

2. Literature Review

2.1. Machine Learning

A fundamental branch of Artificial Intelligence (AI), enables systems to learn from data and perform tasks automatically without being explicitly programmed [4]. The field is broadly categorized into three main approaches: supervised learning, unsupervised learning, and reinforcement learning. Although it has been widely applied across various sectors, research indicates that the application of machine learning is currently most dominant in the medical field [5].

2.2. Anomaly Detection

Research by Fang 2024, highlights that anomaly detection often utilizes Unsupervised Learning techniques, which are effective as they do not require labeled data [6]. Methods in this category, such as Unsupervised Anomaly Detection (UAD), operate by assigning a continuous score to measure the degree of strangeness for each data point. Additionally, Clustering techniques play a significant role by grouping data, where anomalies can be identified as points that form small, isolated clusters or are distant from large, normal clusters. The integration of these unsupervised techniques can complement each other, enhancing the overall effectiveness and accuracy of detecting unusual data patterns.

2.3. Isolation Forest Algorithm

According to research by Liu 2008, Isolation Forest (iForest) is an anomaly detection method that operates by isolating anomalies rather than profiling normal instances [7]. This algorithm is based on the principle that anomalies are few and different, making them more susceptible to isolation. In practice, iForest constructs an ensemble of isolation trees (*iTrees*) and identifies anomalies as the data points with the shortest average path length from the root of the trees, indicating they are the easiest to isolate.

2.4. Database System

Research by Syahputri 2023, states that a key aspect of the development of information and communication technology is the integration of database systems, which are instrumental in efficiently storing, managing, and processing data [8]. A database system consists of a collection of interconnected data stored on a specific medium, designed to minimize redundancy and enhance efficiency in data retrieval and management.

2.5. MySQL Database

Research by Kalsum Siregar 2024, defines the MySQL database as a relational database management system (RDBMS) that utilizes the SQL language to manage and manipulate data [9]. As open-source software distributed under the GNU General Public License (GPL), MySQL allows users to store, retrieve, and manage data in an organized structure without licensing fees. The software itself is distinct from SQL, which is the standard language for interacting with databases, whereas MySQL is the DBMS application that implements SQL for data management.

2.6. Query Performance

According to research by Andriyani & Pujianto 2024, query performance in databases faces challenges due to the rapid accumulation of transactional data and high update frequencies [10]. Despite improvements in system configuration and query tuning, volumetric growth in transaction tables affects query speed. Therefore, maintaining optimal query performance is essential to support business activities and decision-making. The methods used in their research, namely table reconstruction and data archiving, were proven to significantly improve query performance, especially on tables with large transaction volumes.

2.7. Performance Schema

The MySQL Performance Schema is a feature for low-level, real-time monitoring of server execution, focusing on performance data rather than metadata like the INFORMATION_SCHEMA. It monitors events such as SQL statement stages (e.g., parsing, sorting), file I/O, and table locking. Operating entirely in-memory, its data is accessible via SELECT queries but is cleared upon server shutdown. This data is collected efficiently through instrumentation points in the server's source code without requiring additional threads [11].

2.8. Python Flask

Research by Mufid 2019, notes that Python Flask, while a lightweight and flexible framework, does not enforce a Model-View-Controller (MVC) architectural pattern by default, which can lead to disorganized project structures [12]. To address this, their study designed a generator system to automatically create an MVC-based folder structure for Flask applications. The results demonstrated that implementing the MVC pattern not only helps users manage projects in a more structured manner but also improves development efficiency and accelerates the overall application loading time.

3. Analysis And Design

3.1. Research Methodology

This study employs a quantitative research method through a computational experiment, where machine learning techniques are applied to numerically analyze MySQL query performance data from the Performance Schema. The Isolation Forest algorithm is utilized to detect anomaly patterns based on relevant parameters. The output from this model is then visualized in a web application, which was developed using the Waterfall model of the Software Development Lifecycle (SDLC).

3.2. System Requirement Analysis

Based on the problems identified in the problem statement, the system model must be designed to address these issues with an efficient and automated approach. The following are the primary requirements that the system must fulfill:

a) Automated Query Anomaly Detection
   The system shall be capable of automatically detecting queries with abnormal (anomalous) performance without requiring manual intervention from a database administrator. To achieve this, the system will implement the Isolation Forest algorithm to identify queries whose execution patterns deviate significantly from the majority.
b) Visualization of Query Performance Data
   The system must provide a graphical interface that allows administrators to easily understand query performance trends. Data retrieved from the Performance Schema shall be visualized in the form of a scatter plot, where each data point is highlighted based on its anomaly score.
c) Early Warning for Query Performance Anomalies
   The system must be able to display alerts and send email notifications if queries with high anomaly scores are detected, serving as an early warning of potential performance issues.

3.3. Model Implementation

This research utilizes the Isolation Forest algorithm, which excels in detecting data anomalies. The detection model is trained on a dataset from the Performance Schema database, which is directly relevant to query performance. The results from the detection model are then visualized in a web-based interface to facilitate easy monitoring, featuring alerts and email notifications for any detected anomalies. The implementation process of the anomaly detection model in this study is illustrated in the following flowchart:

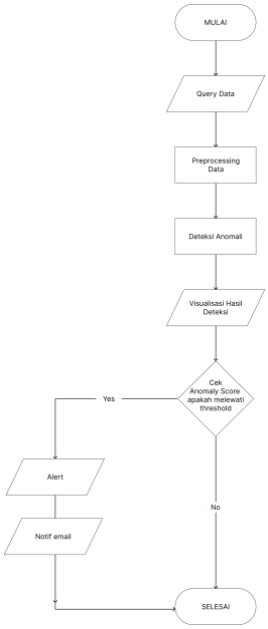

Fig. 1: Flowchart

Information :

1. Start: The anomaly detection process begins.
2. Query Data: Data is retrieved from the performance_schema database.
3. Data Preprocessing: Values are converted to align with the query performance context.
4. Anomaly Detection: The model analyzes patterns to determine if a data point is an anomaly.
5. Visualize Detection Results: All data points are displayed along with their anomaly scores.
6. Check if Anomaly Score Exceeds Threshold: If Yes (Anomaly Detected) The system displays an alert with the details of the predicted anomaly and sends an email notification to the administrator. If No (No Anomaly) The process ends if no anomaly is detected.

3.3.1. Data Preparation

The data collection phase aims to extract relevant quantitative metrics related to query performance. The primary data for this research is sourced from the Performance Schema database in MySQL, specifically from the events_statements_summary_by_digest table. An SQL

query is used to aggregate and select eight primary variables that serve as the features for training the anomaly detection model. These eight selected variables and their relevance to anomaly detection are summarized in the table below:

Table 1: variable from performance schema

Variable Name	Description	Relevance to Anomaly Detection
execution_count	The total number of times the query was executed.	An extremely high count can indicate an uncontrolled loop or an N+1 problem.
total_execution_time_seconds	The total cumulative time (in seconds) for all executions.	A high total time can cause a system bottleneck.
avg_execution_time_seconds	The average execution time per query.	A consistently high average suggests an inefficient query.
max_execution_time_seconds	The longest execution time among all executions.	An extreme maximum value can signify an unusual load or database locking.
total_rows_examined	The total number of rows scanned by the database engine.	A very large number often indicates missing indexes or poor joins.
total_rows_sent	The total number of rows sent as a result to the application.	A large number can strain the network and the client application.
total_errors	The total number of errors that occurred during execution.	An increase in errors suggests issues with the query syntax or data integrity.

After the query performance data is collected, the data preparation stage is performed to ensure its quality and readiness for modeling. This process includes handling missing values, where any null values are replaced with zero to maintain the dataset's numerical integrity. As the data collected from performance_schema represents a snapshot in time, it is merged with existing historical data from the query_performance_dataset.csv file to build a richer dataset. To keep the data current, duplicate queries are handled by retaining the most recent record, ensuring each query's statistics reflect its latest performance. This final, clean, and updated dataset is then ready for implementation in the anomaly detection algorithm.

3.3.2. Model Detection

The prepared dataset is then implemented in the Isolation Forest algorithm. All eight relevant numerical features are used as input to train the model. The key contamination parameter is set according to experimental results (e.g., 0.1) to provide the model with an initial assumption about the proportion of anomalies in the data. The model then constructs an ensemble of isolation trees to separate each data point. Based on how quickly a query can be isolated, the model provides two outputs: an anomaly_score, which is a numerical score indicating the degree of strangeness, and an anomaly label, which is a binary classification (-1 for anomaly, 1 for normal). These outputs form the basis for the visualization and notification system.

3.4. Interface Design

The output of the anomaly detection model is presented through a web-based user interface designed as an interactive dashboard. This dashboard summarizes the necessary data for monitoring and analyzing anomalies. The main components of this interface include:

1. Graphical Visualization: An interactive scatter plot (in both 2D and 3D) that maps key performance metrics, where each data point is colored according to its status (normal or anomaly) to facilitate the visual identification of outliers.
2. Top Anomalies Table: A table that displays the top three queries with the lowest (i.e., most anomalous) anomaly scores, sorted to highlight the most critical performance issues.
3. Query Detail Modal: An interactive pop-up window (modal) that displays all the detailed performance measurement parameters for a selected query from the table, allowing for in-depth analysis.

Fig. 2: Dashboard Interface

4. Discucssion And Implementation

This section presents the results from the implementation and experimentation of the anomaly detection model. The primary experiment focuses on determining the most optimal contamination parameter for the *Isolation Forest* model.

4.1. Experimental Results

Experiments were conducted with three different contamination value scenarios: 0.1, 0.3, and 0.5 with a total of 322 data sets. Model performance in each scenario was measured using Precision, Recall, and F1-Score metrics for the anomaly class (-1), with the results summarized in the table below:

Table 2: contamination value test result

No	Contamination	Precision	Recall	F1-Score
1	0.1	0.24	1	0.39
2	0.3	0.08	1	0.15
3	0.5	0.05	1	0.09

The most notable result is the model's ability to achieve 100% Recall in all scenarios, meaning the system successfully identified all true anomalies present in the test data. With maximum Recall achieved, the primary deciding metrics for model performance become Precision and F1-Score. Based on the table, contamination=0.1 shows the highest F1-Score (0.39) and the highest Precision (0.24).

The data visualization in Figure 1 (3D Scatter Plot) effectively shows how the model successfully separates the anomalous data points (colored red) from the cluster of normal data (colored blue), even with complex metrics.

4.2. Discussion

The experimental results show a clear trade-off between *Recall* and *Precision*. Although contamination values of 0.3 and 0.5 achieved perfect Recall, this was accomplished by drastically sacrificing Precision. The extremely low precision rates (8% and 5%) indicate that the majority of generated alerts are *false positives*. In a real-world implementation, this would lead to *alert fatigue*, where users would begin to ignore notifications from the system.

Therefore, contamination = 0.1 was chosen as the most optimal configuration. This configuration not only found all anomalies but also did so with the highest precision rate among the options, making it the most efficient and reliable.

A case study of the detected anomalies also demonstrates the system's ability to identify complex performance issues such as the N+1 Query Problem. The model successfully flagged a query as an anomaly not because of its slow execution time, but due to the combination of a fast execution time with an extremely high execution_count a pattern often missed in manual analysis. This proves that the model's multi-dimensional approach is superior to single-metric monitoring.

5. Conclusions And Recommendations

5.1. Conclusion

This research has successfully designed and implemented an automated MySQL query performance anomaly detection system using the Isolation Forest algorithm. The system proved capable of proactively identifying queries with unusual behavior by leveraging data from the *Performance Schema*. Based on the experimental results, the optimal model configuration was found with the contamination=0.1 parameter, which yielded the best balance between the ability to find all anomalies (100% Recall) and the accuracy of alerts (24% Precision), with the highest F1-Score (0.39). The developed system offers an efficient solution to help database administrators maintain system stability and performance in a preemptive manner.

5.2. Recommendations

For future development of this research, several suggestions can be considered:

1. Ground Truth Enhancement: Involve validation from a Database Administrator (DBA) to create a more accurate ground truth, not just one based on a simple time threshold.
2. Algorithm Exploration: Compare the performance of *Isolation Forest* with other unsupervised algorithms such as *Local Outlier Factor (LOF)* or *Autoencoder* models to seek potential improvements in precision.
3. Scalability of the notification and scheduling system for very large-scale production environments. It is recommended to upgrade the scheduling system from a simple script to a more robust task queue, such as the Redis message broker, to ensure more robust and distributed task execution.

References

[1] K. E. Permana, K. Sophan, A. Muntasa, and A. B. Rahmat, "Perbandingan Kinerja Query Sql Join Tables Dengan Menggunakan Index Performance Comparison of Query Sql Join Tables Using Index," *J. SimanteC*, vol. 11, no. 2, pp. 241–248, 2023, [Online]. Available: https://github.com/datacharmer/test_db.

[2] G. Yudhy Kusuma and U. Y. Oktiawati, "Perancangan Sistem Monitoring Performa Aplikasi Menggunakan Opentelemetry dan Grafana Stack," *J. Internet Softw. Eng.*, vol. 3, no. 1, pp. 26–35, 2022, [Online]. Available: http://34.128.121.13:5000/v1/campaigns

[3] R. Pamungkas, "Optimalisasi Query Dalam Basis Data My Sql Menggunakan Index," *Res. Comput. Inf. Syst. Technol. Manag.*, vol. 1, no. 1, p. 27, 2018, doi: 10.25273/research.v1i1.2453.

[4] Wijoyo A, Saputra A, Ristanti S, Sya'ban S, Amalia M, and Febriansyah R, "Pembelajaran Machine Learning," *OKTAL (Jurnal Ilmu Komput. dan Sci.*, vol. 3, no. 2, pp. 375–380, 2024, [Online]. Available: https://journal.mediapublikasi.id/index.php/oktal/article/view/2305

[5] A. Roihan, P. A. Sunarya, and A. S. Rafika, "Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper," *IJCIT (Indonesian J. Comput. Inf. Technol.*, vol. 5, no. 1, pp. 75–82, 2020, doi: 10.31294/ijcit.v5i1.7951.

[6] Z. Fang *et al.*, "Towards a Unified Framework of Clustering-based Anomaly Detection," 2024, [Online]. Available: http://arxiv.org/abs/2406.00452

[7] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 413–422, 2008, doi: 10.1109/ICDM.2008.17.

[8] K. Syahputri, M. Irwan, and P. Nasution, "Peran Database Dalam Sistem Informasi Manajemen," *J. Akunt. Keuang. dan Bisnis*, vol. 1, no. 2, pp. 54–58, 2023, [Online]. Available: https://jurnal.ittc.web.id/index.php/jakbs/article/view/36

[9] U. Kalsum Siregar, T. Arbaim Sitakar, S. Haramain, Z. Nur Salamah Lubis, U. Nadhirah, and F. Sains dan Teknologi, "Pengembangan database Management system menggunakan My SQL," *SAINTEK J. Sains, Teknol. Komput.*, vol. 1, no. 1, pp. 8–12, 2024.

[10] W. Andriyani and P. Pujianto, "Maintaining Query Performance through Table Rebuilding & Archiving," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 18, no. 1, p. 73, 2024, doi: 10.22146/ijccs.90062.

[11] dev.mysql.com, "Chapter 29 MySQL Performance Schema," dev.mysql.com. Accessed: Feb. 01, 2025. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/performance-schema.html

[12] M. R. Mufid, A. Basofi, M. U. H. Al Rasyid, I. F. Rochimansyah, and A. rokhim, "Design an MVC Model using Python for Flask Framework Development," in *2019 International Electronics Symposium (IES)*, 2019, pp. 214–219. doi: 10.1109/ELECSYM.2019.8901656.