Journal of Intelligent Decision Making and Information Science

www.jidmis.org
eISSN: 3079-0875

Predictive Detection and Resolution of SQL Server Performance Bottlenecks Using Generative AI And Dynamic Management Views (DMVS)

Krishnan Thiruvengadam[1]*, Yousef Elsheikh[2], Fabio Rossi[3]

1AWS DBA Lead AI, Tata Consultancy Services, Pune, Maharashtra, INDIA
2Software Engineering, Faculty of Information Technology, Applied Science Private University, Amman,
3Lead Infrastructure & MLOps Engineer, IronWay SubInfra Digital Systems, London, United Kingdom

ARTICLE INFO

Article history:
Received 12 April 2026
Received in revised form 20 May 2026
Accepted 24 June 2026

Keywords: SQL Server performance management; Dynamic Management Views; bottleneck prediction; machine learning; generative AI; systems management

ABSTRACT

Resource pressure is a recurring issue in SQL Server performance management because users only notice it when they experience slow response times, blocking, timeouts, or service instability. Manual interpretation of Dynamic Management Views (DMVs) is typically reactive and requires specialist database-administration expertise to interpret the details of the telemetry, which can be accessed via DMVs This study proposes and tests a predictive model to identify and resolve SQL Server performance issues based on the DMVs telemetry, supervised machine learning, and guarded Generative AI suggestions. The records were gathered from a balanced SQL Server 2025 Developer environment in five different operational states: normal, CPU bottleneck, I/O bottleneck, blocking bottleneck, and missing-index bottleneck with the AdventureWorks2022 database. Logistic Regression, Random Forest, Gradient Boosting and XGBoost classifiers were built using Encoded DMV features. Random Forest and Gradient Boosting obtained the highest test accuracy (83.64%) and macro F1-score (83.89%), and the five-fold cross-validation showed good stability. The blocking session count, the number of tasks waiting, wait time, elapsed time, number of logical reads, worker time, and physical reads were the most important features identified by the feature-importance analysis. The GenAI recommendation layer generated root cause explanations and safe resolution actions that are understandable to the DBA. Contribution is a proactive and explainable framework for SQL Server performance management.

1. Introduction

SQL Server is a fundamental platform for transactional, analytical, and enterprise information systems and has a direct impact on the reliability of services, customer experience, operational continuity, and administration costs [1]. In managed environments, any amount of time that it takes to query, block, put pressure on the storage, or overload the CPU can lead to application timeouts and business disruption [2]. Performance management in such systems is no longer focused on simply tuning the database; now it is a systems-management need, which brings together workload

[1] *Corresponding author.*
E-mail address: kriz1111@gmail.com

behaviour, infrastructure resources, observability and incident response. Dynamic Management Views (DMVs) in Microsoft SQL Server allow you to see the wait statistics, execution statistics, active requests, session state, missing indexes and other data that can be used to diagnose issues with the workload at runtime [3]. Other features of Query Store and associated monitoring tools can be used for post-execution analysis as well, but there are still some operational decisions that rely on the database administrator's skill in analysing raw metrics in a timely fashion.

The main issue that this study aims to tackle is that the troubleshooting of SQL Server is mostly reactive. Most of the time, when the problem starts to appear in the applications, administrators realise that it is a degradation, and most of the time, traditional alerts report the threshold violation, but provide no clue as to which bottleneck class it is or what should be done about it. While manual DMV analysis is very powerful, it is also very time-consuming, requires a high level of expertise to master, and can be inconsistent when symptoms are the same in several different scenarios, such as CPU pressure, I/O latency, memory allocation, lock contention, missing indexes, and inefficient query plans [4]. For instance, a high elapsed time could be due to blocking, storage reads or query plan inefficiency or due to CPU waiting; likewise, the same could be the case when logical reads increase with worker time due to a missing index that leads to repeated scans of the same data. This uncertainty in diagnosis leads to a disconnect between monitoring and intervention/management.

In the past few years, however, a lot of work has been done on autonomous database management and tuning using machine learning and models that can learn from workload telemetry to make some configuration or optimisation decisions [5]. The general trend for AI-for-database technologies is also reflected in the surveys, where more were found to be interested in learned query optimisation, index selection, configuration tuning, and workload prediction, all of which are related technologies [6]. There is a relatively limited amount of empirical research that has been done linking evidence from SQL Server DMVs with supervised classifications of bottlenecks and a Generative AI layer which transforms the predictions into explanations that are safe for DBAs to consume. A monitoring-only dashboard will alert administrators that a metric has changed, and a predictive-and-explanatory dashboard will alert administrators which bottleneck is starting to occur, the evidence behind why it is a bottleneck, and which remediation steps to review.

This research aims to create and test a system to detect SQL Server performance issues using SQL Server metrics and provide analytical assistance with Generative AI to resolve them, based on information from the DMV. The research goals are to retrieve performance information from SQL Server DMVs, find out indicators of performance bottlenecks, train the classification models, interpret the model's key features, create guarded recommendations and evaluate the quality of the predictions and recommendations. The study includes four questions: What are the best indicators in a DMV that point to SQL Server bottlenecks? How well can different machine-learning models classify these DMV bottlenecks? Can the integrated framework be used to determine the meaning of the DMV information and use Generative AI to transform it into useful guidance to resolve the bottleneck? How well does this integrated framework support the decision-making process of a DBA?

What is new is the ability to integrate three parts that are normally distinct: Live DMV telemetry, supervised bottleneck prediction and rule-constrained GenAI explanation. This framework not only

targets automatic knob tuning in the context of AI but also interpretable systems management for SQL Server operations. It models the state of the resources based on the signals emitted by the various system components involved in using the DMV, trains the models to recognise the states of the resources, and generates explanations that adhere to the safety rules in the database administration. The experimental database is 825 balanced records gathered from SQL Server 2025 Developer and AdventureWorks2022 under various conditions, namely, normal, CPU, I/O, blocking and missing-index. The outcomes prove that Random Forest and Gradient Boosting can attain an accuracy of 83.64%, while feature significance discovers indicators that have value in operation. This offers both a practical contribution towards proactive database administration and an academic contribution towards the research of systems management assisted by AI.

The problem is not that it is a problem with database tuning, but a problem with predictive systems management. A SQL Server instance is a Managed Service, where workloads, memory, storage, lock duration, waits, scheduler waits and index design (all in combination) impact availability [7]. Thus, the framework proposed is designed to be a human-in-the-loop management assistance tool, to enhance observability, predict degradation and to normalise initial diagnosis. This framing also justifies the study not being limited to the accuracy of their models. A useful operational framework should be able to offer evidence that is reliable, indicators that are easy to understand, and remediation advice that is safe. This study touches on the entire decision cycle from signal collection to actionable response by integrating statistical learning and GenAI recommendations within DBA guardrails. A unified contribution is critical for two reasons: First, enterprise administrators have few other metrics they need; they need one that allows them to translate telemetry into actionable evidence in real time that is governed and assigned to humans.

This type of organisation aids the study in having a clear functional identity. It considers prediction as a part of a broader management process which requires monitoring, diagnosis, explanation and action to be accountable for working together. This type of design is particularly useful in situations where digital services are impacted by the availability of the database and where database administrators need to make tuning decisions based on tangible information and data, instead of intuition. In this research work, the gap is addressed by a supervised classification layer and a guarded GenAI recommendation module.

2. Literature Review

2.1 SQL Server bottlenecks and DMV monitoring

A SQL Server performance bottleneck is a condition at runtime when the concurrency, execution pattern, or dominant resource limits the performance of the database [8]. Frequently occurring reasons for bottlenecks are CPU pressure, delays in allocating memory, I/O latency, blocking and locking, lack of indexes, inefficient query plans, and contention on tempdb. Wait analysis, which is based on the DMV, is still valuable as it provides information about where sessions spend time while execution statistics provide information about the time spent by the workers, elapsed time, logical reads, physical reads and frequency of execution of the query [9]. These indicators are particularly

relevant in systems management as they relate latency that users experience to the behaviour of resources.

Common methods of monitoring are manual DMV scripts, SQL Server Profiler, Extended Events, Query Store, performance counters and rule-based alerts [10]. They are useful, but in most cases, they need to be interpreted after the fact by the more experienced DBAs. Query Store gives a history of plans and runtime actions, but it does not automatically convert heterogeneous evidence to a predicted bottleneck class [40]. Monitoring based on rules can trigger numerous alerts, but not the most likely root cause in practice [11].

2.2 Machine learning and AIOps prediction

Machine learning has increasingly been applied to database management. Previous work by [12] illustrated the ability to use historical tuning information to optimise the configuration of a DBMS, and subsequent testing revealed the potential and challenges involved in introducing ML-based tuners to real database systems. GPTuner incorporated large-language-model guidance and Bayesian optimisation into database tuning decisions [13]. Several studies in machine learning for databases can be categorised into optimisation, indexing, cost estimation, and performance prediction [14, 15]. Recent research also showed that deep learning can help execute queries and adaptively manage database systems, but generalisation of the approach in a wide variety of workloads is still a challenge [16]. A few of these studies are used to support the use of learned models, but most of them are only for tuning actions and not for classifying SQL Server DMV bottleneck states [17, 18].

There are many methods available for predictive monitoring, such as classification and anomaly detection. XGBoost is another gradient boosted tree method for structured data that performs well, and when there are non-linear interactions between features, ensemble learning and Random Forest models work well [19]. There have been some methods based on deep time-series, such as the LSTM-based model, for anomaly detection. The AIOps surveys reveal that current operations research is more and more based on metrics, logs, traces, anomaly detection and root-cause analysis [20]. This literature can be used to inform proactive failure management, but there is limited development of the SQL Server classification of DMVs [42].

The second capability of Generative AI is that it can turn technical evidence into explanatory guidance. Structured or retrieved evidence can be used to generate responses in language systems that synthesise them with transformer models or large-scale few-shot learning or retrieval-augmented generation [21]. However, there are still significant threats of hallucination and unsafe automation. Another important aspect highlighted in explainability research is making the outputs of the models connected with understandable evidence [22]. The gap is not just the absence of monitoring data; it also represents the absence of an integrated framework that can predict the bottlenecks that can be identified in the DMV and generate evidence-based, human-validated recommendations.

2.3 Generative AI, explainability, and gap

The other stream is explainable and maintainable machine-learning systems, which is also relevant. D. Sklavenitis and D. N. Nahar et al. stated that the management of data, models, monitoring, and

operational assumptions is inextricably linked and is a source of technical debt for an ML system [23]. D. Sklavenitis and D. N. Nahar et al. also noted that algorithm selection is not the only engineering practice in production ML; there are others, such as collaboration between the domain expert and developer, data management, and validation [23]. Especially for database administration, this is particularly significant as unsafe recommendations can have an impact on availability, cost and data integrity. In the context of ExAI research, feature importance and recommendation evidence interpretation are thus useful using the lens of explainable AI research [22].

There are other terms used to differentiate between prediction, root cause analysis and remediation, as can be seen in the literature. AIOps research has progressed beyond identifying abnormal metrics to identifying failure sources and helping in incident response [24]. There has been much research on automatic configuration and index tuning in database-management systems. Explanation and synthesis, on the other hand, are now easier in the context of generative AI research, and yet come with risks of hallucination, privacy and automation [25]. This present study is a part of both streams. It is not an attempt to replace a DBA, nor does it do independent production tuning. Rather, it leverages the DMV evidence to determine the most probable states causing bottlenecks and then proposes limitations on the recommendations that can be discussed with a human operator [41].

With this positioning, the literature gap was identified. While existing SQL Server monitoring exposes metrics, ML-based database studies tend to optimise configurations, AIOps work is often targeted at distributed services, and GenAI studies on language synthesis focus on that. Across the 14 recent works reviewed in Sections 2.2 and 2.3, none jointly integrates SQL Server DMV telemetry, supervised bottleneck prediction, model-level explanation, and guarded resolution guidance within one reproducible experimental framework.

Therefore, literature provides support for the components of the approach proposed in this study, but does not fully combine these components. Provided by the monitoring capability from DMV, classification ability from machine learning, transparency from explainability, and accessible guidance from GenAI. The contribution of this study is to combine these elements into a conscious, experimental SQL Server performance management pipeline.

3. Methodology

3.1 Research design and framework layers

This is an experimental research design used to build, mark and test a performance dataset for SQL Server within a controlled workload. The test environment comprised Microsoft SQL Server 2025 Developer Edition, SQL Server Management Studio and the AdventureWorks2022 sample database. The snapshots obtained from the DMV in the repeated workload situations were saved to the research database, DMV_AI_Research. It focused on the systems-management aspect of the design, which was essential to the proactive monitoring and decision-support that the paper emphasised, and was based on the outline's data-collection, feature-engineering, predictive-modelling, recommendation and validation layers.

The proposed scheme is divided into five layers. The data-collection layer saves SQL Server telemetry information (including wait statistics, query execution statistics, active requests, etc.) from DMVs The

feature-engineering layer converts the raw output from the DMVs to numeric and categorical data that can be used for predicting: waiting task count, wait time, signal wait time, worker time, elapsed time, logical reads, physical reads, execution count, blocking session count, missing-index impact, and wait type. The predictive-modelling layer categorises each snapshot according to a bottleneck category. The anticipated class and chosen evidence from the DMV go through the GenAI recommendation layer, which outputs an explanation for the root cause and a list of DBA remediation activities. In the validation layer, metrics of the model, the results of the confusion matrix, feature importance, and safety alignment with recommendations are presented, as shown in the table below.

Table 1: Framework layers and analytical functions.

Layer	Main function	Output
Data collection	Capture DMV telemetry at intervals	Raw DMV snapshot
Feature engineering	Prepare wait, query, blocking, and index features	Model-ready feature set
Predictive modelling	Classify the bottleneck category using ML	Predicted class
GenAI recommendation	Generate evidence-aligned DBA guidance	Root-cause explanation and action list
Validation and reporting	Report metrics, errors, explanations, and safeguards	Tables, figures, and recommendation audit

3.2 Experimental environment and DMV collection

The following five scenarios were designed: normal, CPU-bottleneck, I/O-bottleneck, blocking-bottleneck, and missing-index bottleneck. The AdventureWorks2022 database has been created using Standard joins to create a normal workload. The joins were computationally expensive operations that were used to create CPU pressure. Physical reads were forced (buffer clearing and retrieval), and evidence of I/O bottlenecks was collected. Blocking was caused by an explicit transaction that held a lock while a second session waited. The missing-index behaviour was triggered by copying SalesOrderDetail into a heap table and performing a filter on ProductID without using an index repeatedly, where the index was not present. The top scores for each DMV Wait were added to the research table for each snapshot. The final data set had 825 data points, divided into 165 data points per class.

The target variable used is the scenario_label. The snapshot identifier and the snapshot time stamp were not used for training, as they identify the order of collections instead of bottleneck behaviour. One-hot encoding was performed on wait_type, and numeric features were retained. The data used was divided into an 80% training data set and a 20% testing data set using stratified sampling while keeping 33 test observations for each class. Four classifiers were trained: Logistic Regression

(baseline), Random Forest, Gradient Boosting, and XGBoost. Five-fold stratified cross-validation was used to test the robustness.

The accuracy, macro precision, macro recall, macro F1 score, classification report and confusion matrix were used to measure model performance. Macro averaging was chosen as this gave equal weight to all the bottleneck categories for decision support. The importance of each feature in the best-performing ensemble model was retrieved to determine the most important DMV indicators for prediction. The GenAI module was assessed qualitatively against correctness, actionability, clarity, safety, and evidence alignment. Table 2 instead summarises the DMV feature groups used by the predictive model. Recommendations were made with a few caveats: Do not terminate any sessions, deploy no indexes, do not expose sensitive query text, and do not modify the production configuration without DBA permission. This guardrail design is inspired by those in the literature on risk and risk assessment for AI governance and ML operations.

Table 2: DMV feature groups and purposes.

Feature group	Variables	Purpose
Wait statistics	waiting_tasks_count, wait_time_ms, signal_wait_time_ms	Capture accumulated waiting pressure
Query execution	total_worker_time, total_elapsed_time, execution_count	Represent query cost and duration
I/O evidence	total_logical_reads, total_physical_reads	Identify access-path and storage load
Concurrency evidence	blocking_session_count	Detect blocking and lock contention
Index evidence	missing_index_impact	Represent missing-index signal
Categorical wait	wait_type	Encode dominant wait category

3.3 Modelling and GenAI evaluation

The primary data-collection stored procedure inserted the 15 highest-ranked non-idle wait records from each collection event into dbo.DMV_Performance_Dataset. An initial collection generated 15 records for each scenario. The automated collection procedure then invoked the same snapshot routine ten additional times per scenario at short intervals, producing 150 further records for each scenario. Consequently, every scenario contained 165 records (15 + 10 × 15), and the five scenarios produced 825 records overall. Internal idle and background waits, including sleep, broker, Query Store background, SQL trace, and diagnostic waits, were excluded to retain operationally relevant evidence.

The modelling pipeline was done in Python/Jupyter. The exported CSV file was checked for missing values, class balance and consistency of the columns. There were no missing values found. Wait_type was one-hot encoded because it is a categorical feature in the DMV. After trying the models on the

scaled numeric features, we observed that scale differences were not an issue for tree models like Random Forest, Gradient Boosting and XGBoost, so we did not proceed with scaling these numeric features. Logistic Regression was used as a clear-cut baseline.

The GenAI recommendation module was not intended to be a conversational bot, but rather a template-based, evidence-based layer on which additional customisation will be possible via AI models. The GenAI recommendation module is intended to be a template and evidence-based layer which can be further tailored with the use of AI models as a layer, not as a free-flowing chatbot. The module matched the bottleneck label with the highest DMV evidence for each of the predicted classes, matched the root cause explanation, suggested DBA actions, and safety guardrails. This design is designed to be "conservative," meaning the system is able to suggest, explain, and prioritise, but the role of human oversight is always needed in production changes.

In order to avoid overgeneralisation, all the labels were assumed to be scenario labels created in experimental conditions. This method could be used for an initial supervised-learning study as it provides ground truth. The methodology, however, does not leave too much out of the way and can be reproduced, followed up by other versions of SQL Server, and even compared with other enterprise telemetry collections in the future.

4. Results and Findings

4.1 DMV data summary and distribution

The data set had 825 data points from the DMV and 14 columns. All five scenario labels had 165 observations, which led to an even distribution of observations among the five classifications. This balance is significant since it will not influence the evaluation to be dominated by the results of the records in the normal state and will enable a macro-level portion of the precision, recall and F1-score to be somewhat representative of each bottleneck class. Extracted fields included wait behaviour, indicators of execution resources, concurrency evidence and index-related evidence. Snapshot time was retained solely for traceability and was deliberately excluded from model training because it identifies collection order rather than the operational mechanisms of a bottleneck.

The class-distribution figure shows that the number of normal, CPU, I/O, blocking and missing-index states was equal. The balanced totals are reported in Table 3 and Figure 1. Stratified 80/20 sampling subsequently assigned 132 records to training and 33 records to testing within each class, ensuring that every class was evaluated using the same number of previously unseen observations.

Table 3: Balanced bottleneck class distribution.

Scenario label	Records
Blocking bottleneck	165
CPU bottleneck	165
I/O bottleneck	165
Missing-index bottleneck	165
Normal	165

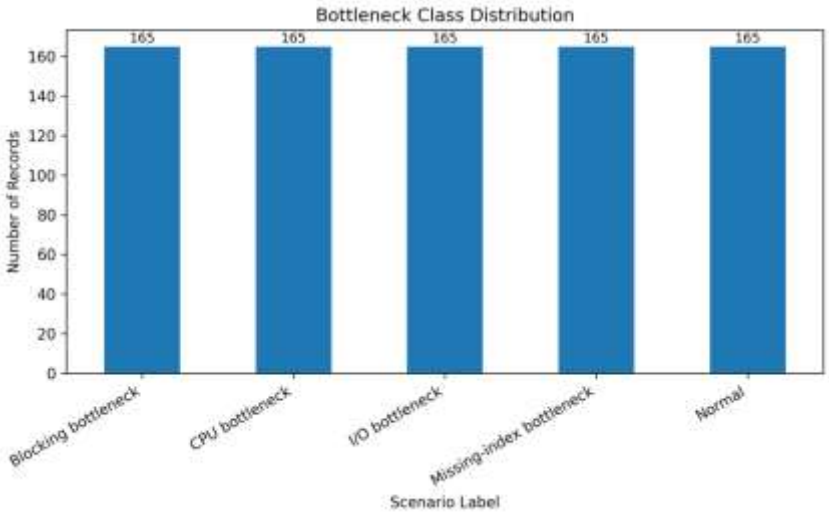

Figure 1: Balanced distribution of labelled bottleneck classes

4.2 Model performance and cross-validation

The best test performance was obtained by models of Random Forest and Gradient Boosting. Both models reached 83.64% accuracy, 85.75% macro precision, 83.64% macro recall, and 83.89% macro F1-score. XGBoost was very close with 82.42% accuracy and 82.73% macro F1 score, followed by Logistic Regression with 54.55% accuracy and 45.22% macro F1 score. The failure to linearly separate the selected DMV features, as demonstrated by the weaker baseline, indicates that the baseline states were not linearly separable. The ensemble models worked better as they can model interactions among wait type, wait tasks, worker time, elapsed time, logical reads, physical reads and blocking indicators as illustrated in Table 4. These identical aggregate test scores represent a tie on the fixed held-out partition and do not imply that the two algorithms are equivalent. Their five-fold cross-validation means differed (0.8376 for Random Forest and 0.8363 for Gradient Boosting), indicating modest performance variation across folds despite the shared test-set summary.

The ranking of the test sets was confirmed by the cross-validation. The mean macro F1_score and standard deviation of Random Forest were 0.8376 and 0.0238, respectively. Gradient Boosting had a similar mean value of 0.8363, XGBoost had a value of 0.8261, and Logistic Regression had a value of 0.4565. The small standard deviation of the cross-validations for the ensemble models shows that it was not a result of one favourable split of the train and test sets, as indicated in Table 5. It is a good starting point for additional enterprise validation for an exploratory DMV dataset.

The confusion matrix gives more detailed diagnostic details as shown in Figure 2. The test set was completely classified for both 'bottleneck' and 'IO bottleneck'; there were 33 correct predictions for each. This indicates that there were unique patterns found for blocking_session_count and I/O-execution evidence. The site of the missing index got 23 positive classifications from a possible 33: seven were classified as CPU bottleneck, and three were classified as Normal. Normal got 28 correct classifications out of 33, five of which were incorrectly classified as CPU bottleneck. The class that had the highest number of errors was CPU bottleneck; 21 correct classifications and 12 records predicted as Normal (Table 6). These errors are common due to moderate CPU pressure and wait

time characteristics of concurrent and normal executions and to missing-index workloads that may cause CPU and elapsed-time metrics to increase due to repeated scanning.

Table 4: Comparative machine-learning model performance.

Model	Accuracy	Precision_macro	Recall_macro	F1_macro
Random Forest	0.8364	0.8575	0.8364	0.8389
Gradient Boosting	0.8364	0.8575	0.8364	0.8389
XGBoost	0.8242	0.8420	0.8242	0.8273
Logistic Regression	0.5455	0.4271	0.5455	0.4522

Table 5: Stratified five-fold cross-validation results.

Model	CV_F1_macro_mean	CV_F1_macro_std
Random Forest	0.8376	0.0238
Gradient Boosting	0.8363	0.0259
XGBoost	0.8261	0.0233
Logistic Regression	0.4565	0.0156

Table 6: Classification report for Random Forest.

Class	precision	recall	f1-score	support
Blocking bottleneck	1.0000	1.0000	1.0000	33.0000
CPU bottleneck	0.6364	0.6364	0.6364	33.0000
I/O bottleneck	1.0000	1.0000	1.0000	33.0000
Missing-index bottleneck	1.0000	0.6970	0.8214	33.0000
Normal	0.6512	0.8485	0.7368	33.0000

4.3 Error Analysis and Feature Importance

Figure 3 and Table 7 show the feature-importance analysis results, with blocking_session_count being the largest influence, followed by waiting_tasks_count, wait_time_ms, total_elapsed_time, total_logical_reads, total_worker_time, total_physical_reads, signal_wait_time_ms, execution_count, and missing_index_impact. The results are feasible and workable. Contention of locks can be directly distinguished by the blocking session count. Waiting tasks and wait time are

indicative of the buildup of pressure. Elapsed time measures the time that the user has experienced. Access-path cost and storage demand are captured by logical and physical reads. The worker time is used to measure CPU usage. The missing-index impact was less than anticipated, but informative, most likely due to the fact that the test query also had an impact on elapsed time, logical reads and worker time. The lack of indexes in classification was based on a set of features and not just one DMV value.

The GenAI recommendation layer was used to transform the predicted bottleneck categories into root causes and DBA actions. For blocking, the module suggested identifying blocking sessions, looking at transaction duration, making transactions smaller and not wanting to kill production sessions automatically. It is advised to look at expensive queries, execution plans, statistics and index usage as the first step to scaling up hardware for CPU pressure. It suggested the following for I/O pressure: physical reads, storage latency, indexing, and file layout. It suggested validating missing-index DMV suggestions with execution plans (before creating indexes) for missing indexes. In the normal state, it is recommended for baseline monitoring only. The recommendations were carefully drafted to be conservative and take into account the belief that the use of GenAI should complement the DBAs' judgment, not supplant it.

The primary empirical results are that DMV telemetry can be used to make useful predictions of the presence of multiple bottlenecks as long as it is collected in the presence of labelled workload scenarios. Secondly, it is found that ensemble models are more appropriate than a linear baseline for this task. The third result is that feature importance makes sense from a database administration point of view, enhancing the trustworthiness. The fourth discovery is that predictive class and DMV evidence can provide interpretability to the recommendations that the GenAI makes. The recommendation layer, though, should not be taken lightly, as the wrong recommendation (hallucination or unsafe) might have negative effects on production systems.

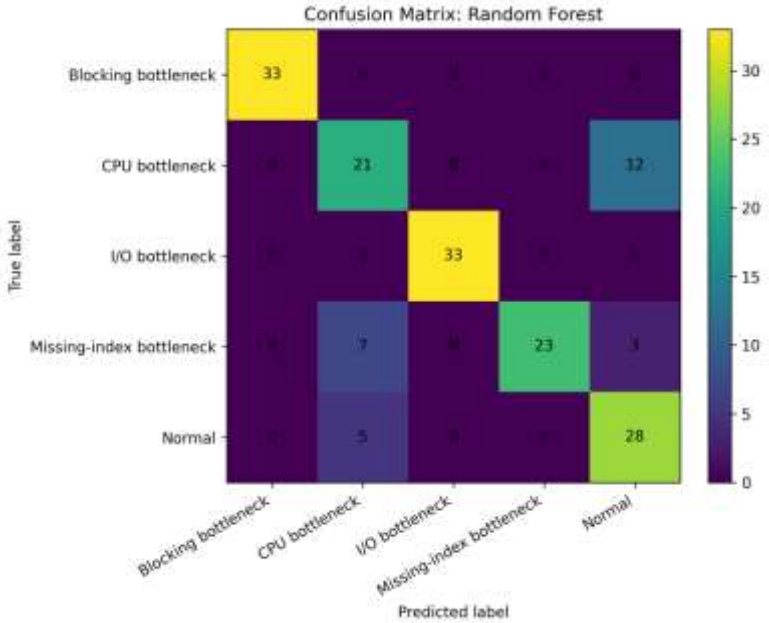

Figure 2: Random forest confusion matrix performance overview

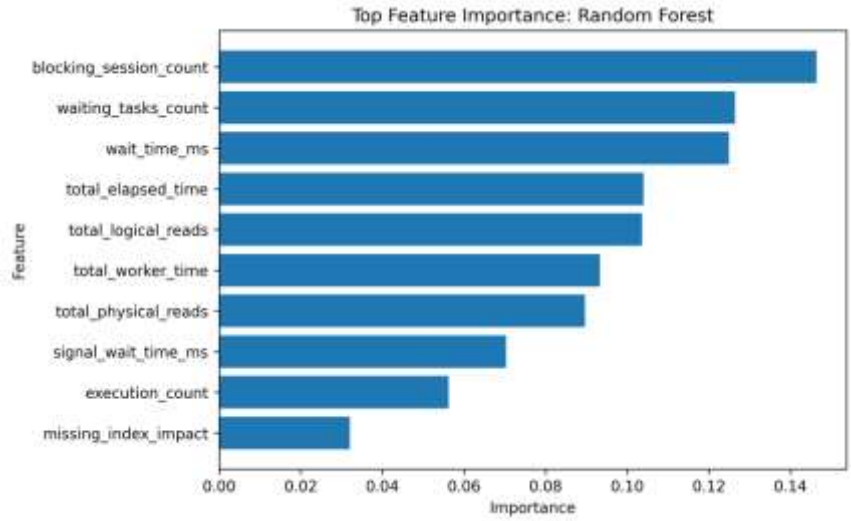

Figure 3: Random forest DMV feature importance ranking

Table 7: Top ten Random Forest feature importances.

feature	importance
blocking_session_count	0.1464
waiting_tasks_count	0.1265
wait_time_ms	0.1250
total_elapsed_time	0.1040
total_logical_reads	0.1037
total_worker_time	0.0933
total_physical_reads	0.0896
signal_wait_time_ms	0.0703
execution_count	0.0562
missing_index_impact	0.0320

4.4 GenAI recommendation results

In the DMV evidence summary, there were clear trends in evidence across classes. The rows that were blocked by the locks contained a non-zero amount of blocking-session evidence in the lock experiment, and the missing-index rows had the highest missing_index_impact score of 81.48 compared to 74.07 in the other experiments collected after DMV state accumulation. Some differences were seen between CPU and I/O classes in terms of worker, elapsed, logical-read and physical-read patterns, but there was some overlap due to the fact that query workloads may impact multiple resource dimensions. This overlap is why it is important to consider the experiment as a multi-class operational classification and not anomaly separation.

The model-performance table is used to establish the difficulty of the prediction task and also whether it is learnable [26]. Logistic Regression was problematic in this part due to the non-linear and interactive nature of the relationships between the DMV indicators and the labels of the bottlenecks. Random Forest and Gradient Boosting had decider paths that were a combination of concurrency, wait and execution metrics, which better captured these interactions. In the dataset, XGBoost did not perform as well as Random Forest, but was still close enough that it suggests that boosted trees are still viable candidates to be further tuned with more data. There was a tie on all test metrics between Random Forest and Gradient Boosting, so Random Forest was chosen as the model to interpret, as it is easier to explain to a DBA audience, and the feature importance is easy to extract.

Additional information is given in terms of class-level precision and recall. In the test set, the values for precision, recall and F1-score for both blocking and I/O were equal to 1.00, which means that they were separated completely. The precision and recall of the model to predict the missing index were 1.00 and 0.70, respectively, that is, when the model predicted a missing index, it was correct, but some missing indices were predicted as CPU or Normal. Among all of the classes, CPU had the lowest precision and recall of 0.64, indicating that it was the least stable class. The other way round some CPU cases fell in Normal was the same, with Normal having a precision of 0.65 and a recall of 0.85. One helpful operation hint: Always see CPU pressure before performing any DBA action, along with other indicators such as scheduler utilisation, runnable queue length, plan cost, resource governor information, etc.

The feature importance ranking can also justify the experimental design. Not surprisingly, Blocking_session_count was the highest-rated measure, since it directly measures the contention for locks. Waiting_tasks_count and wait_time_ms were second and third, indicating that waiting is a crucial aspect of behaviour and is necessary for performance classification. Other extremely high-ranked measures were total_elapsed_time and total_logical_reads, which are more associated with the time taken by a query and the cost of its access path, respectively. CPU and I/O aspects were captured in their parameters: Total_worker_time and total_physical_reads. Although it may not be as severe as missing_index_impact, it is still a factor and proves that the missing index diagnosis is not purely determined by the missing index suggestions provided by the DMV, which is known to require DBA validation. Rather, logical reads, elapsed time and execution count should be taken into consideration together with the evidence for missing indexes.

The figures and tables that are given as a result of this, complement each other. Data were balanced as observed in Figure 1, which helps in the fair evaluation of the models. Figure 2 shows the locations of the prediction error, which would help inform decisions to improve the model. Predictions are connected to operationally meaningful DMV evidence, aiding explainability, in Figure 3. The information about the dataset structure, the model's metrics, the stability of this model across the folds in the cross-validation, the report for each class and the recommendations are given in the tables.

The GenAI recommendation table is not referred to as an autonomous remediation engine. Instead, it demonstrates how classes that are likely to be a pain can be transformed into reviewable and

standardised guidance for responding to them, which makes sense for the DBA. For junior administrators, incident responders and service-management teams, this is especially critical, as they might become familiar with symptoms but not be immediately aware of the next step in the SQL Server diagnostic process. The recommendation layer thus takes the classification solutions one step further and is intended to support decision-making.

What these results mean for the practical situation is that a tool that is used by a DBA should not show only the predicted label. It should show the predicted label, important discriminating symptoms and a brief explanation of the overlapping symptoms. However, in this instance, if the CPU bottleneck is identified as a snapshot, and logical reads and missing-index are also identified as high, it may be a good idea to look at the execution plan and index design before deciding on a hardware scaling requirement. Similarly, if the two classes, normal and CPU, are comparable, the system should be designed to compare to a historical normal class. This "evidence-driven" presentation helps to minimise overreaction to one measure.

Based on the results, it was clear that a balanced data set was needed. If the normal state had been predominant in the sample, the classifier might have been able to give false accuracy due to overprediction of the normal state. The equal class design was used to provide equal weight to all scenarios, and the confusion matrix proved to be more helpful in making decisions about management. These outputs can be used together to make a case for how DMV telemetry can be translated to knowledge when added with modelling, interpretation and recommendation layers for SQL Server teams, specifically.

5. Discussion

5.1 Interpretation of predictive performance

The results show that SQL Server DMVs are able to provide useful signals for the purpose of proactive bottleneck classification, particularly when combined with wait statistics, execution and blocking evidence and evidence of missing indexes. This finding extends the existing Microsoft technical stance that DMVs can be utilised as a method to troubleshoot workload or performance issues [1–3] and also demonstrates that DMVs snapshots can be utilised as supervised learning inputs. The highest accuracy of 83.64% and macro F1-score of 83.89% were achieved by the best models that are suitable for decision support but not for unattended automation.

The results of Random Forest and Gradient Boosting are in line with the results from the literature on machine learning, which highlights the fact that tree-based ensembles are performing well on structured operational data where there are non-linear interactions among features [27]. There is a dependency between the DMV metrics. A missing index can cause an increase in Logical reads, elapsed time and Worker time; a blocking can cause an increase in elapsed time and in the number of waiting tasks; I/O pressure can have an impact on elapsed time and Wait duration [28]. Ensemble models are thus more appropriate than Logistic Regression, as they can use combinations of indicators to split, instead of having to use a linear boundary.

5.2 Practical and academic contribution

The results for the features are also of academic significance. Explainability studies support that making links with understandable evidence to the outputs of a predictive system is important to make the system more credible [29, 30]. The most important variables in this study are not arbitrary but are variables that are meaningful and can be used to explain the operation, such as blocking_session_count, waiting_tasks_count, wait_time_ms, logical reads, physical reads, and worker time. This alignment helps in making a suitable framework for DBAs' decision-making, and also helps the junior administrators to understand the reason behind the prediction of a class.

One real-life problem that is illustrated in the confusion matrix is the problem of overlapping bottlenecks. Sometimes, CPU, normal and missing-index records could be mistaken for each other, as the same SQL workload could say that it has been operating under CPU pressure, normal pressure or missing-index pressure. This is aligned with the database-performance literature that shows that, depending on the workload context, plan shape and the environment state, it is not possible to make tuning decisions based on the value of individual counters [31]. The outcome therefore supports, rather than weakens, the paper's argument because an explainable framework should include evidence and uncertainty instead of assuming that every bottleneck has a unique signature. The GenAI part is adding its value by enabling it to tackle one of the weaknesses of monitoring-only solutions. Having a dashboard that indicates that wait time is getting longer does not give an indication to a DBA. Some methods can generate human-readable instructions, but there are well-documented dangers of hallucination and unsafe advice [32]. This study addresses that risk by limiting the recommendations by stipulating the DBA rules. The framework is not a self-healing database in this regard; it is a protected decision support system to manage systems.

5.3 Comparison with traditional management methods

In comparison, this work is limited in tuning scope but more specific in explaining the bottleneck in terms of DMVs in SQL Server as compared with other tuning systems such as OtterTune, LlamaTune, and the like. It takes the point of view of the database and provides a database-specific experimental workflow with labelled DMV classes, compared to the anomaly detection research for AIOps [33]. The benefits of the practical implementation are the decrease in manual triage effort. The academic contribution is an integrated architecture that connects the telemetry, prediction, interpretability and safety of recommendations. Integration proves to be the most important aspect of the novelty of the study. There had been previous studies that used databases to learn settings or select settings that were successful [34, 35]. This work, is not about using SQL Server DMVs as indicators for possible bottlenecks, but about using them as operational telemetry to classify and explain possible bottlenecks. The previous studies on AIOps are focused on anomaly detection and root cause analysis at the service level, which do not include a SQL Server-specific DMV workflow [36, 37]. Hallucination and safety issues have been observed in previous GenAI studies, which also show great language synthesis powers [25, 38]. The present framework merges these streams to form a limited pipeline: gather telemetry data, identify a bottleneck, provide evidence of a bottleneck, and provide a recommended safe path forward. This is relevant from a systems-management perspective because organisations increasingly expect databases to function as observable, resilient, and proactive platforms. DBA work is not limited to indexing or query plans; it also encompasses incident triage,

communication with application teams, capacity planning, and governance [39]. The model can be used to determine the probable bottleneck classes and point to important DMV evidence especially when several symptoms occur at the same time - which can help decrease the mean time to diagnosis. However, the study also suggests that there is no such thing as using a supervised prediction to replace professional judgement. The CPU class was challenging since there is some CPU-related evidence intermingled with the normal and missing-index. This further emphasises the importance of confidence scores and explanations of features and human approval. It is also possible to suggest methodological improvements for future use, based on the results. The more granular the labels, the longer the collection window and the more sources from the DMV, the less the misclassification. Time-series features may not represent just a single particular "snapshot" of a trend, but rather the trend itself. Recommendations could be used in conjunction with organisation-specific runbooks using retrieval-augmented GenAI. The extensions would take the framework from an experimental decision support to a deployable AIOps for database platforms. It is consistent with modern human-in-the-loop AIOps operational reality: automated alerts help to accelerate the triage process and still allow for accountability to experts and human action. Because of this, databases must be able to maintain this balance as incorrect changes can have an impact on performance, availability, security and compliance all at the same time.

6. Strengths and Limitations

The principal strength of this study is its empirical design: it generates a DMV-based dataset, maintains balanced bottleneck classes, compares multiple machine-learning models, analyses feature importance, and incorporates a guarded recommendation layer. This integrated design enables prediction and explanation to be evaluated together. The workload scenarios, features, labels, and model metrics are documented sufficiently to support replication. Nevertheless, the study has several limitations. The controlled SQL Server 2025 Developer Edition and AdventureWorks2022 environment may not represent enterprise systems with larger schemas, higher concurrency, mixed OLTP/OLAP workloads, or cloud-specific resource patterns. The timestamp variable was excluded from modelling, and memory-pressure and tempdb-contention classes were not included in the final balanced dataset. GenAI recommendations were assessed using a predefined rubric rather than by multiple independent DBAs. Future deployments must also protect potentially sensitive DMV content, including query text, database names, user identifiers, and workload patterns, and must retain human review before any operational action.

7. Future Recommendations and Implications

Future research would need to further test the framework with more enterprise SQL Server workloads over a more extended monitoring window, while also allowing for increased concurrent sessions and other types of bottleneck classes, including memory pressure, tempdb contention, query-plan regression, and deadlock risk. The dataset needs to be enriched with Query Store, extended events, performance counters, storage latency metrics, etc., to enable prediction to use a richer set of observability signals with the DMV snapshots. A real-time dashboard needs to be created to display predicted class, confidence score, top evidence and GenAI recommendations. Pre-publication versions should run the time series models that are LSTM, autoencoder, and transformer

on time-series data for early warning prior to the degradation being apparent. In practice, organisations should start in read-only monitoring mode, look at the predictions and incident tickets and add role-based access, audit trails, model drift checks, and a feedback loop of accepted/rejected recommendations for future prompts and thresholds. Simulation, rollback, security, and governance controls are other areas where reinforcement learning could be considered later. They would enhance the generalizability of these developments and enable the framework to be more useful for managed service providers, administrators and enterprise reliability teams. They would also enable comparative benchmarking or traditional threshold-based alerting. Predictive latency is one indicator that can be measured to show the impact of predictive management, as well as administrator trust, recommendation acceptance, and incident-resolution time.

8. Conclusion

In this study, a predictive model to identify and resolve SQL Server performance bottlenecks has been designed and tested with the help of Generative AI and Dynamic Management Views. A balanced sample was taken of 825 DMV records for normal, CPU, I/O, blocking and missing-index scenarios. Random Forest and Gradient Boosting were both the most promising classifiers with performance of 83.64% accuracy and 83.89% macro F1 score. Blocking session count, waiting task count, wait time, elapsed time, logical reads, worker time, and physical reads were the most important indicators shown by the feature importance. The GenAI recommendation module helped provide root cause explanations of the classes predicted and safe DBA actions. The study brings a proactive, explanatory and guarded decision support approach for Systems Management of SQL Server systems. Though done under controlled conditions, the results indicate that DMV telemetry can be used to aid in the meaningful prediction and guidance for resolving bottlenecks. The study provides a plausible chain of steps from SQL Server observability to operational intelligence that leverages AI for database research and networked systems management practice. Its principal contribution lies in combining these elements within a single reproducible systems-management workflow for SQL Server environments.

Funding

No funding information has been provided. Authors should update this statement before submission.

Data Availability Statement

The experimental DMV dataset and Python/Jupyter analysis files can be made available on reasonable request or submitted as supplementary material.

Conflicts of Interest

Declare conflicts of interest or state “The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.” Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. Any role of the funders in the design of the study; in the collection, analyses or interpretation

of data; in the writing of the manuscript, or in the decision to publish the results must be declared in this section. If there is no role, please state “The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results”.

References

[1].Vellanki, R. B. (2025). SQL Server administration and maintenance: Best practices for modern environments. Journal of Computer Science and Technology Studies, 7(6), 839–848.

[2].Dai, T., He, J., Gu, X., & Lu, S. (2018). Understanding real-world timeout problems in cloud server systems. In 2018 IEEE International Conference on Cloud Engineering (IC2E) (pp. 1–11). IEEE.

[3].LaRock, T., & van de Laar, E. (2023). Querying SQL Server wait statistics. In Pro SQL Server 2022 wait statistics: A practical guide to analyzing performance in SQL Server and Azure SQL Database (pp. 27–63). Springer.

[4].Oliveira, G. C. d. (2025). Investigation of AI-based image, video, and voice analysis to assess clinical symptoms.

[5].Oloruntoba, N. (2025). AI-driven autonomous database management: Self-tuning, predictive query optimization, and intelligent indexing in enterprise IT environments. World Journal of Advanced Research and Reviews, 25(2), 1558–1580.

[6].Wu, Y., Zhou, X., Zhang, Y., & Li, G. (2024). Automatic index tuning: A survey. IEEE Transactions on Knowledge and Data Engineering, 36(12), 7657–7676.

[7].Narasayya, V., & Chaudhuri, S. (2021). Cloud data services: Workloads, architectures and multi-tenancy. Foundations and Trends in Databases, 10(1), 1–107.

[8].Nuriev, M., Zaripova, R., Potapov, A., & Kuznetsov, M. (2023). Achieving new SQL query performance levels through parallel execution in SQL Server. In E3S Web of Conferences (p. 04005). EDP Sciences.

[9].LaRock, T., & van de Laar, E. (2023). Wait statistics internals. In Pro SQL Server 2022 wait statistics: A practical guide to analyzing performance in SQL Server and Azure SQL Database (pp. 3–25). Springer.

[10].Lopes, P., & Lahoud, P. (2024). Learn T-SQL querying: A guide to developing efficient and elegant T-SQL code. Packt Publishing Ltd.

[11].Steenwinckel, B., De Paepe, D., Vanden Hautte, S., Heyvaert, P., Bentefrit, M., Moens, P., et al. (2021). FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning. Future Generation Computer Systems, 116, 30–48.

[12].Van Aken, D., Yang, D., Brillard, S., Fiorino, A., Zhang, B., Bilien, C., et al. (2021). An inquiry into machine learning-based automatic configuration tuning services on real-world database management systems. Proceedings of the VLDB Endowment, 14(7), 1241–1253.

[13].Lao, J., Wang, Y., Li, Y., Wang, J., Zhang, Y., Cheng, Z., et al. (2023). GPTuner: A manual-reading database tuning system via GPT-guided Bayesian optimization. arXiv preprint arXiv:2311.03157.

[14].Li, G., Zhou, X., & Cao, L. (2021). Machine learning for databases. In Proceedings of the First International Conference on AI-ML Systems (pp. 1–2).

[15].Othman, R. S., & Yasin, H. M. (2025). AI-driven database optimization: Machine learning applications in database management systems.

[16].Xing, S., Wang, Y., & Liu, W. (2025). Self-adapting CPU scheduling for mixed database workloads via hierarchical deep reinforcement learning. Symmetry, 17(7), 1109.

[17].Sirimalla, A. (2024). ML-driven performance tuning framework for heterogeneous unified databases. Frontiers in Computer Science and Artificial Intelligence, 3(1), 69–79.

[18].Valavala, M., & Alhamdani, W. (2021). Automatic database index tuning using machine learning. In 2021 6th International Conference on Inventive Computation Technologies (ICICT) (pp. 523–530). IEEE.

[19].Fatima, S., Hussain, A., Amir, S. B., Awan, M. G. Z., Ahmed, S. H., & Aslam, S. M. H. (2023). XGBoost and random forest algorithms: An in-depth analysis. Pakistan Journal of Scientific Research, 3(1), 26–31.

[20].Zhong, Z., Fan, Q., Zhang, J., Ma, M., Zhang, S., Sun, Y., et al. (2023). A survey of time series anomaly detection methods in the AIOps domain. arXiv preprint arXiv:2308.00393.

[21].Zhu, Y., Yuan, H., Wang, S., Liu, J., Liu, W., Deng, C., et al. (2025). Large language models for information retrieval: A survey. ACM Transactions on Information Systems, 44(1), 1–54.

[22].Famiglini, L., Campagner, A., Barandas, M., La Maida, G. A., Gallazzi, E., & Cabitza, F. (2024). Evidence-based XAI: An empirical approach to design more effective and explainable decision support systems. Computers in Biology and Medicine, 170, 108042.

[23].Sklavenitis, D., & Kalles, D. (2025). A scoping review and assessment framework for technical debt in the development and operation of AI/ML competition platforms. Applied Sciences, 15(13), 7165.

[24].Cheng, Q., Sahoo, D., Saha, A., Yang, W., Liu, C., Woo, G., et al. (2023). AI for IT operations (AIOps) on cloud platforms: Reviews, opportunities and challenges. arXiv preprint arXiv:2304.04661.

[25].Ciubotaru, B.-I. (2025). Generative AI and large language models: A comprehensive scientific review.

[26].Tran, R., Lan, J., Shuaibi, M., Wood, B. M., Goyal, S., Das, A., et al. (2023). The Open Catalyst 2022 (OC22) dataset and challenges for oxide electrocatalysts. ACS Catalysis, 13(5), 3066–3084.

[27].Taha, K. (2025). Tree-based ensemble learning models for protein-protein interactions detection: A review and experimental evaluation. BioData Mining.

[28].Li, A., Sudvarg, M., Liu, H., Yu, Z., Gill, C., & Zhang, N. (2022). Polyrhythm: Adaptive tuning of a multi-channel attack template for timing interference. In 2022 IEEE Real-Time Systems Symposium (RTSS) (pp. 225–239). IEEE.

[29].Matta, S. S., & Bolli, M. (2023). Trustworthy AI: Explainability and fairness in large-scale decision systems. Review of Applied Science and Technology, 2(04), 54–93.

[30].Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., et al. (2024). Interpreting black-box models: A review on explainable artificial intelligence. Cognitive Computation, 16(1), 45–74.

[31].Zbarcea, A., Tudose, C., & Boicea, A. (2025). Extending the migration from asynchronous to reactive programming in Java: A performance analysis of caching, CPU-bound, and blocking scenarios. Applied Sciences, 16(1), 90.

[32].Milani, A., Franzoni, V., Florindi, E., Omarbekova, A., Bekmanova, G., & Yergesh, B. (2025). When AI is fooled: Hidden risks in LLM-assisted grading. Education Sciences, 15(11), 1419.

[33].Friedrichs, M. (2025). Automated graph-based data integration and information fusion: An open-source ecosystem of tools with a focus on maintainability and usability.

[34].Aminzadeh, A., Sattarpanah Karganroudi, S., Majidi, S., Dabompre, C., Azaiez, K., Mitride, C., et al. (2025). A machine learning implementation to predictive maintenance and monitoring of industrial compressors. Sensors, 25(4), 1006.

[35].Vistorte, A. O. R., Deroncele-Acosta, A., Ayala, J. L. M., Barrasa, A., López-Granero, C., & Martí-González, M. (2024). Integrating artificial intelligence to assess emotions in learning environments: A systematic literature review. Frontiers in Psychology, 15, 1387089.

[36].Saha, A., & Hoi, S. C. (2022). Mining root cause knowledge from cloud service incident investigations for AIOps. In Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice (pp. 197–206).

[37].Szandała, T. (2025). AIOps for reliability: Evaluating large language models for automated root cause analysis in chaos engineering. In International Conference on Computational Science (pp. 323–336). Springer.

[38].Yigit, Y., Buchanan, W. J., Tehrani, M. G., & Maglaras, L. (2024). Review of generative AI methods in cybersecurity. arXiv preprint arXiv:2403.08701.

[39].Humphrey, T. N. (2025). Guardrails for growth: Governing open-source software risk in healthtech startups. University of Missouri–St. Louis.

[40].Olasupo, A., Ishola, B., Ajiboye, T. O., Esan, A. O., Idris, M. O., Lawal, B., & Suah, F. B. M. (2025). Sewage sludge: a source of renewable energy and resources for circular bioeconomy. Chemical Engineering Journal Green and Sustainable, 1, 100015. https://doi.org/10.1016/j.cejgas.2025.100015

[41].Ishola, B., & Heyes, C. (2025). Scientific generative AI and materials science: From prediction to physical intelligence. MRS Bulletin, 50(9), 1118–1119. https://doi.org/10.1557/s43577-025-00981-0

[42].Paul, J. E. J. D. (2023). Investigating blended instruction using Zearn Math in rural elementary second grade classrooms [Doctoral dissertation, Southern University and A&M College].