Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077691

ARTICLE

Multi-Agent Large Language Model-Based Decision Tree Analysis for Explainable Electric Vehicle Drive Motor Fault Diagnosis

Jaeseung Lee[1] **and Jehyeok Rew**[2,*]

[1]School of Electrical Engineering, Korea University, Seoul, Republic of Korea
[2]Department of Data Science, Duksung Women's University, Seoul, Republic of Korea

*Corresponding Author: Jehyeok Rew. Email: jhrew@duksung.ac.kr

Received: 15 December 2025; Accepted: 12 March 2026; Published: 09 April 2026

ABSTRACT: The accelerating transition toward electrified mobility has positioned electric vehicles (EVs) as a primary technology in modern transportation systems. In this context, ensuring the reliability of EV drive motors (EVDMs) becomes increasingly critical, given their central role in propulsion performance and operational safety. Accurate and interpretable fault diagnosis of EVDMs is therefore essential for enabling effective maintenance and supporting the broader sustainability and resilience of EVs. This study presents a novel framework that combines decision tree-based fault classification with a multi-agent large language model (LLM) interpretation architecture to deliver transparent and human-readable diagnostic explanations. The proposed framework integrates domain-specific decision rules derived from sensor measurements and utilizes specialized LLM agents to translate tree-based decision logic into coherent narratives. The multi-agent architecture decomposes complex diagnostic reasoning into modular subtasks, allowing for enhanced interpretability and facilitating practical understanding for vehicle engineers. Experimental results on a publicly available EVDM dataset demonstrate that the proposed framework maintains high classification accuracy while significantly improving explanation quality and trustworthiness relative to conventional rule-based and single-agent approaches. By coupling symbolic decision models with LLM-driven reasoning, this work contributes to the advancement of trustworthy artificial intelligence for energy and mobility systems, particularly in predictive maintenance and explainable fault diagnosis. The findings highlight the value of integrating classical machine learning with multi-agent LLMs to support reliable, transparent, and human-centered EV infrastructures.

KEYWORDS: Electric vehicle; drive motor; large language model; decision tree; explainable artificial intelligence; AI agent; fault diagnosis

1 Introduction

In recent years, the global automotive industry has undergone a significant transition driven by environmental sustainability concerns and the depletion of fossil fuels [1]. As a result, electric vehicles (EVs) have emerged as a promising alternative to conventional internal combustion engine vehicles. This transformation is not only reshaping transportation systems but also catalyzing advancements in the design, monitoring, and control of core EV components such as drive motors, batteries, and power electronics [2].

Among the critical components of EVs, the EV drive motors (EVDMs) play a vital role in vehicle propulsion by converting electrical energy into mechanical torque [3]. The performance of EVs is directly governed by the operational health of EVDMs. However, due to their continuous operation under varying thermal, mechanical, and electrical stresses, EVDMs are susceptible to various types of faults, including

phase-to-phase short circuits, insulation failures, and overloading [4]. When faults in the EVDMs remain undetected or are identified too late, they can lead to performance degradation and system-level failures. These issues not only compromise vehicle safety but also result in substantial maintenance costs and operational downtime. Therefore, developing intelligent and robust fault diagnosis systems for EVDMs is essential, particularly as the scale and complexity of EV deployment continue to grow [5].

Conventional fault diagnosis approaches for EVDMs have relied heavily on rule-based methods that utilize predefined thresholds, expert knowledge, and deterministic logic [3]. While these methods are simple to implement and interpret, they suffer from limited adaptability under complex operating conditions and unforeseen fault scenarios. Furthermore, they require extensive manual tuning and often fail to generalize across different motor types or operational environments.

To address these limitations, recent studies have increasingly employed machine learning (ML)-based techniques [6,7]. These models have demonstrated high diagnostic accuracy and the capability to detect subtle or incipient faults that are difficult to capture using conventional rule-based systems [8]. Despite their performance advantages, most ML models operate as 'black box' models. Their internal decision-making processes are opaque, making it challenging for vehicle engineers to trust or validate the predictions in safety-critical applications. This lack of interpretability poses a significant barrier to the widespread adoption of ML-based diagnostic models in industrial EV settings, where transparency and explainability are crucial for regulatory compliance, model validation, and real-time decision-making [9].

In response to the interpretability challenges of black-box models, various explainable artificial intelligence (XAI) techniques have been proposed [10], including Shapley additive explanations (SHAP) [11] and local interpretable model-agnostic explanations (LIME) [12]. These techniques aim to provide post-hoc explanations by quantifying the contribution of each input feature to a specific prediction [13,14]. Although such model-agnostic approaches have gained attention for their flexibility, they inherently operate as indirect interpretative tools that approximate, rather than reveal, the model's internal decision logic [15]. As a result, discrepancies may arise between the surrogate interpretation and the true reasoning process of the original model, particularly when nonlinear dependencies or complex feature interactions are involved.

In contrast, decision tree provides inherent interpretability, as their structure naturally encodes a sequence of logical rules that lead to a decision [16]. Unlike post-hoc methods, the decision-making process in a decision tree can be directly traced from root to leaf, allowing vehicle engineers to precisely observe how input features influence the final outcome. Moreover, the hierarchical structure of decision trees facilitates an intuitive understanding of feature importance, decision thresholds, and classification boundaries.

However, interpreting complex decision trees still requires considerable human effort. Vehicle engineers must manually trace decision paths and analyze multiple branching conditions to extract diagnostic insights, which can be both labor-intensive and cognitively demanding, especially when the tree is large or the rules are highly domain-specific. Furthermore, while the structure of decision trees is transparent, their interpretations often lack contextualization in terms of confidence estimation, causal inference, and practical recommendations, factors that are crucial for high-stakes domains such as EVDM fault diagnosis.

Recently, large language models (LLMs) have emerged as powerful tools capable of complex reasoning across diverse domains [17]. Owing to their ability to process structured inputs, capture logical relationships, and generate coherent textual outputs, LLMs are increasingly being explored as autonomous agents within multi-step analytical pipelines. This paradigm, known as the multi-agent LLM approach, enables the decomposition of complex problems into modular subtasks handled by specialized reasoning agents [18]. In the context of XAI, such LLM agent architecture provides strong potential for automating interpretive tasks that would otherwise require considerable human expertise and manual effort.

In this paper, we propose a novel framework that leverages multi-agent LLMs to automate the interpretation of decision tree-based classification models for EVDM fault diagnosis. The proposed framework decomposes the interpretation task into a structured pipeline of specialized LLM agents, each designed to a specific reasoning function. These agents collaboratively interpret decision paths, translate numerical thresholds into domain-relevant expressions, and analyze counterfactual scenarios. They further ensure consistency between local and global reasoning, generate structured diagnostic reports, and verify factual accuracy. This enables a high degree of automation and interpretability across diverse diagnosis contexts.

The main contributions of this paper are summarized as follows:

1. We propose a novel multi-agent LLM-based interpretive framework that explains decision tree-based fault diagnosis models for EVDMs by integrating structural parsing, logic extraction, uncertainty reasoning, and generating domain-level explanations.
2. We design prompt-driven reasoning agents that decompose the model's decision-making logic into interpretable and verifiable narratives, thereby enhancing transparency and usability for domain practitioners such as vehicle engineers and technicians.
3. We validate the proposed framework on a publicly available EVDM fault classification dataset, demonstrating its ability to generate coherent, faithful, and context-aware explanations.

The remainder of this paper is organized as follows. Section 2 reviews related works on EVDM fault diagnosis, XAI, and LLM-based agents. Section 3 presents the architecture of the proposed framework. Section 4 describes dataset, experimental setup, and evaluation metrics. Section 5 reports the experimental results, and Section 6 provides a detailed discussion of the findings. Finally, Section 7 concludes the paper and outlines directions for future research.

2 Related Works

2.1 Electric Vehicle Drive Motor Fault Diagnosis Using Machine Learning

With the increasing adoption of EVs, ensuring the reliability and safety of EVDMs has become a critical concern for both manufacturers and researchers. ML models have shown outstanding performance for enabling automated and accurate fault diagnosis by learning patterns from large-scale sensor datasets [19,20]. Recent studies have explored various ML-based approaches for detecting and classifying a wide range of EVDM faults under diverse operating conditions, outperforming conventional rule-based methods in both flexibility and diagnostic performance.

Thirunavukkarasu et al. [5] proposed an ML-based fault classification method for EVDMs under six operating conditions. Using advanced data transformations including Yeo–Johnson and Hyperbolic Sine, and evaluating multiple classifiers, they found that CatBoost achieved the highest accuracy. Xu et al. [21] proposed a deep learning-based fault diagnosis method for EVDMs using thermographic images and Inception V3 model. By integrating a Squeeze-and-Excitation attention mechanism and applying contrast-limited adaptive histogram equalization for contrast enhancement, their approach achieved high accuracy across 11 fault types. Dettinger et al. [9] presented a digital twin-based fault detection method for EV powertrains, deployed at the edge and connected via 5G. Using a 1-Nearest Neighbor algorithm trained on synthetic simulation data, the method enabled real-time detection and mitigation of faults such as demagnetization and switch failures.

Despite their high diagnostic accuracy, most ML models operate as black-box systems, providing limited insight into their internal reasoning processes. This lack of interpretability poses a significant limitation for safety-critical applications, where understanding the rationale behind a model's prediction is as important as the prediction itself.

2.2 Fault Diagnosis Using Explainable Artificial Intelligence

With the increasing complexity of ML models used for fault diagnosis, the lack of interpretability has emerged as a critical barrier to adoption in safety-critical domains such as EV systems. XAI techniques have been introduced to address this limitation by providing insights into the internal reasoning processes of predictive models [22]. Recent studies have explored various XAI methods, such as SHAP, LIME, and gradient-weighted class activation mapping (Grad-CAM), integrated into ML and deep learning models to improve fault attribution and interpret model decisions in both time-series and image-based diagnostic applications [14,23].

Haque et al. [3] proposed EnsembleXAI-Motor, a lightweight and interpretable framework for EVDM fault diagnosis. Their approach combined recursive feature elimination, ensemble learning, and LIME to achieve high diagnostic performance with minimal computational overhead. The model demonstrated generalizability across the EVDM dataset, while providing transparent explanations for its predictions. Jang et al. [24] proposed a fault diagnosis framework that integrates an adversarial autoencoder with SHAP values to enhance the interpretability of deep learning-based fault detection in industrial processes. Unlike conventional methods which rely solely on reconstruction error, their approach leverages both latent and reconstruction spaces, enabling more accurate fault attribution and deeper insight into variable contributions. Brito et al. [25] proposed FaultD-XAI, an interpretable fault diagnosis framework for rotating machinery that combines transfer learning from augmented synthetic data with Grad-CAM-based explainability. Their method used a 1-dimensional convolutional neural network to classify vibration signals and leveraged Grad-CAM to visually highlight influential signal regions, thereby increasing trust in model predictions.

While existing XAI techniques have enhanced model transparency through visual explanations such as feature importance plots and activation maps, they rely heavily on manual interpretation. Vehicle engineers need to manually analyze graphical outputs to derive meaningful insights, which can be cognitively demanding and prone to subjective bias. This highlights the need for more autonomous and semantically rich explanation methods that can translate visual information into human-interpretable textual insights.

2.3 Large Language Model-Based Agent

LLMs have rapidly evolved from passive information processors to autonomous agents capable of engaging in reasoning and decision-making [26]. LLM-based agents can dynamically interpret instructions and adapt their behavior using natural language [27]. These agents are increasingly being adopted in multi-step analytical pipelines, where each agent performs a specialized role within a large problem.

Holland and Chaudhari [28] demonstrated the practical application of LLM-based agents in industrial process planning for fiber composite structures. By integrating OpenAI GPT-4 with LangChain, they developed an autonomous agent capable of performing key planning tasks such as cycle time estimation and resource allocation. Peng et al. [29] proposed an adaptive LLM-based fault diagnosis framework for railway vehicle on-board controllers (VOBC) to address the limited suitability of general-purpose LLMs in railway-specific scenarios. Their method, named RFD-LLM, leverages low-rank adaptation and instruction tuning to align the model with VOBC fault patterns, achieving accurate identification of seven fault types. This work demonstrates the potential of domain-adapted LLMs as diagnostic agents in safety-critical railway systems. Lukens et al. [30] investigated the use of LLM-based agents as human-in-the-loop copilots for prognostics and health management tasks. Motivated by labor shortages and the need to preserve domain expertise, their framework integrated LLM-based agents into standard maintenance workflows to assist in data processing, failure-mode identification, and maintenance-recommendation generation.

Building on these developments, our study explores the use of multi-agent LLM systems to enhance the interpretability of safety-critical diagnostic models, in particular decision tree classifiers for EVDM fault diagnosis. In contrast to prior applications that primarily focused on task automation, we design modular agent architectures tailored specifically for explaining model behavior at multiple reasoning levels. To clarify the methodological differences and design motivations, Table 1 provides a qualitative comparison between post-hoc XAI methods, intrinsic tree-based explanations, and the proposed multi-agent LLM-based explanation framework for EVDM fault diagnosis.

Table 1: Comparison of explanation approaches for electric vehicle drive motor fault diagnosis.

Aspect	Post-Hoc XAI (e.g., SHAP and LIME)	Tree-Based Models	Proposed Multi-Agent LLM Framework
Explanation type	Post-hoc approximation	Intrinsic and rule-based	Intrinsic and language-based synthesis
Faithfulness to model	Approximate	Exact	Exact (structure-preserving)
Instance-level reasoning	Feature attribution scores	Explicit decision paths	Explicit decision paths with semantic interpretation
Global-local consistency	Not explicitly addressed	Implicit	Explicitly audited via dedicated agent
Counterfactual reasoning	Limited and heuristic	Implicit via thresholds	Explicit, minimal, and physically constrained
Human readability	Requires expert interpretation	Moderate (rule tracing)	High (natural language and structured reports)
Automation of interpretation	Low	Low	High (fully automated multi-agent pipeline)
Explanation validation	Not supported	Not supported	Explicit fact verification agent
Suitability for engineering diagnosis	Medium	Medium	High

3 Proposed Framework

This section describes the overall structure of the proposed framework. Fig. 1 presents overview of the proposed framework.

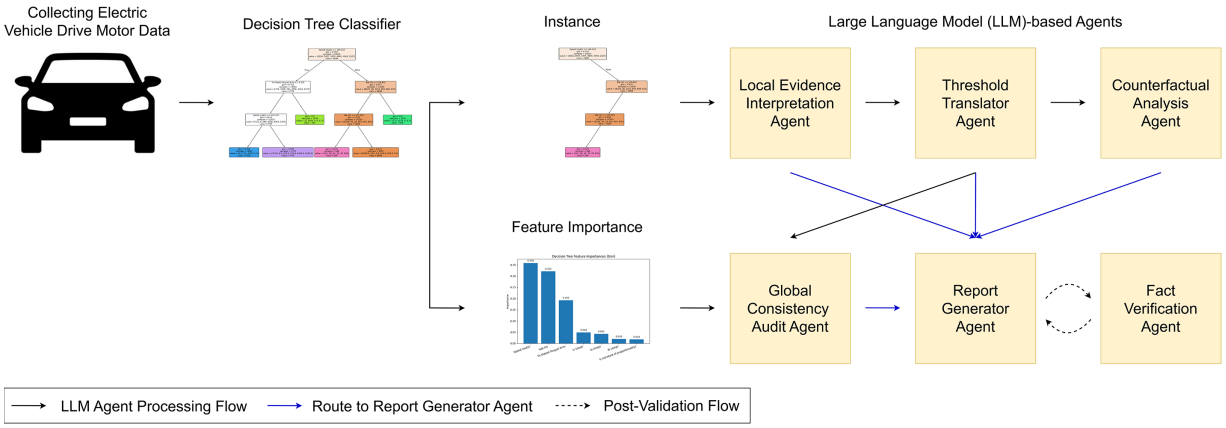

Figure 1: Overview of the proposed framework.

3.1 Electric Vehicle Drive Motor Fault Diagnosis Using Decision Tree

In this study, we employ a decision tree-based classification model to diagnose fault types in EVDMs. Fig. 2 describes the structure of the decision tree. The decision tree algorithm is well-suited for this task due to its hierarchical and rule-based decision-making structure [31]. Each prediction is derived from a sequence of conditional statements that trace a unique path from the root node to leaf node, thereby enabling transparent and interpretable reasoning. This property makes decision trees highly valuable in safety-critical domains, where understanding the decision rationale is as important as the prediction accuracy.

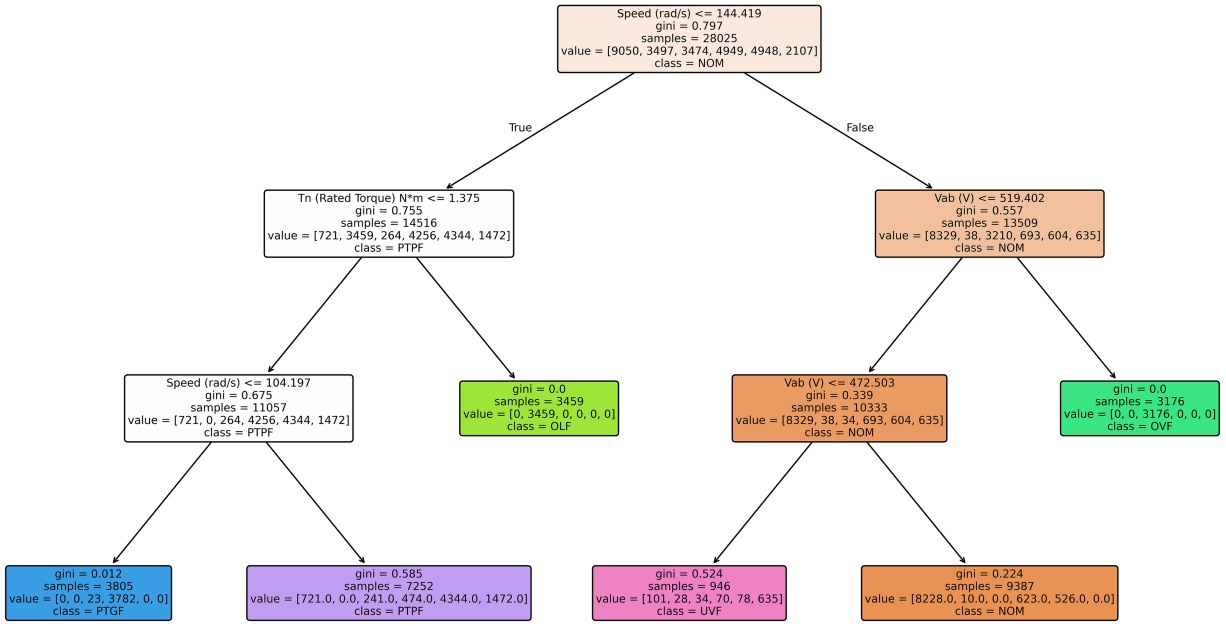

Figure 2: Visualization of the decision tree structure.

To optimize predictive performance while maintaining practical interpretability, hyperparameter tuning was conducted using the training and validation sets derived from the predefined train-validation-test split [32]. Among the tunable parameters, the maximum tree depth plays a central role, as it determines not only the expressive capacity of the classifier but also the structural complexity of the resulting decision paths.

While decision trees are inherently interpretable in a formal sense, increasing tree depth lengthens root-to-leaf decision paths and increases the number of conditional splits involved in each prediction [33]. This growth in structural complexity does not primarily impose a computation burden, given the lightweight nature of decision trees, but instead affects interpretability at the cognitive and explanatory levels. Deep trees yield highly fragmented decision logic with long rule chains, which can hinder human comprehension and reduce the stability and clarity of downstream LLM-based explanation generation.

Conversely, overly shallow trees tend to underfit the data, producing coarse decision boundaries that fail to capture subtle distinctions among fault types, thereby limiting both diagnostic accuracy and explanatory usefulness. The selected tree depth therefore reflects a balance between sufficient model expressiveness for reliable fault discrimination and a constrained decision structure that remains tractable for human interpretation and multi-agent LLM reasoning.

During the tuning process, candidate depths ranging from 3 to 12 are evaluated, and the optimal configuration is selected based on the macro-averaged f1-score on the validation set. The final decision tree achieves a strong balance between classification performance and structural simplicity, maintaining a compact hierarchy of decision rules. This structure serves as the analytical foundation for the subsequent multi-agent LLM-based interpretation framework, where specialized LLM agents systematically analyze, summarize, and contextualize the decision paths to generate human-readable diagnostic explanations.

3.2 Multi-Agent Large Language Model-Based Interpretation Framework

To automate and structure the interpretation of decision tree-based classification models for EVDM fault diagnosis, we propose a multi-agent LLM-based framework composed of six specialized agents. Each agent performs a distinct cognitive function to collectively derive a transparent and human-interpretable understanding of the model's decision process. Table 2 describes the overall structure of the proposed framework. The following subsections describe the functionality and reasoning flow of each agent in detail. Due to the length, prompt templates used to guide each agent are provided in Appendix A.

Table 2: Overall structure of multi-agent large language model-based interpretation framework.

Agent Name	Primary Objective	Main Inputs	Expected Outputs	Core Reasoning Tasks
Local evidence interpretation agent	Explain how each node's feature-threshold pair contributes to the final class by analyzing the root-to-leaf decision path.	- Decision path Javascript object notation (JSON) (feature, threshold, value, impurity, direction, node samples) - Predicted class	- Node-wise feature contributions (impurity drop, probability) - Textual explanation of feature impacts	- Quantify local evidential strength - Rank key features Generate interpretable "If–Then" rules per path
Threshold translator agent	Translate numeric split thresholds into domain-relevant linguistic expressions to improve human interpretability.	- Node-level numeric thresholds and sample values - Global feature statistics (mean, standard deviation (SD), interquartile range)	- Natural-language threshold expressions (e.g., "moderately high vibration") - Quantitative reference summary	- Map numeric values to semantic ranges - Preserve numerical traceability - Harmonize feature terminology

(Continued)

Table 2 (continued)

Agent Name	Primary Objective	Main Inputs	Expected Outputs	Core Reasoning Tasks
Counterfactual analysis agent	Identify minimal and plausible feature perturbations that would reverse the current prediction.	- Decision path (features, thresholds) - Feature ranges or physical limits	- Counterfactual candidates (feasibility, new class) - Summary of decision boundary sensitivity	- Detect threshold features - Compute minimal perturbations - Evaluate realism
Global consistency audit agent	Compare the local feature explanations with global feature importance patterns in the model.	- Local feature importance - Global model-level feature importances	- Quantitative alignment (Jaccard similarity, rank correlation) - Qualitative audit report	- Compare local vs. global rankings - Flag inconsistencies in reasoning patterns
Report generator agent	Integrate all agent outputs into a unified textual diagnostic report explaining the model's reasoning process.	- Outputs from above all four agents - Predicted class and probability	- Structured report - Sections: prediction summary, key rules, thresholds, counterfactuals, consistency, final summary	- Synthesize evidence - Generate cohesive, traceable narrative - Maintain scientific tone and structure
Fact verification agent	Verify factual of the report and refine linguistic clarity for publication readiness.	- Generated report from report generator agent - Model metadata (features, probabilities, global feature importance)	- Validated and corrected report - Log of inconsistencies and style refinements	- Cross-check fidelity - Ensure logical consistency - Standardize technical language and tone

3.2.1 Local Evidence Interpretation Agent

The local evidence interpretation agent is designed to explicitly reconstruct the instance-specific reasoning process embedded in the decision tree, serving as the foundational interpretability layer of the proposed framework. Unlike conventional post-hoc explanation methods that approximate model behavior through feature attribution scores, this agent operates directly on the symbolic structure of the trained decision tree, ensuring faithful and traceable explanations.

Given the root-to-leaf decision path of a specific sample, the agent analyzes each decision node to determine how the observed feature value satisfies or violates the corresponding split condition. For every node along the path, it quantitatively measures both the impurity reduction and the associated shift in class probability. This dual analysis enables the agent to capture not only which features contribute to the prediction, but how and when they influence the decision during the hierarchical reasoning process.

A key motivation for this agent is to decouple model-intrinsic causal evidence from downstream interpretive tasks such as domain contextualization or counterfactual reasoning. The local evidence interpretation agent intentionally focuses on explaining what the model used to reach its decision, without introducing domain semantics or speculative interpretations. This separation ensures that subsequent agents operate on a stable explanation backbone rather than aggregated importance measures.

Each decision node is translated into a human-readable 'If-Then' rule, and these rules are ranked according to a composite evidential score combining impurity decrease and probability gain. The resulting ordered rule set provides a transparent and structured account of the decision tree's internal logic, forming a standardized intermediate representation that is reused by all subsequent agents in the pipeline. In this

way, the local evidence interpretation agent contributes not merely as an engineering aid, but as a structural explanation engine that anchors the entire multi-agent interpretability framework.

3.2.2 Threshold Translator Agent

The threshold translator agent transforms numeric split thresholds of the decision tree into distribution-aware and domain-aligned semantic descriptions, thereby addressing a key gap between model-internal decision logic and human diagnostic reasoning. Rather than performing a superficial numeric-to-text conversion, this agent interprets each threshold relative to the empirical distribution of the corresponding feature, enabling consistent and context-sensitive explanations across operating regimes.

The agent takes as input the node-level outputs generated by the local evidence interpretation agent in Section 3.2.1, including feature identities, threshold values, and instance-specific measurements. For each split condition, it evaluates the position of the sample value and threshold within the global feature distribution using summary statistics such as the mean, SD, and interquartile range. Based on this comparison, the agent maps numeric boundaries to percentile-informed semantic categories (e.g., low, medium, and high), ensuring that linguistic descriptions reflect the data-driven context in which the model operates.

To avoid conflating model-internal causality with semantic interpretation, this agent decouples structural causality from semantic contextualization. While the local evidence interpretation agent explains which rules were applied, the threshold translator agent explains what those rules imply in terms of physical or operational conditions. This separation prevents premature domain interpretation from contaminating model-faithful explanations, while still enabling downstream agents to reason in a human-centered manner.

Each semantic description is paired with its original numeric threshold and sample value, preserving traceability to the underlying model logic. By providing standardized and distribution-aware linguistic anchors, the threshold translator agent supports subsequent agents, and ensures that explanations remain interpretable and consistent across different instances and diagnostic contexts. In this way, the agent contributes not merely to readability, but to the semantic robustness and cross-agent coherence of the overall multi-agent interpretation framework.

3.2.3 Counterfactual Analysis Agent

The counterfactual analysis agent analyzes the local decision stability of the trained decision tree by identifying minimal and physically plausible perturbations that would alter the model's predicted fault class. Rather than generating generic 'What-If' explanations, this agent operates directly on the decision tree's branching structure to reveal how sensitive a specific prediction is to changes in individual features.

The agent receives as input the interpreted decision path and semantically contextualized threshold information generated by the threshold translator agent in Section 3.2.2. For each decision node along the path, it evaluates how close the instance lies to the corresponding split boundary and systematically explores alternative routing scenarios by modifying one feature at a time near its critical threshold. This process identifies the smallest directional change required to redirect the instance to a different leaf node associated with an alternative fault class.

The inclusion of this agent reflects a deliberate architectural decision to separate causal explanation from decision robustness analysis. While preceding agents explain why a prediction was made, the counterfactual analysis agent explains how easily it could change. Each counterfactual scenario is evaluated under physical feasibility constraints derived from EVDM operational limits, ensuring that suggested perturbations correspond to realizable system states rather than abstract mathematical artifacts.

For each valid counterfactual, the agent reports the magnitude and direction of the required feature change, along with the expected class transition. These scenarios are ranked according to minimality and interpretability, prioritizing changes that are both actionable and diagnostically meaningful. By exposing the local decision boundaries of the model, the counterfactual analysis agent provides insights into fault sensitivity and design confidence, thereby extending interpretability beyond static explanations to include dynamic decision behavior.

3.2.4 Global Consistency Audit Agent

The global consistency audit agent assesses the reliability and representativeness of instance-level explanations by examining their alignment with the model's global decision behavior. While local explanation agents reveal how a specific prediction was produced, they do not indicate whether the resulting reasoning pattern is consistent with the dominant mechanisms learned by the model. This agent addresses the gap by performing a meta-level interpretability audit.

The agent takes as input the locally salient features identified along the decision path, together with global feature importance profiles extracted from the trained decision tree. It quantitatively evaluates consistency using ranking-based metrics, such as overlap ratios and rank correlation, to determine the degree to which features emphasized locally also play a significant role in the model's overall fault discrimination strategy.

Beyond numerical comparison, the agent performs a qualitative assessment to interpret the nature of any detected discrepancy. Rather than treating all mismatches as errors, it distinguishes between fault-specific local specializations and potentially unreliable explanations driven by low-support branches. When inconsistencies arise, the agent annotates them with interpretive notes, signaling to vehicle engineers that the explanation may reflect localized behavior not representative of the model's global logic.

By auditing coherence between local fidelity and global interpretability, the global consistency audit agent strengthens trust in the explanatory pipeline and prevents overreliance on locally persuasive but globally unrepresentative reasoning patterns. This step elevates the framework from providing descriptive explanations to delivering validated and confidence-aware interpretability for EVDM fault diagnosis.

3.2.5 Report Generator Agent

The report generator agent performs interpretive synthesis and structural alignment of the heterogeneous outputs generated by the preceding agents in Sections 3.2.1–3.2.4. While earlier agents generate complementary but fragmented explanations, covering local causality, semantic contextualization, decision boundary sensitivity, and global consistency, these outputs should be coherently integrated to support reliable human understanding and decision-making.

The agent organizes the final diagnostic report according to a standardized explanation schema, consisting of: (1) a summary of the predicted fault class and its probability, (2) key decision rules with their respective feature contributions, (3) translated threshold interpretations, (4) counterfactual insights describing boundary conditions, (5) a consistency statement referencing the model's global importance profile and (6) a final summary of the report. Each section is linked to its underlying numerical evidence, ensuring traceability between the textual explanation and the decision tree's internal computations.

A key motivation for this agent is to decouple explanation generation from explanation synthesis. Rather than merely concatenating outputs, the report generator agent resolves potential redundancies or emphasis conflicts across agents, prioritizes diagnostically salient evidence, and ensures logical coherence

across explanation layers. This process standardizes explanation structure across instances, improving comparability, reproducibility, and interpretive clarity.

By transforming distributed reasoning artifacts into a unified and validated narrative, the report generator agent elevates the framework beyond fragmented explanation delivery. It enables vehicle engineers to systematically understand not only what the model predicted, but how multiple strands of evidence jointly support that conclusion, thereby facilitating informed and trustworthy decision-making in EVDM fault diagnosis.

3.2.6 Fact Verification Agent

The fact verification agent is introduced as a model-faithfulness enforcement mechanism that ensures the integrity of the final diagnostic explanation generated by the proposed framework. While preceding agents focus on generating, contextualizing, auditing, and synthesizing explanations, the explicit enforcement of numerical and logical consistency with the underlying decision tree computations is not addressed at earlier stages. This agent is designed to fill the gap.

The agent validates every quantitative statement in the final report by cross-referencing it against the original model artifacts, including impurity reductions, class probabilities, threshold values, and counterfactual perturbations. By checking that all reported figures correspond exactly to the decision tree's internal statistics, the agent prevents subtle numerical drift and cumulative inconsistencies that can arise from multi-stage LLM-based reasoning.

Beyond numerical verification, the agent performs explanation-level consistency checks to ensure that linguistic claims do not exceed what is supported by the model structure or data distribution. Rather than introducing new interpretations, it constrains the explanation space by removing unsupported assertions, harmonizing technical terminology, and standardizing phrasing across sections. When uncertainty or limited evidential support is detected, the agent annotates the corresponding statements with confidence indicators, signaling potential reliability limitations to human users.

The motivation for this agent lies in separating interpretation generation from interpretation validation. By acting as a final anchoring stage that rebinds natural language explanations to model-grounded evidence, the fact verification agent enhances explanation credibility, reproducibility, and deployment readiness. This final verification step ensures that the proposed framework delivers explanations that are not only interpretable, but also faithful, auditable, and trustworthy for EVDM fault diagnosis.

4 Experimental Setup

4.1 Dataset

To evaluate the effectiveness of the proposed framework, we utilize the publicly available EVDM dataset that encompasses various operating and fault conditions of a three-phase induction motor [34]. The dataset comprises time-series measurements of electrical, mechanical, and control variables. It is formatted as a tabular dataset for classification purposes.

As shown in Table 3, the dataset includes seven input variables, which represent a combination of physical measurements and control parameters. Electrical quantities contain the three-phase currents Ia (Amp), Ib (Amp), Ic (Amp), and the line voltage Vab (V), alongside mechanical feature such as rotational speed. Control parameters, including the Tn (rated torque) Nm and k (constant of proportionality), are also included. All input variables are continuous. Table 4 presents the summary statistics for the input variables of the dataset. It is noted that three missing values were identified in the Ic (Amp). Therefore, the corresponding records were removed, resulting in a total of 40,037 valid samples being used for subsequent analysis.

The output variable is a categorical label that indicates the fault status of the motor. As summarized in Table 5, the dataset contains six classes: one representing normal operation (NOM) and five representing distinct fault types, including phase-to-phase fault (PTPF), phase-to-ground fault (PTGF), overvoltage fault (OVF), overload fault (OLF), and undervoltage fault (UVF).

Table 3: Input variables of electric vehicle drive motor dataset.

Input Variable	Description	Variable Type
Tn (rated torque) Nm	Torque in Newton meters	Continuous
k (constant of proportionality)	Constant used in the drive model	Continuous
Ia (Amp)	Current in phase A in Amperes	Continuous
Ib (Amp)	Current in phase B in Amperes	Continuous
Ic (Amp)	Current in phase C in Amperes	Continuous
Vab (V)	Voltage between phases A and B in Volts	Continuous
Speed (rad/s)	Speed in radians per second	Continuous

Table 4: Summary statistics of the input variables of electric vehicle drive motor dataset.

Input Variable	Count	Mean	Standard Deviation	Minimum Value	25%	50%	75%	Maximum Value
Tn (rated torque) Nm	40,040	1.138	0.371	0.800	0.918	1.037	1.181	2.500
k (constant of proportionality)	40,040	0.001	0.000	0.000	0.001	0.001	0.001	0.002
Ia (Amp)	40,040	17.897	10.673	0.000	11.470	13.746	23.355	78.639
Ib (Amp)	40,040	17.764	10.472	0.000	11.470	13.783	22.572	79.113
Ic (Amp)	40,037	18.467	10.509	0.000	11.741	14.274	24.037	78.738
Vab (V)	40,040	432.839	123.434	0.000	389.693	474.566	474.984	649.516
Speed (rad/s)	40,040	123.521	49.801	−112.258	132.843	142.900	147.744	182.599

Table 5: Output variables of electric vehicle drive motor dataset.

Output Variable	Number of Counts	Description
NOM	13,013	Normal operation
PTPF	7007	Phase-to-phase fault (phase B and phase C)
PTGF	7007	Phase-to-ground fault (phase A and ground)
OVF	5005	Overvoltage fault
OLF	5005	Overload fault
UVF	3003	Undervoltage fault

4.2 Experimental Settings

The performance of the proposed framework was evaluated in two stages: the training and selection of the base classification model, and explanation generation using a multi-agent LLM pipeline.

For the classification model, we employ a decision tree classifier [16] implemented with the scikit-learn library [35]. To balance predictive performance with interpretability, hyperparameter tuning was conducted on the maximum tree depth, which directly controls the complexity of the decision structure and the number of rule branches. The dataset was partitioned into training, validation, and test sets with a 70:10:20 ratio. The training and validation sets were used to perform a grid search over the maximum depth parameter, ranging from 3 to 12 in increments of 1. For each candidate depth, the model was trained on the training set and evaluated on the validation set using the macro-average F1-score. The configuration achieving the best validation performance was selected, and the final model was evaluated on the test set.

For the interpretation phase, all agents in the proposed framework are powered by OpenAI GPT-5, accessed via the official OpenAI application programming interface (API) [36]. Each agent operates with a role-specific system prompt and receives structured inputs derived from the trained decision tree, including decision paths, feature thresholds, impurity values, and class probability distributions. To ensure stable and reproducible outputs, deterministic decoding was employed with the temperature fixed at 0.2 and stochastic sampling disabled. The agent outputs were constrained to structured formats such as JSON and tabular representations, enabling downstream aggregation, verification, and evaluation.

The agents are executed sequentially in a fixed pipeline without shared hidden states, ensuring modularity, traceability, and reproducibility of the interpretation process. This design allows each agent to focus on a clearly defined reasoning task while maintaining transparent links between the generated explanations and the underlying model computations.

4.3 Evaluation Metrics

To assess both the performance of the decision tree classifier and the effectiveness of the proposed framework, we use a combination of quantitative and qualitative evaluation metrics.

For the classification task, we employ four standard metrics commonly used in multi-class fault diagnosis: precision, recall, F1-score, and accuracy [37]. Precision indicates the ratio of correctly predicted positive samples to all instances predicted as positive by the model. Recall reflects the model's capability to capture actual positive samples among all true positives. The F1-score, defined as the harmonic mean of precision and recall, serves as a balanced indicator when both measures are of equal importance. Accuracy expresses the proportion of correctly identified samples relative to the total number of observations. The mathematical formulations of these metrics are given in Eqs. (1)–(4).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{1}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + False\ Negative + True\ Negative} \tag{4}$$

To comprehensively evaluate the interpretability of the proposed framework, we adopt a hybrid qualitative evaluation approach that integrates both LLM-as-a-Judge [38] and human expert evaluation [39]. In this dual assessment scheme, the LLM and domain experts independently evaluate the quality of generated explanations using the same predefined criteria summarized in Table 6.

Table 6: Evaluation criteria of large language model-generated explanations assessed on a 5-point Likert scale.

Criterion	Description
Explanation accuracy	Does the generated explanation correctly describe the decision logic of the model?
Logical consistency	Is the reasoning within the explanation coherent and free from contradictions?
Faithfulness to model behavior	Does the explanation accurately reflect the internal reasoning process of the decision tree?
Domain relevance	Does the explanation appropriately use terminology and reasoning relevant to the domain?
Clarity and interpretability	Is the explanation clear, concise, and easy to understand for vehicle engineers?

The LLM-as-a-Judge leverages GPT-5 as an autonomous evaluator capable of providing consistent judgements on textual explanations. This approach ensures linguistic precision, thereby minimizing subjective bias and enabling the reliable assessment of textual coherence. In parallel, a human evaluation was conducted by domain experts specializing in EVDM systems. The experts evaluated the same set of explanations to provide practical perspectives on their interpretability and diagnostic usefulness. Both the LLM and human evaluators were assigned using a five-point Likert scale [40], where a score of 1 denotes very poor quality and 5 denotes excellent quality.

In addition to numerical ratings, human evaluators provided open-ended feedback regarding both the strengths and areas for improvement in the generated explanations. These responses were analyzed to identify subjective perceptions and contextual insights that might not be captured through quantitative scoring.

5 Experimental Results

5.1 Evaluation of Electric Vehicle Drive Motor Fault Diagnosis

We evaluate the classification performance of the decision tree-based EVDM fault diagnosis model by varying the maximum tree depth from 3 to 12. Table 7 summarizes the results in terms of precision, recall, F1-score, and accuracy. The boldface values indicate the best performance. As the tree depth increases, the model shows improved classification performance across all metrics, with F1-score and accuracy both rising from depth 3 to depth 10.

At lower depths, including maximum depths of 3 and 4, the model exhibits modest performance, which can be attributed to underfitting resulting from coarse decision boundaries. As the depth increases to 5 and beyond, the model becomes more expressive and capable of capturing finer-grained decision patterns, leading to consistent improvements across all metrics. However, beyond a certain threshold, the performance gains begin to plateau, suggesting diminishing returns from further increases in model complexity.

The best overall performance is obtained when the maximum depth is set to 10. Beyond this point, further increases in depth do not yield significant improvements and may introduce unnecessary model complexity. Therefore, we select maximum depth of 10 as the final configuration for all subsequent experiments. This configuration provides an effective balance between predictive accuracy and model interpretability, enabling a tractable decision structure that facilitates subsequent analysis.

Table 7: Fault diagnosis performance of the electric vehicle drive motor fault diagnosis model.

Maximum Depth	Precision	Recall	F1-Score	Accuracy
3	0.8646	0.7969	0.8105	0.8438
4	0.8698	0.8679	0.8515	0.8603
5	0.9005	0.9009	0.8974	0.8945
6	0.9471	0.9037	0.9210	0.9096
7	0.9490	0.9226	0.9340	0.9210
8	0.9457	0.9314	0.9382	0.9247
9	0.9440	0.9389	0.9413	0.9271
10	**0.9498**	**0.9437**	**0.9466**	**0.9328**
11	0.9488	0.9423	0.9453	0.9319
12	0.9460	0.9435	0.9447	0.9298

5.2 Multi-Agent Large Language Model-Based Decision Tree Analysis

This section describes the interpretation outputs produced by each agent in the proposed framework. Fig. 3 presents the visualization of the instance-specific decision path in the decision tree that we analyzed in this section. Due to the length, detailed results are provided in Appendix B.

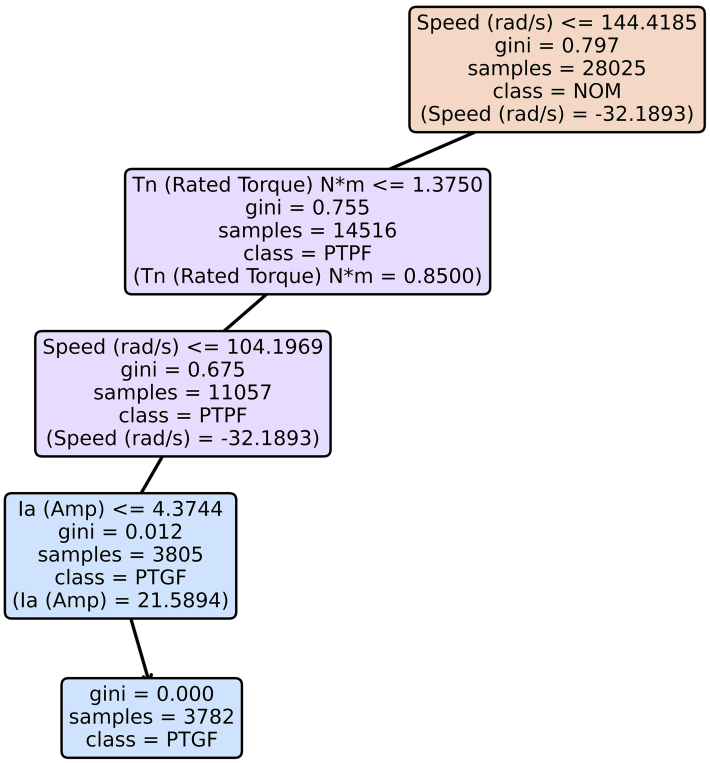

Figure 3: Visualization of the instance-specific decision path in the decision tree.

5.2.1 Local Evidence Interpretation Agent

We evaluated the performance of the local evidence interpretation agent in generating textual explanations for a specific instance. Tables A7–A9 present the complete local interpretation process performed

by the local evidence interpretation agent for a representative sample instance that was classified as a PTGF. Table A7 lists the original decision path extracted from the trained decision tree, detailing each node's feature, threshold, impurity, and class assignment. Table A8 reformats this information into a structured JSON representation, which serves as the standardized input format for the proposed framework. Table A9 presents the output generated by the agent, including node-wise impurity drops, probability shifts, and textual explanations that describe how each decision rule contributes to the final classification outcome.

The agent parsed the decision trajectory from the root to the leaf node and generated interpretable textual explanations for each feature and threshold condition. For sample index 69, the diagnostic path excluded nominal and non-ground fault conditions through sequential threshold evaluations. The initial condition, Speed (rad/s) $\leq$ 144.4185, indicated abnormally low or negative rotation, directing the instance away from normal operation. The following split on Tn (Road Torque) N$*$m $\leq$ 1.375 reflected a low-torque state associated with potential fault zones. A second speed-based threshold, Speed (rad/s) $\leq$ 104.1969 reinforced the connection between reduced speed and ground-related faults. The decisive factor emerged at node 3, where Ia (Amp) = 21.5894 exceeded the threshold (4.3744 A), resulting in a complete impurity reduction and a probability shift of 1.0 toward PTGF.

Collectively, these results demonstrate that the interaction between abnormally low rotational speed and elevated Ia (Amp) serves as the dominant diagnostic evidence for the PTGF class. The agent successfully transformed symbolic tree rules into coherent and human-readable reasoning chains, thereby enhancing both the interpretability and diagnostic transparency of the underlying model.

5.2.2 Threshold Translator Agent

The threshold translator agent refines the numerical split conditions produced by the decision tree into domain-relevant linguistic expressions to enhance interpretability for vehicle engineers. This agent serves as an intermediate reasoning layer that connects numerical decision boundaries with domain semantics, enabling the decision tree's rules to be interpreted in physically meaningful terms.

Table A10 presents the translated outputs generated by the threshold translator agent. The agent interprets the decision path as a sequence of semantically enriched diagnostic conditions. Specifically, the rotational speed of −32.19 rad/s ($\leq$144.4185 rad/s) is described as abnormally low or negative, indicating a non-operational state far below the nominal range. The rated torque of 0.85 N$*$m ($\leq$1.375 N$*$m) is expressed as below nominal torque, reflecting a light-load or near-idle operation. A subsequent speed threshold ($\leq$104.20 rad/s) is interpreted as persistently very low rotational speed, reinforcing the diagnosis of an off-nominal and quasi-stationary condition. Finally, the Phase-A current of 21.5894 A (>4.3744 A) is translated as abnormally high, signifying a severe overcurrent or grounding anomaly within phase A.

The agent's final narrative summary consolidates these interpretations, stating that the examined motor operates at near-zero mechanical load and extremely low speed while exhibiting a severe phase, an overcurrent anomaly, conditions collectively aligned with a PTGF diagnosis. Through this translation process, the threshold translator agent bridges quantitative model logic and qualitative engineering reasoning, producing human-understandable diagnostic statements without sacrificing numerical precision.

5.2.3 Counterfactual Analysis Agent

The counterfactual analysis agent identifies the minimal and physically plausible perturbations in input features that would alter the model's current prediction. By exploring 'what-if' scenarios within the learned decision boundaries of the tree, the agent provides actionable insight into the stability of model decisions.

Table A11 presents the outputs of counterfactual analysis agent. The agent determined four feasible perturbations capable of reversing the PTGF decision. The smallest adjustment is a modest increase in rated torque from 0.85 to 1.3751 N∗m, which redirects the decision path toward PTPF. Alternatively, a reduction in phase-A current from 21.5894 to 4.3744 A also eliminates the PTGF signature, rerouting the instance to a non-ground fault region dominated by the PTPF. Larger increases in rotational speed yield more drastic decision changes: surpassing 104.1969 rad/s (node 2) reclassifies the sample as PTPF, while exceeding 144.4185 rad/s (root node) results in NOM.

The summary of the agent indicates that decision reversibility is most sensitive to early-path thresholds and to the decisive current-based split at node 3. Among the identified counterfactuals, small torque adjustments and current reductions are considered both physically feasible and operationally meaningful, whereas large speed increases are less practical under real-world conditions. This analysis enhances interpretability by linking feature-level perturbations to tangible diagnostic actions.

5.2.4 Global Consistency Audit Agent

The global consistency audit agent evaluates the alignment between the local explanation derived from a specific decision path and the global feature importance distribution. The agent assesses whether the locally dominant features for a given instance align with the globally influential variables.

Table A12 presents the consistency evaluation result generated by global consistency audit agent. The local top features identified along the decision path are Ia (Amp), Speed (rad/s), and Tn (Rated Torque) Nm, whereas the global model importances highlight Speed (rad/s), Vab (V), and Tn (Rated Torque) Nm as the most influential. The resulting Jaccard similarity of 0.5 indicates moderate overlap, while the rank correlation of −0.4 reveals divergent ordering, primarily due to the strong local dominance of Ia (Amp) that is globally underrepresented.

Qualitatively, this discrepancy arises because the global importances are averaged across multiple fault types, elevating features such as Speed (rad/s) and Vab (V) that are broadly predictive across classes, whereas Ia (Amp) is highly concentrated in the PTGF decision subspace. The agent interprets this outcome as partial consistency: while the local reasoning path appropriately relies on operational parameters such as Speed (rad/s) and Tn (Rated Torque) Nm that align with global importance features, it also demonstrates an additional dependence on Ia (Amp) that is comparatively underrepresented in the global analysis.

From a diagnostic perspective, this finding underscores the complementary natures of local and global explanations. While global analyses capture general operating regimes of instance-specific paths such as PTGF depend on localized evidence concentrated in single phases or current channels. As a result, integrating path-level interpretive analyses into global summaries can provide a more comprehensive representation of fault-specific mechanisms, reinforcing the transparency of diagnostic reasoning.

5.2.5 Report Generator Agent

The report generator agent integrates the outputs from all preceding agents in Sections 5.2.1–5.2.4 into a unified and human-readable report that explains the model's reasoning process. This agent combines quantitative evidence, linguistic threshold translations, counterfactual reasoning, and global-local consistency assessments into a coherent narrative suitable for expert interpretation.

Table A13 presents the generated report. The prediction summary describes the sequential rule evaluations on rotational speed, torque, and phase-A current that progressively guided the sample toward the PTGF leaf node, culminating in a zero-impurity decision. The key rules highlight the critical thresholds, abnormally low speed (−32.1893 rad/s ≤ 144.4185 rad/s), low torque (0.85 N∗m <= 1.375 N∗m), and elevated phase-A

current (21.5894 A > 4.3744 A) that collectively define the fault condition. Correspondingly, the threshold interpretations provide natural-language descriptions contextualized to the EVDM domain, explaining how these values represent counter-rotating, low-load, and overcurrent states, respectively.

The counterfactual analysis section summarizes minimal feature adjustments capable of reversing the classification outcome. Small increases in torque (+0.5251 N∗m) or moderate reductions in phase-A current (−17.2151 A) redirect the instance toward PTPF, while large speed increases (+176.6079 rad/s) transition it into the NOM branch. These findings highlight the model's sensitivity to torque-current interactions and its physical interpretability in fault-state transitions. The global consistency audit further indicates partial alignment between local and global explanations (Jaccard = 0.50, $p = -0.40$), showing that while mechanical variables (speed, torque) are consistently influential, the PTGF pathway depends strongly on localized current evidence, Ia (Amp), that is underrepresented in global profiles.

Overall, the consolidated narrative of the report generator agent provides a transparent and traceable account of the model's diagnostic logic. By combining evidence across multiple reasoning dimensions, it transforms fragmented analytical outputs into a cohesive and interpretive artifact that enhances explainability, domain trust, and human-machine interpretive alignment in EVDM fault diagnosis.

5.3 Ablation Study

To address the proposed framework, we conducted an ablation study in which the entire interpretation process was delegated to a single LLM agent, without role-specific decomposition. Table A14 presents the ablation study results, in which the entire interpretation process is performed by a single LLM agent. The table reports the resulting explanation for the same sample instance shown in Table A13, allowing direct comparison with the proposed multi-agent LLM framework.

The ablation study reveals notable qualitative differences between the proposed framework and the single LLM agent. The prediction summary generated by the single LLM agent is more abstract, describing the final decision in broad terms without referencing the specific node-level branching features that informed the classification. In contrast, the proposed framework explicitly incorporates the internal structure of the decision tree, thereby providing a more transparent account of the underlying reasoning process. Second, although the single LLM agent can identify the correct key rules along the decision path, its subsequent threshold interpretations differ markedly from the proposed framework. The proposed framework contextualizes thresholds in relation to the sample's actual input values, enabling semantically grounded interpretations of why specific splits were triggered. The single LLM agent, however, provides threshold descriptions primarily from the perspective of the tree's structural logic, without connecting them to the sample's operational state, resulting in less informative and less user-centered explanations.

The counterfactual analyses produced by the single LLM agent remain reasonably coherent, suggesting that it can still infer plausible decision-boundary inversions when given the full structured input. Nevertheless, the most pronounced limitation emerges in the global consistency section. The single LLM agent fails to articulate how the instance-level reasoning aligns with the tree's global behavior, providing only a generalized statement rather than connecting fault regions and features observed at the dataset level. This omission contrasts with the multi-agent LLM framework, which provides detailed global-local integration by linking low speed, low torque, and high phase current conditions to the broader fault distribution learned by the model. Finally, the interpretive summary produced by the single LLM agent is considerably less comprehensive. Whereas the multi-agent LLM framework synthesizes threshold logic and domain semantics into a cohesive narrative, the single LLM agent summary provides a comparatively shallow description that does not capture the multi-level reasoning chain underlying the PTGF classification.

Overall, the results demonstrate that the proposed framework derives its interpretive strength from its role-specialized decomposition. This design enables detailed, consistently structured, and domain-grounded explanations, which a single LLM agent fails to replicate.

5.4 Evaluation of Explanatory Quality and User Satisfaction

To assess the explanatory quality of the proposed multi-agent LLM framework, we conducted two complementary evaluation procedures, an LLM-as-a-Judge assessment and a user study involving domain practitioners. In both evaluations, the explanations generated by the proposed multi-agent framework were directly compared with those produced by the single agent configuration used in the ablation study. Table 8 summarizes the results of the LLM-based evaluation in terms of mean score and SD for each criterion, where GPT-5 served as an autonomous evaluator applying the scoring criteria described in Section 4.3. GPT-5 was selected as the evaluation model because the evaluation criteria are inherently language and reasoning-centric, requiring strong semantic and contextual understanding to be evaluated consistently at scale.

Table 8: LLM-as-a-Judge evaluation results based on a 5-point Likert scale.

Criterion	Ablation Study (Mean ± SD)	Proposed Framework (Mean ± SD)
Explanation accuracy	3.93 ± 0.36	4.10 ± 0.48
Logical consistency	4.30 ± 0.46	4.36 ± 0.49
Faithfulness to model behavior	4.13 ± 0.34	4.26 ± 0.44
Domain relevance	4.16 ± 0.46	4.40 ± 0.49
Clarity and interpretability	4.06 ± 0.44	4.20 ± 0.40

We acknowledge that using the same LLM for both explanation generation and evaluation may arise concerns regarding potential self-evaluation bias. To mitigate this risk, the evaluation process was designed with strict role separation: the evaluator operates under independent prompts and deterministic decoding settings, and assigns scores based solely on predefined criterion-level Likert scales. As a result, the proposed framework outperformed the single-agent ablation study across all five criteria. The performance gains were moderate yet consistent, indicating that the multi-agent architecture enhances the structural quality and contextual appropriateness of generated explanations without introducing instability or verbosity.

As summarized in Table 9, user study results further support the effectiveness of the proposed framework. A total of ten participants took part in the evaluation. Overall, users rated the proposed framework higher than the ablation study, particularly in explanation accuracy and logical consistency. Importantly, the use of human evaluators provides an independent validation of the LLM-as-a-Judge results, alleviating concerns that performance improvements are artifacts of model self-assessment. The magnitude of improvement remained moderate, reflecting realistic user expectations and indicating that interpretability enhancements were perceptible without introducing exaggerated stylistic changes. Furthermore, domain relevance and clarity also showed improvement, although several participants noted that domain-specific phrasing could be further refined to better align with expert diagnostic conventions.

Table 10 provides a qualitative summary of user feedback. Positive comments highlighted the step-by-step traceability of decision paths and the usefulness of counterfactual reasoning in identifying actionable control adjustments. Users reported that these characteristics reduced the cognitive effort required to interpret the model's internal logic and increased confidence in the diagnostic decision. On the other hand, improvement suggestions emphasized adaptive reporting styles tailored to different expertise levels and the

integration of visual aids for decision path interpretation. These findings indicate that the LLM-as-a-Judge evaluation serves as a scalable and reproducible complement to human judgment, rather than a replacement, and that consistent trends across both evaluation modalities strengthen the reliability of the reported results.

Table 9: Evaluation of proposed framework through the user study.

Criterion	Ablation Study (Mean ± SD)	Proposed Framework (Mean ± SD)
Explanation accuracy	3.43 ± 0.31	3.71 ± 0.30
Logical consistency	3.40 ± 0.40	3.69 ± 0.32
Faithfulness to model behavior	3.57 ± 0.35	3.68 ± 0.35
Domain relevance	3.33 ± 0.47	3.62 ± 0.39
Clarity and interpretability	3.55 ± 0.41	3.78 ± 0.29

Table 10: Opinions of users on the proposed framework.

Type	Opinion
Positive feedback	The step-by-step explanation of the decision path made it easy to understand why a specific fault was predicted.
	The counterfactual analysis is useful because it suggests what parameter changes would prevent a fault.
	The multi-agent structure helps produce consistent explanations without requiring prior expert tuning.
	Explanations reflect actual operational behavior and match domain expectations, increasing trust.
	The explanation format helps reduce the cognitive effort required to interpret the model's internal logic.
Improvement suggestion	The generated reports could be adaptable based on the user's level of expertise.
	Visualization of the decision path would further enhance interpretability.
	Support for real-time inference in embedded or vehicle-mounted systems would be beneficial.
	Extending the framework to multi-modal sensor data would increase diagnostic coverage.
	Automating comparison of multiple fault instances would support faster diagnostic workflows in operations.

5.5 Comparison with Representative Published Studies

To contextualize the contribution of the proposed framework within existing research on EVDM fault diagnosis, Table 11 provides a comparative overview of the proposed framework and representative published studies addressing related diagnostic tasks. The comparison focuses on the target application, core modeling techniques, datasets, and the mechanisms adapted to support interpretability. This analysis aims to clarify how the proposed framework differs from and complements existing methods, especially in terms of explanation fidelity and practical usability for safety-critical maintenance.

Table 11: Performance and interpretability comparison with representative published studies.

Study	Target	Core Model	Dataset	Reported Performance	Explainability Provided	Explainability Mechanism	Key Limitations vs. Our Study
Thirunavukkarasu et al. (2024) [5]	EVDM fault classification	CatBoost and Bayesian optimization	Same with our study	Accuracy 94.1%	X	Not a primary focus (optimization-centric)	High-performing but limited explanation traceability. Does not provide structured instance-level reasoning artifacts
Haque et al. (2025) [3]	EVDM fault classification	Recursive feature elimination with cross-validation and soft voting ensemble	Same with our study	Accuracy 94.5%	O	LIME for local explanations	Use LIME for transparency but explanations are feature-attribution style
Ul Islam Khan et al. (2024) [41]	Fault classification in EV brushless direct current motor	Decision tree and voting classifier	Faulty dataset using parameters including currents and voltage	Accuracy 98.3%	X	Not emphasized as a central contribution	Limited discussion of explanation quality beyond classifier comparison
Rocha et al. (2024) [42]	Fault detection and isolation in dynamic industrial systems	Genetic programming-induced decision tree	Tennessee Eastman benchmark (process industry)	Average accuracy 80.3%	O	Tree structure offers transparency	Focus on fault detection rather than human-readable and domain-grounded explanation generation
Zhang et al. (2024) [43]	Anomaly diagnosis in EV battery	Decision tree and isolated forest	Authentic EV battery dataset	Mean relative error 0.317%. Calculate the abnormal score of each battery cell	X	Anomaly detection logic, not explanation logic	Lack decision path explanation for EV faults
Our study	EVDM fault diagnosis with explainable decision tree reasoning	Decision tree and multi-agent LLM interpretation pipeline	EVDM fault diagnosis dataset	Accuracy 93.2%	O	Structured, role-specialized, and decision path faithful explanations using LLMs	Focus extends beyond accuracy to traceable and consistent interpretability suitable for safety-critical maintenance contexts

Several prior studies have reported strong diagnostic performance for EVDM fault classification using advanced ML and optimization techniques. Thirunavukkarasu et al. [5] employed ensemble-based models, including CatBoost with Bayesian optimization, and achieved high classification accuracy on the same dataset used in this study. However, interpretability was not treated as primary design objective, and structured instance-level reasoning was not provided. Haque et al. [3] proposed a lightweight ensemble

framework with recursive feature elimination and soft voting, supplemented by LIME for local post-hoc explorations. While this improves transparency compared to purely black-box models, the resulting explanations remain feature-attribution oriented and are not explicitly aligned with the classifier's internal decision logic.

Other representative studies address related EV diagnostic contexts. Ul Islam Khan et al. [41] investigated fault classification in brushless direct current motors using decision trees and voting classifiers, focusing on robustness rather than explanation quality. Rocha et al. [42] applied genetic programming-induced decision trees to fault detection and isolation in an industrial benchmark, emphasizing detection performance over human-readable explanation generation. Zhang et al. [43] targeted anomaly diagnosis in EV battery systems using decision tree-based prediction and isolation forests, addressing a different subsystem and diagnostic objective. Consequently, decision path-level explanations for EVDM faults are not a central focus in these works.

In contrast, the proposed framework explicitly treats interpretability as a first-class objective. By combining a decision tree classifier with a multi-agent LLM interpretation pipeline, the framework generates structured, role-specialized explanations that remain faithful to the underlying decision path. As summarized in Table 11, this design complements competitive diagnostic performance with traceable and consistent interpretability tailored to safety-critical EVDM maintenance.

6 Discussions

6.1 Practical Limitations and Deployment Challenges of Large Language Model-Based Agents

Despite the interpretability advances demonstrated in this study, several practical limitations of LLM-based diagnostic frameworks remain to be addressed. First, the proposed framework relies on cloud-hosted LLMs, which introduces dependencies on external service providers [36]. This reliance entails computational costs and inference latency, particularly in a multi-agent configuration where sequential agent invocations can accumulate response delays [26]. Such characteristics pose constraints for resource-constrained environments in on-board diagnostic systems [14].

To mitigate these constraints, the proposed framework adopts modular architecture that enables selective activation of agents according to deployment requirements. For instance, lightweight configurations that exclude counterfactual or global audit agents could be employed when lower latency is required, while full multi-agent reasoning can be reserved for maintenance planning. Future implementations can leverage edge-deployed LLMs or caching strategies to further reduce end-to-end latency and operational cost [44].

In addition, the robustness of LLM-driven reasoning remains a concern. Although role-specific prompting and deterministic decoding were employed to stabilize agent outputs, LLM responses may still exhibit sensitivity to input representations [45]. This sensitivity can be alleviated through prompt standardization, template-based instruction design, and the use of structured intermediate representations such as JSON-based decision paths, which constrain the reasoning space and reduce variability across runs [46]. Furthermore, while the inclusion of a fact verification agent reduces the likelihood of erroneous statements, it does not fully eliminate the risk of domain-level overgeneralization or implicit physical assumptions that are not explicitly supported by the underlying model structure or data distribution. These risks suggest that LLM-generated explanations should be treated as assisted interpretations rather than authoritative physical diagnoses.

Overall, these limitations do not undermine the contribution of the proposed framework but rather delineate its appropriate application domain. By explicitly recognizing deployment constraints and outlining feasible mitigation strategies, this work provides a realistic assessment of how LLM-based diagnostic agents

can be effectively integrated into existing engineering workflows. It also identifies promising directions for future optimization toward real-time or embedded diagnostic applications.

6.2 Reproducibility and Model-Agnostic Design of Large Language Model-Based Agents

Another important consideration concerns the reliance of the proposed framework on a specific LLM, namely GPT-5. While this choice was motivated by its strong reasoning capabilities and long-context handling, it inevitably raises concerns regarding reproducibility and long-term sustainability. Variations in model versions or API updates can affect the reproducibility of results if such dependencies are not explicitly managed [47].

To mitigate these concerns, the proposed framework was designed to be model-agnostic at the architectural level. The core contribution of this work lies not in the selection of a particular LLM, but in the decomposition of interpretive reasoning into modular and role-defined agents with standardized input and output interfaces. The agent responsibilities, prompt templates, and structured JSON-based formats are explicitly specified, allowing the entire framework to be re-instantiated using alternative language models with minimal structural modification.

Moreover, several measures were adopted to enhance reproducibility within the current implementation. These include deterministic decoding strategies, fixed prompt versions, and explicit role constraints for each agent [48,49]. Such design choices reduce stochastic variability in generated explanations and facilitate consistent qualitative evaluation across repeated runs. In future work, open-source LLMs such as HyperCLOVA X [50] and LLaMA [51] can be integrated to further improve reproducibility and deployment flexibility.

6.3 Extension to Multimodal Fault Diagnosis

The current implementation of the proposed framework focuses primarily on tabular electrical and mechanical measurements, such as phase currents, voltage, torque, and rotational speed. While these variables are highly informative for EVDM fault scenarios, complex real-world failures often arise from the interaction of multiple physical phenomena that may not be fully captured by a single data modality [52,53]. For example, mechanical imbalances can be reflected in vibration signals, thermal degradation can be observed through temperature sensors or thermal imaging, and acoustic signatures can indicate bearing or insulation faults.

The proposed framework is well suited for extension to multimodal fault diagnosis. Owing to its modular agent design, modality-specific interpretation agents can be introduced to process heterogeneous inputs independently. For example, a vibration interpretation agent can be designed to analyze frequency-domain features. A thermal reasoning agent can focus on spatial or temporal temperature distributions, and an acoustic agent can examine spectral patterns associated with anomalous noise signatures. The outputs for these specialized agents can then be integrated by the report generator agent to form a unified and cross-modal diagnostic narrative. This design enables the framework to scale from single-modality explanations to multi-factor reasoning altering the underlying decision model.

Furthermore, incorporating multimodal evidence has the potential to improve both fault detection coverage and explanation depth, particularly for early-stage faults where electrical indicators alone can be ambiguous. By explicitly representing how different modalities contribute complementary evidence, the framework can better support root-cause analysis and reduce uncertainty in complex diagnostic cases. Although such extensions were beyond the scope of the present study, they represent a promising direction

for future work. They will leverage the complementary strengths of symbolic decision models and LLM-based reasoning to address complex EVDM fault scenarios.

6.4 Prompt Engineering and User-Adaptive Explanation Design for Large Language Model-Based Agents

While the proposed framework demonstrates strong capability in generating coherent and faithful diagnostic explanations, the quality and usability of such explanations remain closely tied to prompt engineering strategies and domain alignment [46,54]. In the present implementation, prompts were designed to emphasize clarity and traceability, adopting a model-centric tone. However, this style may not fully align with the terminology and heuristic reasoning patterns commonly used by practicing vehicle engineers in real-world diagnostic settings.

To address this gap, future work will focus on domain-adaptive prompt engineering, incorporating industry-standard diagnostic terminology, fault codes, and maintenance conventions into agent instructions [55,56]. This can be achieved by conditioning prompts on curated domain glossaries and historical maintenance reports, thereby reducing semantic friction and cognitive load for end users. Such alignment is important in safety-critical environments, where subtle differences in phrasing can influence interpretive confidence and decision-making efficiency.

In addition, the framework can be enhanced through adaptive explanation design tailored to diverse stakeholder groups [29]. Different users, including maintenance technicians, system engineers, and managerial decision-makers, require varying levels of technical detail and abstraction. The report generation agent can dynamically adjust explanation granularity, generating simplified summaries for non-experts and detailed context for domain experts. This adaptive reporting capability will improve usability and accessibility without compromising the underlying interpretability of the model.

6.5 Fleet-Level Analysis and Cross-Instance Diagnostic Intelligence with Large Language Model-Based Agents

The analysis presented in this study primarily focuses on instance-level interpretability, demonstrating how the proposed framework can generate faithful and structured explanations for individual EVDM fault cases. While such fine-grained analysis is essential for understanding specific failure events, practical maintenance and operational decision-making in real-world deployments require fleet-level and cross-instance insights [57]. Vehicle engineers are tasked with identifying recurring fault patterns, correlating failures across similar operating conditions, and prioritizing maintenance actions at the system or fleet scale rather than at the level of isolated instances [58,59].

The proposed framework can be extended to support cross-instance diagnostic intelligence by leveraging the structured outputs generated by the multi-agent pipeline. Instance-level explanations, including decision paths, dominant features, counterfactual sensitivities, and consistency metrics, can be aggregated and compared across multiple fault cases. This enables the automatic identification of common diagnostic signatures, recurring decision rules, and shared counterfactual triggers associated with specific fault types. Such aggregation will allow vehicle engineers to move beyond *ad hoc* inspection toward systematic pattern discovery across flees of vehicles.

At the fleet level, this capability can support a range of maintenance and reliability applications, including early detection of emerging fault trends, prioritization of high-risk components, and validation of maintenance strategies. By transforming individual explanations into a collective knowledge base, the framework can facilitate proactive maintenance planning and reduce diagnostic redundancy. Although fleet-level automation was beyond the scope of the present study, these extensions highlight the potential of

LLM-based multi-agent interpretation frameworks to evolve from single-case explanation tools into scalable diagnostic intelligence platforms for large-scale EV operations.

6.6 Knowledge Graph-Augmented Explanation for Mitigating Hallucination in Large Language Model-Based Fault Diagnosis

While the proposed framework emphasizes model-faithful interpretation by operating on decision tree structures, it relies on implicit domain knowledge embedded within LLM. As with other LLM-driven diagnostic systems, this reliance introduces a residual risk of hallucination, including unsupported causal claims or overgeneralized fault semantics that are not grounded in verified engineering knowledge.

Recent studies on knowledge graph (KG)-driven fault diagnosis provide a promising direction for addressing this limitation. Prior work in aviation equipment fault diagnosis has demonstrated that integrating structured KGs with LLMs improves explanation reliability by grounding generation in fault entities, relationships, and causal chains extracted from maintenance manuals and fault logs [60]. In such settings, KGs function as external knowledge anchors that constrain reasoning to validated domain facts, thereby reducing unsupported inferences in safety-critical applications.

Complementary research on KG-enabled heuristic reasoning further shows that KGs can function as active reasoning substrates rather than static knowledge repositories [61]. By formalizing domain rules and semantic relationships, these frameworks support interpretable and reusable reasoning processes while preserving the flexibility of natural language explanations. This highlights the value of structured knowledge grounding for improving explanation faithfulness in complex diagnostic workflows.

In the context of EVDM fault diagnosis, integrating a KG-driven approach into the proposed framework would provide multiple benefits. A fault KG encoding components, failure modes, causal dependencies, maintenance actions, and sensor-fault relationships could serve as a semantic verification layer for explanation agents, enabling cross-checking of generated interpretations against validated fault mechanisms. Moreover, formalized heuristic rules, such as associations between fault types and sensor thresholds, can improve reasoning consistency and efficiency while reducing the cognitive load for vehicle engineers. Although KG integration is beyond the scope of the present study, it represents a promising direction for future work toward more grounded and trustworthy EVDM fault diagnosis.

7 Conclusion

In this study, we proposed a novel multi-agent LLM-based interpretation framework for explaining decision tree classifiers applied to EVDM fault diagnosis. The proposed framework organizes the interpretation process into a structured sequence of specialized agents, each dedicated to extracting decision rules, assessing reliability, contextualizing domain knowledge, and synthesizing coherent explanations. By leveraging the reasoning capabilities of GPT-5, the framework automates the generation of comprehensive explanations that bridge the gap between model transparency and diagnostic practicality.

Experimental evaluations using a publicly available EVDM dataset demonstrated that the decision tree classifier achieved robust predictive performance, with a maximum depth of 10 providing the optimal balance between accuracy and interpretability. The proposed framework effectively produced coherent, faithful, and domain-relevant explanations, as verified through both quantitative analysis and qualitative assessment involving LLM-as-a-Judge and human expert evaluation. These results confirm the potential of multi-agent LLM frameworks to enhance the interpretability in safety-critical diagnostic applications.

The primary contributions of this work lie in introducing a modular agent architecture for structured model interpretation, integrating domain-specific reasoning with generalizable XAI techniques, and establishing an evaluation protocol combing automated and human assessment for explanation quality. These contributions provide a foundation for future research on scalable and XAI systems in industrial domains.

Future extensions of this study will explore the integration of multi-modal sensor data (e.g., vibration or thermal imaging) and hybrid decision models to further enhance interpretability across heterogeneous diagnostic contexts [62]. In parallel, adapting the proposed framework to real-time monitoring and in-vehicle diagnostic systems will be an important step toward achieving trustworthy and deployable artificial intelligence-driven maintenance in next-generation EVs [2,63]. Furthermore, future work will focus on advancing the multi-agent LLM framework itself through refined prompt engineering strategies and adaptive task orchestration environments [18]. These developments are expected to improve the agents' reasoning coherence, collaborative efficiency, and contextual adaptability, ultimately enabling a more autonomous and explainable diagnostic intelligence for safety-critical EV systems.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization, Jaeseung Lee and Jehyeok Rew; methodology, Jehyeok Rew; software, Jaeseung Lee; validation, Jehyeok Rew; formal analysis, Jehyeok Rew; investigation, Jaeseung Lee; resources, Jehyeok Rew; data curation, Jaeseung Lee; writing—original draft preparation, Jaeseung Lee; writing—review and editing, Jehyeok Rew; visualization, Jaeseung Lee; supervision, Jehyeok Rew; project administration, Jehyeok Rew; funding acquisition, Jehyeok Rew. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support of the findings of this study are openly available in Machine Learning Based Fault Diagnosis of Electric Drives at https://github.com/HassanMahmoodKhan/Machine-Learning-Based-Fault-Diagnosis-of-Electric-Drives.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

API	Application programming interface
EV	Electric vehicle
EVDM	Electric vehicle drive motor
Grad-CAM	Gradient-weighted class activation mapping
JSON	Javascript object notation
KG	Knowledge graph
LIME	Local interpretable model-agnostic explanations
LLM	Large language model
ML	Machine learning
NOM	Normal operation
OLF	Overload fault
OVF	Overvoltage fault
PTGF	Phase-to-ground fault
PTPF	Phase-to-phase fault
SD	Standarad deviation

SHAP	Shapley additive explanations
UVF	Undervoltage fault
VOBC	Vehicle on-board controller
XAI	Explainable artificial intelligence

Appendix A

Appendix A provides the complete prompt templates used for each agent in the proposed framework. These templates define the role instructions, input structure, output formatting requirements, and domain-specific reasoning constraints that guide the behavior of each agent. By presenting these prompts in full, we aim to ensure the reproducibility of the proposed framework and enable researchers and practitioners to adapt the framework to alternative diagnostic domains.

Tables A1–A6 list the prompt templates for each agent in the order in which they operate within the interpretation pipeline: (1) Local Evidence Interpretation Agent, (2) Threshold Translator Agent, (3) Counterfactual Analysis Agent, (4) Global Consistency Audit Agent, (5) Report Generator Agent, and (6) Fact Verification Agent.

Table A1: Prompt template for local evidence interpretation agent.

Type	Prompt detail
System instruction	<System Instruction> You are a domain expert in electric vehicle drive motor (EVDM) fault diagnosis and explainable artificial intelligence (XAI). Your goal is to interpret the decision path of a trained Decision Tree model for a single sample. You must identify which features and thresholds most influenced the predicted fault class and describe their effects precisely and objectively. </System Instruction>
Context	<Context> Model Type: Decision Tree Classifier Domain: Electric Vehicle Drive Motor Fault Diagnosis Input Features: 1. **Tn (rated torque) [N·m]**—Torque produced by the motor in Newton meters (continuous) 2. **k (constant of proportionality)**—Constant used in the drive model (continuous) 3. **Ia [Amp]**—Current in phase A (continuous) 4. **Ib [Amp]**—Current in phase B (continuous) 5. **Ic [Amp]**—Current in phase C (continuous) 6. **Vab [V]**—Voltage between phases A and B in volts (continuous) 7. **Speed [rad/s]**—Rotational speed in radians per second (continuous) Output (Target) Classes: - **NOM**—Normal operation - **PTPF**—Phase-to-phase fault (between phases B and C) - **PTGF**—Phase-to-ground fault (between phase A and ground) - **OVF**—Overvoltage fault - **OLF**—Overload fault - **UVF**—Undervoltage fault Predicted Class: {predicted_class} Prediction Probability: {predicted_probability} </Context>
Decision path data	<Decision Path Data> {path_json} </Decision Path Data>

(Continued)

Table A1 (continued)

Type	Prompt detail
Task instruction	<Task instruction> 1. For each node in the decision path, describe: •The feature and its threshold condition. •Whether the sample value satisfies (≤) or exceeds (>) the threshold. •How this condition increases or decreases the probability of the predicted class. •The impurity reduction or class confidence gain at this step. 2. Summarize the most influential nodes (top 2–3) that strongly supported the final class. 3. Avoid speculation; base explanations only on provided numeric evidence. 4. Write in formal scientific tone suitable for a diagnostic report. 5. Use concise sentences and technical terminology (e.g., "impurity reduction," "class confidence," "threshold proximity"). 6. Present the result in two sections: (a) **Node-wise Evidence Table** (b) **Interpretive Summary** </Task instruction>

Output format	<Output format>
	{
	"node_evidence": [
	{
	"node_id": <int>,
	"feature": "<feature_name>",
	"rule": "<feature_name> <= <threshold> (sample: <value>)",
	"direction": "LEFT/RIGHT",
	"impurity_drop": <float>,
	"probability_shift": <float>,
	"interpretation": "<concise sentence>"
	}, . . .
	],
	"summary": "<paragraph summarizing how feature interactions led to the predicted class>"
	}
	</Output format>

Table A2: Prompt template for threshold translator agent.

Type	Prompt Detail
System instruction	<System instruction> You are a domain expert in electric vehicle drive motor (EVDM) fault diagnosis and explainable artificial intelligence (XAI). Your role is to interpret decision-path evidence provided by a previous analytical agent, known as local evidence interpretation agent (LEIA), and translate the numerical thresholds, sample values, and logical directions into human-understandable physical and operational meanings. You must describe the condition of the motor (e.g., torque level, phase current, rotational speed) in concise technical language. </System instruction>
	<Context> Domain: Electric Vehicle Drive Motor (EVDM) Fault Diagnosis Model Type: Decision Tree Classifier Input Variables and Physical Meanings (from dataset schema): 1. **Tn (rated torque) [N·m]**—Torque output of the motor (continuous) 2. **k (constant of proportionality)**—Constant parameter in the drive model (continuous) 3. **Ia [Amp]**—Current in phase A (continuous) 4. **Ib [Amp]**—Current in phase B (continuous)

(Continued)

Table A2 (continued)

Type	Prompt Detail
Context	5. **Ic [Amp]**—Current in phase C (continuous) 6. **Vab [V]**—Voltage between phases A and B (continuous) 7. **Speed [rad/s]**—Rotational speed in radians per second (continuous) Output Fault Classes: - **NOM**: Normal operation - **PTPF**: Phase-to-phase fault (between phases B and C) - **PTGF**: Phase-to-ground fault (phase A and ground) - **OVF**: Overvoltage fault - **OLF**: Overload fault - **UVF**: Undervoltage fault Predicted Class: {predicted_class} Prediction Probability: {predicted_probability} Input from Previous Agent (LEIA): Node-wise decision path evidence with numeric reasoning. {input_from_local_evidence_interpretation_agent} Each node record contains: - feature_name - threshold - sample_value - direction (LEFT/RIGHT) - impurity_drop - probability_shift - interpretation (numeric-based reasoning) </Context>
Task instruction	<Task instruction> 1. For each node in the received decision path: •Compare the 'sample_value' with its 'threshold' and, if available, with global statistics (mean ± std or percentile range). •Determine whether the feature's value is low, normal, or high relative to typical operating conditions. •Express this comparison in a physical, domain-relevant way (e.g., "abnormally low motor speed," "below-nominal torque," "excessive phase-A current"). •Include a concise quantitative note (e.g., "sample: 21.59 A > threshold: 4.37 A"). 2. Preserve objectivity—avoid speculation, emotion, or causal claims beyond the data. 3. After translating all nodes, compose a **Narrative Summary** that describes how these semantic thresholds collectively characterize the sample's operational state leading to the predicted fault. 4. The summary should highlight patterns (e.g., "low speed combined with elevated phase-A current indicates a ground-related imbalance") while maintaining a formal and diagnostic tone. 5. Structure the output strictly in JSON format with the following schema: </Task instruction>
Output format	<Output format> { "translated_nodes": [{ "node_id": <int>, "feature": "<feature_name>", "numeric_rule": "<feature_name> <=/> <threshold> (sample: <value>)", "semantic_expression": "<technical phrase expressing operational meaning>", "relative_position": "<low/normal/high or percentile estimate>", "quantitative_context": "(sample: <value>, threshold: <threshold>, impurity_drop: <value>, probability_shift: <value>)" }, ...

(Continued)

Table A2 (continued)

Type	Prompt Detail
	], "summary": "<concise technical paragraph summarizing how the observed feature conditions (speed, torque, current) indicate the {predicted_class} predicted by the model.>" } </Output format>

Table A3: Prompt template for counterfactual analysis agent.

Type	Prompt detail
System instruction	<System instruction> You are a reasoning agent specialized in counterfactual analysis for Decision Tree models within the domain of Electric Vehicle Drive Motor (EVDM) fault diagnosis. Your task is to determine the minimal feature perturbations (Δx) that would alter the current fault prediction to a different class while keeping all changes physically plausible. </System instruction>
Context	<Context> Domain: Electric Vehicle Drive Motor (EVDM) Fault Diagnosis Model Type: Decision Tree Classifier Input Variables and Physical Meanings (from dataset schema): 1. **Tn (rated torque) [N·m]**—Torque output of the motor (continuous) 2. **k (constant of proportionality)**—Constant parameter in the drive model (continuous) 3. **Ia [Amp]**—Current in phase A (continuous) 4. **Ib [Amp]**—Current in phase B (continuous) 5. **Ic [Amp]**—Current in phase C (continuous) 6. **Vab [V]**—Voltage between phases A and B (continuous) 7. **Speed [rad/s]**—Rotational speed in radians per second (continuous) Output Fault Classes: - **NOM**: Normal operation - **PTPF**: Phase-to-phase fault (between phases B and C) - **PTGF**: Phase-to-ground fault (phase A and ground) - **OVF**: Overvoltage fault - **OLF**: Overload fault - **UVF**: Undervoltage fault Predicted Class: {predicted_class} Prediction Probability: {predicted_probability} Alternative Classes: {alternative_classes} Input Variables and Physical Meanings: - **Tn (rated torque) [N·m]**—Motor torque output (Mean = 1.138, SD = 0.371, Range ≈ 0.8–2.5) - **k (constant of proportionality)**—Drive-model constant (Mean = 0.001, SD ≈ 0.0003, Range ≈ 0.000–0.002) - **Ia [Amp]**—Phase A current (Mean = 17.897, SD = 10.673, Range ≈ 0–79 A) - **Ib [Amp]**—Phase B current (Mean = 17.764, SD = 10.472, Range ≈ 0–79 A) - **Ic [Amp]**—Phase C current (Mean = 18.467, SD = 10.509, Range ≈ 0–79 A) - **Vab [V]**—Voltage between phases A and B (Mean = 432.839, SD = 123.434, Range ≈ 0–650 V) - **Speed [rad/s]**—Rotational speed (Mean = 123.521, SD = 49.801, Range ≈ –112 to 183 rad/s) Feature Constraints (derived from Table 3 and physical plausibility): - Tn ∈ [0.0,2.5], Ia/Ib/Ic ∈ [0.0,80.0], Vab ∈ [0.0,650.0], Speed ∈ [–120.0, 200.0], k ∈ [0.0,0.002] Interpretive Inputs (from threshold translator agent): Each node record includes a semantically interpreted threshold relation, relative position ("low", "high", etc.), and quantitative context (sample vs. threshold). Use these as the basis for identifying near-boundary features and plausible adjustments.

(Continued)

Table A3 (continued)

Type	Prompt detail
	{output_of_threshold_translator_agent} </Context>
Task instruction	<Task instruction> 1. Identify features positioned close to decision thresholds or having large influence (high impurity_drop or probability_shift). Focus on those that could feasibly be adjusted to reach a different leaf node. 2. For each such feature: •Compute or qualitatively estimate the minimal change (Δx) needed to cross its decision boundary (e.g., if Speed ≤ 104.20, increasing speed slightly above this threshold). •Specify the direction of change (increase/decrease). •Predict the likely new class that would be reached after the change, based on the alternate branch rule. •Assess the physical feasibility of this change within the operational limits of the EVDM system. 3. Rank counterfactuals by increasing Δx magnitude (minimal first) and interpretability. 4. Describe each counterfactual scenario with concise diagnostic reasoning (e.g., "A 5% increase in torque could redirect classification from PTGF to PTPF, representing recovery from ground fault to inter-phase imbalance"). 5. Provide a final summary explaining which features dominate decision reversibility and how small parameter variations affect fault classification sensitivity. 6. Maintain scientific tone suitable for an engineering diagnostic report. </Task instruction>

Output format	<Output format>
	{
	"counterfactuals": [
	{
	"feature": "<feature_name>",
	"current_value": <float>,
	"threshold": <float>,
	"new_value": <float>,
	"delta": <float>,
	"direction": "<increase/decrease>",
	"predicted_new_class": "<class_name>",
	"feasibility": "<feasible/unrealistic>",
	"interpretation": "<brief diagnostic rationale>"
	},
	...
	],
	"summary": "<technical paragraph explaining how small parameter adjustments (Δx) near decision thresholds could lead to alternate fault classifications and what this implies about the model's sensitivity and physical plausibility.>"
	}
	</Output format>

Table A4: Prompt template for global consistency audit agent.

Type	Prompt Detail
System instruction	<System instruction> You are an auditing agent responsible for evaluating the consistency between local (sample-specific) and global (model-level) interpretive evidence in a Decision Tree classifier. You specialize in the Electric Vehicle Drive Motor (EVDM) fault diagnosis domain. Your task is to verify whether the locally influential features for a specific prediction align with the globally dominant factors learned by the model. Provide both quantitative and qualitative assessments suitable for a diagnostic report.

(Continued)

Table A4 (continued)

Type	Prompt Detail
Context	</System instruction> <Context> Model Type: Decision Tree Classifier Domain: Electric Vehicle Drive Motor (EVDM) Fault Diagnosis Predicted Class: {predicted_class} Local Explanation (from previous agents—translated node evidence): Each entry contains: - feature_name - impurity_drop (information gain) - probability_shift (ΔP toward predicted class) - semantic_expression (physical meaning of the local behavior) Example local input: {translated_nodes_json} Global Feature Importances (model-level): Each entry contains: - feature_name - importance_value (normalized between 0 and 1) Example global input: {global_feature_importance} Notes on domain interpretability: - In EVDM systems, **Phase currents (Ia, Ib, Ic)** and **rotational speed** are typically the most influential indicators of fault state. - Torque (Tn) and voltage (Vab) serve as secondary contributors depending on load and operational phase. These domain priors may assist in reasoning about alignment or deviation between local and global evidence. </Context>
Task instruction	<Task instruction> 1. Rank local features by combined saliency, using a weighted measure such as: saliency_score = (impurity_drop normalized) + (|probability_shift|). Extract the top-k most influential features (typically k = 3). 2. Rank global features by importance_value and extract their top-k set. 3. Compute quantitative alignment metrics: •**Jaccard similarity** between top-k feature sets (0–1 scale). •**Spearman rank correlation** between ranked lists (optional). 4. Provide a qualitative analysis: •Do the most dominant local features (e.g., Speed, Ia) correspond to globally important ones? •If misalignment exists, discuss potential causes such as: - Context-specific local patterns - Global averaging across fault types - Model depth bias or node-specific impurity variance 5. Conclude the audit as one of the following: - **Globally consistent**—local and global importance strongly align - **Partially consistent**—overlap exists but ranking differs - **Inconsistent**—local importance diverges significantly from global trend 6. Express results in structured JSON format (below), followed by a short, formal summary paragraph interpreting what the consistency means for diagnostic reliability. </Task instruction>

Output format	<Output format>
	{
	"consistency_metrics": {
	"local_top_features": ["<feature1>", "<feature2>", "<feature3>"],
	"global_top_features": ["<feature1>", "<feature4>", "<feature2>"],
	"jaccard_similarity": <float>,

(Continued)

Table A4 (continued)

Type	Prompt Detail

	"rank_correlation": <float>
	},
	"qualitative_assessment": {
	"alignment": "<High/Moderate/Low>",
	"explanation": "<technical reasoning about alignment or deviation>"
	},
	"summary": "<concise paragraph summarizing how closely the local feature influences for fault class match the model's global explanatory pattern, and what this implies for interpretability consistency.>"
	}
	</Output format>

Table A5: Prompt template for report generator agent.

Type	Prompt Detail
System instruction	<System instruction> You are a scientific report-writing agent specialized in explainable fault diagnosis. Your task is to integrate the analytical outputs of multiple interpretation agents into a single coherent narrative that describes the reasoning behind a Decision Tree's prediction. </System instruction>
Context	<Context> Domain: Electric Vehicle Drive Motor (EVDM) Fault Diagnosis Model Type: Decision Tree Classifier Input Variables and Physical Meanings (from dataset schema): 1. **Tn (rated torque) [N·m]**—Torque output of the motor (continuous) 2. **k (constant of proportionality)**—Constant parameter in the drive model (continuous) 3. **Ia [Amp]**—Current in phase A (continuous) 4. **Ib [Amp]**—Current in phase B (continuous) 5. **Ic [Amp]**—Current in phase C (continuous) 6. **Vab [V]**—Voltage between phases A and B (continuous) 7. **Speed [rad/s]**—Rotational speed in radians per second (continuous) Output Fault Classes: - **NOM**: Normal operation - **PTPF**: Phase-to-phase fault (between phases B and C) - **PTGF**: Phase-to-ground fault (phase A and ground) - **OVF**: Overvoltage fault - **OLF**: Overload fault - **UVF**: Undervoltage fault Predicted Class: {predicted_class} Prediction Probability: {predicted_probability} </Context>
Inputs from previous agents	<Inputs from previous agents> 1. Local Evidence Interpretation Agent Output: {local_evidence_json} 2. Threshold Translator Agent Output: {threshold_translation_json} 3. Counterfactual Analysis Agent Output: {counterfactual_json} 4. Global Consistency Audit Agent Output: {consistency_json} </Inputs from previous agents>

(Continued)

Table A5 (continued)

Type	Prompt Detail
Task instruction	<Task instruction> 1. Integrate the above outputs into a well-structured technical report containing: (a) **Prediction Summary**—present the predicted class and confidence. (b) **Key Decision Rules**—summarize the decisive feature-threshold pairs and their effects. (c) **Threshold Interpretations**—explain what each numeric threshold means in human terms. (d) **Counterfactual Scenarios**—describe minimal feature changes that would alter the prediction. (e) **Global Consistency Statement**—summarize how local reasoning aligns with global model patterns. 2. Maintain a formal academic tone and avoid speculative or causal language. 3. Each section should reference supporting numeric evidence (e.g., impurity drop, Δprobability, threshold proximity). 4. End with a concise **Interpretive Summary** paragraph linking all findings. 5. Present output as structured JSON to support traceability and automated rendering. </Task instruction>

Output format	<Output format>
	{
	“report”: {
	“prediction_summary”: “<text>”,
	“key_rules”: [
	{“feature”: “<feature_name>”, “threshold”: <float>, “sample_value”: <float>, “effect”: “<textual interpretation>”}
	],
	“threshold_interpretations”: [
	{“feature”: “<feature_name>”, “semantic_expression”: “<natural-language threshold explanation>”}
	],
	“counterfactuals”: [
	{“feature”: “<feature_name>”, “delta”: <float>, “new_value”: <float>, “new_class”: “<alt_class>”, “interpretation”: “<summary>”}
	],
	“global_consistency”: “<short text on alignment or discrepancy>”,
	“interpretive_summary”: “<final narrative integrating all reasoning steps>”
	}
	}
	</Output format>

Table A6: Prompt template for fact verification agent.

Type	Prompt Detail
System instruction	<System instruction> You are a verification and refinement agent responsible for auditing the final diagnostic report produced by a multi-agent interpretation system. Your goals are to: (1) verify numerical consistency and factual accuracy, (2) ensure internal logical coherence, and (3) refine linguistic clarity and style for technical publication. </System instruction>
Context	<Context> Domain: Electric Vehicle Drive Motor (EVDM) Fault Diagnosis Model Type: Decision Tree Classifier Verified Data Sources:

(Continued)

Table A6 (continued)

Type	Prompt Detail
	- Decision Tree Metadata: {tree_metadata_json} - Path JSON (sample-level evidence): {path_json} - Global Feature Importances: {global_importances_json} Input Report (from Report Generator Agent): {generated_report_json} </Context>
Task instruction	<Task instruction> 1. **Numerical Verification** •Cross-check all numeric values (thresholds, sample values, impurity drops, probability scores) against the provided model data. •Flag any mismatches beyond ±0.01 tolerance. •Ensure probability values sum to 1 (if applicable) and units are consistent. 2. **Logical Consistency** •Verify that described feature directions (e.g., "higher than threshold") match their numeric comparison. •Ensure class names, feature names, and relationships remain internally coherent. 3. **Linguistic Refinement** •Improve clarity, conciseness, and grammatical accuracy. •Standardize terminology (e.g., "threshold," "feature contribution," "impurity reduction"). •Remove redundancy and speculative phrasing while preserving technical meaning. 4. **Output Requirements** •Return corrected and validated report text. •List detected inconsistencies and their corrections. •Include a final confidence score (0–1) indicating factual completeness. </Task instruction>

Output format	<Output format>
	{
	"validation_results": {
	"numerical_checks": [
	{"field": "<name>", "status": "<passed/failed>", "original_value": <float>, "verified_value": <float>, "correction": "<text if any>"}
	],
	"logical_checks": [
	{"description": "<inconsistency detected>", "resolution": "<how it was corrected>"}
	],
	"style_refinements": [
	{"before": "<original_phrase>", "after": "<revised_phrase>", "reason": "<clarity/grammar/consistency>"}
	]
	},
	"validated_report": "<final corrected report text>",
	"confidence_score": <float>
	}
	</Output format>

Appendix B

Appendix B presents the complete interpretation artifacts generated by the proposed framework for a representative EVDM fault sample. It includes the full reasoning outputs to ensure transparency and reproducibility of the interpretation process. Tables A7–A13 document each stage of the multi-agent pipeline in the exact order of execution: the raw decision path extracted from the trained decision tree, its JSON-formatted representation, and the respective outputs of the local evidence interpretation agent, threshold

translator agent, counterfactual analysis agent, global consistency audit agent, and report generator agent. These tables allow readers to trace how input features, split thresholds, and model states are incrementally transformed into a coherent diagnostic explanation.

Furthermore, Table A14 provides the complete ablation study results generated by a single-agent baseline, enabling direct comparison with the proposed multi-agent design. By making full outputs available, Appendix B supports independent verification of interpretive fidelity and provides a practical reference for extending the framework to alternative datasets, diagnostic tasks, or agent configurations.

Table A7: Original decision path for sample instance.

Result
=== Decision Path for sample index 69 === Input features: - Tn (Rated Torque) N*m: 0.8500 - k (constant of proportionality): 0.0010 - Ia (Amp): 21.5894 - Ib (Amp): 39.6349 - Ic (Amp): 18.0468 - Vab (V): 362.4316 - Speed (rad/s): −32.1893 ---------- Class index mapping (index -> name): {0: 'NOM', 1: 'OLF', 2: 'OVF', 3: 'PTGF', 4: 'PTPF', 5: 'UVF'} ---------- [Node 0] if Speed (rad/s) <= 144.4185 - sample value: Speed (rad/s) = −32.1893 -> go LEFT (next: Node 1) - node impurity: 0.7968 - node samples: 28025, pred: index = 0, name = NOM ---------- [Node 1] if Tn (Rated Torque) N*m <= 1.3750 - sample value: Tn (Rated Torque) N*m = 0.8500 -> go LEFT (next: Node 2) - node impurity: 0.7546 - node samples: 14516, pred: index = 4, name = PTPF ---------- [Node 2] if Speed (rad/s) <= 104.1969 - sample value: Speed (rad/s) = −32.1893 -> go LEFT (next: Node 3) - node impurity: 0.6749 - node samples: 11057, pred: index = 4, name = PTPF ---------- [Node 3] if Ia (Amp) <= 4.3744 - sample value: Ia (Amp) = 21.5894 -> go RIGHT (next: Node 5) - node impurity: 0.0120 - node samples: 3805, pred: index = 3, name = PTGF ---------- [Leaf 5] - node impurity: 0.0000

(Continued)

Table A7 (continued)

Result
- node samples: 3782 - predicted: index = 3, name = PTGF ========== Final prediction: index = 3, name = PTGF Class probabilities (by index): {'0': 0.0, '1': 0.0, '2': 0.0, '3': 1.0, '4': 0.0, '5': 0.0} Class probabilities (by name): {'NOM': 0.0, 'OLF': 0.0, 'OVF': 0.0, 'PTGF': 1.0, 'PTPF': 0.0, 'UVF': 0.0}

Table A8: JSON decision path for sample instance.

Result


```
{
  "sample_index": 69,
  "input_features": {
    "Tn (Rated Torque) N*m": 0.85,
    "k (constant of proportionality)": 0.001,
    "Ia (Amp)": 21.5894,
    "Ib (Amp)": 39.6349,
    "Ic (Amp)": 18.0468,
    "Vab (V)": 362.4316,
    "Speed (rad/s)": −32.1893
  },
  "class_index_mapping": {
    "0": "NOM",
    "1": "OLF",
    "2": "OVF",
    "3": "PTGF",
    "4": "PTPF",
    "5": "UVF"
  },
  "decision_path": [
   {
     "node_id": 0,
     "impurity": 0.7968,
     "samples": 28025,
     "predicted_class_index": 0,
     "predicted_class_name": "NOM",
     "feature": "Speed (rad/s)",
     "threshold": 144.4185,
     "sample_value": −32.1893,
     "direction": "LEFT",
     "next_node": 1
```


(Continued)

Table A8 (continued)

Result

},
{
"node_id": 1,
"impurity": 0.7546,
"samples": 14516,
"predicted_class_index": 4,
"predicted_class_name": "PTPF",
"feature": "Tn (Rated Torque) N*m",
"threshold": 1.375,
"sample_value": 0.85,
"direction": "LEFT",
"next_node": 2
},
{
"node_id": 2,
"impurity": 0.6749,
"samples": 11057,
"predicted_class_index": 4,
"predicted_class_name": "PTPF",
"feature": "Speed (rad/s)",
"threshold": 104.1969,
"sample_value": −32.1893,
"direction": "LEFT",
"next_node": 3
},
{
"node_id": 3,
"impurity": 0.012,
"samples": 3805,
"predicted_class_index": 3,
"predicted_class_name": "PTGF",
"feature": "Ia (Amp)",
"threshold": 4.3744,
"sample_value": 21.5894,
"direction": "RIGHT",
"next_node": 5
},
{
"node_id": 5,
"impurity": 0.0,
"samples": 3782,

(Continued)

Table A8 (continued)

Result


```
        "predicted_class_index": 3,
        "predicted_class_name": "PTGF",
        "leaf": true
      }
    ],
    "final_prediction": {
      "predicted_class_index": 3,
      "predicted_class_name": "PTGF",
      "class_probabilities_by_index": {
        "0": 0.0,
        "1": 0.0,
        "2": 0.0,
        "3": 1.0,
        "4": 0.0,
        "5": 0.0
      },
      "class_probabilities_by_name": {
        "NOM": 0.0,
        "OLF": 0.0,
        "OVF": 0.0,
        "PTGF": 1.0,
        "PTPF": 0.0,
        "UVF": 0.0
    }
  }
}
```


Table A9: Output of local evidence interpretation agent.

Result


```
{
    "node_evidence": [
      {
        "node_id": 0,
        "feature": "Speed (rad/s)",
        "rule": "Speed (rad/s) <= 144.4185 (sample: −32.1893)",
        "direction": "LEFT",
        "impurity_drop": 0.0422,
        "probability_shift": 0.0,
        "interpretation": "The sample's negative speed is well below the 144.4185 rad/s threshold, directing
it to the left branch dominated by non-nominal cases, thereby excluding normal operation (NOM) early."
```


(Continued)

Table A9 (continued)

Result

},
{
"node_id": 1,
"feature": "Tn (Rated Torque) N*m",
"rule": "Tn (Rated Torque) <= 1.375 (sample: 0.85)",
"direction": "LEFT",
"impurity_drop": 0.0797,
"probability_shift": 0.0,
"interpretation": "Torque below the threshold guided the path toward a subset enriched in fault categories (notably PTPF), slightly reducing impurity but not yet isolating PTGF."
},
{
"node_id": 2,
"feature": "Speed (rad/s)",
"rule": "Speed (rad/s) <= 104.1969 (sample: −32.1893)",
"direction": "LEFT",
"impurity_drop": 0.0797,
"probability_shift": 0.0,
"interpretation": "The second speed-based split reinforces that extremely low or negative speeds are associated with ground-related faults, narrowing the population further toward PTGF."
},
{
"node_id": 3,
"feature": "Ia (Amp)",
"rule": "Ia (Amp) > 4.3744 (sample: 21.5894)",
"direction": "RIGHT",
"impurity_drop": 0.6629,
"probability_shift": 1.0,
"interpretation": "Phase-A current significantly exceeding 4.3744 A sharply reduced impurity to zero and decisively classified the instance as PTGF, indicating that high phase-A current under low-speed conditions is a defining indicator of a phase-to-ground fault."
}
],

(Continued)

Table A9 (continued)

Result
“summary”: “The decision tree’s diagnostic trajectory for sample #69 demonstrates that the model progressively excluded nominal and non-ground fault conditions through sequential threshold evaluations. Initially, low rotational speed (−32.19 rad/s ≤ 144.42 rad/s) shifted classification away from normal operation. Low torque (0.85 ≤ 1.375 N·m) further constrained the sample to fault-prone operational zones. Reiterated low-speed confirmation (≤104.20 rad/s) directed the path toward faults associated with grounded circuits. The decisive factor emerged at node 3, where high phase-A current (21.59 A > 4.37 A) produced complete impurity reduction and yielded a unit probability (1.0) for the PTGF class. Thus, the interaction of abnormally low rotational speed and elevated phase-A current served as the dominant diagnostic evidence supporting the phase-to-ground fault prediction.” }

Table A10: Output of threshold translator agent.

Result
{ “translated_nodes”: [{ “node_id”: 0, “feature”: “Speed (rad/s)”, “numeric_rule”: “Speed (rad/s) <= 144.4185 (sample: −32.1893)”, “semantic_expression”: “The motor exhibits abnormally low or negative rotational speed, indicating a non-operational or counter-rotating condition far below nominal range.”, “relative_position”: “very low (below 10th percentile of normal operating range)”, “quantitative_context”: “(sample: −32.19 rad/s, threshold: 144.42 rad/s, impurity_drop: 0.0422, probability_shift: 0.0)” }, { “node_id”: 1, “feature”: “Tn (Rated Torque) N*m”, “numeric_rule”: “Tn (Rated Torque) <= 1.375 (sample: 0.85)”, “semantic_expression”: “The output torque is below the nominal threshold, representing a light-load or near-idle operational state where the motor produces minimal mechanical output.”, “relative_position”: “low (below nominal torque level)”, “quantitative_context”: “(sample: 0.85 N·m, threshold: 1.375 N·m, impurity_drop: 0.0797, probability_shift: 0.0)” }, { “node_id”: 2, “feature”: “Speed (rad/s)”, “numeric_rule”: “Speed (rad/s) <= 104.1969 (sample: −32.1893)”,

(Continued)

Table A10 (continued)

Result

“semantic_expression”: “The second speed evaluation confirms the persistently low rotational speed, reinforcing an off-nominal, quasi-stationary state typically associated with drive interruptions or grounding anomalies.”,
“relative_position”: “very low (persistently below threshold range)”,
“quantitative_context”: “(sample: −32.19 rad/s, threshold: 104.20 rad/s, impurity_drop: 0.0797, probability_shift: 0.0)”
},
{
“node_id”: 3,
“feature”: “Ia (Amp)”,
“numeric_rule”: “Ia (Amp) > 4.3744 (sample: 21.5894)”,
“semantic_expression”: “Phase-A current is abnormally high, exceeding the threshold by a large margin, indicating an overcurrent condition in phase A consistent with a phase-to-ground fault.”,
“relative_position”: “very high (above 95th percentile of typical range)”,
“quantitative_context”: “(sample: 21.59 A, threshold: 4.37 A, impurity_drop: 0.6629, probability_shift: 1.0)”
}
],
“summary”: “The examined motor instance exhibits a combination of extremely low rotational speed (−32.19 rad/s) and low output torque (0.85 N·m), indicating near-stationary operation with minimal mechanical load. Despite this low-speed state, the phase-A current is markedly elevated (21.59 A), surpassing the diagnostic threshold by approximately fivefold. This current anomaly under low mechanical output conditions signifies a severe electrical imbalance, characteristic of current leakage or grounding within phase A. These conditions collectively describe a motor state consistent with a phase-to-ground fault (PTGF), aligning with the model’s unit-probability classification outcome.”
}

Table A11: Output of counterfactual analysis agent.

Result
{ “counterfactuals”: [{ “feature”: “Tn (rated torque) [N·m]”, “current_value”: 0.85, “threshold”: 1.375, “new_value”: 1.3751, “delta”: 0.5251, “direction”: “increase”, “predicted_new_class”: “PTPF”, “feasibility”: “feasible”,

(Continued)

Table A11 (continued)

Result

"interpretation": "Raising mechanical load slightly above the torque split (>1.375 N·m) diverts the path at node 1 into the alternative subtree, which is dominated upstream by phase-to-phase fault patterns; this is the smallest single-feature change that exits the PTGF leaf."
},
{
"feature": "Ia [Amp]",
"current_value": 21.5894,
"threshold": 4.3744,
"new_value": 4.3744,
"delta": 17.2151,
"direction": "decrease",
"predicted_new_class": "PTPF",
"feasibility": "feasible",
"interpretation": "Reducing phase-A current to at or below the node-3 boundary removes the high-Ia signature that finalized PTGF, sending the instance to the alternate child (non-PTGF) under the same low-speed/low-torque regime; given nodes 1–2 are enriched for PTPF, the likely reassignment is PTPF."
},
{
"feature": "Speed [rad/s] (node 2 split)",
"current_value": −32.1893,
"threshold": 104.1969,
"new_value": 104.1970,
"delta": 136.3863,
"direction": "increase",
"predicted_new_class": "PTPF",
"feasibility": "feasible",
"interpretation": "Increasing speed just above the second speed threshold (>104.1969 rad/s) reroutes the path at node 2 into a subtree whose parent prediction is PTPF, replacing the ground-related pattern with a regime more consistent with inter-phase imbalance."
},
{
"feature": "Speed [rad/s] (node 0 split)",
"current_value": −32.1893,
"threshold": 144.4185,
"new_value": 144.4186,
"delta": 176.6079,
"direction": "increase",
"predicted_new_class": "NOM",
"feasibility": "feasible",

(Continued)

Table A11 (continued)

Result
"interpretation": "Driving speed above the root threshold (>144.4185 rad/s) sends the sample to the opposite top-level branch where the node's prediction is nominal operation, effectively bypassing all downstream fault-oriented splits." }], "summary": "Decision reversibility is most sensitive to thresholds encountered early in the path or to the decisive split at node 3. The smallest perturbation is a modest torque increase of ≈0.53 N·m, which flips the branch at node 1 and likely reclassifies the instance toward PTPF. Eliminating the PTGF signature directly requires decreasing Ia by ≈17.22 A to cross node 3's boundary; this is physically plausible and redirects away from PTGF under unchanged low-speed conditions. Larger speed increases move the sample into alternate regimes: a +136.39 rad/s increase surpasses the node-2 threshold and likely yields PTPF, while a +176.61 rad/s increase crosses the root split and leads to NOM. Overall, small adjustments in torque or a substantial reduction in phase-A current are the most direct and interpretable routes to alter the PTGF decision, whereas very large speed increases are feasible but less practical from an operational standpoint." }

Table A12: Output of global consistency audit agent.

Result

{
"consistency_metrics": {
"local_top_features": ["Ia (Amp)", "Speed (rad/s)", "Tn (Rated Torque) Nm"],
"global_top_features": ["Speed (rad/s)", "Vab (V)", "Tn (Rated Torque) Nm"],
"jaccard_similarity": 0.5,
"rank_correlation": −0.4
},
"qualitative_assessment": {
"alignment": "Moderate",
"explanation": "Local saliency is dominated by Ia due to a decisive split (normalized impurity_drop ≈ 1.00 plus probability_shift = 1.0), while Speed and Tn provide earlier gating with smaller normalized impurity drops (Speed total ≈ 0.184; Tn ≈ 0.120). Globally, Speed and Vab rank highest, with Tn third and Ia relatively low. The overlap of Speed and Tn in the top-k sets supports partial alignment, but the absence of Vab locally and the strong local dominance of Ia indicate divergence. Likely causes include class-averaging in global importances (elevating Vab and Speed across multiple faults), node-specific impurity concentration for PTGF on Ia, and depth effects where Ia appears less frequently across the full tree but is decisive on the PTGF path."
},

(Continued)

Table A12 (continued)

Result
"summary": "Local evidence for the PTGF instance is led by Phase-A current (Ia), with Speed and Tn as supporting gates, whereas global importances emphasize Speed and Vab with Tn as secondary. The top-k overlap (Speed, Tn) yields a Jaccard of 0.50, but the rank correlation is negative (−0.40), reflecting that Ia is locally decisive yet globally modest. Overall, the explanation is partially consistent: key operational factors (Speed, Tn) align, while PTGF-specific reliance on Ia is underrepresented in the model's global profile." }

Table A13: Output of report generator agent.

Result

{
"report": {
"prediction_summary": "The Decision Tree classifier predicted the fault class as PTGF (Phase-to-Ground Fault) with a confidence of 1.0, indicating complete certainty in this classification. The decision path involved sequential rule evaluations on rotational speed, torque, and phase-A current, progressively narrowing the instance toward the PTGF leaf node through impurity reductions culminating in a zero-impurity, single-class prediction.",
"key_rules": [
{
"feature": "Speed (rad/s)",
"threshold": 144.4185,
"sample_value": −32.1893,
"effect": "Extremely low rotational speed excluded the normal operation regime and initiated classification into fault-related zones."
},
{
"feature": "Tn (Rated Torque) N*m",
"threshold": 1.375,
"sample_value": 0.85,
"effect": "Low torque confirmed minimal load conditions, further constraining the decision path toward fault-prone states."
},
{
"feature": "Speed (rad/s)",
"threshold": 104.1969,
"sample_value": −32.1893,
"effect": "Reinforcement of persistently low rotational speed signaled an off-nominal, quasi-stationary state commonly linked to grounding-related faults."
},
{

(Continued)

Table A13 (continued)

Result

"feature": "Ia (Amp)",
"threshold": 4.3744,
"sample_value": 21.5894,
"effect": "Phase-A current far above the critical threshold triggered the decisive rule leading to PTGF classification by eliminating residual impurity."
}
],
"threshold_interpretations": [
{
"feature": "Speed (rad/s)",
"semantic_expression": "The motor operated at a very low or negative rotational speed (−32.19 rad/s), indicating a near-stall or counter-rotation far below nominal range."
},
{
"feature": "Tn (Rated Torque) N*m",
"semantic_expression": "The torque output of 0.85 N·m reflects an under-loaded condition, characteristic of idle or no-load operation."
},
{
"feature": "Ia (Amp)",
"semantic_expression": "Phase-A current (21.59 A) substantially exceeds the normal operating threshold (4.37 A), signifying a phase-to-ground overcurrent anomaly."
}
],
"counterfactuals": [
{
"feature": "Tn (rated torque) [N·m]",
"delta": 0.5251,
"new_value": 1.3751,
"new_class": "PTPF",
"interpretation": "A slight increase in torque above 1.375 N·m would redirect the classification path toward a phase-to-phase fault (PTPF), reflecting sensitivity to load level."
},
{
"feature": "Ia [Amp]",
"delta": 17.2151,
"new_value": 4.3744,
"new_class": "PTPF",
"interpretation": "Reducing phase-A current to approximately 4.37 A removes the defining overcurrent condition, leading to a reclassification as PTPF under unchanged low-speed conditions."
},

(Continued)

Table A13 (continued)

Result

{
"feature": "Speed [rad/s]",
"delta": 136.3863,
"new_value": 104.1970,
"new_class": "PTPF",
"interpretation": "Raising speed slightly above 104.20 rad/s moves the instance into a higher-speed regime where inter-phase fault patterns dominate."
},
{
"feature": "Speed [rad/s]",
"delta": 176.6079,
"new_value": 144.4186,
"new_class": "NOM",
"interpretation": "Increasing rotational speed beyond 144.42 rad/s transitions the instance into the normal-operation branch, bypassing all fault-oriented nodes."
}
],
"global_consistency": "The local decision pathway emphasized Phase-A current (Ia) as the dominant determinant, supported by Speed and Torque. Globally, the model prioritizes Speed, Voltage (Vab), and Torque. The Jaccard similarity between local and global top features is 0.50, with a negative rank correlation (−0.40), indicating partial consistency—core mechanical factors align, but current-based indicators are underrepresented in the global profile.",
"interpretive_summary": "The Decision Tree's reasoning for the PTGF classification centers on a distinctive pattern: extremely low rotational speed (−32.19 rad/s), low mechanical load (0.85 N·m), and an abnormally high Phase-A current (21.59 A). This combination reflects a grounded phase under non-rotational conditions, which the model identifies with maximal certainty. Counterfactual analysis shows that modest increases in torque or speed, or a substantial reduction in Ia, could shift the classification toward PTPF or NOM, underscoring the model's sensitivity to current and load dynamics. The global audit reveals partial alignment between local and global explanatory trends—while mechanical parameters such as Speed and Torque are consistently influential, the PTGF pathway's dependence on current-specific evidence (Ia) highlights localized fault specialization not fully captured by global importance averages. Overall, the integrated analysis confirms the model's capacity to identify class-specific electrical anomalies with high precision while emphasizing the need for class-level interpretability audits for comprehensive diagnostic transparency."
}
}

Table A14: Result of ablation study

Result
{
“report”: {
“prediction_summary”: “The decision tree classified the sample (index 69) as a Phase-to-Ground Fault (PTGF) with a predicted probability of 1.0. The sample traversed a five-node path in which impurity decreased from 0.7968 at the root to 0.0 at the leaf, indicating a fully homogeneous PTGF region.”,
“key_rules”: [
{
“feature”: “Speed (rad/s)”,
“threshold”: 144.4185,
“sample_value”: −32.1893,
“effect”: “Values below this threshold route the instance toward a subsystem where faults, rather than normal operation, dominate (impurity 0.7968 → 0.7546).”
},
{
“feature”: “Tn (Rated Torque) N*m”,
“threshold”: 1.375,
“sample_value”: 0.85,
“effect”: “A torque lower than 1.375 N·m steers the classification away from high-torque fault modes and into a region primarily separating PTPF and PTGF (impurity 0.7546 → 0.6749).”
},
{
“feature”: “Speed (rad/s)”,
“threshold”: 104.1969,
“sample_value”: −32.1893,
“effect”: “Remaining below this secondary speed threshold places the instance into a low-speed subregion dominated by PTGF vs. PTPF discrimination (impurity 0.6749 → 0.012).”
},
{
“feature”: “Ia (Amp)”,
“threshold”: 4.3744,
“sample_value”: 21.5894,
“effect”: “Crossing above this current threshold leads directly to a pure PTGF leaf node (impurity 0.012 → 0.0).”
}
],
“threshold_interpretations”: [
{
“feature”: “Speed (rad/s)”,
“semantic_expression”: “The model treats speeds lower than 144.4 rad/s as operationally distinct, grouping them with conditions more frequently associated with non-normal classes.”
},
{

(Continued)

Table A14 (continued)

Result

“feature”: “Tn (Rated Torque) N*m”,
“semantic_expression”: “Torque values below 1.375 N·m delineate a lower-load regime where the model historically observed specific fault separations, particularly between PTPF and PTGF.”
},
{
“feature”: “Speed (rad/s)”,
“semantic_expression”: “Remaining below 104.2 rad/s signals an even more restricted behavior regime in which the model predominantly distinguishes between PTGF and PTPF outcomes.”
},
{
“feature”: “Ia (Amp)”,
“semantic_expression”: “Exceeding approximately 4.37 A on phase A current places the sample into a region consistently associated with PTGF within the training data.”
}
],
“counterfactuals”: [
{
“feature”: “Ia (Amp)”,
“delta”: −17.215,
“new_value”: 4.3744,
“new_class”: “PTPF”,
“interpretation”: “Reducing Ia to the threshold boundary would move the sample from the PTGF-pure leaf back to the parent node, where PTPF was the majority prediction (predicted class at node 2).”
},
{
“feature”: “Speed (rad/s)”,
“delta”: 136.5862,
“new_value”: 104.1969,
“new_class”: “PTPF”,
“interpretation”: “Raising speed to the secondary threshold could revert the traversal to a region where PTPF dominates before the Ia split.”
},
{
“feature”: “Tn (Rated Torque) N*m”,
“delta”: 0.525,
“new_value”: 1.375,
“new_class”: “PTPF”,
“interpretation”: “Increasing torque to the split boundary at node 1 would reroute the sample upward to a path where the predicted class shifts from PTGF toward PTPF.”
}

(Continued)

Table A14 (continued)

Result
], "global_consistency": "The local reasoning path aligns with global model behavior: PTGF predictions frequently emerge in low-speed, low-torque, and elevated phase-A-current regions. The monotonic impurity drop along the path matches this pattern, culminating in a pure PTGF leaf.", "interpretive_summary": "The model's PTGF prediction arises from a sequence of threshold-based decisions that progressively isolate a low-speed, low-torque, and high-Ia operational regime. Each split consistently narrows the classification space, reducing impurity from 0.7968 to 0.0, and ultimately placing the instance in a region exclusively associated with PTGF in the training data. Counterfactual analyses indicate that modest adjustments to Ia, torque, or speed would redirect the sample into regions dominated by PTPF, highlighting the boundary-sensitive nature of the decision. Overall, the decision tree's reasoning for this sample coherently reflects its global structural tendencies toward using current and kinematic state thresholds to delimit PTGF conditions." } }

References

1. Umayal R, Darapaneni N, Aditya V, Paduri AR. Machine learning based remaining useful life prediction of lithium-ion batteries in electric vehicle battery management system. In: Proceedings of the 2023 International Conference on Communication, Security and Artificial Intelligence (ICCSAI); 2024 Nov 23–25; Greater Noida, India. p. 547–51. doi:10.1109/ICCSAI59793.2023.10421014.
2. Gopalakrishnan R, Vidhya DS, Bharath P, Kaviyan P, Sivaselvam P. Battery state estimation and control system for mobile charging station for electric vehicles. In: Proceedings of the 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS); 2023 Mar 23–25; Erode, India. p. 985–9. doi:10.1109/ICSCDS56580.2023.10104624.
3. Haque ME, Zabin M, Uddin J. EnsembleXAI-motor: a lightweight framework for fault classification in electric vehicle drive motors using feature selection, ensemble learning, and explainable AI. Machines. 2025;13(4):314. doi:10.3390/machines13040314.
4. Khaneghah MZ, Alzayed M, Chaoui H. Fault detection and diagnosis of the electric motor drive and battery system of electric vehicles. Machines. 2023;11(7):713. doi:10.3390/machines11070713.
5. Thirunavukkarasu S, Karthick K, Aruna SK, Manikandan R, Safran M. Optimized fault classification in electric vehicle drive motors using advanced machine learning and data transformation techniques. Processes. 2024;12(12):2648. doi:10.3390/pr12122648.
6. Zhu S, Tan MK, Chin RKY, Chua BL, Hao X, Teo KTK. Engine fault diagnosis using probabilistic neural network. In: Proceedings of the 2021 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET); 2021 Sep 13–15; Kota Kinabalu, Malaysia. p. 1–6. doi:10.1109/iicaiet51634.2021.9573654.
7. Akbalık F, Yıldız A, Ertuğrul ÖF, Zan H. Engine fault detection by sound analysis and machine learning. Appl Sci. 2024;14(15):6532. doi:10.3390/app14156532.
8. Lee J, Baik N, Rew J. Multi-scale graph neural network for multivariate time-series anomaly detection. J Korea Soc Ind Inf Syst. 2025;30(1):51–65. doi:10.9723/jksiis.2025.30.1.051.
9. Dettinger F, Jazdi N, Weyrich M, Brandl L, Reuss HC, Pecha U, et al. Machine-learning-based fault detection in electric vehicle powertrains using a digital twin. In: SAE technical paper series. Stuttgart, Germany: SAE International; 2023. doi:10.4271/2023-01-1214.
10. Mersha M, Lam K, Wood J, AlShami AK, Kalita J. Explainable artificial intelligence: a survey of needs, techniques, applications, and future direction. Neurocomputing. 2024;599(5):128111. doi:10.1016/j.neucom.2024.128111.

11. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. Adv Neural Inf Process Syst. 2017;30:4766–75.
12. Ribeiro M, Singh S, Guestrin C. "Why should I trust you?": explaining the predictions of any classifier. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. Stroudsburg, PA, USA: ACL; 2016. p. 97–101. doi:10.18653/v1/n16-3020.
13. Lee J, Rew J. Large language model-based SHAP analysis for interpretation of remaining useful life prediction of lithium-ion battery. J Korea Soc Ind Inf Syst. 2024;29(5):51–68. doi:10.2139/ssrn.5537792.
14. Lee J, Rew J. Vision-language model-based local interpretable model-agnostic explanations analysis for explainable in-vehicle controller area network intrusion detection. Sensors. 2025;25(10):3020. doi:10.3390/s25103020.
15. Saranya A, Subhashini R. A systematic review of explainable artificial intelligence models and applications: recent developments and future trends. Decis Anal J. 2023;7(5):100230. doi:10.1016/j.dajour.2023.100230.
16. Izza Y, Ignatiev A, Marques-Silva J. On explaining decision trees. arXiv:2010.11034. 2020.
17. Minaee S, Mikolov T, Nikzad N, Chenaghlu M, Socher R, Amatriain X, et al. Large language models: a survey. arXiv:2402.06196. 2024.
18. Guo T, Chen X, Wang Y, Chang R, Pei S, Chawla NV, et al. Large language model based multi-agents: a survey of progress and challenges. arXiv:2402.01680. 2024.
19. Bari BS, Yelamarthi K, Ghafoor S. Intrusion detection in vehicle controller area network (CAN) bus using machine learning: a comparative performance study. Sensors. 2023;23(7):3610. doi:10.3390/s23073610.
20. Alalwany E, Mahgoub I. Classification of normal and malicious traffic based on an ensemble of machine learning for a vehicle CAN-network. Sensors. 2022;22(23):9195. doi:10.3390/s22239195.
21. Xu L, Teoh SS, Ibrahim H. A deep learning approach for electric motor fault diagnosis based on modified InceptionV3. Sci Rep. 2024;14(1):12344. doi:10.1038/s41598-024-63086-9.
22. Hassija V, Chamola V, Mahapatra A, Singal A, Goel D, Huang K, et al. Interpreting black-box models: a review on explainable artificial intelligence. Cogn Comput. 2024;16(1):45–74. doi:10.1007/s12559-023-10179-8.
23. Zereen AN, Das A, Uddin J. Machine fault diagnosis using audio sensors data and explainable AI techniques-LIME and SHAP. Comput Mater Contin. 2024;80(3):3463–84. doi:10.32604/cmc.2024.054886.
24. Jang K, Pilario KES, Lee N, Moon I, Na J. Explainable artificial intelligence for fault diagnosis of industrial processes. IEEE Trans Ind Inform. 2025;21(1):4–11. doi:10.1109/TII.2023.3240601.
25. Brito LC, Susto GA, Brito JN, Duarte MAV. Fault Diagnosis using eXplainable AI: a transfer learning-based approach for rotating machinery exploiting augmented synthetic data. Expert Syst Appl. 2023;232(4):120860. doi:10.1016/j.eswa.2023.120860.
26. Lee J, Rew J. Memory-augmented large language model for enhanced chatbot services in university learning management systems. Appl Sci. 2025;15(17):9775. doi:10.3390/app15179775.
27. Lee J, Rew J. Automated change history analysis of software bill of materials (SBOM) using large language model-based structural analysis and forgery detection agents. J Korea Soc Ind Inf Syst. 2025;30(4):39–60. doi:10.9723/jksiis.2025.30.4.039.
28. Holland M, Chaudhari K. Large language model based agent for process planning of fiber composite structures. Manuf Lett. 2024;40(2):100–3. doi:10.1016/j.mfglet.2024.03.010.
29. Peng C, Peng J, Wang Z, Wang Z, Chen J, Xuan J, et al. Adaptive fault diagnosis of railway vehicle on-board controller with large language models. Appl Soft Comput. 2025;185(1):113919. doi:10.1016/j.asoc.2025.113919.
30. Lukens S, McCabe LH, Gen J, Ali A. Large language model agents as prognostics and health management copilots. Annu Conf PHM Soc. 2024;16(1):3906. doi:10.36001/phmconf.2024.v16i1.3906.
31. Blockeel H, Devos L, Frénay B, Nanfack G, Nijssen S. Decision trees: from efficient prediction to responsible AI. Front Artif Intell. 2023;6:1124553. doi:10.3389/frai.2023.1124553.
32. Yu T, Zhu H. Hyper-parameter optimization: a review of algorithms and applications. arXiv:2003.05689. 2020.
33. Moshkov M. On the depth of decision trees over infinite 1-homogeneous binary information systems. Array. 2021;10(7):100060. doi:10.1016/j.array.2021.100060.

34. H. Mahmood. Khan. Machine learning based fault diagnosis of electric drives. GitHub. [cited 2026 Jan 20]. Available from: https://github.com/HassanMahmoodKhan/Machine-Learning-Based-Fault-Diagnosis-of-Electric-Drives.
35. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. J Mach Learn Res. 2011;12:2825–30.
36. OpenAI. GPT-5 system card. arXiv:2601.03267. 2026.
37. Vujovic ŽĐ. Classification model evaluation metrics. Int J Adv Comput Sci Appl. 2021;12(6):0120670. doi:10.14569/ijacsa.2021.0120670.
38. Chiang WL, Gonzalez J, Li D, Li Z, Lin Z, Sheng Y, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In: Proceedings of the Advances in Neural Information Processing Systems 36; 2023 Dec 10–16; New Orleans, LA, USA: Neural Information Processing Systems Foundation, Inc. (NeurIPS); 2023. p. 46595–623. doi:10.52202/075280-2020.
39. Rong Y, Leemann T, Nguyen TT, Fiedler L, Qian P, Unhelkar V, et al. Towards human-centered explainable AI: a survey of user studies for model explanations. IEEE Trans Pattern Anal Mach Intell. 2024;46(4):2104–22. doi:10.1109/tpami.2023.3331846.
40. Joshi A, Kale S, Chandel S, Pal D. Likert scale: explored and explained. Br J Appl Sci Technol. 2015;7(4):396–403. doi:10.9734/bjast/2015/14975.
41. Ul Islam Khan M, Pathan MIH, Mominur Rahman M, Islam MM, Arfat Raihan Chowdhury M, Anower MS, et al. Securing electric vehicle performance: machine learning-driven fault detection and classification. IEEE Access. 2024;12(12):71566–84. doi:10.1109/access.2024.3400913.
42. Rocha RCN, Soares RA, Santos LI, Camargos MO, Ekel PY, Libório MP, et al. A new fault classification approach based on decision tree induced by genetic programming. Processes. 2024;12(4):818. doi:10.3390/pr12040818.
43. Zhang Z, Dong S, Li D, Liu P, Wang Z. Prediction and diagnosis of electric vehicle battery fault based on abnormal voltage: using decision tree algorithm theories and isolated forest. Processes. 2024;12(1):136. doi:10.3390/pr12010136.
44. Wang X, Tang Z, Guo J, Meng T, Wang C, Wang T, et al. Empowering edge intelligence: a comprehensive survey on on-device AI models. ACM Comput Surv. 2025;57(9):1–39. doi:10.1145/3724420.
45. Sahoo P, Singh AK, Saha S, Jain V, Mondal S, Chadha A. A systematic survey of prompt engineering in large language models: techniques and applications. arXiv:2402.07927. 2024.
46. Vatsal S, Dubey H. A survey of prompt engineering methods in large language models for different NLP tasks. arXiv:2407.12994. 2024.
47. Kalyan KS. A survey of GPT-3 family large language models including ChatGPT and GPT-4. Nat Lang Process J. 2024;6(6):100048. doi:10.1016/j.nlp.2023.100048.
48. Singh H, Verma N, Wang Y, Bharadwaj M, Fashandi H, Ferreira K. Personal large language model agents: a case study on tailored travel planners. In: Proceedings of the EMNLP 2024—2024 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Industry Track. Miami, FL, USA: Association for Computational Linguistics; 2024. p. 486–514.
49. Jia F, Ye Z, Lai S, Shu K, Gu J, Bibi A, et al. Can large language model agents simulate human trust behaviors? arXiv:2402.04559. 2024.
50. Cloud N. HyperCLOVA X THINK technical report. arXiv:2506.22403. 2025.
51. Grattafiori A, Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, et al. The Llama 3 herd of models. arXiv:2407.21783. 2024.
52. Hang J, Qiu G, Hao M, Ding S. Improved fault diagnosis method for permanent magnet synchronous machine system based on lightweight multisource information data layer fusion. IEEE Trans Power Electron. 2024;39(10):13808–17. doi:10.1109/TPEL.2024.3432163.
53. He W, Hang J, Ding S, Sun L, Hua W. Robust diagnosis of partial demagnetization fault in PMSMs using radial air-gap flux density under complex working conditions. IEEE Trans Ind Electron. 2024;71(10):12001–10. doi:10.1109/TIE.2024.3349520.

54. Gu J, Han Z, Chen S, Beirami A, He B, Zhang G, et al. A systematic survey of prompt engineering on vision-language foundation models. arXiv:2307.12980. 2023.
55. Wu C, Sehab R, Akrad A, Morel C. Fault diagnosis methods and fault tolerant control strategies for the electric vehicle powertrains. Energies. 2022;15(13):4840. doi:10.3390/en15134840.
56. Cui X. Research on common faults and maintenance countermeasures of automobile engines. J Educ Teach Soc Stud. 2023;5(2):p68. doi:10.22158/jetss.v5n2p68.
57. Xu X, Hang J, Cao K, Ding S, Wang W. Unified open-phase fault diagnosis of five-phase PMSM system in normal operation and fault-tolerant operation modes. IEEE Trans Transp Electrif. 2025;11(5):12063–75. doi:10.1109/TTE.2025.3584777.
58. Hossain MN, Rahman MM, Ramasamy D. Artificial intelligence-driven vehicle fault diagnosis to revolutionize automotive maintenance: a review. Comput Model Eng Sci. 2024;141(2):951–96. doi:10.32604/cmes.2024.056022.
59. Hua L, Tang J, Zhu G. A survey of vehicle system and energy models. Actuators. 2025;14(1):10. doi:10.3390/act14010010.
60. Li L, Haruna A, Ying W, Noman K, Li Y. Knowledge graph-driven fault diagnosis for aviation equipment: integrating improved joint extraction with large language model. J Ind Inf Integr. 2026;50:101039. doi:10.1016/j.jii.2025.101039.
61. Haruna A, Noman K, Li Y, Makanda ILD, Zubair A, Hasand MJ, et al. Facilitating heuristic reasoning by utilizing knowledge graph and natural language processing. Knowl Based Syst. 2026;334:115153. doi:10.1016/j.knosys.2025.115153.
62. Hasan MJ, Sohaib M, Kim JM. An explainable AI-based fault diagnosis model for bearings. Sensors. 2021;21(12):4070. doi:10.3390/s21124070.
63. Huang Y, Jafari M, Jin PJ. Driving safety prediction and safe route mapping using in-vehicle and roadside data. SSRN J. 2022;59(12):604. doi:10.2139/ssrn.4135994.