Computing (2025) 107:225
https://doi.org/10.1007/s00607-025-01585-x

REGULAR PAPER

Intelligent predictive maintenance: multivariate ML model optimization in an Edge-Fog-Cloud environment

Preeti Jain[1] · **Koustav Kumar Mondal**[1] · **Debasis Das**[1,2] · **Arpit Khandelwal**[1,3]

Received: 1 April 2025 / Accepted: 22 October 2025 / Published online: 5 November 2025
© The Author(s), under exclusive licence to Springer-Verlag GmbH Austria, part of Springer Nature 2025

Abstract

As a cornerstone of the global economy, the automotive industry requires intelligent and reliable maintenance approaches. Predictive maintenance provides a proactive mechanism for timely fault detection, minimized operational interruptions, and prolonged vehicle lifespan. Traditional cloud-based predictive maintenance solutions, however, face latency, bandwidth constraints, and limited scalability challenges in real-time applications. This paper proposes FogBayes, an intelligent predictive maintenance system built on a three-layer Edge–Fog–Cloud architecture to balance computation and minimize latency, employing optimized multivariate machine learning models at its core. The proposed system model achieves a prediction accuracy of 98.88% and an AUC (Area under curve) of 0.9830, outperforming several baseline approaches. To ensure transparency and reproducibility, the complete source code, datasets, and experimental configurations used in this study are publicly available at https://github.com/TakMashhido/FogBayes.

Keywords Predictive maintenance · Vehicle health prediction · Fog and edge computing · Cloud computing · Internet of vehicles (IoV) · Machine learning for vehicular systems

Mathematics Subject Classifications 68T05 · 68M14 · 68U35

Preeti Jain and Koustav Kumar Mondal have contributed equally to this work.

Extended author information available on the last page of the article

1 Introduction

The increasing complexity and connectivity in the automotive industry have made predictive maintenance (PdM) an essential strategy to ensure vehicle reliability, safety, and cost efficiency [1]. Predictive maintenance uses real-time data from vehicle sensors and advanced machine learning (ML) models to proactively detect failures before they occur, thus reducing unexpected downtime and repair costs. However, traditional maintenance methods, such as scheduled inspections or reactive repairs, are often inefficient. Studies reveal that only 18% of vehicle failures are age-related, while 82% occur unpredictably, highlighting the limitations of conventional maintenance strategies [2]. Furthermore, improper monitoring can result in up to 40% of maintenance costs being wasted, underscoring the need for more effective and efficient maintenance solutions. Centralized PdM systems, which rely on cloud-based architectures, have shown potential in analyzing large-scale vehicle data. However, they suffer from inherent challenges such as high latency, energy inefficiency, and scalability issues, particularly in scenarios requiring real-time decision-making [3]. These limitations are exacerbated in dynamic environments like the Internet of Vehicles (IoV), where large volumes of sensor data from connected vehicles need to be processed quickly and accurately. The IoV ecosystem introduces additional complexities such as frequent data transmissions, system congestion, and the need for real-time adaptability, which existing centralized architectures struggle to handle efficiently. In latency-critical Internet of Vehicles (IoV) applications, relying solely on centralized cloud-based solutions often fails to meet real-time safety and efficiency requirements due to inherent communication delays and network uncertainties. For example, in collision avoidance systems, vehicles must process sensor inputs and trigger braking or evasive maneuvers within milliseconds; however, round-trip delays to the cloud–ranging from tens to hundreds of milliseconds–are often too slow to prevent accidents. Similarly, in adaptive traffic signal control, real-time vehicle data must be analyzed instantly to dynamically adjust signal timings and prevent congestion, yet centralized processing cannot keep pace with rapidly changing traffic patterns. Furthermore, in remote vehicle diagnostics and emergency response, events such as brake failure or engine overheating demand immediate, localized decisions to trigger safety protocols, where dependence on distant cloud servers could jeopardize timely interventions. These scenarios clearly demonstrate that Fog/Edge computing provides a decentralized alternative, bringing computation closer to vehicles, reducing latency, ensuring reliability under variable network conditions, and enabling real-time predictive maintenance and safety control mechanisms essential for IoV environments.

To address these challenges, this study proposes a fog-enabled multi-layered architecture that hierarchically distributes computational tasks across the edge, fog, and cloud layers. In this architecture, edge computing performs localized data preprocessing, fog computing enables intermediate decision-making and localized machine learning, and cloud computing provides large-scale model optimization and storage. By leveraging the strengths of distributed computing, the framework minimizes latency, reduces network congestion, and optimizes resource utilization, thus overcoming the limitations of traditional centralized systems.

One of the key components of this study is the integration of a Multivariate ML Model Optimization within the fog computing layer. This ML pipeline improves the adaptability and precision of predictive models by employing Bayesian optimization for hyperparameter tuning. The proposed framework demonstrates significant advancements in predictive maintenance for IoV systems. Experimental results show that the FogBayes pipeline achieves a prediction accuracy of 98.88% and an AUC(Area Under the ROC Curve) of 0.9830, significantly outperforming traditional methods.

1.1 Motivation

The global automotive industry, valued at USD 5.93 billion in 2023 and projected to grow to USD 8.35 billion by 2030 at a CAGR of 27.4% [4], faces increasing risks from outdated maintenance practices. While onboard sensors generate abundant operational data, the lack of real-time analytics limits the effectiveness of traditional approaches.To address these gaps, this study integrates fog computing and Bayesian-optimized machine learning to minimize latency and optimize data processing in vehicular networks, improve the accuracy and scalability of predictive maintenance systems, and enhance vehicle safety, operational efficiency, and component longevity.

1.2 Problem statement

This research addresses several key challenges, including transmission issues and system congestion in Internet of Vehicles (IoV) scenarios, where frequent data transmissions can overwhelm centralized servers, leading to increased latency ($\mathcal{T}_T$) and congestion; this is mitigated through efficient task distribution across edge and fog layers. Additionally, it focuses on engine health monitoring, which requires robust models to minimize false positives (FP) and false negatives (FN) while maximizing precision and recall. Another critical challenge is inefficient data processing, as static, centralized predictive maintenance (PdM) models cannot adapt to real-time changes in vehicular conditions, highlighting the need for distributed frameworks capable of dynamic updates.

1.3 Contributions

The key contributions of this paper are as follows. First, a distributed edge-fog-cloud architecture is proposed to optimize resource utilization and reduce system congestion. Second, the integration of a Bayesian-Optimized Light Gradient Boosting Machine (BOGBM) pipeline enables real-time predictive maintenance by ensuring low-latency processing and high prediction accuracy. Third, the model's performance is further enhanced through the application of advanced Bayesian optimization techniques for effective hyperparameter tuning.

This paper is organized as follows: Sect. 2 reviews related work, Sect. 3 presents the proposed methodology, Sect. 4 details the experimental setup and dataset, Sect. 5 presents the results and analyses, Sect. 6 provides a comprehensive discussion of the experimental results, and Sect. 7 concludes with future directions.

2 Related work

This section reviews existing research on predictive maintenance and computing architectures, highlighting advancements relevant to cloud-fog-edge integration and machine learning applications.

2.1 Machine learning-based predictive maintenance approaches

Various machine learning approaches have been proposed for predictive maintenance in diverse domains [5]. For instance, Xie et al. [6] introduced an ensemble learning method for surface roughness prediction, while Pagano et al. [7] combined Long Short-Term Memory (LSTM) networks with Bayesian inference for monitoring industrial plants. Yang et al. [8] proposed FedPreM, a federated reinforcement learning-based predictive maintenance scheme to address data privacy and non-IID challenges. In the vehicular domain, most studies focus on commercial fleets or autonomous systems. For example, Massaro et al. [9] utilized an MLP-ANN model for analyzing bus fleet driver behavior, and Gomes et al. [10] proposed a hierarchical Bayesian network for health monitoring in autonomous vehicles.

Despite advancements, research on predictive maintenance specifically for personal vehicles remains limited. Recent efforts, such as PreM-FedIoV [11], address data privacy and connectivity issues in IoV scenarios but lack a focus on engine health prediction using distributed architectures. To the best of our knowledge, no prior work utilizes an edge-fog-cloud layered architecture for vehicle health prediction based on engine parameters in the IoV context. This study addresses this gap, integrating distributed computing and machine learning to achieve real-time, efficient predictive maintenance. Table 1 compares FogBayes framework with the most recent publications in IoV-based predictive maintenance. PreM-FedIoV [11] is a federated learning approach optimized for vehicle fleets, but without edge-level preprocessing or per-engine feature selection. IoV-TwinChain [12] employs a federated digital-twin architecture but lacks a true fog layer and does not incorporate engine-specific tuning. SGSFCN [13] employs cloud CNN-based strategy that neither addresses vehicular network constraints nor real-time latency. IntelliPdM [14] which supports edge-cloud collaboration but omits fog-layer adaptation and detailed engine-parameter modeling.

Table 1 Comparison of methodologies for predictive maintenance in IoV

Methodology	IoV Scenario	Architecture	Engine-Specific Parameters	PdM for Vehicles
FogBayes (Our Method)	✓	Edge-Fog-Cloud	✓	✓
PreM-FedIoV [11]	✓	✗	✗	✓
IoV-TwinChain [12]	✓	✗	✓	✓
SGSFCN [13]	✗	✗	✗	✗
IntelliPdM [14]	✗	edge-cloud	✗	✗

2.2 Edge-Fog-Cloud architecture-based solutions

Cloud-fog-edge architectures have been explored across various domains for task offloading, resource management, and predictive analytics. Studies have demonstrated the potential of cloud-fog architectures in task offloading [15], scheduling [16], and resource orchestration [17]. These works leverage fog nodes to optimize resource utilization and reduce cloud dependency [18]. However, most focus on improving efficiency and lack distributed machine learning implementations, limiting their adaptability to data-intensive, real-time applications. The integration of machine learning at the fog layer has enhanced monitoring and diagnostic tasks [19, 20]. Although some works demonstrate improved predictive capabilities using fog nodes [21], few have implemented distributed machine learning across cloud, fog, and edge layers, leading to suboptimal utilization of the layered architecture's potential. Recent studies such as [22] have emphasized the importance of scalable and low-latency inference deployment in distributed environments, aligning with the fog–cloud paradigm. Likewise, [23] proposed energy-efficient and adaptive resource allocation mechanisms for sustainable computing. Moreover, [24] explored decentralized learning in multi-agent systems, offering insights for distributed vehicular intelligence and [25] highlighted trust, privacy, and reliability concerns crucial to data-driven predictive maintenance.

In the vehicular domain, fog computing has been used for predictive maintenance tasks such as real-time health monitoring and failure prediction [26, 27]. While these studies demonstrate the benefits of fog-based solutions in reducing latency and improving reliability, they rely primarily on centralized model training, limiting the efficiency of real-time predictions and adaptation at the edge and fog layers [28, 29]. This study bridges these gaps by incorporating distributed machine learning across edge, fog, and cloud environments, enabling scalable, low-latency, and accurate predictive maintenance in IoV scenarios.

3 Proposed Edge-Fog-Cloud architecture

The proposed architecture integrates a hierarchical Edge-Fog-Cloud framework to facilitate efficient, scalable, and real-time vehicle health prediction. This tri-layered structure as shown in the Fig. 1 ensures optimal resource utilization, reduced latency, and robust scalability by dynamically distributing computational tasks across the Edge, Fog, and Cloud layers. The interactions and dependencies among these layers are pivotal for the seamless operation of the system, as illustrated in Fig. 2. The system design assumes that MQTT-based communication between edge, fog, and cloud components is sufficiently reliable for transmitting sensor data and receiving model updates in near-real time.

3.1 Edge computing

At the foundation of the architecture lies the Edge Computing layer, where each vehicle is equipped with an array of onboard sensors acting as primary data acquisition

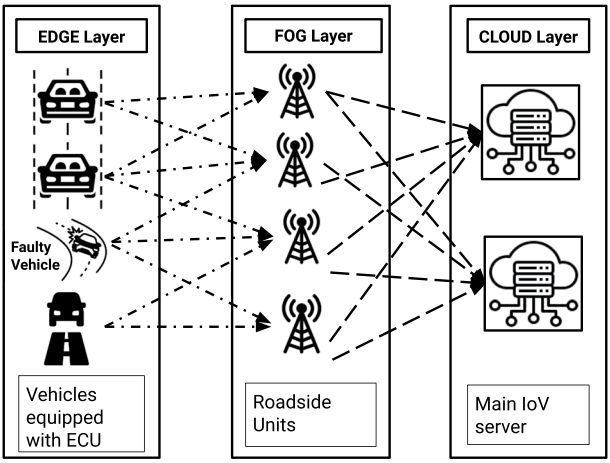


Fig. 1 Three-layered architecture for vehicle health prediction

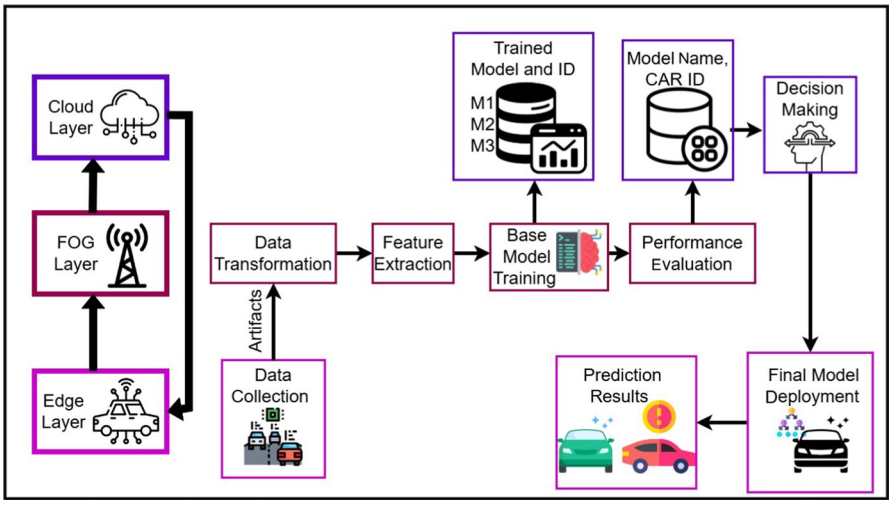


Fig. 2 Edge(Pink colored)–Fog(Dark Red colored)–Cloud(Blue colored) architecture with distinct roles across layers. The Edge and Fog perform real-time prediction and local inference, while the Cloud handles global model training, historical data storage, and model orchestration. We emphasize the FOG layer in uppercase to highlight its central role in the proposed FOGBayes algorithm. Directional arrows and color-coded boxes distinguish real-time and batch operations

nodes. These sensors, including temperature sensors, accelerometers, and GPS modules, continuously generate high-dimensional raw data vectors, as defined in Eq. (1):

$$\mathrm{X}_v(t) = \{x_{v,1}(t), x_{v,2}(t), \ldots, x_{v,n}(t)\}, \quad v \in \{1, 2, \ldots, V\} \tag{1}$$

Raw data $\mathrm{X}_v(t)$ undergoes preprocessing to mitigate noise and standardize the data distribution. The preprocessing pipeline includes noise filtering and normalization. In

the noise filtering step, a low-pass Finite Impulse Response (FIR) filter $\mathcal{F}$ is applied to remove high-frequency noise. This process can be mathematically expressed as Eq. (2).

$$\tilde{\mathbf{X}}_v(t) = \mathcal{F}(\mathrm{X}_v(t)) = \sum_{k=0}^{K-1} h_k x_v(t-k) \tag{2}$$

h_k are the filter coefficients and K is the filter order. Next, normalization is performed to standardize the data to have zero mean and unit variance:

$$\hat{\mathrm{X}}_v(t) = \frac{\mathrm{X}_v(t) - \boldsymbol{\mu}_v}{\boldsymbol{\sigma}_v}, \quad \boldsymbol{\mu}_v = \frac{1}{T}\sum_{t=1}^{T} \mathrm{X}_v(t), \quad \boldsymbol{\sigma}_v = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\mathrm{X}_v(t) - \boldsymbol{\mu}_v)^2}$$

T is the total number of time instances. Post preprocessing, feature extraction techniques distill essential artifacts from the normalized data $\hat{\mathrm{X}}_v(t)$. Let $\Phi : \mathbb{R}^n \to \mathbb{R}^p$ represent the feature extraction function, which maps the preprocessed input to a lower-dimensional feature space, with its operation defined in Eq. (3):

$$\mathrm{F}_v(t) = \Phi(\hat{\mathrm{X}}_v(t)) = \{f_{v,1}(t), f_{v,2}(t), \ldots, f_{v,p}(t)\}, \quad p < n \tag{3}$$

Common feature extraction methods include Principal Component Analysis (PCA), Fourier Transforms, and domain-specific feature engineering. The MQTT protocol facilitates efficient and low-latency communication between vehicles and Roadside Units (RSUs) at the Fog layer.The data transmission can be formalized using the mapping function given in Eq. (4).

$$\mathcal{T} : \mathrm{F}_v(t) \to \mathrm{D}_F(t) \tag{4}$$

$\mathcal{T}$ denotes the transmission function and $\mathrm{D}_F(t)$ represents the aggregated data at the Fog layer.

3.2 Fog computing

The Fog Computing layer, comprising RSUs, serves as an intermediary processing tier that aggregates and preprocesses data from multiple vehicles. This layer bridges the gap between the resource-constrained Edge layer and the centralized Cloud layer, ensuring timely data processing and initial model training. Aggregated preprocessed feature data from V vehicles is denoted as $\mathrm{D}_F(t) = \bigcup_{v=1}^{V} \mathrm{F}_v(t)$.

Advanced transformations are applied to the aggregated data, encompassing feature selection and correlation analysis. Feature selection aims to reduce dimensionality while preserving predictive power. Let $\mathcal{S} \subseteq \{1, 2, \ldots, p\}$ denote the set of selected feature indices, and let $\mathrm{X}_{\mathcal{S}}$ represent the corresponding subset of features. The optimal subset is determined by solving the following optimization problem:

$$\mathcal{S}^* = \arg \max_{\mathcal{S} \subseteq \{1,\ldots,p\}} [\mathcal{I}(\mathcal{S}; y) - \lambda |\mathcal{S}|] \tag{5}$$

$\mathcal{I}(\mathcal{S}; y)$ represents the mutual information between the feature subset $\mathcal{S}$ and the target y, while λ is a regularization parameter that controls the size of the selected feature set, balancing between prediction accuracy and computational efficiency.

To identify interdependencies between features, the Pearson Correlation coefficient $\rho(f_i, f_j)$ as given in Eq. (6) is computed as:

$$\rho(f_i, f_j) = \frac{\mathrm{Cov}(f_i, f_j)}{\sigma_{f_i} \sigma_{f_j}}, \quad f_i, f_j \in \mathrm{F}_v(t) \tag{6}$$

This analysis aids in eliminating feature redundancy and mitigating multicollinearity, thereby enhancing model performance.

The Fog layer is also responsible for training base predictive models M_F for each vehicle using the Light Gradient Boosting Machine (LightGBM). The training objective is formulated in Eq. (7).

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \ell\big(y_i, \hat{y}_i; \theta\big) + \lambda |\theta\ |^2, \tag{7}$$

N is the number of training samples, $\ell(y_i, \hat{y}_i; \theta) = (y_i - \hat{y}_i)^2$ is the Mean Squared Error loss function, θ represents the model parameters, and λ is the regularization coefficient to prevent overfitting. The gradient boosting framework iteratively builds an ensemble of weak learners, $\{H_w\}_{w=1}^{W}$, specifically shallow decision trees, where each new tree attempts to correct the residual errors of the ensemble built thus far. The ensemble $\{H_w\}_{w=1}^{W}$ is trained to minimize a predefined loss function over the training data. The final prediction for the i^{th} instance is given by Eq. (8)

$$\hat{y}_i = \sum_{w=1}^{W} \alpha_w H_w(\mathrm{z}_i) \tag{8}$$

where $\hat{y}_i$ denotes the predicted value for the input feature vector z_i, and H_w represents the w^{th} weak learner, typically a shallow decision tree. Each weak learner H_w is trained to approximate the negative gradient (residual) of the loss function with respect to the current ensemble's output. The scalar α_w is the weight assigned to that learner's contribution.

Post-training, each base model $M_{F,v}$ is evaluated using performance metrics $\mathcal{M}(M_{F,v})$, such as Root Mean Squared Error (RMSE):

$$\mathcal{M}(M_{F,v}) = \mathrm{RMSE}(M_{F,v}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \big(y_{i,v} - \hat{y}_{i,v}\big)^2}, \tag{9}$$

The trained models and their corresponding metrics are encapsulated and transmitted to the Cloud layer as:

$$\mathrm{T}_F = \{(M_{F,v}, \mathcal{M}(M_{F,v}), v)\}_{v=1}^{V} \tag{10}$$

3.3 Cloud computing

The Cloud Computing layer functions as the centralized hub for advanced analytics, global optimization, and ensemble model construction. It orchestrates the integration and refinement of models received from the Fog layer, ensuring optimal predictive performance and scalability. The process begins with the aggregation of all base models from the Fog layer, denoted as $\mathrm{M}_F = \{M_{F,v}\}_{v=1}^{V}$.

To enhance predictive accuracy, hyperparameter optimization is performed using Bayesian Optimization, which fine-tunes the parameters θ of the ensemble model. The objective function G_{obj} is modeled as a Gaussian Process (GP), given by $G_{obj} \sim \mathcal{GP}\left(\mu(\theta), cov(\theta, \theta')\right)$, $\mu(\theta)$ is the mean function and $cov(\theta, \theta')$ is the covariance kernel. The optimization process is guided by an acquisition function $a(\theta) = \mu(\theta) + \kappa\sigma(\theta)$, $\sigma(\theta) = \sqrt{\mathrm{Var}(H(\theta))}$, and $\kappa > 0$ balances exploration and exploitation. At each iteration, the hyperparameters that maximize the acquisition function are selected, and the GP model is updated based on observed performance metrics.

Once optimized, the ensemble model M_C is constructed by integrating the base models through weighted averaging:

$$M_C(\mathrm{z}) = \sum_{v=1}^{V} \beta_v M_{F,v}(\mathrm{z}), \quad \beta_v = \frac{\mathcal{M}(M_{F,v})}{\sum_{u=1}^{V} \mathcal{M}(M_{F,u})} \tag{11}$$

β_v are normalized weights based on the individual model performance metrics $\mathcal{M}(M_{F,v})$.

Finally, the optimized ensemble model M_C is deployed back to the vehicles for real-time inference. This deployment is represented as $\mathcal{D} : M_C \rightarrow \{M_C^v\}_{v=1}^{V}$, M_C^v denotes the personalized model instance deployed to vehicle v.

The Cloud layer facilitates continuous learning by incorporating feedback from deployed models. This involves retraining and updating the ensemble M_C based on new data and performance metrics, ensuring adaptability to evolving vehicular conditions and enhancing predictive accuracy over time.

3.4 Inter-layer connectivity and system dynamics

The seamless operation of the Edge-Fog-Cloud architecture is governed by the intricate connectivity and dynamic interactions among the layers. The flow of data and models between these layers is critical for maintaining system efficiency and real-time performance.

Data flows from the Edge layer to the Fog layer through the transmission function $\mathcal{T}$, and from the Fog layer to the Cloud layer via the aggregated model set T_F. The Cloud layer processes and optimizes the ensemble model M_C, which is then deployed back to the Edge layer, completing the data loop. The end-to-end latency L of the *full feedback loop* (Edge→Fog→Cloud→Fog→Edge) comprises communication, processing, and queueing/handshake delays across all layers. We therefore refine the latency model as

$$L_{\mathrm{hier}} = L_{\mathrm{comm}} + L_{\mathrm{proc}} + L_{\mathrm{queue}}, \tag{12}$$

where L_{comm} captures payload transmission over network legs, L_{proc} aggregates edge/fog/cloud compute times, and L_{queue} captures broker/TLS/MQTT queueing and handshake overheads. The communication component decomposes into the four legs of the hierarchical loop:

$$L_{\mathrm{comm}} = \underbrace{\frac{|\mathrm{F}_v(t)|}{R_{\mathrm{EF}}}}_{L_{\text{Edge-Fog}}} + \underbrace{\frac{|\mathrm{T}_F|}{R_{\mathrm{FC}}}}_{L_{\text{Fog-Cloud}}} + \underbrace{\frac{|\mathrm{M}_C|}{R_{\mathrm{CF}}}}_{L_{\text{Cloud-Fog}}} + \underbrace{\frac{|\mathrm{D}_F(t)|}{R_{\mathrm{FE}}}}_{L_{\text{Fog-Edge}}}.$$

Processing and queueing terms are given by

$$L_{\mathrm{proc}} = P_E + P_F + P_C, \qquad L_{\mathrm{queue}} = Q_{\mathrm{EF}} + Q_{\mathrm{FC}} + Q_{\mathrm{CF}} + Q_{\mathrm{FE}}.$$

Here, $|\cdot|$ denotes payload size in bytes, $R_{\cdot}$ are *effective* data rates (payload bytes/s) measured under the TLS/MQTT stack, $P_{\{E,F,C\}}$ are per-layer processing times, and $Q_{\cdot}$ collect queueing/handshake overheads. For WAN-facing hops, a simple $M/M/1$ approximation yields $Q_i \approx \frac{\rho_i}{\mu_i - \lambda_i}$ with $\mu_i \propto \frac{R_i}{\bar{S}_i}$, so local aggregation that reduces arrival rate λ_i and average payload $\bar{S}_i$ lowers Q_i. In the following M/M/1 queueing approximation, we use the subscript 'i' to index the properties of an individual network hop.

To ensure real-time performance, the total latency L must satisfy:

$$L \leq \tau_{\max} \tag{13}$$

with $\tau_{\max}$ the maximum allowable end-to-end time for the *full loop*.

Comparison with a direct Edge↔Cloud loop. For reference, the direct two-leg loop (no fog) is

$$L_{\mathrm{dir}} = \frac{|\mathrm{F}_v(t)|}{R_{\mathrm{EC}}} + \frac{|\mathrm{M}_C|}{R_{\mathrm{CE}}} + P_E + P_C + Q_{\mathrm{EC}}, \tag{14}$$

whereas the hierarchical loop is $L_{\mathrm{hier}} = L_{\mathrm{comm}} + L_{\mathrm{proc}} + L_{\mathrm{queue}}$ as in Eq. (12). Their difference is

$$\begin{aligned}\Delta L \equiv L_{\text{hier}} - L_{\text{dir}} = |\mathrm{F}_v(t)| \left(\frac{1}{R_{\text{EF}}} - \frac{1}{R_{\text{EC}}}\right) \\ + \frac{|\mathrm{T}_F|}{R_{\text{FC}}} + \frac{|\mathrm{D}_F(t)|}{R_{\text{FE}}} \\ + |\mathrm{M}_C| \left(\frac{1}{R_{\text{CF}}} - \frac{1}{R_{\text{CE}}}\right) + P_F \\ + \big(Q_{\text{EF}} + Q_{\text{FC}} - Q_{\text{EC}}\big).\end{aligned} \tag{15}$$

If the local links dominate the WAN in throughput ($R_{\text{EF}} \geq R_{\text{EC}}$, $R_{\text{FE}} \geq R_{\text{CE}}$), the fog transmits models/summaries rather than raw features ($|\mathrm{T}_F| \ll |\mathrm{F}_v(t)|, |\mathrm{D}_F(t)| \ll |\mathrm{M}_C|$), and WAN queueing relief satisfies $Q_{\text{EF}} + Q_{\text{FC}} \leq Q_{\text{EC}} - \epsilon$ for some $\epsilon > P_F$, then $\Delta L < 0$ and hence $L_{\text{hier}} < L_{\text{dir}}$. This matches our measurement protocol for Fig. 10, which reports median full-loop latencies. Resource allocation across the Edge, Fog, and Cloud layers is posed as a constrained optimization to *maximize* system throughput while respecting the latency budget with L defined in Eq. (12):

$$\max_{\mathcal{R}} \Theta_t = \sum_{v=1}^{V} \omega_v \quad \text{subject to} \quad L(\mathcal{R}) \leq \tau_{\max}, \mathcal{R} \in \mathcal{C}.$$

ω_v is the throughput per vehicle and $\mathcal{R}$ denotes the resource allocation strategy within the feasible set $\mathcal{C}$. The predictive accuracy $\mathcal{A}(M)$ of a model M remains a function of its hyperparameters θ and the quality of input features F_v,

$$\mathcal{A}(M; \theta, \mathrm{F}_v) = \mathbb{E}_{\mathrm{F}_v, y} \left[\mathcal{L}(y, M(\mathrm{F}_v; \theta))\right], \tag{16}$$

minimize the prediction error (or equivalently, maximize the prediction accuracy) improves predictive performance under the same latency budget.

3.5 Security and privacy enhancements

Given the sensitivity of vehicular data, the architecture integrates robust security and privacy mechanisms across all layers. All data transmissions between layers are secured using Advanced Encryption Standard (AES) given by Eq. (17)

$$\text{Ciphertext} = \text{AES}_{\text{key}}(\text{Plaintext}) \tag{17}$$

ensuring data confidentiality and integrity. Mutual authentication protocols (e.g., TLS/SSL) are employed to verify the identities of communicating entities, preventing unauthorized access and ensuring secure interactions. Before transmission, sensitive data can be anonymized using techniques such as Differential Privacy given by Eq. (18) mathematically.

$$\mathcal{M}(M_{F,v}) + \mathcal{N}(0, \sigma^2) \tag{18}$$

$\mathcal{N}(0, \sigma^2)$ is Gaussian noise added to protect individual privacy while preserving data utility.

This Edge-Fog-Cloud architecture for vehicle health prediction is underpinned by mathematical frameworks.The seamless integration and dynamic interactions among the Edge, Fog, and Cloud layers, coupled with continuous learning mechanisms, enhance the system's adaptability and long-term reliability.

4 Proposed prediction framework

Building upon the hierarchical Edge-Fog-Cloud architecture delineated in Sect. 3, we introduce our proposed prediction framework designed for monitoring vehicle health and predicting engine failures.

4.1 System overview

The Prediction Framework operates within the established Edge-Fog-Cloud paradigm, leveraging data flows and computational resources across all three layers to achieve robust and scalable vehicle health monitoring. The prediction framework comprises three core modules: The Data Engineering Module, which is responsible for data collection, preprocessing, and feature selection; the Prediction Module, which utilizes machine learning models to predict vehicle health status; and the Hyperparameter Optimization Module, which employs Bayesian Optimization to fine-tune model hyperparameters for optimal performance.

4.2 Data engineering module

The Data Engineering Module ensures the quality and relevance of the data utilized for model training and prediction. It comprises three sub-components: Dataset Collection, Data Preprocessing, and Latent Feature Selection.

4.2.1 Dataset collection

The dataset used consists of 10,000 samples representing engine-related parameters. The dataset [30] used in this study is a toy dataset inspired by the machine failure dataset, adapted to simulate vehicle-like engine behavior for preliminary analysis and commonly used in predictive maintenance research. Although originally designed for industrial machinery, it is conceptually similar to vehicle engine systems, as it includes parameters such as torque, rotational speed, air temperature (short for Engine Intake Air Temperature), and process temperature – all of which are also relevant indicators of vehicle engine health. While this dataset does not originate from real vehicular environments, it serves as a suitable proxy for simulating and validating the proposed framework in a controlled setting. This allows us to demonstrate the feasibility and effectiveness of the FogBayes architecture for engine health prediction,

with the understanding that future work will incorporate real-world vehicular data for deployment and validation. Data is assumed to be aggregated from the On-Board Unit (OBU) of each vehicle, encompassing sensor readings such as air temperature, process temperature, rotational speed, torque, and tool wear. Let D denote the dataset collected from V vehicles over T time instances:

$$D = \{(d_v(t), f_v(t), v) \mid v \in \{1, 2, \ldots, V\},\ t \in \{1, 2, \ldots, T\}\} \tag{19}$$

$d_v(t) \in \mathbb{R}^d$ represents the diagnostic data for vehicle v at time t, $f_v(t) \in \mathbb{R}^f$ denotes the feature set extracted at time t, and v is the unique vehicle identifier.

The dataset used in this study includes multivariate numerical data with continuous values, presenting detailed statistics for key parameters such as air temperature, process temperature, rotational speed, torque, tool wear, and machine failure. Table 2 provides a comprehensive overview of the dataset's descriptive statistics, including the mean, minimum, and maximum values for each parameter.

The statistics presented in Table 2 offer insights into the data distribution, highlighting the range and variability of each parameter. For instance, the rotational speed ranges from 1168 RPM to 2886 RPM, with a mean value of 1538.78 RPM, while the torque varies between 3.80 Nm and 76.60 Nm. These values form the basis for feature engineering and model training in subsequent stages of the proposed framework.

4.2.2 Data preprocessing

Preprocessing enhances data quality through noise reduction, normalization, and correlation analysis. The preprocessing steps are formalized as follows:

$$\hat{\mathrm{X}}_v(t) = \frac{\mathcal{F}(\mathrm{X}_v(t)) - \boldsymbol{\mu}_v}{\boldsymbol{\sigma}_v} \tag{20}$$

$\mathcal{F}$ represents the Finite Impulse Response (FIR) filter applied to mitigate high-frequency noise, while $\boldsymbol{\mu}_v$ and $\boldsymbol{\sigma}_v$ denote the mean and standard deviation vectors, respectively, computed over the training dataset for normalization.

The Pearson Correlation coefficient $\rho(f_i, f_j)$ is utilized to assess feature interdependencies:

$$\rho(f_i, f_j) = \frac{\mathrm{Cov}(f_i, f_j)}{\sigma_{f_i}\sigma_{f_j}}, \quad \forall f_i, f_j \in \mathrm{F}_v(t) \tag{21}$$

Table 2 Descriptive data statistics of the dataset

Parameter	Air Temperature (AT) [K]	Process Temperature (PT) [K]	Rotational Speed (RS) [rpm]	Torque (TQ) [Nm]	Tool Wear (TW) [min]	Machine Failure (MF)
Mean	300.00	310.01	1538.78	39.99	107.95	0.0339
Min.	295.30	305.70	1168.00	3.80	0.00	0.00
Max.	304.50	313.80	2886.00	76.60	253.00	1.00

This analysis identifies relationships among features, facilitating the elimination of redundant features to reduce multicollinearity and enhance model interpretability. The results of the correlation analysis are visualized in Fig. 3, where a heatmap is used to determine which pairs of variables are most closely related.

The heatmap in Fig. 3 provides a visual representation of feature interdependencies, helping to pinpoint redundant features that can be removed in subsequent analysis. This step is crucial in ensuring that only the most informative features are retained for model training, leading to improved accuracy and reduced computational complexity.

4.2.3 Latent feature selection

Latent feature selection is a critical step in optimizing the predictive performance of the model while minimizing computational complexity. The feature selection process identifies the most informative features by maximizing their mutual information with the target variable y, while imposing a sparsity constraint. This process is formulated as Eq. (5)

Using this method, the following key features were identified as the most significant predictors of vehicle health: PT, AT, RS, TQ, TW. These features were selected due to their strong correlations with the target variable and their ability to explain the variance in the dataset. The feature importance is visualized in Fig. 4, which presents a partial dependence plot showcasing the relative importance of each selected feature.

As shown in Fig. 4, TQ emerged as the most critical feature, RS and PT. These features exhibited the highest predictive power for vehicle health. Specifically, Torque (TQ) demonstrated the highest importance among the selected features, emphasizing its pivotal role in assessing engine load and mechanical wear. Rotational Speed (RS) also emerged as a key indicator, closely correlating with mechanical stress and the likelihood of potential faults. Parameters such as Process Temperature (PT) and Air Temperature (AT) (short for Engine Intake Air Temperature)highlighted the influence of thermal dynamics on engine performance and overall longevity. Additionally, Tool



Fig. 3 Correlation heatmap: identifying feature relationships

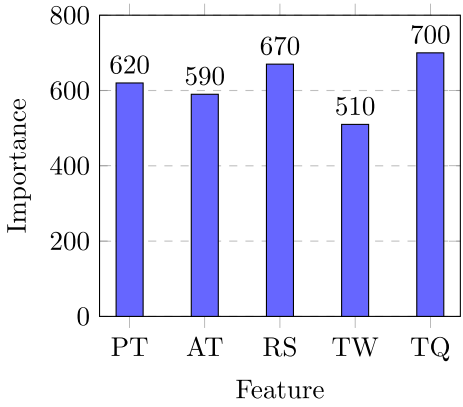


Fig. 4 Feature importance for selected parameters

Wear (TW) captured the cumulative degradation over time, offering valuable insights into the vehicle's maintenance needs.

This analysis ensures that only the most relevant features are retained for model training, reducing dimensionality and enhancing interpretability while preserving the model's predictive power. The selected features form the basis for further preprocessing and predictive modeling.

4.3 Prediction module

The Prediction Module employs the Light Gradient Boosting Machine (LightGBM) to forecast the vehicle health status V_{hs}. The predictive model is defined as:

$$V_{hs} = \begin{cases} 1, \text{if the vehicle is unfit} \\ 0, \text{if the vehicle is fit} \end{cases} \tag{22}$$

4.3.1 Model architecture

The predictive function $\hat{P}$ estimates the probability of a vehicle being healthy based on diagnostic data $d_v(t)$, feature set f, and vehicle identifier v:

$$\hat{P}(\text{healthy vehicle}) = c(q(d_v(t), f), e(v)) \tag{23}$$

- $q : \mathbb{R}^{d\times(2L_{\text{lag}}+2)} \rightarrow \mathbb{R}^m$ is the diagnostic encoding function that transforms diagnostic data and features into an m-dimensional latent space.where L_{lag} represents the number of temporal lags used to capture historical context in the diagnostic data.
- $e(v) \in \mathbb{R}^d$ is a learned embedding representing vehicle v. The embedding is learned directly from the vehicle identifier during training. This allows the model to capture vehicle-specific signal patterns from the diagnostic features–air temperature (AT), process temperature (PT), rotational speed (RS), torque (TQ), and

tool wear (TW)–even in the absence of additional contextual information. In future work, this embedding could be extended to incorporate richer vehicle metadata such as model type, manufacturing year, or usage patterns to further improve predictive performance and personalization.

- $c : \mathbb{R}^{m+d} \to \mathbb{R}$ is the classification head that outputs the probability of the vehicle being healthy.

4.3.2 Training procedure

The training of the predictive model is carried out through a series of well-defined steps that leverage advanced optimization techniques to enhance the efficiency and accuracy of the model. The complete training process is illustrated in Fig. 5, providing a comprehensive overview of the methodology.

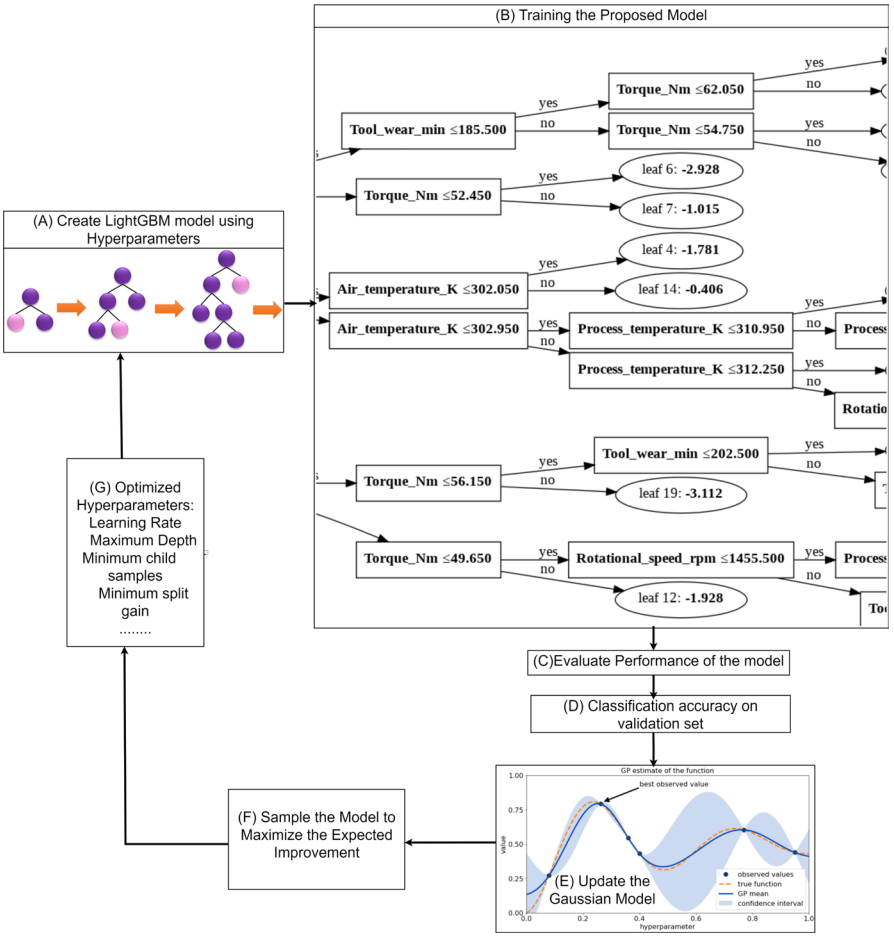


Fig. 5 Workflow of the predictive model training process. Representative portion of a gradient boosting decision tree used in our model. Due to space constraints, the full tree is not shown; this diagram illustrates the typical structure, feature usage, and depth patterns learned during training

1. *Exclusive Feature Bundling (EFB):* To mitigate the challenges posed by high-dimensional datasets, mutually exclusive features are grouped into bundles. This reduces the dimensionality of the feature space, thereby optimizing computational efficiency without compromising information retention. This step ensures the model remains scalable for real-world IoV scenarios.
2. *Gradient-based One Side Sampling (GOSS):* GOSS prioritizes instances with larger gradients during training, as these carry more information regarding the optimization direction. The GOSS algorithm is defined mathematically in Eq. (24)

$$\tilde{V}_j(\boldsymbol{\delta}) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(\boldsymbol{\delta})} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(\boldsymbol{\delta})}\right) \quad (24)$$

- g_i denotes the gradient of the loss function with respect to the prediction of the *i*-th data sample.
- A_l and A_r represent the subsets of data samples (with large gradients) that fall into the left and right child nodes, respectively, after a split.
- B_l and B_r represent the subsets of data samples (with small gradients) that fall into the left and right child nodes, respectively, after a split.
- *a* is the proportion of large-gradient data samples retained.
- *b* is the proportion of small-gradient data samples sampled.
- *n* is the total number of data samples considered at the node.
- $n_l^j(\boldsymbol{\delta})$ and $n_r^j(\boldsymbol{\delta})$ represent the weighted sample counts at the left and right child nodes, respectively, for split dimension *j* and threshold $\boldsymbol{\delta}$.

3. *Model Training:* The model iteratively builds an ensemble of decision trees to minimize the objective loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \mid\theta\mid^2 \quad (25)$$

y_i represents the true vehicle health status, while $\hat{y}_i$ denotes the predicted value. The model parameters are represented by θ, and λ is the regularization coefficient. Each iteration updates the model parameters as:

$$\theta_m(\chi) = \theta_{m-1}(\chi) + \theta'_m(\chi) \quad (26)$$

$\theta'_m(\chi)$ represents the newly added decision tree at iteration m.

4. *Model Evaluation:* The trained models $M_{F,v}$ are evaluated using metrics such as Root Mean Squared Error (RMSE) and R-squared (R^2):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{27}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{28}$$

$\bar{y}$ is the mean of the true values y_i.

5. *Model Transmission:* The trained models and their performance metrics are encapsulated into a structured data format and transmitted to the cloud layer for global optimization and deployment:

$$\mathrm{T}_F = \{(M_{F,v}, \mathcal{M}(M_{F,v}), v)\}_{v=1}^{V} \tag{29}$$

Fig. 5 visually captures the training process, including model creation, hyperparameter tuning, decision tree construction, performance evaluation, and sampling to maximize expected improvement. This systematic workflow ensures the development of robust predictive models tailored for the IoV scenario, achieving a balance between computational efficiency and predictive accuracy.

4.4 Hyperparameter optimization module

Optimizing hyperparameters is critical for enhancing the predictive performance of machine learning models. In this work, Bayesian Optimization (BO) is employed to efficiently navigate the hyperparameter space Θ_{HS}, minimizing the objective function $H(\theta)$:

$$H(\theta) = \mathcal{A}(M; \theta, \mathrm{F}_v) = \mathbb{E}_{\mathrm{F}_v, y}\left[\mathcal{L}(y, M(\mathrm{F}_v; \theta))\right] \tag{30}$$

$\mathcal{A}(M; \theta, \mathrm{F}_v)$ denotes the expected predictive accuracy of model M given hyperparameters θ and features F_v. The hyperparameters optimized in this work are detailed in Table 3, which includes key parameters such as the learning rate, number of leaves,

Table 3 FogBayes hyperparameter configuration

Parameter	Value	Parameter	Value
Type of boosting	GBDT	Number of leaves	96
Weight of class	None	Random state	969
Colsample for tree	1.0	L1 regularization gain α	6.633×10^{-8}
Importance type	Split	L2 regularization gain λ	6.165×10^{-10}
Learning rate	0.0834	Silent	Warn
Maximum depth	-1	Subsample	1.0
Minimum child samples	10	Subsample for bin	200,000
Minimum child weight	0.001	Subsample frequency	0
Minimum split gain γ	0.4395	Feature fraction	0.9618
Number of estimators	83	Bagging fraction	0.7716

regularization coefficients, and feature fractions. The parameter random state in Table 3 fixes the random seed of the pseudo-random number generator to ensure reproducibility of experiments. In stochastic algorithms such as Gradient Boosting Decision Trees, randomness appears in processes like data sampling and feature selection. By setting the random state to a fixed value (969 in our experiments), we guarantee that results remain consistent across multiple runs, thereby enabling reproducibility and fair comparison of outcomes.

Note: The value of `max_depth = -1` disables the depth constraint, allowing tree depth to be determined dynamically based on other constraints such as `num_leaves` and `min_child_samples`. The setting `subsample_freq = 0` disables subsampling per boosting iteration, using the full dataset in each round. This is suitable for moderately sized datasets when regularization parameters (λ, α) are used to prevent overfitting.

4.4.1 Bayesian optimization process

The Bayesian Optimization process is formalized as follows:

$$p(\Upsilon \mid A_{1:t}) = \frac{p(A_{1:t} \mid \Upsilon)\, p(\Upsilon)}{p(A_{1:t})}, \tag{31}$$

$$y_t = \Upsilon(x_t) + \epsilon_t, \tag{32}$$

$$A_{1:t} = \{(x_\tau, y_\tau)\}_{\tau=1}^{t}, \tag{33}$$

Υ represents the objective function that reflects model performance, y_t denotes the observed performance at iteration t, and x_t is the hyperparameter configuration at iteration t. The term ε_t represents the noise in the observation, and $A_{1:t}$ is the set of all observations up to iteration t. The Bayesian Optimization algorithm iteratively updates a Gaussian Process (GP) surrogate model to approximate the objective function and employs an acquisition function $\mathrm{a}(\theta)$ to balance exploration and exploitation:

$$\mathrm{a}(\theta) = \mu(\theta) + \kappa\sigma(\theta) \tag{34}$$

$\mu(\theta)$ represents the mean predicted by the Gaussian Process (GP) at hyperparameter θ, $\sigma(\theta)$ denotes the standard deviation predicted by the GP at θ, and $\kappa > 0$ is the trade-off parameter that balances exploration and exploitation. The steps of Bayesian Optimization are summarized in Algorithm 1.

Input: Hyperparameter space Θ, Target score function $H(\theta)$, Max number of evaluations $n_{\max}$
Select an initial configuration $\boldsymbol{\phi}_0 \in \Theta$
Evaluate the initial score $y_0 = H(\boldsymbol{\phi}_0)$
Set $\boldsymbol{\phi}^* = \boldsymbol{\phi}_0$, $y^* = y_0$, and $S_0 = \{\boldsymbol{\phi}_0, y_0\}$
for $n = 1$ to $n_{\max}$ **do**
    Select a new hyperparameter configuration $\boldsymbol{\phi}_n$ by maximizing the acquisition function:

$$\boldsymbol{\phi}_n = \arg\max_{\boldsymbol{\phi} \in \Theta} \mathbf{a}(\theta)$$

    Evaluate H at $\boldsymbol{\phi}_n$ to obtain $y_n = H(\boldsymbol{\phi}_n)$
    Augment the dataset: $S_n = S_{n-1} \cup \{\boldsymbol{\phi}_n, y_n\}$
    Update the Gaussian Process surrogate model with S_n
    if $y_n < y^*$ **then**
        $\boldsymbol{\phi}^* = \boldsymbol{\phi}_n$ and $y^* = y_n$
    end if
end for
Output: $\boldsymbol{\phi}^*$ and y^*

Algorithm 1 Bayesian optimization algorithm

4.4.2 Ensemble construction

Upon optimizing hyperparameters, the Cloud layer aggregates the base models into an ensemble M_C through weighted averaging:

$$M_C(\mathrm{z}) = \sum_{v=1}^{V} \beta_v M_{F,v}(\mathrm{z}), \quad \beta_v = \frac{\mathcal{M}(M_{F,v})}{\sum_{u=1}^{V} \mathcal{M}(M_{F,u})} \tag{35}$$

β_v represents the normalized weight based on the performance metric $\mathcal{M}(M_{F,v})$ of each base model $M_{F,v}$, and $\mathcal{M}(M_{F,v})$ denotes metrics such as RMSE or R^2 used to evaluate $M_{F,v}$. The ensemble model M_C leverages the strengths of individual base models while reducing their weaknesses, ensuring robust and accurate predictions for the IoV environment.

4.5 Integration and deployment

The integrated Prediction Framework operates in tandem with the Edge-Fog-Cloud architecture to deliver real-time vehicle health predictions. The workflow is as follows:

The data processing workflow involves several key stages across different layers. Initially, vehicles in the Edge layer collect and preprocess sensor data, extracting key features $\mathrm{F}_v(t)$. This data is then aggregated by Roadside Units (RSUs) in the Fog layer, where advanced feature transformations are performed, and base predictive models M_F are trained using LightGBM. The trained models T_F are sent to the Cloud layer, where Bayesian Optimization is employed to fine-tune hyperparameters and construct an optimized ensemble model M_C. This ensemble model is then

deployed back to the vehicles in the Edge layer for real-time health status predictions V_{hs}. Finally, the Cloud layer facilitates continuous learning and feedback, utilizing insights from deployed models to retrain and update the ensemble, ensuring its adaptability to evolving vehicular conditions.

5 Experimental analysis

This section delineates the experimental testbed employed to evaluate the proposed FogBayes. The testbed is meticulously designed to emulate a real-world vehicular network, encompassing three hierarchical layers: Edge, Fog, and Cloud. Each layer is equipped with specific hardware and software configurations to facilitate seamless data processing, communication, and model deployment. Figure 6 provides a schematic representation of the experimental setup. "The one-to-one correspondence is only illustrative. In practice, a fog node typically supports multiple edge nodes."

5.1 Environment setup

The experimental setup comprises three interconnected layers, each serving distinct roles in the data acquisition, processing, and analysis pipeline. The following subsections provide an in-depth description of each layer's hardware and software components.

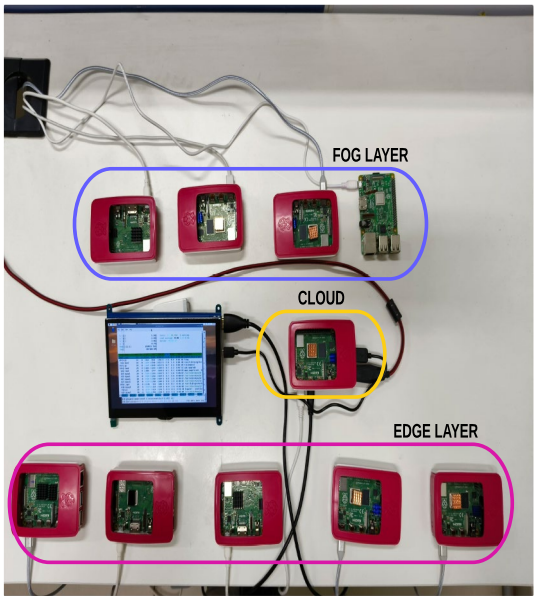


Fig. 6 Overview of the experimental testbed

5.1.1 Edge layer

The Edge layer is responsible for data acquisition and preliminary processing and comprises five Raspberry Pi 3 Model B units. Each unit is equipped with a 1.2 GHz Broadcom BCM2837 quad-core ARM Cortex-A53 processor and 1 GB of LPDDR2 RAM to support basic computational tasks. For local data storage, they utilize 32 GB microSD cards, while the operating system is Raspberry Pi OS (formerly Raspbian), optimized for lightweight performance to ensure efficient operation. To ensure efficient data management and rapid retrieval, each Edge device deploys a Redis distributed database. Redis facilitates low-latency data access and supports various data structures essential for real-time applications. Communication between the Edge layer and the Fog layer is managed using Node-RED, a flow-based development tool tailored for IoT applications. Node-RED enables the creation of custom data pipelines and supports multiple communication protocols, including MQTT, HTTP, and WebSockets, ensuring interoperability with diverse vehicular systems.

5.1.2 Fog layer

The Fog layer serves as an intermediary processing node, aggregating and preprocessing data received from the Edge layer before transmitting it to the Cloud layer. The Fog layer consists of five Raspberry Pi 4 Model B units, each equipped with a 1.5 GHz quad-core ARM Cortex-A72 processor and 4 GB of LPDDR4 RAM. These units feature dual-band wireless capabilities, supporting both 2.4 GHz and 5 GHz frequencies to enhance connectivity. For storage, each Raspberry Pi is connected to a 64 GB SSD via NAS, enabling faster wireless data access and efficient storage management. Software components within the Fog layer include Docker Swarm for container orchestration, enabling the deployment, scaling, and management of containerized applications. Docker Swarm ensures high availability and fault tolerance of Fog services by distributing containers across multiple nodes. Secure communication between the Edge and Fog layers is established using Transport Layer Security (TLS) protocols, providing end-to-end encryption and data integrity. Additionally, the Fog layer incorporates a dedicated Network Address Translation (NAT) gateway, which facilitates secure and controlled access to the Cloud layer while masking the internal network structure from external entities. This setup enhances security by preventing unauthorized access and mitigating potential cyber threats.

5.1.3 Cloud layer

The Cloud layer is emulated on a high-performance laptop featuring an Intel Core i7 13th-generation processor with 8 cores and 16 threads, operating at a base frequency of 2.2 GHz. It is equipped with an NVIDIA RTX 4060 GPU to support accelerated parallel processing tasks. The laptop also includes 16 GB of DDR4 RAM for efficient multitasking and a 1 TB NVMe SSD, providing high-speed data storage and rapid retrieval capabilities. Containerization within the Cloud layer is managed using Docker, facilitating the deployment of scalable and consistent machine learning services. The integration of Docker allows for the encapsulation of software depen-

dencies, ensuring uniform application performance across different environments. The computational capabilities of the Cloud layer, augmented by the GPU, enable efficient training and optimization of complex machine learning models, including the proposed FogBayes.

5.2 Scalability and performance considerations

The experimental testbed is engineered to evaluate the scalability and performance of the FogBayes architecture under varying operational loads. Key considerations include:

- *Scalability:* The use of Docker Swarm in the Fog layer allows for dynamic scaling of containerized applications in response to fluctuating data volumes. This ensures that the system can handle increasing numbers of vehicles without degradation in performance. The scalability is mathematically modeled by analyzing the system's ability to maintain linear or sub-linear growth in resource utilization as the number of connected vehicles V increases:

$$\lim_{V \to \infty} \frac{U}{V} \leq C \tag{36}$$

- U represents resource utilization (CPU, memory) and C is a constant.
- *Latency Minimization:* By processing data locally at the Fog layer, the system reduces the latency associated with transmitting large volumes of raw data to the Cloud. This is critical for real-time vehicle health monitoring applications where timely predictions are essential. Latency L defined by Eq. (12) constrained by Eq. (13) is modeled as the sum of communication delays between layers.
- *Resource Optimization:* The hierarchical distribution of computational tasks across Edge, Fog, and Cloud layers ensures optimal utilization of available resources. This balance prevents bottlenecks and enhances overall system efficiency. Resource optimization is quantitatively assessed by monitoring CPU and memory usage across all layers under different load conditions.
- *Fault Tolerance and Reliability:* The redundancy provided by multiple Edge and Fog nodes ensures that the system remains operational even if individual nodes fail. Fault tolerance is mathematically represented by the system's reliability function $R(t)$ as shown in Eq. (37), which denotes the probability that the system performs without failure up to time t:

$$R(t) = \prod_{i=1}^{n} \left(1 - F_i(t)\right),^{1} \tag{37}$$

- [1] $F_i(t)$ is the failure probability of the i-th node.

[1] Here,$i = 1 \ldots n$ refers to the indexing of nodes for the system's reliability function $R(t)$.

This experimental testbed, encompassing the Edge, Fog, and Cloud layers, provides a robust platform for evaluating the proposed FogBayes architecture. The integration of advanced hardware specifications, containerization technologies, efficient communication protocols, and stringent security measures ensures that the testbed accurately emulates a real-world vehicular network environment. This setup facilitates comprehensive assessment of the FogBayes's performance, scalability, and efficiency, laying the groundwork for subsequent experimental analyses.

6 Results and discussion

In this section, we present a comprehensive analysis of the experimental results obtained from evaluating the proposed FogBayes.

6.1 Comparative performance analysis

Table 4 presents a comprehensive comparative analysis of proposed FogBayes against various established machine learning classifiers, shown in bold in the Table 4. The models evaluated include Light Gradient Boosting Machine (Light-GBM), CatBoost, Random Forest (RF), Extreme Gradient Boosting (XGBoost), Decision Tree (DT), AdaBoost (ADA), Logistic Regression (LR), K-Nearest Neighbors (KNN), Naive Bayes (NB), and Support Vector Machine (SVM). The evaluation metrics encom-

Table 4 Performance comparison between the proposed FogBayes model and other established machine learning classifiers. The best-performing results, corresponding to FogBayes, are shown in bold

Model	Accuracy	AUC	Recall	Precision	F1	Kappa	MCC	TT (Sec)
Proposed FogBayes	**0.9888**	**0.9830**	**0.8148**	**0.8462**	**0.8302**	**0.8244**	**0.8245**	**0.129**
Light Gradient Boosting Machine (Light-GBM) [31]	0.9824	0.9692	0.6170	0.8360	0.7076	0.6988	0.7086	0.135
CatBoost Classifier (Catboost) [32]	0.9833	0.9698	0.6213	0.8591	0.7190	0.7107	0.7217	2.670
Random Forest (RF) Classifier [33]	0.9817	0.9588	0.5762	0.8568	0.6846	0.6756	0.6919	0.745
Extreme Gradient Boosting (XGBoost) [31]	0.9813	0.9640	0.6215	0.7959	0.6972	0.6877	0.6937	0.401
Decision Tree Classifier (DT) [34]	0.9721	0.8052	0.6258	0.6046	0.6109	0.5966	0.5988	0.023
AdaBoost Classifier (ADA) [35]	0.9701	0.9380	0.4115	0.6080	0.4895	0.4747	0.4850	0.165
Logistic Regression (LR) [36]	0.9696	0.8845	0.1977	0.7575	0.3077	0.2976	0.3716	0.337
K-Nearest Neighbors Classifier (KNN) [37]	0.9681	0.7474	0.2097	0.6267	0.3125	0.3004	0.3492	0.126
Naive Bayes (NB) [38]	0.9589	0.8982	0.1895	0.3322	0.2397	0.2204	0.2303	0.013
Support Vector Machine (SVM) [39]	0.8671	0.6921	0.2532	0.2475	0.2503	0.2238	0.2246	0.032

pass Accuracy, Area Under the Curve (AUC), Recall, Precision, F1 Score, Kappa, Matthews Correlation Coefficient (MCC), and Training Time (TT). As illustrated in Table 4, the proposed FogBayes significantly outperforms all benchmark models across multiple performance metrics. Specifically, FogBayes achieves an Accuracy of 98.88% and an AUC of 0.9830, surpassing the next best model, CatBoost, which attains an Accuracy of 98.33% and an AUC of 0.9698 [32]. The Recall, Precision, and F1 Score of FogBayes are 81.48%, 84.62%, and 83.02%, respectively, indicating its robust ability to correctly identify positive instances while maintaining high precision and balance between precision and recall.

The Kappa statistic and Matthews Correlation Coefficient (MCC) for FogBayes are 0.8244 and 0.8245, respectively, reflecting strong agreement beyond chance and reliable predictive performance. Furthermore, FogBayes achieves these outstanding results with a minimal Training Time (TT) of 0.129 s, demonstrating its computational efficiency compared to models like CatBoost, which, while accurate, require significantly longer training times of 2.670 s [32].

6.2 ROC curve analysis

Figure 7 illustrates the Receiver Operating Characteristic (ROC) curves for FogBayes and selected benchmark models. The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings, providing a visual assessment of each model's classification performance.

As depicted in Fig. 7, FogBayes exhibits the highest AUC closely followed by Light-GBM whereas traditional classifiers like SVM and Naive Bayes lag significantly. The steep ascent of the ROC curve for FogBayes indicates excellent sensitivity and specificity, affirming its capability to accurately distinguish between healthy and faulty vehicle states.

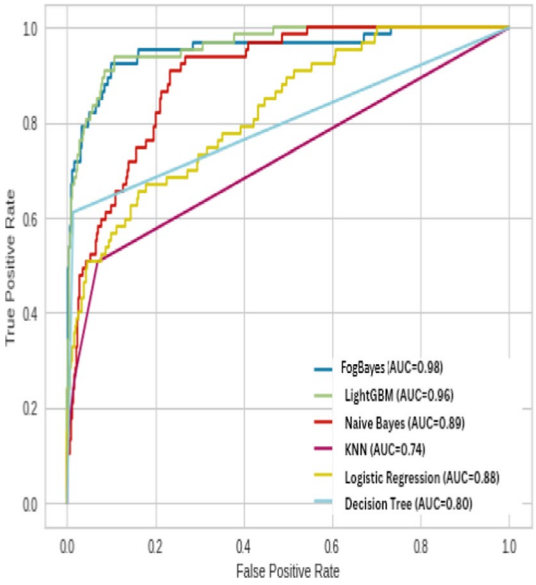


Fig. 7 Comparative analysis of ROC curves

6.3 Distributed framework performance

Beyond model-centric metrics, the distributed Edge-Fog-Cloud framework's performance is evaluated in terms of efficiency, energy consumption, network throughput, and latency.

6.3.1 Processing efficiency across layers

Figure 8 summarizes per–layer **Processing efficiency,** $\eta_\ell = \frac{\text{successful tasks}}{\text{total tasks}}$ for $\ell \in \{\text{Edge}, \text{Fog}, \text{Cloud}\}$ using violin plots, while Fig. 9 reports per–iteration *energy consumption* $E_\ell(k)$ in joules. Visually, the Processing efficiencyefficiency distributions are centered highest at the Edge (median $\approx 60\%$), then Fog ($\approx 50\%$), then Cloud ($\approx 40\%$). To relate Processing efficiency to energy, we convert the raw energy traces to a robust normalized utilization curve per layer.

$$U_\ell(k) = \frac{E_\ell(k)}{E_{\ell,95}} \times 100\%, \qquad E_{\ell,95} \triangleq 95\text{th percentile of } \{E_\ell(k)\}_k, \tag{38}$$

and we report the interquartile range (IQR) of U_ℓ across iterations as the "percentage bands" in the text. Using Fig. 9, the IQR bands are Edge ≈ 40–80%, Fog ≈ 30–65%, and Cloud ≈ 10–75%. Correlating Figs. 8 and 9, the Edge exhibits the highest Processing efficiency alongside the lowest normalized energy utilization, the Fog sits in the middle on both axes, and the Cloud shows the lowest Processing efficiency effi-

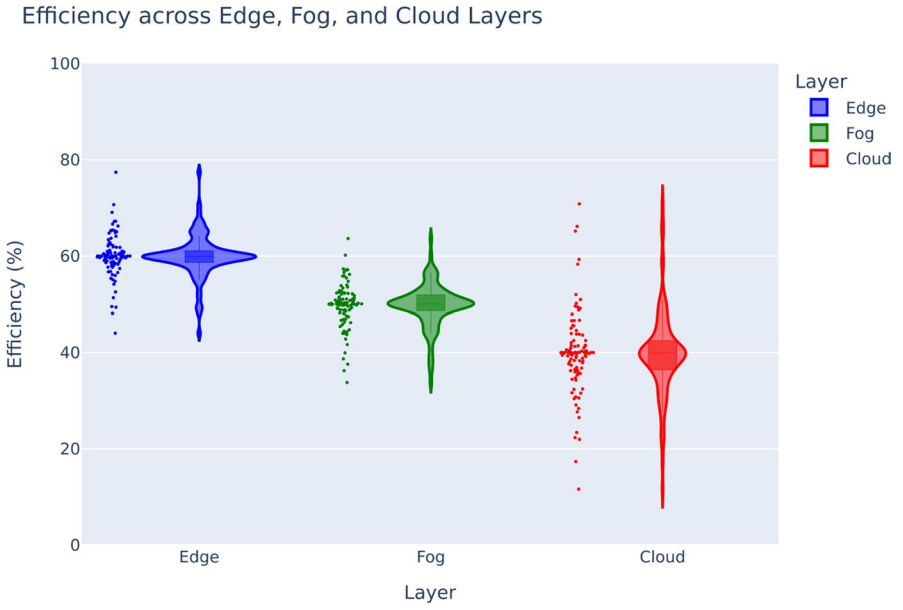


Fig. 8 Violin plots of inference-task processing efficiency at Edge (40–80 %), Fog (30–65 %), and Cloud (10–75 %) layers, illustrating how compute-resource variability impacts throughput

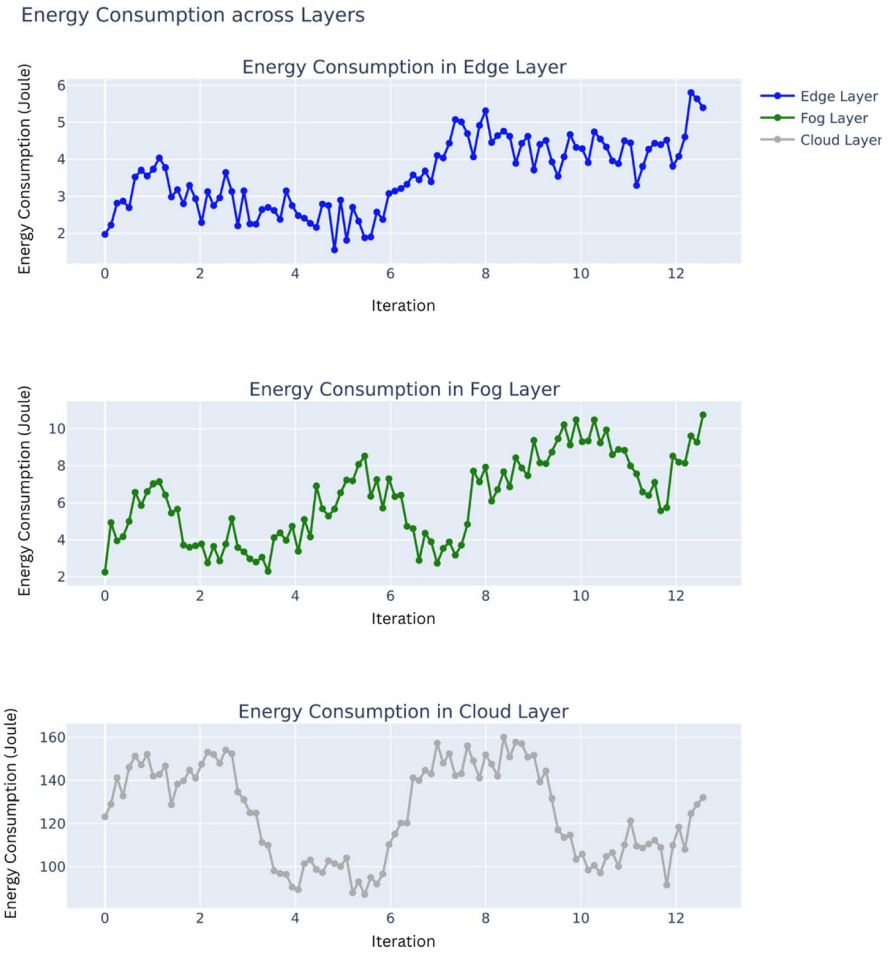


Fig. 9 Normalized energy usage during inference and model updates at Edge (∼2–6 J), Fog (∼3–11 J), and Cloud layers (∼90–160 J), demonstrating scalable energy management across the hierarchy. The X-axis represents the progression of end-to-end experimental cycles (Iterations). The integer values mark the completion of a cycle, while the data points show the metric sampled at high frequency throughout the duration of each cycle

ciency with the highest (and most variable) energy–consistent with locality (Edge), aggregation (Fog), and orchestration/queuing overheads (Cloud).

6.3.2 Energy consumption

Figure 9 shows that the absolute per–iteration energy scales with the role of each layer: Edge ∼2–6 J, Fog ∼3–11 J, and Cloud ∼90–160 J. Because layers have different power envelopes, we interpret energy comparatively via the normalized utilization $U_\ell(k) = E_\ell(k)/E_{\ell,95}$ (percent of a robust per–layer budget). The resulting IQR bands are Edge ≈40–80%, Fog ≈30–65%, and Cloud ≈10–75%. These bands, when

read alongside the efficiency violins in Fig. 8, reveal a consistent layerwise trend: the Edge attains higher computational efficiency at lower normalized energy (local inference on small payloads), the Fog achieves moderate efficiency with moderate energy (aggregation and model selection), and the Cloud exhibits lower efficiency with higher energy and variance (centralized orchestration, batching, and WAN–induced waiting). In short, the hierarchical design shifts work to the energetically favorable Edge/Fog tiers, reducing Cloud energy intensity while preserving accuracy, and the efficiency–energy relationship observed across layers matches this allocation strategy.

6.3.3 Network throughput and latency

Figures 10 and 11 illustrate the network throughput and latency across different communication topologies within the Edge-Fog-Cloud architecture.

The Edge-Fog-Cloud topology achieves an average latency of 150 ms, significantly lower than the direct Edge-Cloud transmission latency of 300 ms. This reduction is attributed to localized data processing at the Fog layer, which minimizes the volume of data transmitted to the Cloud. Network throughput is enhanced in the dis-

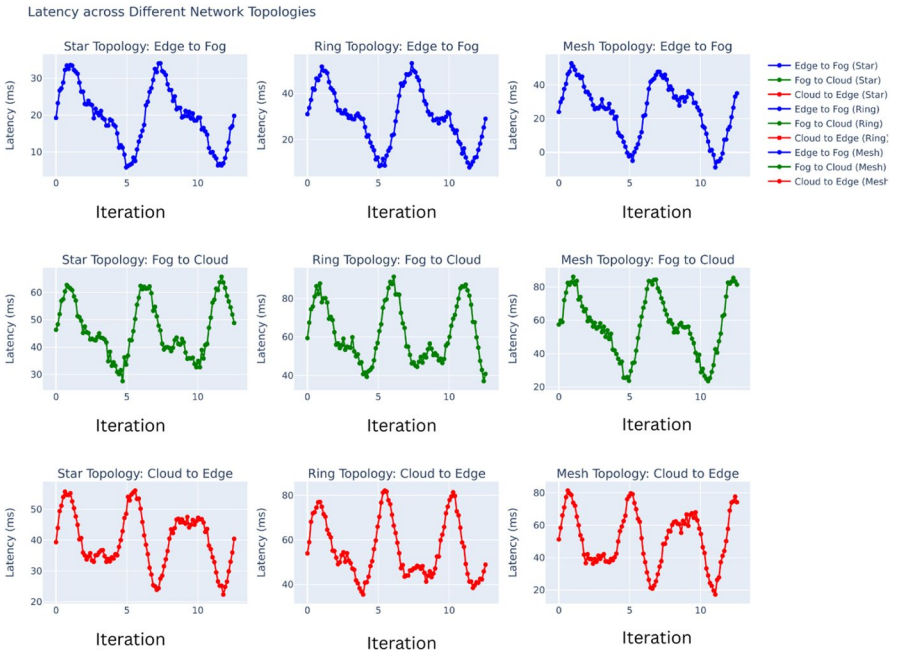


Fig. 10 End-to-end latency for direct Edge→Cloud (∼300 ms), Edge→Fog→Cloud (∼150 ms), and the full feedback loop, showing that Fog mediation roughly halves transmission delay. The X-axis represents the progression of end-to-end experimental cycles (Iterations). The integer values mark the completion of a cycle, while the data points show the metric sampled at high frequency throughout the duration of each cycle

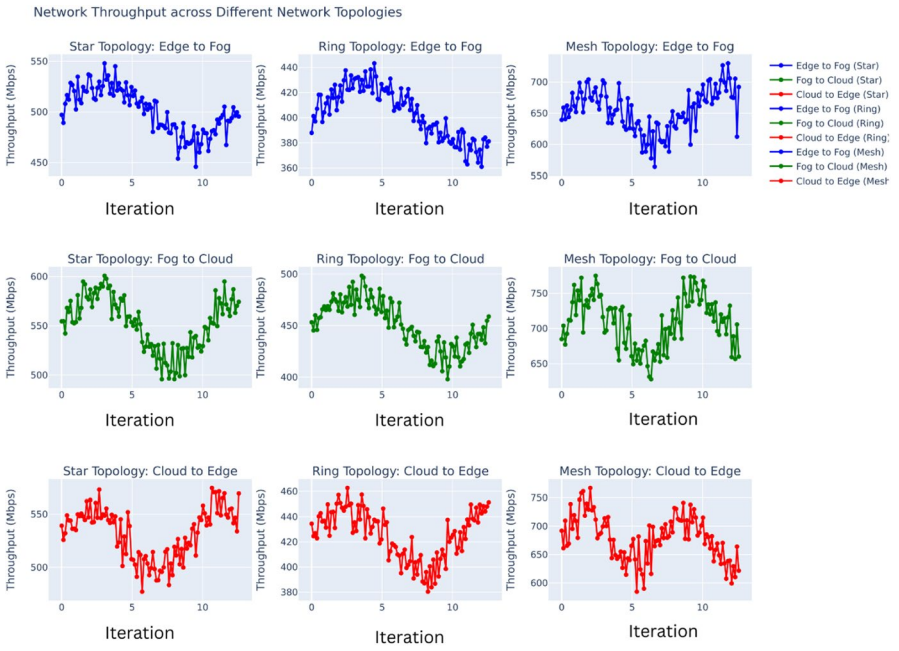


Fig. 11 Throughput (Mbps) comparisons for direct Edge→Cloud, Edge→Fog→Cloud, and full feedback loop topologies, with the hierarchical setup achieving the highest median rate via local aggregation. The X-axis represents the progression of end-to-end experimental cycles (Iterations). The integer values mark the completion of a cycle, while the data points show the metric sampled at high frequency throughout the duration of each cycle

tributed setup, as evidenced by higher data transfer rates within the Edge-Fog-Cloud topology compared to the baseline.

7 Conclusion

This research addresses the pivotal challenge of predicting vehicle health within modern transportation systems by integrating hyperparameter optimization with a distributed Edge-Fog-Cloud computing framework. We introduced the FogBayes, which employs Bayesian Optimization and distributed architecture to enhance the performance of gradient boosting models. Our empirical evaluation demonstrates that FogBayes significantly outperforms conventional classifiers, achieving an Accuracy of 98.88% and an AUC of 0.9830. Additionally, the distributed architecture optimizes energy consumption, network throughput, and latency, highlighting the efficiency and scalability of our approach. These findings contribute to the intersection of machine learning and distributed computing, providing a robust framework for real-time vehicle health monitoring. Future work will explore further optimization techniques and expand the framework to support larger-scale vehicular networks. While the current study focuses on core engine-related features for vehicle health classification, we acknowledge that the dataset does not account for variability across vehicle types,

operating environments, or usage patterns. This limitation may affect the generalizability of the proposed framework in diverse real-world scenarios. As part of future work, we aim to extend the dataset and model to incorporate broader contextual features such as vehicle category, load conditions, road type, weather variations, and driving behavior, to enhance the robustness and applicability of the approach.

Author contributions P.J. has spearheaded the study by formulating the core idea, identifying and defining the research problem, conducting an extensive literature survey, developing and refining the methodology, actively participating in experiment execution and validation, ensuring the accuracy and reliability of results, and performing a thorough analysis of the experimental outcomes and writing the main manuscript text. K.K.M. has assisted in formulating the methodology, writing the manuscript, and conducting the experiments. D.D. and A.K. provided invaluable supervision, offering critical insights, guiding discussions, and shaping the research direction through meaningful intellectual contributions.

Funding This work is partially supported by the TIH Foundation for IoT & IoE (TIH-IoT) (Project No.: TDP06-A-26).

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Coughlin JF, Reimer B, Mehler B (2011) Monitoring, managing, and motivating driver safety and well-being. IEEE Pervasive Comput 10:14–21
2. Torell W, Avelar V (2004) Mean time between failure: explanation and standards. White Paper 78:6–7
3. Niu G, Liu E, Wang X, Zhang B (2024) A hybrid bearing prognostic method with fault diagnosis and model fusion. IEEE Trans Industr Inf 20:864–872
4. NextMSC (2023) Predictive maintenance market analysis. *NextMSC Reports*. https://www.nextmsc.com/report/predictive-maintenance-market
5. Huang C et al (2024) Prognostics and health management for predictive maintenance: a review. J Manuf Syst 75:78–101
6. Xie S et al (2024) A novel interpretable predictive model based on ensemble learning and differential evolution algorithm for surface roughness prediction in abrasive water jet polishing. J Intell Manuf 35:2787–2810
7. Pagano D (2023) A predictive maintenance model using long short-term memory neural networks and Bayesian inference. Decis Anal J 6:100174
8. Yang L, Tham C-K, Guo S (2023) Fedprem: a novel federated reinforcement learning framework for predictive maintenance. GLOBECOM 2023–2023 IEEE Global communications conference pp 7556–7561
9. Massaro A, Selicato S, Galiano A (2020) Predictive maintenance of bus fleet by intelligent smart electronic board implementing artificial intelligence. IoT 1:12
10. Gomes IP, Wolf DF (2021) Health monitoring system for autonomous vehicles using dynamic Bayesian networks for diagnosis and prognosis. J Intell Robot Syst 101:1–21
11. Yang L et al (2024) Prem–Fediov: a novel federated reinforcement learning framework for predictive maintenance in IoV. IEEE Trans Mob Comput 23(12):11954–11970
12. Iqbal M et al (2025) Iov–Twinchain: predictive maintenance of vehicles in internet of vehicles through digital twin and blockchain. Internet Things 30:101514

13. Cui L, Shen Q, Xiao Y, Liu D, Wang H (2025) Sparse graph structure fusion convolutional network for machinery remaining useful life prediction. Reliab Eng Syst Saf 254:110592
14. Somu N, Dasappa NS (2025) An edge-cloud iiot framework for predictive maintenance in manufacturing systems. Adv Eng Inform 65:103388
15. Dai X et al (2023) Task offloading for cloud-assisted fog computing with dynamic service caching in enterprise management systems. IEEE Trans Indus Inf 19:662–672
16. Peng H, Xiong R, Feng T (2022) A cloud–fog based adaptive framework for optimal scheduling of energy hubs. IEEE Trans Indus Inf 18:5681–5688
17. Ghafouri S et al (2022) Mobile-kube: mobility-aware and energy-efficient service orchestration on kubernetes edge servers. 2022 IEEE/ACM 15th International conference on utility and cloud computing (UCC) pp 82–91
18. Sahil Sood SK, Chang V (2024) Fog-cloud-IoT centric collaborative framework for machine learning-based situation-aware traffic management in urban spaces. Computing 106:1193–1225. https://doi.org/10.1007/s00607-022-01120-2
19. Li Z et al (2024) A reference framework for the digital twin smart factory based on cloud-fog-edge computing collaboration. J Intell Manuf. https://doi.org/10.1007/s10845-024-02424-0
20. Jabeen N et al (2022) Autonomous vehicle health monitoring based on cloud-fog computing. 2022 International Conference on emerging trends in electrical, control, and telecommunication engineering (ETECTE) pp 1–6
21. Hassan M et al (2024) Optimizing structural health monitoring systems through integrated fog and cloud computing within iot framework. IEEE Access 12:89628–89646
22. Aslani A, Ghobaei-Arani M (2025) Machine learning inference serving models in serverless computing: a survey. Computing 107:47
23. Wang H, Mathews KJ, Golec M, Gill SS, Uhlig S (2025) Amazonaicloud: proactive resource allocation using amazon chronos based time series model for sustainable cloud computing. Computing 107:77
24. Ahmed I, Syed MA, Maaruf M, Khalid M (2025) Distributed computing in multi-agent systems: a survey of decentralized machine learning approaches. Computing 107:2
25. Ali TE, Ali FI, Dakić P, Zoltan AD (2025) Trends, prospects, challenges, and security in the healthcare internet of things. Computing 107:28
26. Rahim MA et al (2022) An intelligent risk management framework for monitoring vehicular engine health. IEEE Trans Green Commun Netw 6:1298–1306
27. Gu K, OuYang X, Wang Y (2024) Malicious vehicle detection scheme based on spatio-temporal features of traffic flow under cloud-fog computing-based iovs. IEEE Trans Intell Transp Syst 25:11534–11551
28. Joseph Chukwudi I et al (2024) An ensemble deep learning model for vehicular engine health prediction. IEEE Access 12:63433–63451
29. Rahman MA et al (2022) A secure and intelligent framework for vehicle health monitoring exploiting big-data analytics. IEEE Trans Intell Transp Syst 23:19727–19742
30. Amidstdebug (2023) Machine failure prediction model dataset. https://github.com/amidstdebug/Machine-Failure-Prediction-Model/tree/main. Accessed: 2025-08-18
31. Chen T et al (2015) Xgboost: extreme gradient boosting. R package version 0.4-2 1:1–4
32. Hancock JT, Khoshgoftaar TM (2020) Catboost for big data: an interdisciplinary review. J Big Data 7:94
33. Heath D, Kasif S, Salzberg S (1993) Induction of oblique decision trees. IJCAI 1993:1002–1007
34. Du W, Zhan Z (2002) Building decision tree classifier on private data. Electrical Eng Comput Sci
35. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55:119–139
36. LaValley MP (2008) Logistic regression. Circulation 117:2395–2399
37. Peterson LE (2009) K-nearest neighbor. Scholarpedia 4:1883
38. Webb GI, Keogh E, Miikkulainen R (2010) Naïve bayes. Encycl Mach Learn 15:713–714
39. Suthaharan S (2016) Support vector machine. Machine learning models and algorithms for big data classification: thinking with examples for effective learning pp 207–235

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Preeti Jain[1] · Koustav Kumar Mondal[1] · Debasis Das[1,2] · Arpit Khandelwal[1,3]

✉ Debasis Das
debasis@iitj.ac.in

Preeti Jain
preeti.1@iitj.ac.in

Koustav Kumar Mondal
mondal.4@iitj.ac.in

Arpit Khandelwal
arpitkhandelwal@iitj.ac.in

1 Centre for AIoT and Applications, Inter-Disciplinary Research Division, Indian Institute of Technology Jodhpur, Jodhpur, Rajasthan, India

2 Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur, Jodhpur, Rajasthan, India

3 Department of Electrical Engineering, Indian Institute of Technology Jodhpur, Jodhpur, Rajasthan, India