IntelliMaint: An Intelligent Component-Agnostic Framework for Health Indicator Generation, and Prognostics for ElectroMechanical Systems

Ramesh Krishnamurthy[1], Shweta S[1], Rachana Sreedhar[1] and Archana Chandrashekar[1]

[1] *Intellipredikt, Bengaluru, Karnataka, 560078, India*

ramesh@intellipredikt.com

shweta.s@intellipredikt.com

rachana.s@intellipredikt.com

archana.s@intellipredikt.com

ABSTRACT

Predictive maintenance of complex mechanical systems requires robust health monitoring capabilities that can generalize across diverse components and operating conditions. We present a novel component-agnostic framework that unifies Health Indicator (HI) generation and Remaining Useful Life (RUL) prediction through an integrated pipeline comprising of: (a) advanced feature engineering, (b) unsupervised health baseline modeling, (c) monotonicity and trendability learning (d) probabilistic degradation detection with confidence-aware RUL estimation. We validate our framework on two distinct industrial applications: (1) We monitor tool insert wear in a CNC machine using vibration and spindle current signatures. Our framework achieves early detection of tool life with RUL prediction within 10% of actual failure time. Flank wear (VB) was measured to evaluate tool wear. (2) We utilize the IMS bearing dataset to demonstrate fault detection earlier than traditional threshold methods with 95% confidence intervals.

Both case studies show strong HI monotonicity $> 60\%$ and reliable uncertainty quantification, establishing the foundation for scalable and explainable predictive maintenance solutions. The framework's component-agnostic design enables rapid deployment across heterogeneous assets without extensive reconfiguration, while its interpretable architecture facilitates root cause analysis and maintenance decision support. These results demonstrate significant advances in scalable and explainable predictive maintenance, offering practitioners a unified solution for diverse industrial health monitoring challenges.

1. INTRODUCTION

Failures in industrial machines can cause delays, rack up costs, and can even be catastrophic (Schwendemann, Amjad, & Sikora, 2021). In addition, they can also precipitate the failure of other components. Thus, being able to anticipate failures early and respond accordingly is crucial. Conventional methods include frequent manual inspections to check the health of the component. These can involve pausing the machines and measuring health, which is time consuming and inefficient (Rombach, Michau, Bürzle, Koller, & Fink, 2024). Intelligent RUL estimation maintains the efficiency and safety of industrial machines through proactive maintenance scheduling, which directly reduces delays caused by worn-out components and lowers costs of repairs (Lei et al., 2018). However, the broad application of effective predictive maintenance systems remains challenging due to fundamental limitations in current approaches. Different operating conditions mean that several current approaches cannot be used in a different environment, with different loads, speeds, or materials. In addition, many solutions cater only to specific components or systems that can require extensive reconfiguration or retraining for other new components/systems. They may also require labeled data from previous failures which is hard to obtain, as many times failures are not recorded or the creation of labeled data can be time consuming and impractical.

To address the above limitations, we propose IntelliMaint, a novel, intelligent framework that can be applied on a diverse array of components to monitor, diagnose and prognosticate

in different domains. It's adaptive learning methodologies don't require labeled failure data as it understands the underlying structure of normal operating patterns and provides monotonicity enforcement to ensure interpretable and realistic degradation trends. The framework simplifies the development of PHM applications, making it easier to adapt the library to various components and operating environments.

2. RELATED WORKS

The increasing availability of data has provided ways to predict future events in the industrial maintenance sphere. However, due to the large diversity of systems, materials and conditions, a generic and comprehensive framework that makes it accessible and convenient to create PHM applications is needed. During the creation of the library, we focused on 4 requirements: (1) Unsupervised Health Indicator creation (HI), (2) Unsupervised RUL estimation, (3) Component agnostic, and (4) End to end workflow. We explored current literature pertaining to these requirements.

2.1. Health Indicator

Health Indicators (HI) are useful as they are a real time representation of the health of the component. HIs are different from condition indicators in that while condition indicators are usually a single feature that reflects the health, HIs are usually combinations of features (Zhou et al., 2022). They can distinguish between degraded and normal functioning of systems. Traditional approaches to calculate condition indicators are statistical in nature like Root Mean Square (RMS) or Spectral Kurtosis. However, separately, each signal may not represent the complete health status of the component (Zhou et al., 2022). To counter this, a HI is created by aggregating several statistical features that more accurately reflects the health (Li et al., 2023). There are supervised and unsupervised approaches to this type of feature fusion. (Wang, Zhang, Guo, & Lin, 2022) creates a health indicator from multiple time domain features by employing a Supervised Depth network. However, it still encounters the challenge of often requiring labeled data and limited transferability between components.

Unsupervised approaches extracts time-domain features from bearing vibration signals and fusing them into a single HI using Principal Component Analysis (PCA) (Gao, Li, Huang, & Wang, 2021). However, PCA doesn't capture nonlinear structures. Self Organizing Maps (SOM) is often utilized as a type of unsupervised feature fusion (Hong, Zhou, Lu, Heart, & Zhao, 2015). SOM is an unsupervised neural network with the advantage that it can be trained on healthy data and does not require any faulty samples. Since many industrial systems lack extensive historical failure data, our framework uses SOM to learn a baseline "healthy" state from normal operating data, allowing it to detect degradation and anomalies without requiring a large number of complete run-to-failure cycles. This is a significant advantage over supervised models that demand extensive labeled data for training.

2.2. RUL Estimation

Remaining Useful Life (RUL) estimation is a critical component of predictive maintenance, enabling the assessment of operational longevity for industrial assets through sensor data and degradation modeling (Wu, Wu, Tan, & Xu, 2024). Broadly, RUL methodologies fall into physics-based models (reliant on domain-specific mathematical formulations) and data-driven approaches (leveraging historical and real-time operational data). Modern research increasingly focuses on data-driven techniques due to their adaptability across systems. Deep learning models like LSTM networks excel at capturing temporal dependencies in sensor time-series data, with bidirectional variants (BiLSTM) enhancing feature extraction through forward-backward sequence analysis (Nie, Zhang, Xu, Cai, & Yang, 2022). Particle Filtering (PF) employs Monte Carlo simulations to recursively update state estimates in dynamic systems, particularly effective for nonlinear degradation processes with uncertainty (Fan, Yang, Li, & Wang, 2015). Gaussian Process Regression (GPR) provides probabilistic RUL forecasts by modeling stochastic degradation trajectories, offering inherent uncertainty quantification (Hong et al., 2015) and outperforms PF in scenarios requiring probabilistic uncertainty quantification. In our work, we employ GPR based models to provide probabilistic RUL predictions with confidence intervals, enabling risk-based maintenance decisions rather than deterministic point estimates.

2.3. Component agnostic framework

PHM is necessary for any machines with electro-mechanical signals and parts because it enables early detection and prediction of faults and can reduce downtime and maintenance costs. This makes a generic and complete workflow necessary but also challenging to create. Most current solutions for RUL estimation are specific to a certain system or type of component or signal and they may require complete re-engineering when applied to new components or operating environments. This limitation could prevent organizations from scaling up in their predictive maintenance implementations. The specificity of many existing approaches could be because of several factors: (1) Component-Level Focus: Much of the research has traditionally concentrated on component-level PHM for particular parts, such as bearings (Kim, Kim, & Choi, 2021), rather than comprehensive system-level PHM. (2) Model-Based Limitations: While physics-based methods can offer high accuracy by employing physical and damage propagation models, they are often restricted to specific components and fault types (Qi, Zhu, Liu, Mauricio, & Gryllias, 2024). (3) Uncertainty Handling: While advanced Deep Learning (DL) models can

generate accurate RUL predictions, their inherent ability to account for uncertainties is often limited, necessitating hybrid approaches (Aizpurua et al., 2022). There are a few solutions that are scalable and generic enough to extend to a variety of usecases and scenarios. Progpy (Teubert, Jarvis, Corbetta, Kulkarni, & Daigle, 2023) is a robust set of support packages for development of PHM applications. The library focuses on modeling and prediction aspects. Another framework (Kansanaho & Kärkkäinen, 2019) is generic across different systems and components providing core PHM functionalities and modular architecture. However, this framework is limited to tribotronic systems only. Given the prevalence and need for prognostics across different systems such as electrochemical and electro-mechanical ones, Intellimaint seeks to provide prognostic solutions for a diverse array of systems by separating core logic from domain-specific configurations.

2.4. End to end workflow

An efficient and end to end predictive framework enables developers to efficiently deploy robust, scalable, and sophisticated predictive maintenance solutions for various systems. Progpy (Teubert et al., 2023), while being a robust framework for prognostics and predictive maintenance, relies on external tools for data clean up, signal processing and feature extraction. (Zhuang, Xu, & Wang, 2023) worked on a general predictive maintenance framework based on Bayesian deep learning. It addresses uncertainty quantification, and the frameworks aim to manage complexity and uncertainty but doesn't support the entire life cycle. In IntelliMaint, we choose to address all aspects of the Predictive Maintenance (PdM) pipeline, from data acquisition to post processing artifacts such as visualizable health indicators, degradation trends, and confidence bounds that enable maintenance personnel to understand and trust the system's recommendations.

3. PROPOSED FRAMEWORK

We propose a single framework that can effectively monitor, diagnose, and prognosticate across diverse components, signals, and operational domains.

3.1. Key Features

- Adaptive Data Driven Learning: The core library employs unsupervised approaches like self organizing maps (SOM), Autoencoders, Transformers methods for adaptive unsupervised learning. In this paper, we've considered and applied SOM to learn from the normal operation of the component, thus eliminating the need for labeled failure data. Because of this, it can be generalized to different operating conditions. It learns what constitutes normal for that component and operating environment. Since it has an understanding of baseline operation, it can provide a health metric that tracks when deviation from the baseline occurs. Additionally, it dynamically updates its learning as each new normal data point becomes available. For degradation learning, it employs Gaussian Process Regression based model, which is a probabilistic, Bayesian based framework that models the degradation trend by learning from the health metric derived by SOM. It makes no assumptions of the degradation form, it could be exponential or linear and provides confidence bounds for the predicted degradation.
- Component agnostic by design: The framework has been designed as a flexible and customizable set of modules, due to which it can be employed across different systems, be it tools/bearing/motors/batteries etc. and physical domains such as mechanical, electrical or chemical. The Component Template isolates domain logic while reusing IntelliMaint's core for HI generation, anomaly detection, and RUL prediction.
- Monotonicity: Ensures health indicators decrease over time by leveraging Isotonic regression which provides a mathematical guarantee of monotonic health indicator evolution. Additionally, the system enhances trendability by intelligently filtering noise.
- Probabilistic Uncertainty Estimation: The library provides RUL with quantified uncertainty. For example, it provides explainable and risk aware predictions such as "Failure expected in 120 cycles ± 20 cycles at 95% confidence", helping make risk informed decision. Further, it can provide confidence bounds that narrow as failure approaches. This can help with maintenance planning.
- Explainable, Interpretable Results: IntelliMaint provides intuitive visualizations that empower decision-makers through HI curves, degradation phase identification and feature contribution analysis.
- Scalability and Modularity: IntelliMaint employs a object-oriented design with interchangeable modules which lends it a plug and play extensibility. Different signal types, custom domain-specific features, physics and data driven approaches can be seamlessly integrated.
- Ease of Use: Since the entire PHM workflow is contained in the Component Template, even domain engineers who may not necessarily be data scientists can use the library by choosing an existing model and inheriting the remaining from the Base class. See Table 1 for a summary of advantages IntelliMaint provides from the perspective of the developer, based on internal development experience. Future developer sessions are planned to assess these benefits empirically. The proposed plan is explained in the Appendix.

Feature	Without Template	With IntelliMaint Template
Code per asset	Built from scratch	Only override domain specific methods
Model reuse	Manual integration	Pre-wired models (SOM, GPR)
Developer skill required	High (ML + DSP)	Moderate (domain + config)
Consistency of results	Varies	Standardized + testable pipeline

Table 1. Advantages of IntelliMaint framework from the Perspective of the Developer

3.2. Framework design

The core library is organized around a set of core abstract and base classes, each encapsulating a specific stage of the data-driven maintenance workflow. The base class defines the entire PHM workflow, this way developers don't need to create it from scratch, they only need to create the domain specific methods. The core classes have prebuilt implementations of popular analysis approaches like SOM and GPR. The user can create asset specific monitors by overriding core class methods to handle system/domain specific logic. These overridden classes are brought together by the Component Template which manages the end-to-end process: loading data, extracting features, detecting anomalies, and predicting RUL, while handling errors and logging at each step. Fig 1 and Table 2 contain an overview of the framework. These are the 4 core steps:

- Data Acquisition: The DataAcquisition class handles the discovery and loading of raw sensor data from various file formats (CSV, MAT, etc.). These classes provide methods for listing files, loading individual files, and preprocessing data as needed, supporting both generic and asset-specific data structures. It can be subclassed to serve domain-specific functionality (e.g., For a bearing component, you could make a BearingDataLoader to handle additional requirements such wanting only numeric data etc.).
- Data manipulation: This group of classes includes generic signal processing classes (Signal Separation, Signal Enhancement), feature extraction classes (TimeDomain, FrequencyDomain) and feature selection. These classes provide methods to process and enhance raw signal data, compute statistical, time-domain, and frequency-domain features, as well as specialized features relevant to the monitored asset. Feature extraction can be performed on a per-file or per-window basis, and results are aggregated into structured DataFrames for downstream analysis.
- State Detection: This group of classes include health indicator construction and anomaly detection (SOMAnomalyDetection, COSMOAnomalyDetection) (Rafik, 2019), which implement self-organizing map (SOM) and other algorithms for creation of health indicators and for unsupervised detection of abnormal behavior. These classes are responsible for training baseline models, scoring new data, and visualizing anomaly distributions.
- Prognostics Assessment: For assets where remaining useful life (RUL) estimation is required, the framework provides modelers such as Gaussian Process Regression based models to forecast degradation trends and predict time-to-failure. These components can be further customized for specific assets or degradation patterns.

Additionally, all components are configured via nested Python dictionaries, allowing users to specify file patterns, preprocessing options, feature extraction parameters, and model hyperparameters without changing code. The framework's design encourages extension through subclassing and method overriding, so new asset types or algorithms can be integrated by implementing only the relevant methods. In terms of visualization and reporting, the framework provides detailed visualizations (e.g: anomaly score distributions, health index trajectories, RUL forecasts) and logs throughout the workflow, supporting both interactive exploration (e.g., in Jupyter notebooks) and automated reporting.

4. THEORY

IntelliMaint's mathematical framework outlines a systematic approach to processing raw sensor data, extracting meaningful features, transforming them into a normalized Health Indicator (HI), modeling degradation, and estimating Remaining Useful Life (RUL) probabilistically. Below is a detailed explanation of its mathematical formulations of the components:

4.1. Feature Extraction and Fused Indicator

This initial stage focuses on converting raw, high-dimensional sensor signals into a set of relevant features that capture the system's state and then combining these features into a single fused indicator. The process begins with $x_t \epsilon R^d$, which represents the multivariate signal snapshot captured at a specific time t and d sensor channels. This is the raw sensor data, potentially from multiple sensors or multiple axes (like X, Y, Z vibration data). A feature extractor function $feat\_extract()$ is applied to x_t to derive a set of salient features, denoted as f_t . These features are designed to be

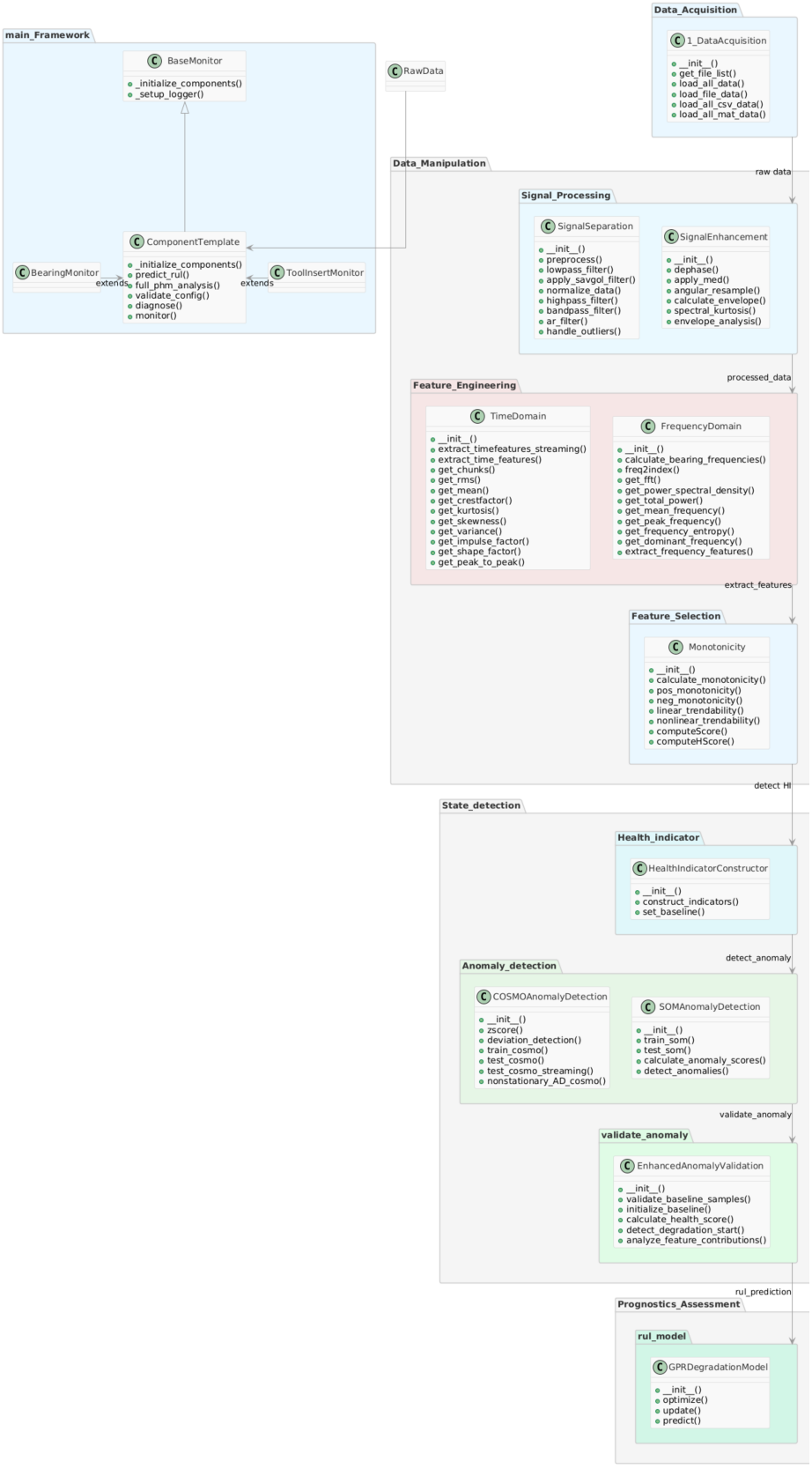


Figure 1. Framework Design

Input	Class	Output
Raw signals	Data Acquisition	Structured data
Structured data	Signal Processing	Processed data
Processed data	Feature Engineering	Derived features
Derived features	Feature selection	Selected features
Selected features	Health Indicator	HI values
HI values, Selected features	Anomaly detection	Anomaly set
Anomaly set	Validate anomaly	Validated anomaly
Selected Features, Anomaly set	Prognostics RUL Model	RUL predicted

Table 2. Framework Design Data Flow

sensitive to degradation and component wear. Examples of features extracted include: Root Mean Square (RMS), Kurtosis, Spectral Kurtosis, Discrete Wavelet Transform (DWT) Coefficients.

$$f_t = feat\_extract(x_t) \tag{1}$$

The framework utilizes a Self-Organizing Map (SOM) for unsupervised learning. SOM is a competitive neural network that performs unsupervised feature mapping through topological preservation. The SOM MQE error e_t is computed as the squared Euclidean distance between the extracted feature vector f_t and $w*$, which represents the weight vector of the Best Matching Unit (BMU) from the trained SOM (Hong et al., 2015). This error measures how well the current feature vector is represented by the learned prototypes of the SOM. A higher error might indicate a departure from the learned baseline or healthy states.

$$e_t = ||f_t - w*||^2 \tag{2}$$

The IntelliMaint framework then creates a "fused indicator" z_t from the SOM MQE error (e_t). This fusion aims to create a more robust single metric that captures different aspects of degradation.

$$z_t = e_t \tag{3}$$

4.1.1. Adaptive HI mapping

The raw fused indicator z_t can be unbounded and may not directly represent degradation in an intuitive way. This step adaptively maps z_t into a normalized Health Indicator (HI), denoted as h_t. The HI is typically a bounded metric (e.g., between 0 and 1) that monotonically increases or decreases with degradation. Four types of mapping functions M are provided:

1. Log Mapping: This mapping uses a logarithmic transformation, useful when the indicator z_t spans several orders of magnitude. α is a scaling parameter, and z_{min} is a reference minimum value.

$$M(z_t) = \exp(-\alpha * (\log(z_t) - \log(z_{min}))) \tag{4}$$

2. Min-Max Mapping: This is a common linear normalization technique that scales the indicator between 0 and 1, assuming degradation increases as z_t moves from z_min to z_max. It can be adapted to scale between 0 and 1 or 1 and 0 depending on whether increasing z_t means worsening health. The formula maps minimum health to 1 and maximum health to 0, or vice versa, depending on the interpretation ofz_{min} and z_{max}.

$$M(z_t) = (z_t - z_{min})/(z_{max} - z_{min}) \tag{5}$$

3. Sigmoid Mapping: The sigmoid function maps any real value to a range between 0 and 1. This can be useful for indicators that exhibit an S-shaped degradation curve.

$$M(z_t) = 1/(1 + \exp(-\beta_0 - \beta_1 * z_t)) \tag{6}$$

4. Isotonic Regression: This is a non-parametric method that learns a monotonic (non-decreasing or non-increasing) mapping function from z_t to h_t . It is data-driven and does not assume a specific functional form.

4.1.2. Monotonicity Enforcement

To ensure consistent degradation trends, the mapped health indicator h_t is enforced to be monotonic. One simple stepwise enforcement is:

$$h_t = \min(h_t, h_{t-1} - \epsilon) \tag{7}$$

4.1.3. Degradation Modeling & RUL Estimation

This final stage models the dynamics of the Health Indicator over time and uses this model to predict the Remaining Useful Life (RUL) through:

1. HI Dynamics Modeled as Gaussian Process (GP): The

Health Indicator (HI) h_t is modeled as a Gaussian Process. A Gaussian Process defines a distribution over functions, meaning it can model the uncertainty in the HI trajectory. It is characterized by a mean function $\mu(t)$ and a covariance (kernel) function $k(t, t')$.

$$h_t \approx GP(\mu_t, k(t, t')) \tag{8}$$

This allows IntelliMaint to not only predict the future HI trajectory but also to provide probabilistic uncertainty bounds around these predictions.

2. RUL Estimation: The Remaining Useful Life (RUL) is estimated as the minimum future time t^* at which the predicted Health Indicator $h(t^*)$ reaches or crosses a predefined failure threshold δ. Below we assume δ is a low health threshold:

$$RUL = min(t^* | h(t^*) \leq \delta) \tag{9}$$

Because the HI dynamics are modeled probabilistically via a GP, the RUL estimation also inherently comes with uncertainty bounds (e.g., 2σ).

This mathematical framework enables IntelliMaint to learn component-agnostic health baselines via unsupervised learning, adaptively map raw metrics to a normalized HI, enforce monotonicity, and predict degradation trajectories and RUL probabilistically with uncertainty bounds.

5. CASE STUDIES

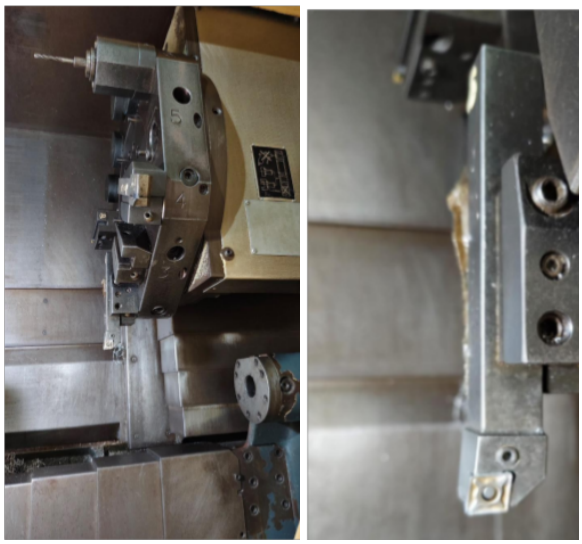
The framework was tested through 2 case studies: The first is a real world experiment conducted on the CNC machine (Turn mill) where the run to failure data of the tool insert was collected. The second study tested the framework on a bearing vibration dataset created by the Center for Intelligent Maintenance Systems (IMS) of the University of Cincinnati (*IMS Bearings*, 2007).

5.1. Tool Insert Monitoring

Vibration data was collected from an accelerometer sensor mounted on the spindle headstock of the CNC machine, over the course of 4 days with gaps. The Flank Wear (VB), Surface Roughness (Ra) and Current were measured every few hours with a digital microscope, profilometer and Hall effect current transducer respectively, see Table 3 and Fig 5b. The purpose of these physical measurements was to provide an accurate timestamp of the tool's degradation and failure in the real world. A run-to-failure experiment was conducted with a single carbide insert operated on Mild Steel work pieces, Fig 3b. The tool performed facing operations on the workpieces under regular industrial working conditions.

5.1.1. Experiment Specifications

- Sensors: NCD Predictive Maintenance accelerometer



(a) Insert in Machine (b) Close up of insert

Figure 2. Tool Insert in CNC Turn Mill Machine

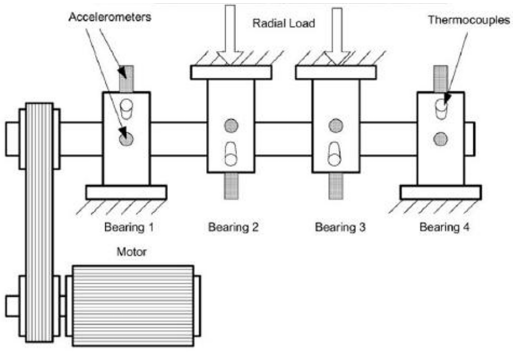
- Insert: Carbide insert - CNMG 120408-DR YBC252
- Work piece: Mild-steel workpieces
- Total number of work pieces operated on: 24 work pieces
- Number of files recorded: 88 files or snapshots
- Machine: DMG CTX 310 eco V3 CNC
- Sampling Rate: 25.6 kHz for 52 milliseconds every 5 minutes
- Facing operation cycle time: 16 minutes
- Depth of cut: 0.5 mm
- RPM: 1500
- Feed rate: 0.3 mm/rev

5.2. IMS Bearing Dataset

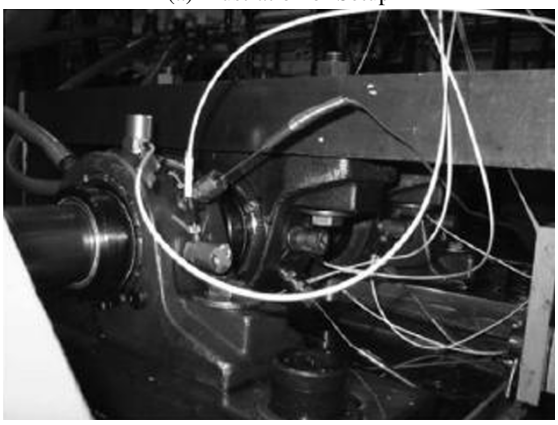
We used a popular dataset from the Center for Intelligent Maintenance Systems (IMS) of the University of Cincinnati (*IMS Bearings*, 2007) in this evaluation. In these run to failure tests, four double row bearings were installed on one shaft. The shaft rotation and the radial load were constant during the test-runs and all bearings were force-lubricated. For each test-run, run-to-failure data was collected. For our evaluation, we used the data from the second set for bearing 1. At the end of the test-to-failure experiment, outer race failure occurred in bearing 1.

5.2.1. Experiment Details

- Sensors: PCB 353B33 accelerometer sensor
- Sampling: 20 kHz for 1 second every 10 minutes
- Loading: 6000 lbs radial load
- Lubrication: Oil circulation system



(a) Illustration of Setup



(b) Sensor placement

Figure 3. Bearing Setup and Sensor placement

- Duration: February 12, 2004 to February 19, 2004
- Bearing: Rexnord ZA-2115 double row

6. RESULTS AND DISCUSSION

6.1. Tool Insert

6.1.1. Overview

Firstly, time domain features like RMS, Skewness, Crest Factor, Kurtosis and frequency domain features like Spectral Kurtosis based features are extracted. The features are then ranked according to their monotonicity scores and the top k features are selected. The value of k is chosen empirically through trial and error to maximize prognostic performance. Secondly, an initial set of data points (20%) is considered as the baseline i.e. normal functioning of the insert. The extracted features are used to compute a health indicator by taking the Euclidean difference between a baseline sequence and the remaining data points. The cumulative wear is then derived by sequentially summing the health indicator values, normalizing the cumulative sum to the [0, 1] interval, and expressing it as a percentage. When the cumulative wear crosses 60% wear, we consider that to be the warning threshold and 80% as the critical threshold.

Algorithm Parameters:

- k: 7
- Baseline for Health Indicator: 20% of data points
- Warning threshold: 60% of cumulative wear
- Critical threshold: 80% of cumulative wear

Next, to estimate RUL with confidence intervals, an ensemble model consisting of Gaussian Process regression (GPR) and Linear regression on polynomial fit is employed. An ensemble approach can leverage the strengths of both GPR and linear regression. While GPR can handle the complex, nonlinear parts of the degradation, linear regression can capture the overall trend. The parameters of the GPR are determined through Hyperparameter grid search and Time Series cross validation. The hyperparameters are optimized with a grid search of different kernel parameters of learning scale and noise levels. Below are the GPR kernels we finally used:

- RBF with length scale 10.0 and bounds of 0.01 to 100
- WhiteKernel with default noise level
- DotProduct with sigma of 10.0

The models are trained using data collected up to the stage when cumulative wear surpasses 60% of its total. The different models in the ensemble training were weighted using dynamic weighting with the GPR being given higher importance closer to failure. The source code underlying this work is proprietary and its access may be granted to interested parties upon request.

6.1.2. Framework customization

Fig 4 depicts the customization of the IntelliMaint framework in the creation of the Tool Insert monitoring.

6.1.3. Analysis of results

Since the amount of data available for tool insert wear monitoring is limited (only 88 files), we focused on using the cumulative wear thresholds of 60% and 80%. Fig 5a shows the cumulative wear in percentage with key prognostic thresholds indicated: a warning threshold at 60% wear, a critical threshold at 80%, and complete failure at 100% wear. The blue curve represents the GPR-predicted wear progression, with the shaded region denoting the 95% confidence interval (CI), capturing uncertainty in the forecast.

The accuracy is calculated by measuring how close the predicted RUL is to the actual amount of time before failure.

$$\text{Accuracy} = \left(1 - \frac{|\text{RUL}_{\text{predicted}} - \text{RUL}_{\text{actual}}|}{\text{RUL}_{\text{actual}}}\right) \times 100\% \tag{10}$$

At the warning threshold, the RUL is estimated to occur after 2.33 hours which is 90% accurate as the actual failure occurs after 2.59 hours. At the critical threshold, the RUL is

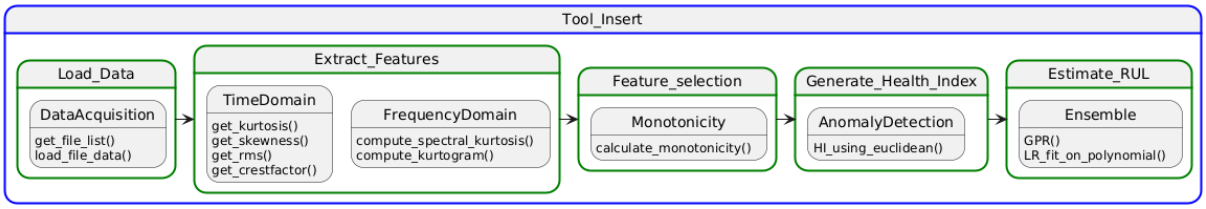


Figure 4. Tool Insert Monitor created with the Component Template

estimated to be 1.08 hours which is 92% accurate as the actual failure occurs after 1 hour. The 95% confidence bounds are shown. The algorithm accurately identifies the onset of rapid wear and provides early predictions of RUL at both warning and critical points, as annotated on the plot. The close alignment between measured and predicted wear, along with quantified uncertainty, demonstrates IntelliMaint's effectiveness in generating robust health indicators and supporting timely maintenance decisions in electromechanical systems. In practical contexts, this enables scheduling maintenance windows hours in advance, potentially avoiding unplanned stops.

Table 3 shows the three key physical measurements: Flank Wear (VB), Surface Roughness (Ra), and Spindle Current over time, while Fig 5b illustrates the correlation between observed cumulative wear percentage and the physical measurements. The observed cumulative wear percentage is derived by sequentially summing the observed wear values, normalizing the cumulative sum to the [0, 1] interval, and expressing it as a percentage. (The wear value at the start i.e. new tool, is mapped to 0 and the wear value at failure i.e. completely worn out tool is mapped to 1). Since the physical measurements were only taken every few hours due to logistical constraints in the testing environment, the intermediate values are interpolated from measured values. As the cumulative wear increases, flank wear and surface roughness exhibit strong upward trends, with surface roughness showing a particularly sharp rise after the 5-hour mark, reflecting degradation in tool quality. Notably, current draw increases initially but plateaus after approximately 4 hours, suggesting that current is a more reliable early indicator than a late-stage predictor. The observed cumulative wear percentage curve closely follows the Flank wear's trend, supporting Flank wear's role as a key wear metric.

6.1.4. Comparison with baseline model

Table 4 presents a comparative evaluation of two RUL (Remaining Useful Life) estimation approaches: an Ensemble Model and a Linear Fit model, based on multiple performance metrics. The Ensemble Method significantly outperforms the Linear Fit in terms of predictive accuracy and robustness. It achieves lower root mean square error (RMSE = 0.0314) and mean absolute error (MAE = 0.0278), indicating more precise RUL predictions. Moreover, its R^2 score of 0.9382 reflects a high degree of fit to the actual wear progression data, compared to 0.7464 for the Linear Fit. When assessed at key degradation thresholds, the Ensemble Method demonstrates higher accuracy at the warning threshold (60% wear) with 89.96% accuracy, and markedly better performance at the critical threshold (80% wear) with 92% accuracy, compared to only 58% for the Linear Fit. Interestingly, both methods achieve high accuracy for failure prediction (97.66% and 98.76%, respectively), suggesting that linear models may suffice near end-of-life but struggle earlier in the degradation cycle. Overall, the results highlight the superior early and mid-life prediction capability of the Ensemble Method, making it more suitable for proactive maintenance and early warning applications within the IntelliMaint framework.

6.2. Bearing

6.2.1. Overview

Time and frequency domain features are extracted from all bearing data files. The extracted features are then ranked according to their monotonicity scores. Feature selection is performed by selecting the top k features, with the value of k determined empirically to maximize prognostic performance. For the below experiment, we selected 13 features with 9 time domain ones: RMS, Peak, Crest Factor, Kurtosis, Skewness, Standard deviation, Peak-to-Peak,Shape Factor and Impulse Factor and 4 feature domain ones: BPFO (Ball Pass Frequency Outer), BPFI (Ball Pass Frequency Inner), FTF (Fundamental Train Frequency) and BSF (Ball Spin Frequency). An initial portion of the feature dataset (20%) is treated as baseline data representing normal operation. A Self-Organizing Map (SOM) neural network is trained and used to find the best matching unit's weight vector. The Minimum Quantization Error (MQE) between each input sample and the corresponding best matching unit is used to construct the Health Indicator (HI). Essentially, for each new feature vector, the MQE value is calculated as the Euclidean distance from the vector to the closest matching neuron on the trained SOM. The point of degradation is also identified based on deviations in the MQE trend. The HI is then constructed by mapping the log(MQE) values to a Health Index using fixed exponential decay.

Algorithm Parameters:

Workpiece	VB(µm)	Ra(µm)	Current (a)
1	78.62	0.613	11.3
8	299.45	1.506	13.28
16	558.85	1.866	13.23
19	570.41	2.276	13.27
21	586.8	3.653	13.325
24	595.46	4.016	13.5

Table 3. Measured values of Flank Wear - VB (µm), Surface Roughness - Ra (µm), and Current (Amps)

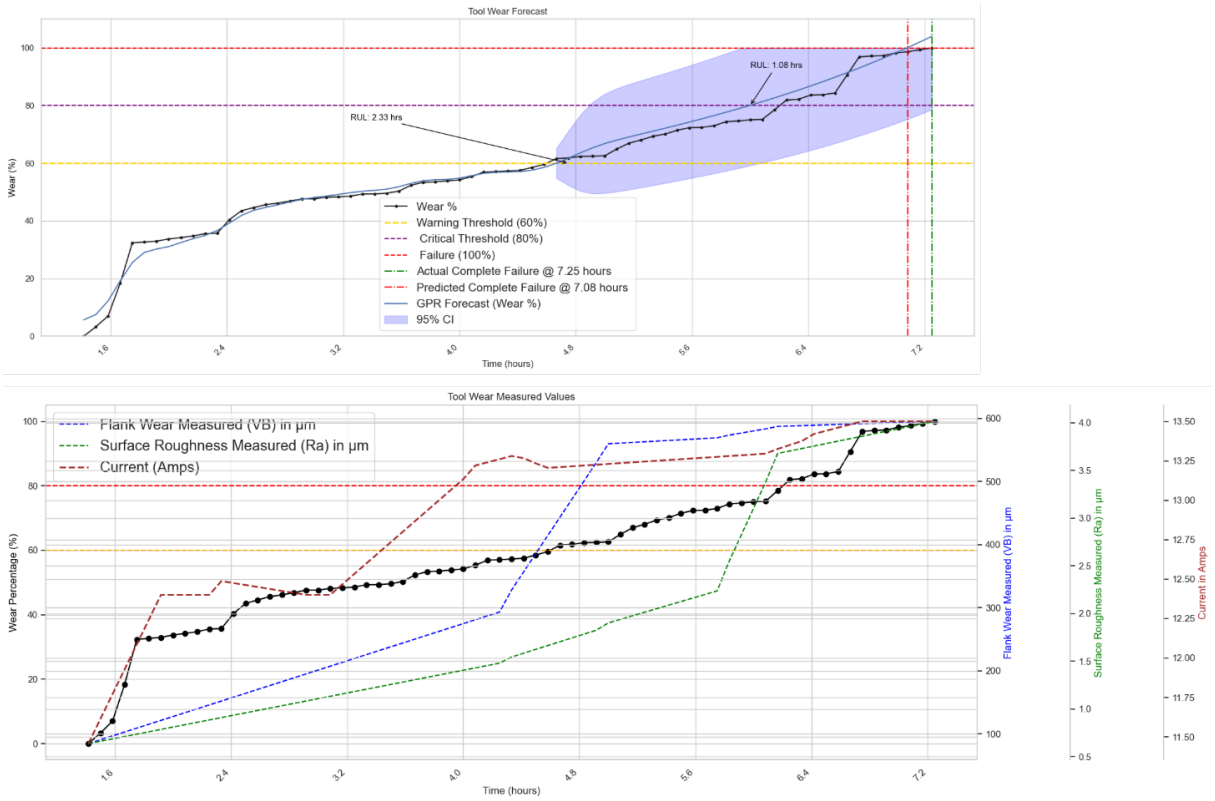


Figure 5. (**a**) Observed Cumulative Wear (black) and Predicted Wear (blue), (**b**) Measured values of Flank Wear (blue), Surface Roughness (green) and Current (brown) plotted against Observed Cumulative Wear (black)

- k: 13 features
- Baseline Data for SOM training: 20%
- SOM Map Size: 50x50
- SOM Training Iterations: 500

From this degradation point onward, Remaining Useful Life (RUL) is estimated using Gaussian Process Regression (GPR) with confidence intervals. GPR is selected for its ability to capture non-linear behavior and provide uncertainty-aware predictions. Data from the first point of occurrence of the degradation is used as training data for GPR. The training is done with a sliding window technique, where for a certain number of data points, the next data point is the label. Below are the kernels we employed for the GPR:

- ConstantKernel,
- RBF with length scale 5.0
- WhiteKernel with noise level 1×10^{-3}

The source code underlying this work is currently proprietary and its access may be granted to interested parties upon request.

6.2.2. Framework customization

Fig 6 depicts the customization of the IntelliMaint framework in the creation of the Bearing monitoring.

RUL estimation method	RMSE	MAE	R^2 score	Alpha Score	Lambda Score	Accuracy of RUL at Warning %	Accuracy of RUL at Critical %	Accuracy of Predicted failure %
Ensemble Method	0.0314	0.0278	0.9382	0.9639	0.0021	89.96139	92	97.65517
Linear Fit	0.0637	0.0571	0.7464	0.922	0.0044	87.25869	58	98.75862

Table 4. Comparison of RUL Estimation Methods for Tool Insert

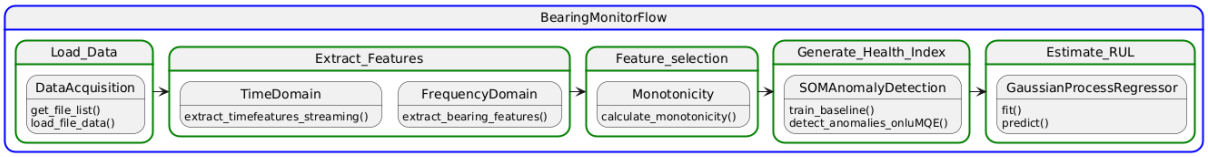


Figure 6. Bearing Monitor created with the Component Template

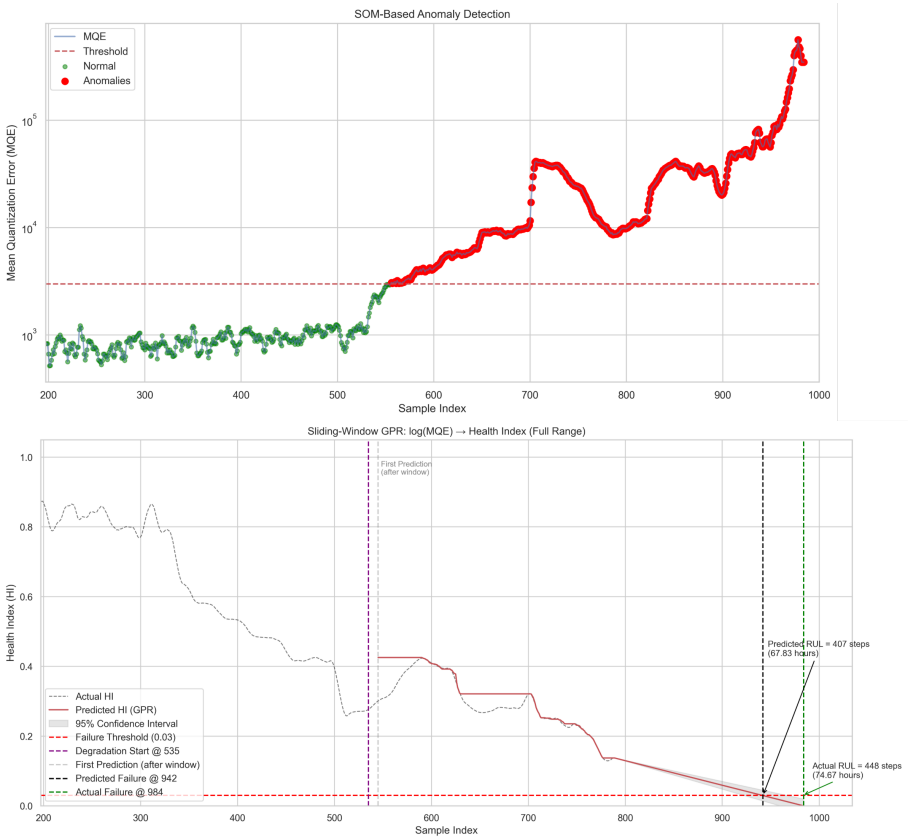


Figure 7. (a) Anomaly Detection and (b) RUL Prediction in Bearing

6.2.3. Analysis of results

Fig 7a shows the Mean Quantization Error (MQE) computed using the trained SOM across the extracted features. Since the initial 20% is set aside for SOM training, only the portion of data used for evaluation is shown in the plot. The green markers represent data points that were identified as normal during testing, while the red markers indicate samples where anomalies were detected. Initially, the MQE remains stable and low, reflecting normal behavior, until a clear shift occurs at sample 535, where MQE crosses the threshold and degradation (anomaly) is detected. Beyond this point, MQE values rise steadily, indicating progressive fault development.

Fig 7b illustrates the Health Index (HI) derived from the MQE values, along with RUL prediction using Gaussian Pro-

cess Regression (GPR). Training and evaluation followed a sliding-window approach. The start of degradation at sample 535 is highlighted, aligning with the anomaly point in the SOM plot. After an initial prediction window, the GPR model begins forecasting HI progression from this point onward. The predicted HI closely follows the actual HI, with a 95% confidence interval shown as the shaded band. As MQE increases, HI decreases, essentially capturing the worsening system health. The HI failure threshold of 0.03 was selected empirically by analyzing the run-to-failure dataset. Specifically, we examined the distribution of HI values across the baseline and degradation phases. A threshold near 0.03 maximized separation between healthy and failed states, while also aligning with the 95th percentile of healthy variation. This ensured that the threshold was sensitive to degradation onset without generating false alarms. The GPR model predicts failure at sample 942, while actual failure occurs at sample 984. This corresponds to a predicted RUL of 407 steps (67.83 hours) versus an actual RUL of 448 steps (74.67 hours), measured from the degradation point. The accuracy was calculated by measuring how close the predicted RUL is to the actual amount of time before failure, achieving approximately 91%.

$$\text{Accuracy} = \left(1 - \frac{|\text{RUL}_{\text{predicted}} - \text{RUL}_{\text{actual}}|}{\text{RUL}_{\text{actual}}}\right) \times 100\% \tag{11}$$

To further evaluate the GPR model's performance, several error metrics were considered, as summarized in Table 5. It achieves a low RMSE and MAE, indicating high precision in RUL prediction and minimal deviation from the actual values. Moreover, its R^2 score of 0.8795 reflects a strong correlation with the observed degradation trend, suggesting the model effectively captures the underlying wear progression dynamics. These results demonstrate IntelliMaint's effectiveness in constructing robust health indicators and delivering accurate, early RUL predictions, thereby supporting timely maintenance decisions in electromechanical systems.

6.2.4. Comparison with Baseline model

Table 5 summarizes the performance of different RUL estimation methods: Gaussian Process Regression (GPR), Support Vector Regression (SVR), and Sparse Variational Gaussian Process (SVGP) on the bearing dataset. Among the three, GPR demonstrates the most balanced and reliable performance across the evaluation metrics. It achieves the lowest MAE and RMSE, along with the highest R^2, indicating both superior accuracy and goodness-of-fit. Importantly, GPR also yields the highest monotonicity score, reflecting its ability to produce smoother, more physically consistent degradation trajectories, which is a crucial property for prognostics. Although SVR and SVGP show marginally higher accuracy at the early stages of degradation, their predictions tend to forecast failure later than observed, whereas GPR anticipates failure earlier, which is advantageous for minimizing unplanned downtime. Overall, SVR and SVGP display reduced monotonicity and increased error rates relative to GPR. Given that predictive maintenance applications require not only accurate but also stable and interpretable degradation trends, GPR emerges as the most effective method, offering the best trade-off between prediction accuracy, error minimization, and monotonic behavior.

6.3. Development Efficiency

The framework enables rapid deployment of new component monitors within two to three days, requiring only specific feature extraction overrides while reusing shared signal processing, anomaly detection, and RUL prediction modules. Testing and field deployment are streamlined via standardized interfaces and configuration-driven parameter tuning. The pseudocode in Fig 8, illustrates the development of the Tool Insert and Bearing monitor by leveraging the framework. Table 6 includes different aspects of the development using the framework vs without. These estimates are based on internal development experience, and engineering logs. While they demonstrate indicative benefits of the framework, a controlled user study and external benchmarking are planned as future work to provide empirical validation. The proposed plan is explained in the Appendix.

6.4. Practical Considerations and Business Impact

The IntelliMaint framework's ability to provide accurate and uncertainty-aware RUL predictions translates directly into tangible business benefits, primarily by enabling a shift from reactive to proactive maintenance strategies. Our case studies demonstrate the framework's effectiveness in providing early warnings. For instance, in the CNC tool wear case study, the framework accurately predicted the failure would occur in 2.33 hours with a 90% accuracy - giving manufacturers time to plan and anticipate failures and reduce downtime. While full-scale quantification by the user company is planned for the upcoming pilot deployment, expected benefits of the IntelliMaint framework in CNC tool monitoring based on ongoing conversations with user companies include: unplanned downtime reduction, tool life extension, scrap/rework reduction, and maintenance cost reduction.

7. FUTURE WORK

Future work will include other deep learning methods for health indicator and RUL estimation and explore the incorporation of multi-modal sensor data such as thermal and acoustic signals, and the adoption of hybrid deep learning architectures. Future directions can also encompass the adoption of transfer learning. The framework's modularity makes it well-suited to leverage pre-trained models from similar as-

RUL estimation method	MAE	RMSE	R^2 Score	Monotonicity	Predicted RUL (hours)	Actual RUL (hours)	Accuracy of RUL at Degradation Start %
GPR	0.0288	0.0396	0.8795	0.6264	70.783	74.833	91.51
SVR	0.0777	0.067	0.61	0.5781	78	74.833	95.76
SVGP	0.0117	0.0454	0.82	0.4665	78.843	74.833	94.64

Table 5. Metrics of RUL Estimation Methods for Bearing

```python
# Import IntelliMaint modules
from IntelliMaint.component_template import
    ComponentTemplate
from IntelliMaint.intelliMaint_core import (
    DataAcquisition, SignalSeparation,
    SignalEnhancement,
    TimeDomain, FrequencyDomain, SOM,
    GPRDegradationModel_
)

class ToolInsertMonitor(ComponentTemplate):

    def monitor(self):

        data_loader = ToolInsertDataLoader(self.
    config)
        feature_extractor =
    ToolInsertFeatureExtractor(self.config)
        feature_selector =
    ToolInsertFeatureSelector(self.config)

        files = data_loader.get_file_list()
        data = data_loader.load_file_data()

        # === Feature Extraction ===
        features = feature_extractor.
    extract_features(data)

        # === Feature selection ===
        selected_features = feature_selector.
    calculate_monotonicity(features)

        # === Estimate HI and anomaly detection
    ===
        anom = AnomalyDetection()
        health_indicator = anom.hi_using_euclidean
    (results[features])

        # === RUL Estimation ===
        gpr= EnsembleModel()
        rul_hours = gpr.predict_rul(
    health indicator)
```

```python
# Import IntelliMaint modules
from IntelliMaint.component_template import
    ComponentTemplate
from IntelliMaint.intelliMaint_core import (
    DataAcquisition, SignalSeparation,
    SignalEnhancement,
    TimeDomain, FrequencyDomain, SOM,
    GPRDegradationModel_
)

class BearingMonitor(ComponentTemplate):

    def monitor(self):

        data_loader = BearingDataLoader(self.
    config)
        feature_extractor =
    BearingFeatureExtractor(self.config)
        feature_selector = BearingFeatureSelector(
    self.config)

        files = data_loader.get_file_list()
        data = data_loader.load_file_data()

        # === Feature Extraction ===
        features = feature_extractor.
    extract_features(data)

        # === Feature selection ===
        selected_features = feature_selector.
    calculate_monotonicity(features)

        # === Estimate HI and anomaly detection
    ===
        som = BearingBaselineEstimator()
        health_indicator = som.
    detect_anomalies_onlyMQE(results[features])

        # === RUL Estimation ===
        gpr= BearingGPRDegradationModel()
        rul_hours = gpr.predict_rul(
    health indicator)
```

Figure 8. Left: (a) Tool Insert pseudocode and Right: (b) Bearing pseudocode

sets or public datasets, allowing it to provide reliable predictions even with limited data on a new machine/environment. Cross-asset learning and cloud-native microservices will be investigated to improve scalability and generalizability. Furthermore, adaptive GPR models and physics-informed approaches will be developed to enhance prediction robustness, supported by expanded validation across diverse electromechanical systems.

8. DEPLOYMENT CHALLENGES

To handle large-scale data, the framework can be deployed in a cloud-native microservices architecture. In this setup,

Aspect	Without Framework	With IntelliMaint
Development Time	Order of weeks (Estimated)	2-3 days
Testing Effort	Full custom validation	Template validation
Maintenance	Component-specific	Standardized

Table 6. Tool Insert Engineering Effort Metrics

each module, such as data acquisition, feature extraction, or prognostics assessment, can be run as a separate service. This allows for horizontal scaling, where multiple instances of a service can be spun up to handle increased data loads without requiring a monolithic application to be re-engineered.

9. LIMITATIONS

Despite its promising results, the proposed framework faces several limitations requiring further research. The primary challenge is that the Gaussian Process Regression (GPR) model can be computationally demanding, which restricts its suitability for deployment in resource-limited distributed or edge computing scenarios. If the GPR model's update time takes longer than the data stream's interval, the system will fall behind. This could lead to a backlog of data points and inaccurate or delayed predictions, which is unacceptable for critical systems like those in industrial maintenance. Moreover, integrating the framework with existing industrial ecosystems is nontrivial and requires the development of robust APIs for seamless interfaces with existing systems.

10. CONCLUSION

We propose IntelliMaint, an intelligent, component-agnostic framework with the critical ability to generalize for different components and operating environments and provide uncertainty-aware predictions. These qualities make it uniquely suited for practical industrial deployment across diverse domains. We validate our framework on two distinct industrial applications: (1) Tool insert wear monitoring using vibration and spindle current. Early detection of tool wear with RUL prediction with accuracy of 97%. (2) IMS bearing dataset used for fault detection, earlier than threshold methods, with 95% confidence intervals with RUL accuracy of 91%. Both cases show HI monotonicity and reliable uncertainty quantification.

REFERENCES

Aizpurua, J., Stewart, B., McArthur, S., Penalba, M., Barrenetxea, M., Muxika, E., & Ringwood, J. (2022). Probabilistic forecasting informed failure prognostics framework for improved RUL prediction under uncertainty: A transformer case study. , *226*, 108676. doi: 10.1016/j.ress.2022.108676

Fan, X., Yang, X., Li, X., & Wang, J. (2015). A particle-filtering approach for remaining useful life estimation of wind turbine gearbox. In *International conference on chemical, material and food engineering* (pp. 198–200). Atlantis Press.

Gao, T., Li, Y., Huang, X., & Wang, C. (2021). Data-driven method for predicting remaining useful life of bearing based on bayesian theory. , *21*(1). doi: 10.3390/s21010182

Hong, S., Zhou, Z., Lu, C., Heart, K., & Zhao, T. (2015). Bearing remaining life prediction using gaussian process regression with composite kernel functions. , *17*, 695–704.

IMS bearings. (2007). Retrieved from `https://data.nasa.gov/dataset/ims-bearings`

Kansanaho, J., & Kärkkäinen, T. (2019). *Software framework for tribotronic systems* (No. arXiv:1910.13764). arXiv. doi: 10.48550/arXiv.1910.13764

Kim, S., Kim, N. H., & Choi, J.-H. (2021). A study toward appropriate architecture of system-level prognostics: Physics-based and data-driven approaches. , *9*, 157960–157972. doi: 10.1109/ACCESS.2021.3129516

Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to rul prediction. *Mechanical Systems and Signal Processing*, *104*, 799-834. doi: https://doi.org/10.1016/j.ymssp.2017.11.016

Li, X., Teng, W., Peng, D., Ma, T., Wu, X., & Liu, Y. (2023). Feature fusion model based health indicator construction and self-constraint state-space estimator for remaining useful life prediction of bearings in wind turbines. , *233*, 109124. doi: https://doi.org/10.1016/j.ress.2023.109124

Nie, L., Zhang, L., Xu, S., Cai, W., & Yang, H. (2022). Remaining useful life prediction of milling cutters based on CNN-BiLSTM and attention mechanism. , *14*(11), 2243. doi: 10.3390/sym14112243

Qi, J., Zhu, R., Liu, C., Mauricio, A., & Gryllias, K. (2024). Anomaly detection and multi-step estimation based remaining useful life prediction for rolling element bearings. , *206*, 110910. doi: 10.1016/j.ymssp.2023.110910

Rafik, M.-R.-B. (2019). *GRAND: Group-based anomaly detection for large-scale monitoring of complex systems.* CAISR - Center for Applied In-

telligent Systems Research. Halmstad University. Retrieved from \url{https://github.com/caisr-hh/group-anomaly-detection}

Rombach, K., Michau, G., Bürzle, W., Koller, S., & Fink, O. (2024). Learning informative health indicators through unsupervised contrastive learning. *IEEE Transactions on Reliability*.

Schwendemann, S., Amjad, Z., & Sikora, A. (2021). A survey of machine-learning techniques for condition monitoring and predictive maintenance of bearings in grinding machines. *Computers in Industry*, *125*, 103380.

Teubert, C., Jarvis, K., Corbetta, M., Kulkarni, C., & Daigle, M. (2023). ProgPy: Python packages for prognostics and health management of engineering systems. , *8*(87), 5099. doi: 10.21105/joss.05099

Wang, H., Zhang, X., Guo, X., & Lin, T. (2022). Remaining useful life prediction of bearings based on multiple-feature fusion health indicator and weighted temporal convolution network. , *33*. doi: 10.1088/1361-6501/ac77d9

Wu, F., Wu, Q., Tan, Y., & Xu, X. (2024). Remaining useful life prediction based on deep learning: A survey. , *24*(11), 3454. doi: 10.3390/s24113454

Zhou, H., Huang, X., Wen, G., Lei, Z., Dong, S., Zhang, P., & Chen, X. (2022). Construction of health indicators for condition monitoring of rotating machinery: A review of the research. , *203*, 117297. doi: 10.1016/j.eswa.2022.117297

Zhuang, L., Xu, A., & Wang, X.-L. (2023). A prognostic driven predictive maintenance framework based on bayesian deep learning. , *234*, 109181. doi: https://doi.org/10.1016/j.ress.2023.109181

APPENDIX: PROPOSED PLAN TO MEASURE DEVELOPMENT EFFICIENCY OF FRAMEWORK

To evaluate the practical utility of the IntelliMaint framework, a structured user study will be conducted involving individual developers. The methodology entails recruiting four to five participants and dividing them into two groups: a control group, which will independently develop a predictive maintenance solution from scratch, and an experimental group, which will utilize the IntelliMaint framework for the same task of creating a monitoring application for a machine component. Evaluation will focus on several key metrics: total time from project initiation to deployment of a functional solution; detailed time allocation for data acquisition, feature engineering, model training, and deployment; and the number of lines of code written by each team. This experimental design will provide quantitative insights into the framework's impact on development efficiency and module reusability, facilitating an evidence-based assessment of its strengths and limitations.