From Prediction to Decision: Uncertainty-Aware Maintenance Scheduling for Commercial Turbofan Engines

Gaoyuan Du[1], Ark Ifeanyi[2], and Anahita Khojandi[3]

[1] *Department of Industrial and Systems Engineering, University of Tennessee, Knoxville, TN 37996, USA*
gdu3@vols.utk.edu

[2] *Department of Nuclear Engineering, University of Tennessee, Knoxville, TN 37996, USA*
aifeanyi@alum.utk.edu

[3] *Department of Industrial Engineering, University of Arkansas, Fayetteville, AR 72701, USA*
akhojandi@uark.edu

ABSTRACT

Accurate prediction of remaining flight cycles is the goal of most turbofan prognostics, but low aggregate error does not by itself guarantee low realized maintenance cost. On the PHM North America 2025 Data Challenge (three maintenance targets of very different event frequencies, scored under an asymmetric, time-weighted cost) we build a conventional prediction pipeline and then ask how reliable its accuracy is. We take a strictly nested temporal analysis, in which calibration data are untouched by fitting or tuning, as the primary result. Across two allocations of the limited pre-deployment history, a nominal 90% split-conformal interval achieves 90–93% coverage on the late-life deployment window for the frequent water-wash, but only 25–29% for the high-pressure-turbine (HPT) shop visit and 19–28% for the high-pressure-compressor (HPC) shop visit. Complementary interval and base-model checks, and all four subsets of the public NASA C-MAPSS benchmark, reproduce the separation between water-wash and the two shop-visit targets. Converting predictions into scheduling decisions, a policy whose safety buffer is the conformal half-width reduces realized cost by 23–41% against trusting the point prediction under the two nested protocols, cutting unplanned failures from 31 to 9–13. Its comparison with a tuned constant buffer changes from a 13% advantage to a 3% disadvantage as the scarce pre-deployment data are apportioned, so four engines do not order the two buffer policies. A cost sweep places their crossover at a failure penalty of roughly 250 forfeited cycles of useful life. On HPC, constant-buffer policies are not robust: the original and Nested Variant I buffers leave all six failures, while the much wider Variant II buffer cuts failures to three at substantial early-service cost. A prediction-free age-based fallback and, under the secondary non-nested calibration, two instance-dependent triggers also reduce failures. Per-component reliability must therefore gate the maintenance decision rather than follow it.

Gaoyuan Du et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

https://doi.org/10.36001/IJPHM.2026.v17i2.5107

1. INTRODUCTION

Unplanned maintenance is a leading cause of commercial-aviation disruption, and the gas-path components of a turbofan engine – particularly the high-pressure compressor (HPC) and high-pressure turbine (HPT) – degrade under cyclic thermal and mechanical stress (Saxena, Goebel, Simon, & Eklund, 2008). Accurate prediction of the remaining useful life (RUL) of these components enables airlines to schedule interventions cost-optimally, reducing both unplanned downtime and unnecessary early servicing.

Most prognostics research frames this as a regression problem: minimize the error between predicted and true RUL (Lei et al., 2018; Li, Ding, & Sun, 2018; Zheng, Ristovski, Farahat, & Gupta, 2017). We argue that, for maintenance, a low aggregate prediction error is necessary but not sufficient. Two issues are routinely under-examined. First, prediction reliability is rarely *uniform*: a single model can be well-calibrated for a frequent, strong-signal event and badly miscalibrated for a rare, slow one, yet aggregate metrics hide this. Second, a prediction is only useful insofar as it drives a low-cost *decision* – when to schedule the maintenance – and the cost structure of that decision (an unplanned in-service failure is far worse than an early service) is asymmetric in a way that a symmetric error metric ignores.

We study these issues on the PHM North America 2025

Data Challenge (PHM Society, 2025), which is unusually well suited to them: it poses three maintenance targets of very different frequencies on the same engines, and it scores predictions with an explicitly asymmetric, time-weighted cost. Our contributions are:

1. **A prediction pipeline** combining engineered degradation features, a stacking ensemble, and a custom loss aligned with the competition's asymmetric cost (Section 4).
2. **A per-component reliability diagnosis** using split-conformal prediction intervals, which quantifies how a late-life temporal distribution shift degrades interval coverage unevenly across the three targets (Section 5).
3. **An uncertainty-aware maintenance-scheduling policy** that uses the conformal interval as a safety buffer. Two claims are supported by the evidence and we confine ourselves to them: buffering in proportion to measured unreliability is substantially cheaper than trusting the point prediction, robustly across calibration protocols, and the conformal half-width requires no additional outcome-tuning split beyond the calibration data, since it is a residual quantile rather than a value tuned against realized decision costs. Whether it also beats a well-tuned constant margin is not resolved by four engines, and we do not claim it (Section 6.9). A cost-sensitivity analysis delimits when the approach is worthwhile (Section 6).

Terminology. Two terms recur throughout and are used in a strictly measurable sense. A prediction is *calibrated* for a target if the achieved coverage of its nominal $(1-\alpha)$ prediction interval, evaluated on a strictly later time window, lies within a stated tolerance of $1-\alpha$; Section 5 measures this quantity per component, and Section 5.6 uses a tolerance of 0.10 to route components. One decision is *better* than another if it attains a lower realized maintenance cost J (Eq. 4) on the same evaluation window at a stated failure penalty C_{fail}. Where we call a prediction trustworthy or a decision good, those words abbreviate exactly these two measurable quantities and carry no further connotation.

Among studies derived from this challenge, ours is, to our knowledge, the only one to treat per-component prediction reliability and the downstream scheduling decision – rather than the leaderboard score – as the object of study; the published entries (Pankaj, Joshi, Peng, Teramoto, & Hearn, 2025; Han & Liang, 2025; Inoue, Wakimoto, Ozeki, Kuriyama, & Masuzaki, 2025) all optimize the challenge metric itself. The broader thesis that low prediction error does not imply good downstream decisions is shared with the decision-focused maintenance-optimization literature reviewed in Section 2; our contribution is the per-component

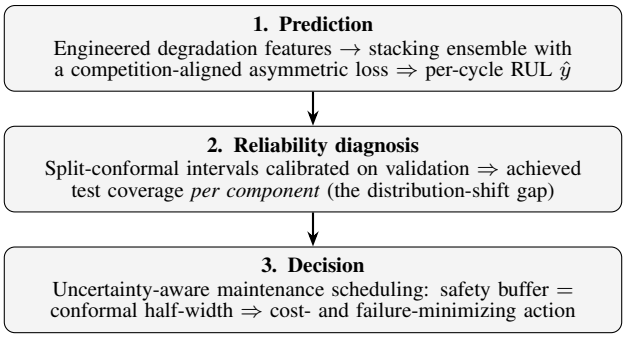


Figure 1. The three-stage prognostics-and-health-management pipeline. Prediction accuracy feeds a per-component reliability diagnosis, which in turn governs an uncertainty-aware maintenance decision. The contribution is stages 2–3; stage 1 is a deliberately simple, reproducible base.

reliability gate that connects a measured conformal-coverage deficit to the choice of decision policy. Where the method fails, we report the failure as a central finding rather than a limitation: on HPC, constant-margin policies are not robust across calibration protocols. The original and Nested Variant I margins leave all six failures, while the much wider Variant II margin is costly and only partly effective; a scheduled fallback and, under the secondary calibration, triggers whose thresholds vary by instance remain viable alternatives. Figure 1 summarizes the three-stage pipeline that organizes the paper. Section 2 reviews related work; Section 3 describes the data and cost structure; Section 4 summarizes the prediction pipeline; Section 5 develops the per-component reliability diagnosis and its robustness checks; Section 6 evaluates the maintenance-scheduling policies; and Sections 7 and 8 discuss practical guidance, limitations, and conclusions.

2. RELATED WORK

2.1. Remaining Useful Life Prediction

RUL prediction for turbofan engines is the canonical prognostics task, anchored by the NASA C-MAPSS benchmark (Saxena et al., 2008; Ramasso & Saxena, 2014). Statistical data-driven approaches were surveyed early by (Si, Wang, Hu, & Zhou, 2011), and the broader prognostics-and-health-management discipline by (Lee et al., 2014; Lei et al., 2018). Deep learning now dominates: convolutional networks (Li et al., 2018; Babu, Zhao, & Li, 2016), long short-term memory networks (Zheng et al., 2017), attention and transformer architectures (Chen et al., 2021), and hybrid or semi-supervised models (Wang, Lei, Li, & Li, 2020; Ellefsen, Bjørlykhaug, Åsøy, Ushakov, & Zhang, 2019) all report strong benchmark accuracy; hybrid physics-and-data approaches have also been advanced in this journal (Kong et al., 2023). O. Fink et al. (2020) review

the potential and open challenges of deep learning in PHM. A complementary line distinguishes inference-based from direct RUL estimation (Ifeanyi, Zanotelli, & Coble, 2024), but the focus remains the prediction itself. Among entries to the same PHM North America 2025 Data Challenge studied here, Pankaj et al. (2025) (first place in the public test phase) combine virtual sensing, LSTM regressors with a competition-adapted loss, and a linear degradation prior; Han and Liang (2025) build interpretable health indices from operating-condition-adjusted measurement residuals; and Inoue et al. (2025) fit a maintenance-cycle model with physics-informed features. All three optimize the challenge score itself. Almost all of this work optimizes an aggregate prediction error and reports a single aggregate metric; per-component reliability and the downstream decision are typically out of scope. Our prediction stage uses gradient-boosted trees (Ke et al., 2017) with automated temporal features (Christ, Braun, Neuffer, & Kempa-Liehr, 2018) and stacking (Wolpert, 1992), not as a contribution but as a deliberately simple, reproducible base on which to study reliability and decisions.

2.2. Maintenance Optimization and Decision Making

Translating a prognostic estimate into a maintenance action is the subject of a large operations-research literature. de Jonge and Scarf (2020) and Olde Keizer, Flapper, and Teunter (2017) review maintenance optimization and condition-based policies for multi-component systems, and Khojandi, Maillart, and Prokopyev (2014) study optimal planning of life-depleting maintenance. Markov decision processes (Puterman, 1994) and, more recently, reinforcement learning (Ong, Niyato, & Yuen, 2020; Compare, Bellani, & Zio, 2017) formalize the when-to-act decision, and decision-making under absent or unreliable ground truth has itself been studied within this journal (W. Fink & Al Hajhog, 2024). Our decision layer connects to this literature but in a deliberately lightweight way: we ask how far a simple uncertainty-aware buffer goes before a full sequential-decision model is warranted, and find (Section 6.7) that under four-engine data the heuristic is in fact the more robust choice.

2.3. Uncertainty Quantification and Distribution Shift

Conformal prediction (Vovk, Gammerman, & Shafer, 2005; Angelopoulos & Bates, 2023) provides distribution-free prediction intervals with finite-sample coverage under exchangeability, and has begun to appear in prognostics (Javanmardi & Hüllermeier, 2023). Its behavior under *covariate and label shift* – precisely our late-life regime – has been studied by Tibshirani, Foygel Barber, Candès, and Ramdas (2019), extended to non-exchangeable data with weighted quantiles by Barber, Candès, Ramdas, and Tibshirani (2023), and addressed adaptively by Gibbs and Candès (2021); dataset shift more broadly is treated by Quiñonero-Candela, Sugiyama, Schwaighofer, and Lawrence (2009), and domain adaptation for RUL specifically by de Oliveira da Costa, Akcay, Zhang, and Kaymak (2020). We use conformal intervals not to claim robustness to shift but to *measure* it per component, turning the coverage gap into a diagnostic that drives the decision. Calibration of probabilistic forecasts in the classical sense is due to DeGroot and Fienberg (1983).

Table 1. Dataset variables. Sensor units follow the challenge documentation; °R denotes degrees Rankine.

Variable	Description	Unit
ESN	Engine serial number	–
Cycles_Since_New	Engine age	cycles
Snapshot	Flight-condition index (1–8)	–
Cumulative_⟨event⟩	Event counts (×3)	–
Sensed_Altitude	Flight altitude	ft
Sensed_Mach	Mach number	–
Pamb, Pt2, Ps3, P25	Pressures (ambient/inlet/HPC/LPC)	psia
TAT, T25, T3, T45, T5	Air-path temperatures	°R
Sensed_WFuel	Fuel flow rate	–
Sensed_VAFN, VBV	Nozzle / bleed-valve positions	–
Fan_Speed, Core_Speed	Spool speeds	rpm
Cycles_to_⟨event⟩	Prediction targets (×3)	cycles

3. DATA AND PROBLEM SETTING

The dataset (PHM Society, 2025) contains time-series sensor readings and maintenance histories for four commercial turbofan engines (ESN 101–104), with 59,702 records covering 0–20,000 flight cycles and up to eight snapshots per cycle representing distinct flight phases. Sixteen raw sensors (pressures, temperatures, fuel flow, spool speeds, valve positions, ambient conditions) are recorded, along with cumulative counts of three maintenance events; Table 1 lists all variables.

The three prediction targets are cycles to the next water-wash (WW; mean 492 cycles, ∼20 events per engine), HPT shop visit (mean 1,681; ∼6 events), and HPC shop visit (mean 3,814; ∼2 events). The targets therefore differ by an order of magnitude in both horizon and event frequency, which is what makes per-component reliability analysis meaningful. Each target follows a characteristic sawtooth degradation profile – the remaining-cycle count decreases approximately linearly until a maintenance event resets it (Figure 2) – with water-wash resetting roughly twenty times per engine life, HPT about six, and HPC only twice.

3.1. Asymmetric Cost Metric

The challenge scores predictions with an asymmetric, time-weighted squared error. For a single prediction,

$$\text{score} = w \cdot (\hat{y} - y)^2, \quad w = \frac{\kappa}{1 + \lambda\, y}, \tag{1}$$

where $\kappa = 2$ if $\hat{y} \geq y$ (a *late* prediction, overestimating remaining life) and $\kappa = 1$ otherwise (*early*), and $\lambda = 0.01$

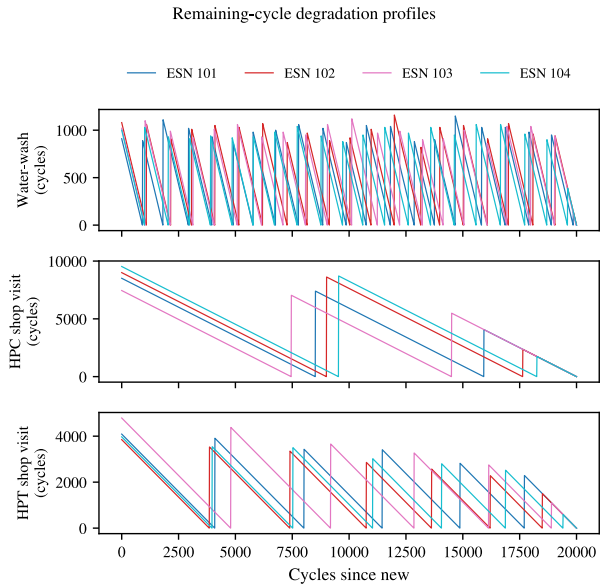


Figure 2. Remaining-cycle degradation profiles by engine. Each target decreases linearly until its maintenance event resets it; the rare HPC reset (twice per engine) leaves few complete degrade-to-event cycles to learn from, foreshadowing its poor reliability under shift.

(we reserve α for the conformal miscoverage level of Section 5). Two asymmetries are encoded: late predictions are penalized twice as heavily as early ones, and errors near the event ($y \to 0$) are weighted more heavily. Both reflect the operational reality that failing to anticipate imminent maintenance is more consequential than being conservative.

3.2. Temporal Splitting

To respect causality, each engine's timeline is split chronologically into train (cycles 0–11,990; 60%), validation (12,000–15,990; 20%), and test (16,000–20,000; 20%). All four engines contribute to each split. The test window is therefore the *late-life* regime of every engine, which, as we show, exhibits degradation patterns not fully represented in training – a realistic and consequential distribution shift (Quiñonero-Candela et al., 2009).

4. PREDICTION PIPELINE

We summarize the prediction stage compactly; it is a means to the reliability and decision analyses that follow, not the contribution.

4.1. Preprocessing

Raw sensor channels contain physically implausible values from transient faults; we cap eight channels at engineering thresholds (e.g. core speed $> 30{,}000$ rpm, $T_3 > 3{,}000\,^\circ$R), setting violations to missing. Missing values (4,003 cells, 0.42% of all sensor cells, at most 1.34% within any single channel) occur in paired channel-dropout patterns (e.g. T_{45} with T_5), indicating sensor-channel failures rather than random loss; we impute by forward- then backward-fill within each (engine, snapshot) group.

Because an online implementation can use only past observations, we state the causal status of every preprocessing and feature-construction step explicitly. Two steps are *not* strictly causal. First, imputation was applied to the complete record before the temporal split rather than independently within each split, and its backward-fill component can fill an earlier gap with a later measurement; a global median absorbed the residual gaps that neither fill direction reached. Both touch only the 0.42% of sensor cells that were missing. Section 5.8 quantifies their influence with a strictly causal alternative: forward fill only, residual gaps filled with the training-window median, and every imputation statistic computed on the training split alone. All remaining steps are strictly causal. Cycle-to-cycle deltas use a backward difference; the multi-scale rolling means ($5/20/50$ cycles) and rolling standard deviations ($10/20$ cycles) use trailing windows only; each tsfresh feature is extracted from the sub-series consisting of the ten cycles up to and including the current cycle; and the cycles-since-last-maintenance counters are derived only from past increments of the cumulative event counts. Feature selection used the first 60% of each engine's timeline, the training window, so no validation or test observation entered it.

4.2. Features

From the 16 raw sensors we engineer 256 candidate features in six families and select 229 by gradient-boosting importance: (i) *per-cycle aggregation* (mean/std/min/max of each sensor across the up-to-eight within-cycle snapshots, compressing flight-phase variation); (ii) *cycle-to-cycle deltas* (degradation rate); (iii) *multi-scale rolling means* over 5/20/50-cycle windows (slow trends relevant to the long-horizon HPC target); (iv) *rolling volatility* (10/20-cycle standard deviations, which often rise before failure); (v) automated tsfresh (Christ et al., 2018) statistics over a 10-cycle window; and (vi) *cycles-since-last-maintenance* counters, derived by detecting increments of the cumulative event counts. A drop-one-category ablation confirms the last family is decisive: removing the three cycles-since-last features raises the validation cost by more than an order of magnitude (the HPC coefficient of determination for the single gradient-boosting model used in this ablation falls from 0.78 to -1.11), while no other family changes the cost by more than a few percent – several even improve slightly when removed, indicating redundancy. SHAP (Lundberg & Lee, 2017) attributions corroborate this (Figure 3): the cycles-since-last feature is the top-ranked predictor for all three targets, and engineered features occupy 39 of the 45 slots across the three top-15 lists.

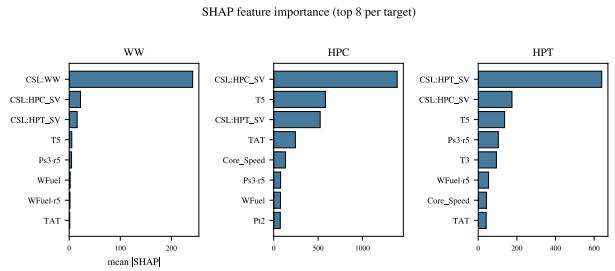


Figure 3. Top features by mean absolute SHAP value for each target. The engineered cycles-since-last-maintenance counter dominates every target, confirming that the informative signal is the time elapsed since the last event rather than any single raw sensor channel.

4.3. Models and Loss

Four base learners (ridge regression, gradient boosting, random forest, and LightGBM (Ke et al., 2017)) are trained per target and combined by a ridge meta-learner (Wolpert, 1992). Hyperparameters are tuned per target with Optuna (Akiba, Sano, Yanase, Ohta, & Koyama, 2019), and SHAP (Lundberg & Lee, 2017) attributions confirm that cycles-since-last-maintenance is the top predictor for all three targets. On the validation window the stacking ensemble reaches a competition score of 2.3, two orders of magnitude below a predict-the-training-mean baseline (score 276) and $4.4\times$ below a single gradient-boosting model trained with symmetric squared error (score 10.2; the "MSE" row of Table 2). The result is stable under retraining: re-running the full base-learner/stacking/calibration pipeline over five random seeds yields a validation score of 2.0 ± 0.2 (mean $\pm$ standard deviation), with per-target validation R^2 varying by less than 0.01. On the late-life test window the same predictions score 310 – a two-orders-of-magnitude degradation that previews the transfer failure dissected in Section 4.4. These scores are computed on our own temporal validation split of the labeled training engines and are not comparable to official leaderboard scores on the challenge's hidden test set (e.g., Pankaj et al., 2025), which use a different normalization and evaluation population.

4.3.1. Competition-aligned training objective

Because the evaluation cost (Eq. 1) is asymmetric, training LightGBM with symmetric squared error misaligns the optimization target with how predictions are scored. We therefore derive a custom objective directly from the cost. With per-sample loss $\ell(\hat{y}, y) = w(\hat{y}, y)\,(\hat{y} - y)^2$, where w is the asymmetric time-weight of Eq. 1, the gradient and Hessian required by the second-order boosting routine are, treating w as locally constant at the current prediction,

$$g = \frac{\partial \ell}{\partial \hat{y}} = 2\,w(\hat{y}, y)\,(\hat{y} - y), \qquad h = \frac{\partial^2 \ell}{\partial \hat{y}^2} = 2\,w(\hat{y}, y). \tag{2}$$

Since $w > 0$, the Hessian is strictly positive and each Newton-style leaf update is well posed. The target-level normalization in Eq. 1 is a constant multiplier on (g, h) that only rescales the effective learning rate and is dropped from training.

Table 2. Loss-function ablation on validation (lower competition score is better). Asymmetry alone hurts; the time-weighting is what aligns training with the asymmetric evaluation cost.

Training loss	Comp. score	Late %	Bias
MSE (symmetric)	10.2	79%	+304
Asymmetric only	11.6	74%	+311
Asymmetric + time-weighted	**10.0**	**70%**	+267

4.3.2. Which part of the cost matters?

The competition cost couples two effects – a $2\times$ *asymmetry* (late worse than early) and a *time-weighting* that emphasizes errors near the event. To see which drives the benefit, we ablate the training loss across three forms (Table 2). The result is informative: the asymmetry-only loss is actually *worse* than plain MSE (competition score 11.6 vs. 10.2), over-correcting the bias on some targets while under-correcting others; only when the time-weighting is added does the loss beat MSE (10.0) and cut the late-prediction rate from 79% to 70%. The *time-weighting*, not the asymmetry, is the component that aligns training with the cost – a finding we carry directly into the decision cost model of Section 6, where the same near-event emphasis governs the safety buffer. Two caveats on Table 2: the aligned loss improves the single-model score only modestly over plain MSE (10.0 vs. 10.2; the headline 2.3 comes from stacking and calibration, not the loss); and even the best loss retains a $+267$-cycle mean bias – a persistent tendency to over-predict remaining life that foreshadows exactly the late-trigger failure mode dissected in Section 6.8.

4.4. The Catch: Validation Accuracy Does Not Transfer

The same pipeline degrades sharply on the late-life test window (Table 3; Figure 4 shows the parity plots). Water-wash holds up reasonably (R^2 falling from 0.91 to 0.73), but HPT falls to $R^2 = 0.18$ with its MAE more than quadrupling, and HPC collapses to $R^2 = -6.5$ – worse than a constant mean predictor – with a test MAE nearly ten times its validation value. This is not a tuning failure but a distribution shift compounded, for HPC, by data scarcity: the test window is the late-life regime of every engine, under-represented in training, and the rare HPC event (only $\sim$2 per engine) gives the

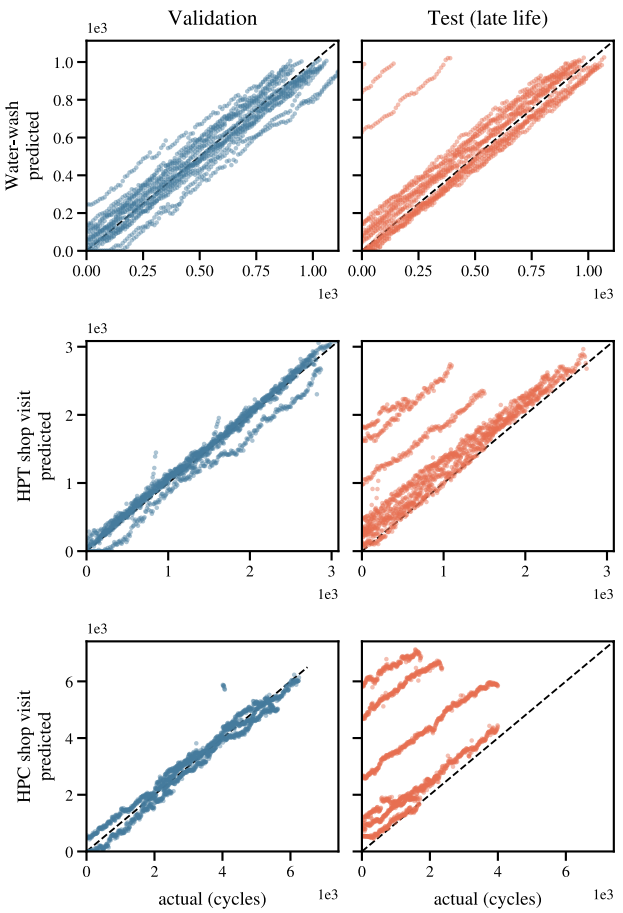


Figure 4. Predicted versus actual remaining cycles on validation (left, blue) and the late-life test window (right, orange), one row per target; the dashed line is the identity. Validation predictions track the diagonal for all three targets; on test, water-wash remains near-diagonal while HPT and HPC drift above it (over-prediction of remaining life), the failure mode quantified in Table 3.

model too few complete degrade-to-event cycles to learn – with two events per engine, shift and scarcity cannot be separated in this dataset, and we do not attempt to. The degradation is ordered exactly by event frequency, and it is why a per-component reliability analysis is needed. The ordering also contains the paper's thesis in miniature: water-wash has the *lowest* validation R^2 of the three targets yet transfers best, while HPC's near-perfect validation fit (0.97) collapses hardest – in-distribution accuracy simply does not rank out-of-distribution calibration. Rather than report this only as a limitation, we make it the object of study: *which* predictions remain calibrated under this shift, and how should a decision maker act on that knowledge?

Table 3. Stacking-pipeline prediction accuracy on validation versus the late-life test window (the same predictions consumed by the reliability and decision analyses of Sections 5–6). Accuracy transfers for the frequent water-wash event but degrades sharply for HPT and collapses ($R^2 < 0$) for HPC. The validation column is in-sample for two stages of the pipeline, since the meta-learner and the bias correction are both fit on that window (Section 5.1), and is therefore optimistic; Table 7 gives out-of-sample figures from a strictly nested design.

	Validation		Test	
Target	MAE	R^2	MAE	R^2
Water-wash	68.4	0.91	84.4	0.73
HPT shop visit	93.4	0.98	421.7	0.18
HPC shop visit	224.3	0.97	2147.6	−6.5

5. PER-COMPONENT RELIABILITY UNDER DISTRIBUTION SHIFT

To quantify calibration per target we use split-conformal prediction intervals (Vovk et al., 2005; Angelopoulos & Bates, 2023), which provide finite-sample coverage guarantees under exchangeability and – crucially – let us *measure* the loss of coverage when exchangeability is broken by shift. The strictly nested protocol of Section 5.6, whose calibration window is untouched by fitting and tuning, is the primary reliability analysis. Sections 5.1–5.3 first retain the original 60/20/20 analysis as a secondary diagnostic so that its predictions can be compared with the submitted manuscript and propagated through the policy-construction ablations. That original calibration window is partially in-sample for the final predictor; accordingly, it is neither independent nor fully held out for the complete pipeline.

5.1. Secondary 60/20/20 Calibration Protocol

For target t, let $\hat{y}(x)$ be the stacking prediction at observation x. We form the absolute-residual nonconformity score $s(x, y) = |y - \hat{y}(x)|$ and evaluate it on the n *validation* observations $\{(x_i, y_i)\}_{i=1}^{n}$. This secondary analysis uses the validation window as its calibration set. Although the four base learners were fit on the training window alone, the complete predictor was not independent of validation. Two qualifications on this choice must be stated explicitly, because they weaken the usual split-conformal justification. First, the validation labels enter model selection: the LightGBM hyperparameters were chosen by 50 Optuna trials scored on the validation competition score (Appendix, Table 18). Second, and more consequentially, two stages of the predictor are themselves fit on the validation window: the ridge meta-learner that combines the four base predictions, and the per-bin bias correction applied to its output (Section 4). The calibration residuals of the complete pipeline are therefore in-sample for those two stages, and the resulting half-widths are narrower than a fully out-of-sample calibration would give. We re-

port the analysis in this form because it is the pipeline whose predictions the decision layer consumes, and we quantify the consequence directly: Section 5.6 repeats the entire reliability analysis under a strictly nested temporal design in which the calibration window is untouched by fitting, by tuning, and by the meta-learner and bias correction alike. The validation column of Table 3 should be read with the same caveat. Accordingly, the values from this protocol are descriptive secondary results, not the paper's primary coverage estimates. For target miscoverage α, the conformal radius is the rank-adjusted empirical quantile

$$q_\alpha = \mathrm{Quantile}\left(\{s_i\}_{i=1}^{n};\ \frac{\lceil (n+1)(1-\alpha) \rceil}{n} \right), \qquad (3)$$

and the prediction interval at a new point is $C_\alpha(x) = [\max(0, \hat{y}(x) - q_\alpha),\ \hat{y}(x) + q_\alpha]$, clipped because RUL is non-negative. Under exchangeability of calibration and test residuals, C_α satisfies the finite-sample guarantee $\Pr(y \in C_\alpha(x)) \geq 1 - \alpha$ (Vovk et al., 2005). The late-life test window breaks exchangeability; we therefore report the *achieved* test coverage $\widehat{\mathrm{cov}} = \frac{1}{m} \sum_j \mathbf{1}[y_j \in C_\alpha(x_j)]$, and treat the gap $(1 - \alpha) - \widehat{\mathrm{cov}}$ as a direct, per-component measure of the shift's severity. This inverts the usual use of conformal prediction: rather than assuming the guarantee holds, we use its violation as a diagnostic.

5.2. Secondary 60/20/20 Result

Table 4 reports achieved test coverage at four nominal levels (90%, 85%, 80%, 75%), and Figure 6 shows the per-instance intervals with test points sorted by true RUL. The tiering is stable across every level: water-wash tracks its nominal target within two points throughout, while HPT and HPC undershoot at all four levels. The three targets separate cleanly into reliability tiers. Water-wash intervals are well-calibrated under shift (95% achieved at nominal 90%); HPT is degraded (47%); and HPC effectively collapses (17%), with a test mean absolute residual (∼2,150 cycles) nearly ten times its validation value (∼220). The order matches event frequency: the more often an event recurs within an engine's life, the more its late-life behavior resembles training, and the better the intervals transfer. Two caveats on interpretation: the conformal guarantee – and hence the achieved coverage we report – is *marginal*, averaged over the test window, so even the well-calibrated water-wash intervals carry no validity claim for any specific RUL region (cf. Javanmardi & Hüllermeier, 2023); and the constant half-width per target means adaptivity across the horizon comes only from the normalized variant of Section 5.3.

This is the paper's key observation: a single model that looks accurate in aggregate is in fact calibrated for one component, marginal for a second, and uncalibrated for a third. A maintenance policy that ignores this – applying the same logic to all three – inherits the worst component's unreliability.

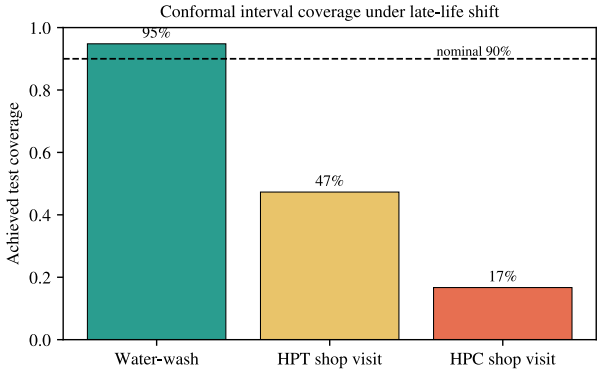


Figure 5. Secondary 60/20/20 analysis: achieved test coverage of a nominal 90% conformal interval, calibrated on validation but partially in-sample for the meta-learner and bias-correction stages. Coverage is retained for the frequent water-wash event but collapses for the rarer HPT and HPC shop visits under the late-life shift; Table 7 gives the primary, strictly nested estimates.

Table 4. Secondary 60/20/20 analysis: achieved coverage on the late-life test window (n = 1,604 snapshots per target) for conformal intervals calibrated on the partially in-sample validation window. Wilson 95% confidence intervals for the nominal-90% rows are $[0.94, 0.96]$ (WW), $[0.45, 0.50]$ (HPT), and $[0.15, 0.19]$ (HPC); snapshots cluster within four engines, so these understate engine-level variation (Section 5.5). A reliable target retains coverage near nominal; an unreliable one loses it. The last column gives the range of achieved coverage on the same deployment window under the two strictly nested calibration protocols of Section 5.6, in which no fitting or tuning step has seen the calibration data; the separation between water-wash and the two shop-visit targets is unchanged.

Target	Nom.	Achieved	Half-width	Nested
Water-wash	90%	**95%**	151	**90–93%**
Water-wash	85%	89%	126	84–89%
Water-wash	80%	83%	110	76–85%
Water-wash	75%	78%	96	51–83%
HPT shop visit	90%	47%	232	25–29%
HPT shop visit	85%	35%	168	20–27%
HPT shop visit	80%	28%	137	14–23%
HPT shop visit	75%	23%	115	10–19%
HPC shop visit	90%	17%	393	19–28%
HPC shop visit	85%	15%	365	18–25%
HPC shop visit	80%	13%	340	17–23%
HPC shop visit	75%	12%	318	16–22%

5.3. The Collapse Is a Property of the Shift, Not the Interval Method

A natural objection is that the coverage collapse might be an artifact of split-conformal's distribution-free construction, and that a different uncertainty-quantification (UQ) method would behave better under shift. We rule this out by repeating the analysis with four alternatives, all calibrated on validation: a parametric Gaussian interval ($\hat{y} \pm z_{1-\alpha/2}\, \hat{\sigma}_{\mathrm{res}}$,

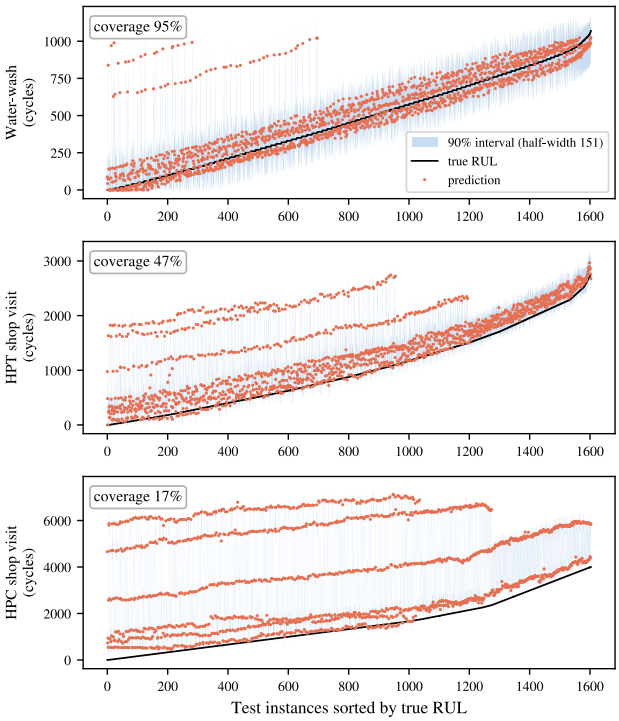


Figure 6. Per-instance 90% conformal intervals in the secondary 60/20/20 analysis on the late-life test window (shaded band; constant half-width per target, calibrated on the partially in-sample validation window), with test instances sorted by true RUL. For water-wash the band tracks the truth; for HPT and especially HPC the predictions (orange) sit systematically *above* the true RUL at low RUL. The constant half-width used here does not compensate for that over-prediction; instance-dependent triggers are examined in Section 6.6.

assuming homoscedastic Gaussian residuals); a normalized (adaptive) conformal interval whose width scales with a difficulty proxy $\sigma(x) = a + b\,\hat{y}$ fit on validation, so that long-horizon predictions receive wider intervals; conformalized quantile regression (CQR) (Romano, Patterson, & Candès, 2019) on gradient-boosted $5\%/95\%$ quantile models; and – most pointedly, since it is designed for exactly the non-exchangeable regime we diagnose – the weighted split-conformal method of Barber et al. (2023) (nex-SCP) with fixed time-decay weights $w_i = 0.99^{\Delta t_i}$, which discounts older calibration residuals and, unlike online adaptive schemes, requires no post-deployment labels. Table 5 shows that all five methods preserve the same reliability ordering: water-wash retains 92–96% coverage, HPT never exceeds 47%, and HPC never exceeds 26%. Even nex-SCP recovers neither HPT nor HPC: down-weighting stale residuals cannot help when the informative late-life residuals do not exist in the calibration window at any weight. The collapse is therefore a property of the late-life distribution shift, not of the particular interval construction. The parametric-Gaussian entry is not a population interval at all but the 90% interval

Table 5. Achieved test coverage of a nominal 90% interval under five UQ methods, all calibrated on validation. The reliable/unreliable separation is invariant to the method; in particular, the non-exchangeable weighted conformal method (nex-SCP; Barber et al., 2023) does not recover coverage on the shifted targets.

Method	WW	HPT	HPC
Split-conformal	0.95	0.47	0.17
Parametric Gaussian	0.93	0.44	0.22
Normalized conformal	0.94	0.47	0.17
CQR	0.96	0.34	0.24
nex-SCP	0.92	0.39	0.26

of a full predictive density, and it fails in the same pattern; Section 5.7 pursues that direction with two constructions that return a predictive distribution proper. The conclusion is also not specific to the stacking ensemble: refitting split-conformal intervals around three heterogeneous base predictors trained from scratch (a single gradient-boosted model, a ridge linear model, and a feed-forward neural network) reproduces the same separation between the reliable and unreliable targets, with water-wash at 82–96% coverage while neither shop-visit target ever exceeds 43%.

A subtler objection targets the feature representation rather than the model: the dominant cycles-since-last-maintenance counter takes its largest values in the late-life window, and tree ensembles cannot extrapolate beyond the training range of that counter, so the collapse might be a feature-design artifact. Two ablations address this. Removing the entire counter family (cycles-since-last, cumulative counts, and engine age) and refitting raises HPC coverage to 0.67 – but only because the calibration residuals explode: the resulting half-width is $2{,}346$ cycles, 61% of HPC's mean horizon, an interval too wide to schedule against. And the ridge learner, which extrapolates linearly in the counter, still collapses on the shop-visit targets (0.39 HPT, 0.24 HPC with the full feature set). Neither removing the non-extrapolating features nor using an extrapolating model recovers a *usable* calibrated interval; the representation choice moves probability mass between miscoverage and vacuous width, which is precisely the trade-off a genuine information deficit forces. We use split-conformal in the decision layer for its distribution-free calibration guarantee on the in-distribution regime.

5.4. Why Coverage Must Degrade under Late-Life Shift

That no interval method recovers HPC's coverage is not a coincidence but a consequence of how conformal validity is established. The finite-sample guarantee $\Pr(y \in C_\alpha(x)) \geq 1 - \alpha$ requires the test nonconformity score $s(x_{\text{test}}, y_{\text{test}})$ to be *exchangeable* with the calibration scores $\{s_i\}$ (Vovk et al., 2005). Our calibration set is the validation window (cycles 12,000–15,990) and the test set is the strictly later window (16,000–20,000); because

every engine degrades monotonically between maintenance events, the test residuals are drawn from a later point on the degradation trajectory than any calibration residual. Exchangeability is therefore *structurally* violated: the two residual populations are ordered in time, not interchangeable. Under such a shift the coverage guarantee weakens to $\Pr(y \in C_\alpha(x)) \geq 1 - \alpha - \mathrm{TV}(P_{\text{cal}}, P_{\text{test}})$, where the total-variation term grows with the residual-distribution shift between the two windows (Barber et al., 2023). This predicts exactly the ordering we observe: water-wash recurs $\sim$20 times per engine, so its late-life residual distribution closely resembles the calibration window and the TV term is small (coverage retained); HPT recurs $\sim$6 times and HPC only $\sim$2, so their late-life residuals are increasingly out-of-distribution and the TV term – hence the coverage loss – grows accordingly. The coverage collapse is thus a measurable signature of how far each component's late-life behavior has drifted beyond what the calibration data can certify. Adaptive conformal schemes (Gibbs & Candès, 2021) that re-estimate the quantile online can partially track such drift, but they require a stream of post-deployment labels that the once-per-life cadence of shop visits does not provide – which is why we treat the coverage gap as a diagnostic to be respected by the decision layer rather than a defect to be corrected.

5.5. Robustness of the Diagnosis

Because the analysis rests on four engines, we probe its stability in three ways. First, *sampling uncertainty*: with $n = 1{,}604$ test snapshots per target, the snapshot-level Wilson 95% confidence intervals on the achieved coverages are $[0.94, 0.96]$ for water-wash, $[0.45, 0.50]$ for HPT, and $[0.15, 0.19]$ for HPC. These intervals treat autocorrelated snapshots as independent and therefore overstate precision; the engine is the appropriate unit of variation. Second, *engine dependence*: recalibrating on three engines and evaluating on the held-out fourth (four leave-one-engine-out folds) preserves the *ordering* in every fold – water-wash coverage spans 0.79–0.99, HPT 0.20–0.78, and HPC never exceeds 0.37 – but the water-wash and HPT ranges overlap at this level, so the tier *boundaries* are not sharply resolved by four engines even though water-wash is always the best-covered target and HPC always at or near the bottom. The diagnosis is likewise insensitive to the calibration-set size: recomputing the nominal-90% coverages from random 50% and 25% subsamples of the calibration window (20 repetitions each) shifts every achieved coverage by less than 0.03.

Third, and most importantly, *external replication*: the mechanism predicts that any run-to-failure dataset should exhibit the same late-life coverage loss, because the approach to failure is by construction the regime least represented earlier in life. We test this on the public NASA C-MAPSS benchmark (Saxena et al., 2008), applying the identical protocol to all four subsets (FD001–FD004; 100–260 units each): a gradient-boosted model trained on the first 60% of each unit's life, conformal calibration on the 60–80% window, evaluation on the final 20% (late-life), with the standard 14 informative sensors and rectified RUL labels capped at 125 (Li et al., 2018; Javanmardi & Hüllermeier, 2023). Nominal 90% intervals achieve only 18–52% late-life coverage across the four subsets (Table 6), while an in-distribution control – calibrating and evaluating on a random unit-stratified split of the same mid-life window – retains 84–92%. The collapse also scales with operating-regime complexity (six-condition FD002/FD004 lose more than single-condition FD001/FD003), mirroring the frequency ordering we observe across components. The phenomenon is therefore not an artifact of our four-engine dataset. We scope the claim carefully: what replicates is the failure of *within-unit chronological* calibration – the protocol forced whenever no fleet history of completed run-to-failure trajectories exists. C-MAPSS, with 100–260 run-to-failure units per subset, lets us test the mature-fleet alternative directly: calibrating on the *late-life* residuals of half the units ("historical engines that already ran to failure") and evaluating on the late-life of the other half restores coverage to 0.90 on all four subsets (per-fold values 0.88–0.92), at the cost of 1.3–1.8$\times$ wider intervals (Table 6, last column). The collapse is thus a *protocol* property, not an intrinsic one: what breaks coverage is calibrating without any late-life examples, and fleet history at matched life stage repairs it. This sharpens the practical reading of our four-engine result – the challenge fleet is the young-fleet regime where no such history exists, which is precisely when the diagnostic is needed as a gate. We also note that both datasets are simulator-generated (C-MAPSS by design; the challenge data's provenance is not disclosed by the organizers), so field validation on operational fleet data remains open.

Table 6. External replication on NASA C-MAPSS: achieved coverage of nominal 90% split-conformal intervals on the final 20% of each unit's life (late-life). Within-unit chronological calibration (the young-fleet protocol, as in the secondary 60/20/20 analysis) collapses in all four subsets, while an in-distribution control on the mid-life window stays near nominal. Cross-unit calibration – using late-life residuals from half the fleet's completed run-to-failure histories to calibrate the other half (two folds averaged) – restores exactly nominal coverage, identifying the collapse as a protocol property.

Subset	Units	Snapshots	Within-unit	Control	Cross-unit
FD001	100	4,163	0.52	0.84	**0.90**
FD002	260	10,854	0.29	0.92	**0.90**
FD003	100	4,985	0.49	0.85	**0.90**
FD004	249	12,346	0.18	0.86	**0.90**

5.6. Primary Strictly Nested Calibration and a Prospective Reliability Gate

Two objections to the analysis so far concern the calibration window rather than the phenomenon. First, as Section 5.1 states, the validation window is not independent of model selection: it scores the Optuna search, and it is the window on which the meta-learner and the bias correction are fit, so the calibration residuals are in-sample for those two stages. Second, the gate of Section 6.8 is demonstrated after the fact: achieved coverage on the test window identifies HPC as unreliable, and the fallback is then evaluated on that same window. We address both with one strictly nested temporal design. Each engine's timeline is cut into five disjoint, time-ordered windows: base-model fitting uses window A; model selection, the ridge meta-learner and the per-bin bias correction use window B; split-conformal calibration uses window C, which no fitting or tuning step has seen; window D supplies the *gate* measurement; and window E is the deployment window, identical to the test window used throughout the paper ($n = 1{,}604$). Every quantity the gate consumes therefore precedes deployment.

Because the four engines afford only so much history, any such design must apportion the 60–80% range between model selection, calibration and the gate, and the apportionment is itself a choice. We therefore ran two variants and report both. Variant I fits on A=0–60%, selects and fits the remaining predictor stages on B=60–70%, calibrates on C=70–75%, measures the gate on D=75–80%, and deploys on E=80–100%. It holds the fitting window at the original analysis's 60%, the same $4{,}800$ cycle-level rows, and leaves 5% each for calibration and the gate. Variant II fits on A=0–40%, uses B=40–55%, C=55–70%, D=70–80%, and the same E=80–100%; it gives calibration and the gate more data (15% and 10%) by shortening the fitting window to 40%. Neither dominates: Variant I matches the original 60/20/20 analysis's training set but estimates the conformal quantile from 400 observations, and Variant II estimates it from $1{,}200$ but trains on a third less data.

Table 7 reports both. On the untouched calibration window the achieved coverage is 0.902–0.905 for every target in both variants; because this same window determines the conformal quantile, that coverage is largely a consequence of the construction, not independent evidence of coverage validity. The nested design is established by the separation of the fitting, tuning and calibration windows, and windows D and E provide the subsequent coverage assessments. The reliability separation then reproduces on the deployment window in both: water-wash retains 0.90 or better, while both shop-visit targets collapse below 0.29. Under Variant I, which keeps the training set of the original analysis, the ordering by event frequency also reproduces ($0.902 > 0.286 > 0.190$ for water-wash, HPT and HPC); under Variant II the two shop-visit targets swap, which we read as the ordering between them not being resolved once the predictor is weakened rather than as evidence against the frequency explanation. The out-of-sample half-widths are in both cases wider than those of Table 4: for HPT, 350 and 457 cycles against 232. That is the direct, quantified consequence of the in-sample calibration noted in Section 5.1.

The gate itself is applied with a tolerance fixed in advance at 0.10: a target is driven by predictions if its coverage on window D is at least $1 - \alpha - 0.10 = 0.80$, and is otherwise routed to the prediction-free age-based policy. Its behaviour is the same in both variants. Water-wash passes (0.963 and 0.974) and deploys at 0.902 and 0.926, so the gate is right; HPC fails (0.507 and 0.432) and is routed to the fallback, which is also right, and on window E that fallback removes all six of its unplanned failures where the naive trigger incurs six. HPT, however, measures 0.840 and 0.816 at the gate and is therefore trusted, yet deploys at 0.286 and 0.248: the gate misses it in both variants. We report this rather than adjust the tolerance after seeing window E, since tuning the threshold on the deployment outcome would reintroduce exactly the retrospective reasoning this experiment is meant to remove.

Because the level of coverage in a single window is only one possible gate statistic, and because distribution shift is progressive, we also asked whether a statistic that looks for *deterioration* would catch HPT. Having found the level rule insufficient, we specified a trend rule on quantities that still precede deployment. We state the order of events plainly, since the idea came from that failure: the pre-deployment windows C and D are cut into four equal blocks in cycle order, achieved coverage is computed in each with the half-width fixed from window C, and a target is flagged when the ordinary-least-squares slope of coverage against block index is significantly negative at the 10% level. The rule caught HPC and, again, did not catch HPT. Water-wash is flat ($1.00, 0.81, 0.93, 1.00$; slope $+0.012$, $p = 0.84$) and is trusted, correctly. HPC deteriorates visibly before deployment ($0.95, 0.86, 0.52, 0.50$; slope -0.170, $p = 0.06$) and is routed to the fallback, correctly and now for a second independent reason. HPT, however, is flat across the same blocks ($0.86, 0.95, 0.81, 0.87$; slope -0.011, $p = 0.76$) and then collapses to 0.286 on the deployment window.

The negative result tells us more than a successful gate would have. HPT's loss of coverage is not progressive within the observable window; it is abrupt at the deployment boundary, so no statistic computed on windows B–D can anticipate it, and the miss is a limit on what the data contain rather than a defect of one gate rule. The mechanism is visible in Table 7: HPT's accuracy does decay gradually across the pre-deployment windows, but its half-width of 350 cycles is wide enough to absorb the growing errors, so coverage holds until it breaks. Two independent prospective statistics, a level

Table 7. Strictly nested temporal design, in two variants that apportion the 60–80% range differently. Base learners are fit on window A, all model selection (Optuna, meta-learner, bias correction) on window B, conformal calibration on window C, the gate measurement on window D, and deployment on window E, which is the test window used elsewhere in the paper. Coverage is of a nominal 90% interval. In both variants the calibration window attains nominal coverage, which is largely a consequence of the construction because that window determines the conformal quantile; windows D and E provide the coverage assessments, and the deployment window reproduces the collapse on both shop-visit targets.

Target	Half-width	Achieved coverage C (cal.)	D (gate)	E (deploy)	Gate decision
Variant I: fit 0–60%, select 60–70%, calibrate 70–75%, gate 75–80%					
Water-wash	382.8	0.905	0.963	**0.902**	predict
HPT s.v.	349.6	0.905	0.840	0.286	predict†
HPC s.v.	492.9	0.905	0.507	0.190	fallback
Variant II: fit 0–40%, select 40–55%, calibrate 55–70%, gate 70–80%					
Water-wash	172.3	0.902	0.974	**0.926**	predict
HPT s.v.	457.2	0.902	0.816	0.248	predict†
HPC s.v.	996.6	0.902	0.432	0.280	fallback

† Gate miss: in both variants HPT clears the pre-specified tolerance of 0.10 on window D but deploys far below nominal.

and a trend, both separate water-wash from HPC and neither separates HPT from either. We therefore present the gate as a screen for unusable components and report the marginal case as beyond prospective resolution on four engines.

Two further observations bound what the gate can do. First, what fails is the absolute threshold, not the gate statistic's *ordering*: in Variant I the gate ranks the three targets exactly as the deployment window does ($0.963 > 0.840 > 0.507$ against $0.902 > 0.286 > 0.190$), and in Variant II it also ranks water-wash first and differs only in the order of the two shop-visit targets, whose deployed coverages are 0.248 and 0.280 and thus not distinguishable. A gate used to *rank* components by trustworthiness would therefore have been correct; what four engines do not supply is the calibration needed to place a cut-off on that ranking. Second, and reported explicitly as a post-hoc sensitivity rather than as a result, a tolerance of 0.05 instead of 0.10, that is a window-D coverage of at least 0.85, classifies all three targets correctly in both variants. We do not adopt it, because it was selected with knowledge of the outcome and because a single fleet cannot establish where the cut-off belongs; we report it because it locates the gate's failure precisely, in the choice of tolerance rather than in the statistic. The gate at a pre-specified tolerance thus catches the target that must be routed away and correctly trusts the target that is calibrated, but has no resolution for the marginal one, consistent with the overlapping leave-one-engine-out ranges of Section 5.5. A prospectively deployable gate is, on this evidence, a coarse instrument: it separates usable from unusable, not good from marginal.

5.7. Predictive Distributions Rather Than Population Intervals

All five constructions of Section 5.3 summarize a population of residuals. A natural question is whether the diagnosis depends on that choice, and what happens when the uncertainty is a full predictive density, as returned by Bayesian methods. We therefore repeat the analysis with two constructions that return a distribution rather than an interval, both fit on the training window only: an MC-dropout multilayer perceptron, in which test-time dropout over 100 stochastic passes approximates a posterior over weights, and a five-member deep ensemble whose networks carry mean and log-variance heads trained by Gaussian negative log likelihood, so that the predictive distribution is a Gaussian mixture with both aleatoric and epistemic parts. For each we report the coverage of the central 90% credible interval, the continuous ranked probability score (CRPS), which scores the whole density rather than one interval, and the probability integral transform (PIT), whose distance from uniformity localizes the miscalibration.

Table 8 shows two things. First, neither density is calibrated even in distribution: on the validation window the central 90% credible interval covers only 0.52–0.87 of observations, far below nominal. The split-conformal coverage on calibration window C (Table 7) is not a comparable figure, since it is largely a consequence of the construction (Section 5.6); windows D and E provide the split-conformal coverage assessments. The difference in kind lies in the guarantee: a credible interval inherits the correctness of its likelihood and prior, while the conformal interval carries a finite-sample coverage guarantee that requires only exchangeability. Second, and more to the point of the paper, the distributions degrade under the late-life shift in exactly the tiered pattern already reported. For water-wash, CRPS is essentially unchanged from validation to deployment ($78 \rightarrow 79$ for MC-dropout). For the shop-visit targets it degrades by a factor of three to eight ($343 \rightarrow 2{,}572$ for HPC), credible coverage falls to 0.19–0.43, and the PIT mass below the 5th percentile rises to 0.54–0.81, which says that the density places its mass at too high a remaining life, the over-prediction that Section 6.8 identifies as the operative failure mode. The conclusion is therefore not an artifact of frequentist interval construction: the deficit is in the calibration data, which contains no late-life residuals, so any construction estimated on that window inherits it. A Bayesian treatment changes what the uncertainty means, not what the data can support, and it forfeits the finite-sample guarantee in the process.

5.8. Sensitivity to Causal Imputation

Section 4 notes that two imputation steps are not strictly causal: the backward-fill component, which can fill an earlier gap with a later measurement, and a global median for whatever the two fill directions leave. To quantify their

Table 8. Predictive-distribution uncertainty quantification, fit on the training window only. MC-dropout is an approximate Bayesian posterior over weights; the Gaussian-NLL deep ensemble returns a Gaussian-mixture predictive density. "Cov" is the coverage of the central 90% credible interval and "PIT$_{<5}$" the fraction of observations falling below the 5th percentile of the predictive distribution. Neither density is calibrated in distribution, and both degrade under the shift in the same tiered pattern as the conformal intervals.

		Validation			Test (late life)		
Method	Target	Cov	CRPS	PIT$_{<5}$	Cov	CRPS	PIT$_{<5}$
MC-drop.	WW	0.52	78	0.23	0.61	79	0.27
MC-drop.	HPT	0.67	162	0.25	0.28	528	0.68
MC-drop.	HPC	0.78	343	0.17	0.19	2,572	0.81
Ensemble	WW	0.65	77	0.20	0.59	100	0.34
Ensemble	HPT	0.80	144	0.17	0.43	483	0.54
Ensemble	HPC	0.87	349	0.11	0.24	2,989	0.76

influence we re-ran the complete pipeline twice, changing nothing but the imputation. The causal variant forward-fills only within each (engine, snapshot) group and fills any residual gap with the median of the training window alone, so no value is ever imputed from a later observation and no imputation statistic is computed across the temporal split. One deviation from a strictly per-split procedure should be stated: the forward fill runs continuously along each engine's timeline rather than restarting at every split boundary. It therefore never uses a future observation, but neither is it computed independently within each split; continuing the fill across a boundary is what an online implementation would in fact do, since at deployment time the preceding cycles are available. The outlier caps are fixed engineering thresholds (core speed above $30{,}000$ rpm, T_3 above $3{,}000\,^\circ$R) rather than quantities estimated from the data, so applying them to the whole record introduces no dependence on the evaluation window. Feature families, the feature-selection rule, the temporal split, the four base learners, the stored hyperparameters, the meta-learner and the bias correction are all held fixed; hyperparameters are deliberately not re-tuned, since re-tuning would confound an imputation effect with a search effect.

The two variants give *identical* results, to the digit, in accuracy and in conformal coverage (Table 9). We verified the reason directly: after outlier capping, $4{,}003$ sensor cells are missing, and forward fill alone resolves every one of them, leaving nothing for the backward fill or the global median to impute. The missingness arises from paired channel dropouts *within* a series rather than at its start, so the non-causal steps are present in the code but inert on this dataset. The sensitivity analysis is therefore exact rather than approximate: no result in this paper depends on non-causal imputation. We retain the causal implementation as the default in the released code, because a dataset whose gaps begin at the first cycle of a channel would not be so forgiving. One caveat on reading Table 9 against Tables 3 and 4: this is a fresh end-to-end execution, including re-running the importance-based feature selection, under Python 3.11 and tsfresh 0.21.2 rather than the Python 3.12 and tsfresh 0.21.1 environment used for the archived 60/20/20 artifacts. It selects the same number of features (229) and reproduces the accuracy figures closely (validation R^2 within 0.01 on all three targets), but the coverage figures move: HPT to 0.568 and HPC to 0.284, against 0.47 and 0.17 in Table 4. The comparison of interest here is between the two columns of Table 9, which are identical; the drift relative to the secondary 60/20/20 analysis is a separate reproducibility observation, and it is another reason we treat the reliable-versus-unreliable separation, rather than the individual coverage values, as the robust finding.

Table 9. Causal-imputation sensitivity. The submitted imputation (forward then backward fill, global median for residual gaps, applied before the temporal split) and a strictly causal alternative (forward fill only, training-window median, computed within the training split) give identical results, because forward fill alone resolves all $4{,}003$ missing cells. Coverage is of a nominal 90% interval on the test window.

Imputation	Target	Val R^2	Test R^2	Half-w.	Cov.
As submitted	WW	0.913	0.697	152.2	0.944
As submitted	HPT	0.969	0.025	280.8	0.568
As submitted	HPC	0.969	-6.711	433.8	0.284
Causal only	WW	0.913	0.697	152.2	0.944
Causal only	HPT	0.969	0.025	280.8	0.568
Causal only	HPC	0.969	-6.711	433.8	0.284

6. UNCERTAINTY-AWARE MAINTENANCE SCHEDULING

We now turn predictions into decisions and ask what conditioning the decision on the measured reliability of Section 5 is worth. The strictly nested decision experiment of Section 6.9 is the primary result. The earlier subsections retain the original 60/20/20 predictions as a secondary policy-decomposition analysis because they support the comparisons among buffer constructions, target-specific costs and the MDP baseline. Two framing statements belong at the outset. First, this section is a *retrospective case study* of what a reliability gate would have decided, not a demonstration of a prospectively deployable one: the achieved coverage that identifies HPC as unreliable is computed against materialized outcomes on the same window used to evaluate the resulting policy. Section 5.6 demonstrates the prospective version separately, using only information that precedes the evaluation window, and reports where that gate succeeds and where it misses. Second, the conformal half-widths used as buffers here inherit the in-sample calibration discussed in Section 5.1; Section 6.9 repeats the whole decision experiment with the wider, out-of-sample half-widths of the nested design and reports how much of the advantage survives.

6.1. Decision and Cost Model

Walking forward through an engine's timeline, a maintenance action is triggered the first time the predicted RUL falls to or below a safety buffer b: maintenance is triggered at the first cycle where $\hat{y} \leq b$. Let r be the *true* RUL at that triggered cycle. The realized cost is

$$J(r) = \begin{cases} c_{\text{early}}\, r, & r > 0 \quad \text{(early service)} \\ C_{\text{fail}} + c_{\text{late}}\, |r|, & r \leq 0 \quad \text{(unplanned failure)} \end{cases} \tag{4}$$

so an early action wastes the r cycles of useful life it forfeits, while acting after the event incurs a large fixed failure penalty C_{fail} plus a late term. We set $c_{\text{early}} = 1$ and $c_{\text{late}} = 2$, mirroring the challenge metric's $2\times$ late asymmetry (Eq. 1), and treat C_{fail} as a domain parameter whose value we sweep. All three cost terms are expressed in one unit, the *forfeited cycle*: with $c_{\text{early}} = 1$ per cycle of remaining life given up, $C_{\text{fail}} = 250$ penalizes an unplanned in-service failure as much as forfeiting 250 cycles of remaining useful life, and $C_{\text{fail}} = 500$ as much as forfeiting 500 cycles. C_{fail} is therefore not a multiple of the cost of "one early service": the cost of an early service is $c_{\text{early}} r$, which depends on how early that service occurs and is not a fixed quantity. The costs are relative and normalized, intended for ranking policies rather than for estimating monetary expenditure; Section 6.5 relaxes the further simplification that the same C_{fail} applies to an on-wing water wash and to a shop visit. The buffer b thus trades the two error modes: a larger b services earlier, converting potential failures into (cheaper) early services at the cost of forfeited life. The optimal b depends on how confidently $\hat{y}$ locates the event – precisely the per-component reliability measured in Section 5. Decisions are evaluated per maintenance cycle: each engine's test timeline is segmented into individual degrade-to-event cycles using resets of the cycles-since-last counter, yielding 37 decision instances (WW 20, HPT 11, HPC 6).

6.2. Policies

We compare three buffer policies, all sharing the trigger rule of Algorithm 1. The *naive* policy uses $b = 0$ (act when the point prediction reaches zero). The *fixed-buffer* policy uses a single constant b per target, tuned on validation. The *uncertainty-aware* policy sets b equal to the conformal 90% half-width $q_{0.1}$ of Section 5, so the buffer automatically widens for unreliable targets and stays tight for reliable ones.

The buffer choices instantiate a spectrum of how much the decision trusts the point prediction: naive trusts it fully, fixed-buffer applies a constant margin, and uncertainty-aware lets the calibrated interval width set the margin per component.

6.3. Secondary 60/20/20 Policy Decomposition

Table 11 reports per-target outcomes at $C_{\text{fail}} = 500$ (an unplanned failure is penalized as much as forfeiting 500 cycles of remaining useful life; we use 500 as the headline point because it sits mid-range of the sweep in Section 6.4). The outcomes track the reliability tiers of Section 5 exactly. On well-calibrated WW, both buffered policies eliminate most failures and are statistically indistinguishable from each other (fixed 3,860 vs. uncertainty-aware 3,870; 4 failures each): when the interval is calibrated, a well-tuned constant margin recovers the same protection. On *marginal* HPT, the naive policy fails on all 11 cycles (its point predictions, biased late under shift, never leave enough lead time); the conformal buffer cuts failures to 6 and roughly halves cost, beating the tuned fixed buffer (4,840; 9 failures) – though we note 6 of 11 cycles still end in unplanned failures, so HPT is managed better, not managed well. On unreliable HPC, the fixed and symmetric-conformal buffers in this secondary analysis each leave all six failures, consistent with its 17% coverage. A prediction-free, age-based scheduled policy (service at a fixed cycle count tuned on validation inter-event intervals) reduces HPC failures (6 to 4, at higher realized cost). Two prediction-driven policies do match or beat it, and both are instance-dependent rather than constant offsets (Section 6.6): the conformalized quantile-regression trigger also reduces HPC failures from six to four, at a realized cost of $4{,}470$, and a conformal buffer whose width is a learned function $\sigma(x)$ of the full feature vector reduces them to four at a cost of $2{,}630$, below the naive trigger's $3{,}000$ and the scheduled policy's $4{,}200$, making it the cheapest HPC policy we found. The claim that no prediction-driven policy helps HPC, made in the submitted version of this paper, is therefore wrong and is corrected here. The failure here is specific to the constant buffers evaluated in this secondary analysis: the prediction never descends to their fixed thresholds before the event. Nested Variant II later shows that a much wider constant buffer can trigger, but it remains costly and only partly effective (Section 6.9). A trigger that varies with the operating point still extracts usable signal from a badly calibrated predictor. The lesson is therefore not to stop predicting, but to avoid treating a single constant margin as robust across calibration protocols.

Aggregating over the two targets with usable signal (WW and HPT), the uncertainty-aware policy reduces total cost by 36% versus naive (failures 23 to 10) and by 14% versus the tuned fixed buffer (failures 13 to 10). The two comparisons answer different questions: the first measures the value of *any* safety

Algorithm 1 Buffered maintenance-scheduling policy

Require: predictions $\hat{y}_1, \ldots, \hat{y}_T$ over a maintenance cycle; buffer b
1: **for** $k = 1$ to T **do**
2: **if** $\hat{y}_k \leq b$ **then**
3: trigger maintenance; **return** cost $J(r_k)$ ▷ Eq. 4
4: **end if**
5: **end for**
6: **return** unplanned-failure cost C_{fail} ▷ event reached, never triggered

margin over trusting the point prediction; the second – the fair test of this paper's contribution – measures the incremental value of a *calibrated* margin over a well-tuned constant one. That second comparison arises on HPT, where the conformal width is wider than a margin tuned on in-distribution validation data, but it rests on two of the four engines and does not survive every calibration protocol, so we report it as directional (Table 10, Section 6.9).

Because these numbers rest on 31 decision instances (WW 20 + HPT 11; HPC's 6 are excluded) clustered within four engines, we quantify sampling uncertainty with an engine-level cluster bootstrap (resampling engines with replacement, 5,000 replicates), which respects the within-engine correlation that an instance-level bootstrap would ignore. The naive-relative reduction is robust: median 36%, 95% interval [19%, 57%]. The fixed-buffer-relative reduction is not resolved at this fleet size: median 14%, 95% interval [−4%, 36%].

We therefore claim only the naive-relative improvement as statistically supported. The advantage over the tuned fixed buffer is directional, and Table 10 resolves it by engine to show how narrowly it is carried: the calibrated buffer is better on ESN 102 and ESN 103 (+21% and +47%) and marginally worse on ESN 101 and ESN 104 (−5% and −4%), so the aggregate 14% rests on two of the four available units rather than on a fleet-wide effect. Two mechanisms explain the pattern. On water-wash the two policies are in effect the same policy: the conformal half-width (150.5 cycles) and the validation-tuned fixed buffer (150 cycles) nearly coincide, so the 0.3% cost difference there is numerical rather than methodological, and the earlier claim that the calibrated buffer is never worse than the fixed one is not supported by these results and has been withdrawn. On HPT the conformal half-width (232 cycles) exceeds the tuned fixed buffer (180 cycles), which is what converts failures into early services on the two engines where the margin was decisive. The defensible reading is therefore narrower than a dominance claim: the calibrated buffer recovers a well-tuned constant margin where the prediction is calibrated and widens it where the prediction is not, without requiring any per-target tuning, and four engines cannot establish that it dominates a well-tuned constant.

6.4. When Is Uncertainty-Awareness Worthwhile?

Because the conclusion depends on how costly a failure is, we sweep C_{fail} (Table 12). When failures are cheap ($C_{\text{fail}} = 100$), the naive policy is best: the early-service waste from buffering exceeds the failures it prevents. From $C_{\text{fail}} = 250$ upward, the uncertainty-aware policy is uniformly best, and its margin widens with C_{fail}. Safety-critical aviation maintenance sits firmly in the high-C_{fail} regime, where an unplanned failure dwarfs the cost of an early shop visit; in that regime the approach is clearly worthwhile. We report the low-C_{fail} crossover rather than hiding it, because it delimits the operating conditions under which the method should be used.

Table 10. Decision outcomes by engine on the test window ($C_{\text{fail}} = 500$, water-wash and HPT combined, 31 decision instances). Cost is in forfeited-cycle units; failures are unplanned in-service events. The conformal buffer's aggregate advantage over the tuned fixed buffer is carried by ESN 102 and ESN 103; on ESN 101 and ESN 104 it is marginally worse. The last column isolates HPT, the target the aggregate advantage derives from, and shows the same two-of-four pattern.

	Naive		Fixed		Conformal		Conf. vs fixed	
ESN	Cost	F.	Cost	F.	Cost	F.	All	HPT
101	2,540	5	**1,710**	2	1,790	2	−5%	−10%
102	3,100	6	2,390	4	**1,900**	**3**	+21%	+33%
103	3,030	6	2,010	3	**1,060**	**1**	+47%	+63%
104	3,070	6	**2,590**	4	2,700	4	−4%	−10%
All	11,740	23	8,700	13	**7,450**	**10**	+14%	+26%

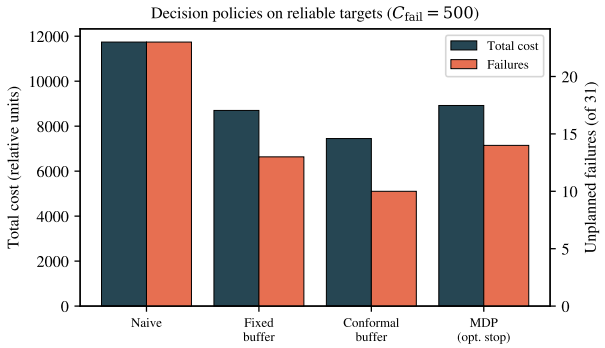


Figure 7. Total maintenance cost (left axis) and unplanned failures (right axis) on water-wash and HPT combined at $C_{\text{fail}} = 500$. The conformal-buffer policy attains the lowest cost and fewest failures; the formal optimal-stopping MDP, limited by noisy four-engine transition estimates, does not improve on it. Its margin over the tuned fixed buffer arises on HPT and on two of the four engines (Table 10), and is not resolved by this fleet (Section 6.9).

6.5. Target-Specific Failure Penalties

The sweep above applies one C_{fail} to all three targets, although the three maintenance actions are not comparable: an HPC water wash is an on-wing procedure measured in hours, whereas an HPT or HPC shop visit requires engine removal and is measured in weeks. A single normalized penalty is defensible for ranking policies within a target but not for transferring a threshold across targets, so we sweep each target over its own range: water-wash over $C_{\text{fail}} \in \{25, \ldots, 500\}$ and the shop visits over $\{250, \ldots, 4000\}$, always with $c_{\text{early}} = 1$ per forfeited cycle so that the unit is unchanged.

The separation of ranges is anchored in the published lifecycle-costing literature rather than chosen freely.

Table 11. Secondary 60/20/20 analysis: maintenance-decision outcomes on the test window ($C_{\text{fail}} = 500$). Cost is in relative units; failures are unplanned in-service events out of the number of maintenance cycles for that target (in parentheses). Bold marks the lowest cost and fewest failures within each target block. For HPC the constant-buffer policies listed in this table coincide; the prediction-free scheduled policy reduces failures. Section 6.6 evaluates instance-dependent prediction-driven triggers that also reduce failures.

Policy	Target	Total cost	Failures
Naive	WW (20)	6,240	12
Fixed buffer	WW (20)	**3,860**	**4**
Uncertainty-aware	WW (20)	3,870	**4**
Naive	HPT (11)	5,500	11
Fixed buffer	HPT (11)	4,840	9
Uncertainty-aware	HPT (11)	**3,580**	**6**
Naive	HPC (6)	3,000	6
Uncertainty-aware	HPC (6)	3,000	6
Scheduled (age-based)	HPC (6)	4,200	**4**

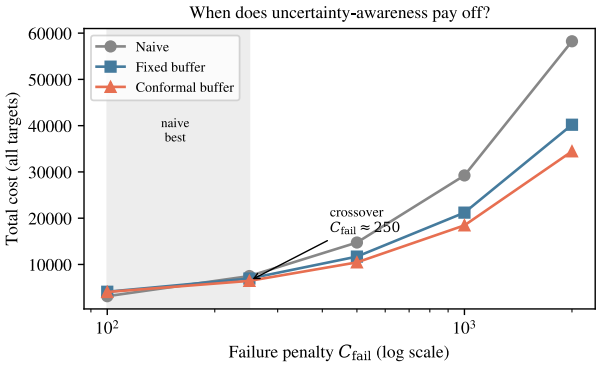


Figure 8. Total maintenance cost versus the failure penalty C_{fail} (log scale, all targets). The conformal-buffer policy is dominated by naive only in the cheap-failure shaded region; beyond the crossover near $C_{\text{fail}} \approx 250$ it is uniformly cheapest, with the gap widening as failures become more costly.

Wehrspohn, Pohya, and Wicke (2021), simulating engine-wash policies over the lifecycle of an A319/A320/A321 fleet, cost an on-wing wash at US$3,000 per engine per wash against a fleet-average direct maintenance cost of US$818 per flight cycle and a lifecycle cost of US$5,854 per flight cycle, with a new CFM56-5B listed at roughly US$10 million. A wash is therefore of the order of a few flight cycles' direct maintenance cost, and about 0.03% of the engine's value, whereas a shop visit is a shop-level refurbishment of the engine core and is necessarily a substantial fraction of that value. The two actions are separated by orders of magnitude, so a common C_{fail} across them is not defensible; if anything the single decade separating our two sweep ranges is conservative relative to the gap the literature implies. We stress that this argument fixes only the relative *ordering* of magnitudes: the costs in this paper remain normalized ratios, and we make no claim about absolute monetary values, which depend on the operator, the contract and the engine variant.

Table 12. Total cost by policy across failure-penalty values (all targets); bold marks the lowest cost per row. The best policy is naive only when failures are cheap; uncertainty-awareness wins for $C_{\text{fail}} \geq 250$.

C_{fail}	Naive	Fixed	Unc.-aware	Best
100	**3,140**	4,100	4,050	naive
250	7,490	6,950	**6,450**	unc.-aware
500	14,740	11,700	**10,450**	unc.-aware
1000	29,240	21,200	**18,450**	unc.-aware
2000	58,240	40,200	**34,450**	unc.-aware

Table 13. Target-specific failure penalties (total cost in forfeited-cycle units; "one-sid." is the one-sided conformal buffer of Section 6.6). The crossover is target-specific: on the expensive HPT shop visit the uncertainty-aware buffer is best from $C_{\text{fail}} = 250$ upward, while on the cheap on-wing water wash it is never strictly best. For HPC the listed constant-buffer policies coincide because none ever triggers.

Target	C_{fail}	Naive	Fixed	Symm.	Best
WW	25	**540**	540	1,970	naive
WW	100	**1,440**	1,580	2,270	naive
WW	250	3,240	2,670	2,870	one-sid. (2,650)
WW	500	6,240	**3,860**	3,870	fixed
HPT	250	2,750	2,750	**2,080**	symm.
HPT	500	5,500	4,840	**3,580**	symm.
HPT	1000	11,000	9,340	**6,580**	symm.
HPT	2000	22,000	18,340	**12,580**	symm.
HPC	any	all coincide (never triggers)			fallback

Table 13 shows that the crossover is target-specific, and that the paper's headline threshold is a shop-visit statement rather than a universal one. For HPT the uncertainty-aware buffer is best at every penalty from 250 upward, and its margin widens with the penalty, exactly as Section 6.4 reports. For water-wash the picture is different: the naive trigger is cheapest up to $C_{\text{fail}} = 100$, the one-sided buffer is cheapest at 250, and the *tuned fixed* buffer is cheapest at 500; the uncertainty-aware buffer is never strictly best on this target, which is the same tie already visible in Table 11 seen across a range of penalties. For HPC the constant-buffer policies evaluated in this secondary sweep coincide at every penalty because none of them triggers, so the comparison is uninformative there and only the fallback and the two instance-dependent triggers of Section 6.6 change the outcome. The practical reading is that uncertainty-aware buffering earns its place where the intervention is expensive relative to a forfeited cycle, the shop visits, and for a cheap on-wing action a simpler policy is preferable.

6.6. Directional and Instance-Dependent Buffers

The buffer used so far is a single symmetric conformal half-width per target. It therefore varies across components but is constant across operating points and RUL regions within a

component, and it is derived from *absolute* residuals although the decision risk is directional: over-predicting remaining life is the error that delays maintenance, while under-predicting merely forfeits life. Three alternatives address these two objections directly, all calibrated on the validation window and evaluated on the test window.

The *one-sided* buffer replaces the absolute-residual quantile with the quantile of the signed over-prediction residuals $\hat{y}-y$, which yields the one-sided guarantee $\Pr(y \geq \hat{y} - q) \geq 1-\alpha$. The residuals are used unclipped. We report it at two levels: at the same nominal α a one-sided quantile is necessarily narrower than a two-sided one, so we add $\alpha = 0.05$ as an additional sensitivity setting. This is not an exactly matched comparison: an interval constructed from absolute residuals does not generally allocate equal miscoverage to its two tails, particularly with asymmetric residuals, so the *lower* bound of a symmetric $1-\alpha$ interval need not sit at the one-sided level $1-\alpha/2$. The *normalized* buffer is the adaptive construction of Section 5.3 propagated into the decision, with per-instance width $q\,\sigma(x)$ and $\sigma(x) = a + b\,\hat{y}$ fit on the calibration window, so that the margin varies within a component. Because a width scaled by $\hat{y}$ alone still cannot vary with the operating point, we add a second version in which $\sigma(x)$ is a gradient-boosted regression of the absolute residual on the full feature vector, fit on the training window, so that the buffer responds to flight condition as well as to remaining life. Finally, the *CQR* policy triggers on the conformalized lower quantile itself rather than on a point prediction offset by a constant, making its threshold fully instance-dependent.

Table 14 reports the outcome, and it is not the one the directional argument predicts. The one-sided buffer is *narrower* than the symmetric one at both levels (113 cycles at $\alpha = 0.10$ and 136 at $\alpha = 0.05$, against 232 for the symmetric buffer on HPT), and it is worse at $C_{\text{fail}} = 500$ in both cases (8,890 and 8,850 against 7,450; 15 and 14 unplanned failures against 10). The reason is not the obvious one: the absolute-residual quantile that the symmetric buffer uses is inflated by the *under*-prediction tail of the calibration residuals, so that buffer is more conservative than the directional risk alone would justify. At a high failure penalty, that incidental conservatism is what pays. Restricting attention to over-prediction, as the directional argument recommends, removes it. Directional risk is real, but the way to buy more protection is a smaller α, not a one-sided quantile.

Instance-dependent width behaves differently for coverage than for cost. Scaling by $\hat{y}$ leaves both essentially unchanged (7,820 against 7,450; coverage 0.471 against 0.473 on HPT). The covariate-dependent $\sigma(x)$ does what it is supposed to do to coverage, raising HPT from 0.473 to 0.565 and HPC from 0.167 to 0.240, the largest coverage gain of any construction we tried. It is nonetheless the most expensive policy on water-wash and HPT combined (9,230), because the wider intervals it assigns near the event convert failures into early services that forfeit a great deal of life. Better calibration and lower realized cost are not the same objective, and on the two calibrated targets a single well-chosen constant remains hard to beat.

Table 14. Buffer constructions propagated into the decision experiment (water-wash and HPT combined, $C_{\text{fail}} = 500$; buffers in cycles, cost in forfeited-cycle units). "Vs fixed" is the engine-level cluster-bootstrap cost reduction relative to the tuned fixed buffer with its 95% interval. On the two calibrated targets neither the directional one-sided buffer, at either level, nor either instance-dependent construction improves on the single symmetric half-width; on the uncalibrated HPC target the instance-dependent constructions are the only ones that reduce failures materially (footnote). All five interval constructions of Section 5.3 are propagated, not only the one the comment requires.

Construction	Buffer WW	Buffer HPT	Cost	Fail	Vs fixed (95% CI)
Naive	0	0	11,740	23	-35%
Tuned fixed	150	180	8,700	13	*ref.*
Symmetric conf.	151	232	**7,450**	**10**	$+14\%$ $[-4, 36]$
One-sided, α=0.10	102	113	8,890	15	-2% $[-15, 11]$
One-sided, α=0.05	131	136	8,850	14	-2% $[-14, 5]$
Normalized, $\sigma(\hat{y})$	151	231	7,820	11	$+10\%$ $[-3, 23]$
Normalized, $\sigma(x)$	181	335	9,230	12	-6% $[-23, 15]$
Parametric Gaussian	142	212	8,250	12	$+5\%$ $[-2, 17]$
nex-SCP (weighted)	119	144	8,700	14	0% $[-13, 9]$
CQR trigger	160	104	8,290	12	$+5\%$ $[-1, 11]$

On HPC, evaluated separately, the constant buffers leave all six unplanned failures (the wider parametric-Gaussian buffer reaches five); the CQR trigger reduces them to four at a cost of 4,470 and the covariate-dependent buffer to four at a cost of 2,630, against 3,000 for the naive trigger and 4,200 for the prediction-free scheduled policy.

On HPC the ordering reverses, which amends a claim of the submitted paper. Both instance-dependent triggers reduce HPC failures from six to four where the constant buffers evaluated in the secondary analysis leave all six, and the covariate-dependent buffer does so at a realized cost of 2,630, below the naive trigger's 3,000, CQR's 4,470 and the prediction-free scheduled policy's 4,200. For the one component whose intervals are badly miscalibrated, then, the remedy that works in this secondary analysis is not to stop predicting but to stop relying on a *constant* margin, although a much wider constant buffer can partly help: under Nested Variant II it reduces HPC failures from six to three, at substantial early-service cost (Section 6.9).

6.7. Does a Formal Sequential-Decision Optimum Do Better?

The buffer policies above are heuristics. A natural question is whether casting the problem as a formal sequential decision – an optimal-stopping Markov decision process (MDP) – improves on them. We formulate one: the state is the discretized predicted RUL bin; the actions are MAINTAIN (terminal) or WAIT; the immediate maintenance cost $m(b)$ is the expected

servicing cost in bin b, estimated from validation $(\hat{y}, y)$ pairs (so it inherits each component's predictive spread); and waiting incurs a forced-failure risk $p_{\text{event}}(b)$, the empirical probability that the true event strikes within one cycle-step, making deferral costly near the event. We solve for the stopping policy by discounted value iteration ($\gamma = 0.99$) on validation and evaluate on test.

The result is worth stating carefully: the MDP policy reduces cost by 24% over naive on WW and HPT – better than naive but *worse* than both the tuned fixed buffer (26%) and the conformal buffer (36%). The reason is informative rather than disappointing: with only four engines, the estimated transition and forced-failure probabilities over 40 RUL bins are noisy, whereas the conformal half-width summarizes the entire calibration-residual distribution in a single, stably estimated quantity. In the data-limited, point-prediction-only regime typical of fielded PHM, a lightweight uncertainty-aware heuristic is therefore the more robust choice. A richer formulation – a partially observed MDP whose belief state is the hidden degradation condition, with the conformal interval supplying the belief width – is the natural way to give a formal optimum enough information to win, but it is data-hungry and we leave it as the primary direction for future work.

6.8. Error Analysis: Why Constant-Buffer Policies Are Unreliable on HPC

The HPC result in the secondary 60/20/20 analysis – $6/6$ unplanned failures under each constant-buffer policy in Table 11 – is worth dissecting, because it exposes a failure mode that a fixed buffer cannot address. We inspect the six HPC decision cycles individually. In every one, the predicted RUL *never falls to the trigger buffer* ($b = 393$ cycles for the conformal policy): the minimum prediction reached within the cycle ranges from 450 to $5{,}791$ cycles, always above b, while the true RUL nonetheless reaches zero. The model does not merely err on HPC; it systematically *over*-predicts remaining life under the late-life shift, so the constant-threshold rules evaluated here are never armed before the event. A sufficiently large constant buffer can force a trigger, as Nested Variant II demonstrates, but it does so by forfeiting substantial useful life; the original 393-cycle buffer cannot compensate for the bias. The instance-dependent results of Section 6.6 show that this is a limitation of constant margins, not proof that the predictor contains no usable scheduling signal. Two of the six cycles are especially stark (minimum predictions of $4{,}660$ and $5{,}791$ cycles at the moment the component actually reaches its shop visit), reflecting the $R^2 = -6.5$ seen in Table 3: the constant-margin trigger receives too little usable late-life signal for HPC. This is the operational meaning of the low conformal coverage: when the prediction is biased beyond a constant interval's reach, the response is either a different information source, a scheduled-maintenance fallback, or a trigger whose threshold adapts to the operating point. The decision layer should therefore consume the reliability diagnosis as a gate away from automated constant-buffer triggering, not necessarily away from every prediction-driven policy.

6.9. Primary Decision Result under Strictly Nested Calibration

The buffers used above are the conformal half-widths of Table 4, which Section 5.1 shows to be optimistically narrow because the meta-learner and the bias correction were fit on the calibration window. Since a buffer is exactly a calibration width, the decision result must be re-examined under the wider, out-of-sample half-widths of the nested design (Section 5.6). We repeat the entire decision experiment on the same deployment window with those widths, tuning the fixed buffer and the scheduled interval on the nested calibration window rather than on the original validation window.

Table 16 reports what survives and what does not. The advantage of uncertainty-aware buffering over trusting the point prediction is robust: on water-wash and HPT combined it is a 23% cost reduction under Variant I and 41% under Variant II, against 36% in the secondary 60/20/20 analysis, with unplanned failures falling from 31 to 9 and 13 respectively.

The comparison against a *tuned fixed* buffer, by contrast, is not identifiable at this fleet size, and the two variants show why. Under Variant I the conformal buffer costs $11{,}860$ against the fixed buffer's $13{,}600$, a 13% improvement; under Variant II it costs $9{,}160$ against $8{,}870$, a 3% deterioration. The sign flips with how the pre-deployment model-selection and calibration range is apportioned, and the reason is straightforward: a tuned constant buffer is a fitted quantity and needs data to fit, so it is well estimated when the available tuning window is generous (Variant II gives it $1{,}200$ observations) and poorly estimated when it is not (Variant I gives it 400). The conformal half-width is a quantile of the same residuals and needs no separate tuning pass. Taken with the engine-level resolution of Table 10, which showed the secondary analysis's improvement resting on two of four units, the supported conclusion is that four engines cannot order these two policies. What the evidence does establish is that either buffer is far better than none, and that the calibrated buffer needs no additional outcome-tuning split beyond its calibration residuals, which is the property that matters when data are as limited as they are here.

Four engines cannot order the two policies, but a public benchmark can, so we put the same question to C-MAPSS, where each subset supplies enough run-to-failure units for the comparison to resolve. Using the mature-fleet protocol of Section 5.5 (half the units fitted in full, a quarter supplying the calibration window, a quarter deployed, calibration and deployment both on the final 20% of a unit's life) gives

Table 15. The conformal-versus-tuned-constant comparison of Table 16 repeated on the public C-MAPSS benchmark under the mature-fleet protocol, pooled over 178 deployment units from the four subsets (one decision instance per unit; cost in forfeited-cycle units). With this sample the ordering resolves, and it crosses over between failure penalties of 1,000 and 1,500; the calibrated buffer leaves no unplanned failures at any penalty.

	Tuned fixed		Conformal		Conf. vs
C_{fail}	Cost	Fail	Cost	Fail	fixed
250	**2,575**	2	4,352	**0**	-69%
500	**3,075**	2	4,352	**0**	-42%
1000	**4,075**	2	4,352	**0**	-7%
1500	5,075	2	**4,352**	**0**	$+14\%$
2000	6,075	2	**4,352**	**0**	$+28\%$
4000	10,075	2	**4,352**	**0**	$+57\%$

The naive trigger costs 37,603 to 600,103 over the same range with 150 unplanned failures throughout. Per subset, the calibrated buffer reduces cost against naive by 93–96%; against the tuned buffer at $C_{\text{fail}} = 500$ it ranges from -158% (FD001) to $+13\%$ (FD002), with a unit-level bootstrap interval of $[-113\%, +5\%]$ pooled.

Table 16. The decision experiment repeated under the strictly nested calibration protocol of Section 5.6 on the same deployment window ($C_{\text{fail}} = 500$; cost in forfeited-cycle units, failures out of the decision instances in parentheses). The improvement over the naive policy is robust to the protocol; the improvement over a tuned fixed buffer is not.

Protocol	Policy	Cost	Failures
Main (Sec. 6)	Naive	11,740	23
	Tuned fixed	8,700	13
	Conformal buffer	**7,450**	**10**
Nested I (Sec. 5.6)	Naive	15,500	31
	Tuned fixed	13,600	25
	Conformal buffer	**11,860**	**9**
Nested II (Sec. 5.6)	Naive	15,500	31
	Tuned fixed	**8,870**	14
	Conformal buffer	9,160	**13**

Water-wash and HPT combined (31 instances). On HPC alone the conformal buffer still never triggers under Variant I (6/6 failures, as in the secondary analysis) and triggers under Variant II, where its wider half-width of 997 cycles reduces failures from 6 to 3; the gated fallback removes all six under both variants.

178 deployment units across the four subsets, one decision instance each, with achieved coverage of 0.90–0.94 as that protocol predicts. Table 15 reports the outcome. Against the naive trigger the calibrated buffer reduces cost by 94% and eliminates all 150 unplanned failures, so the claim we retain from Section 6 is confirmed at a sample size two orders of magnitude larger than our fleet. Against the tuned constant buffer the ordering resolves too, and it depends on the failure penalty: below roughly 1,000 forfeited cycles the tuned buffer is cheaper, because in this well-calibrated regime the conformal half-width (17–25 cycles) is wider than cost-optimal and forfeits life unnecessarily; above roughly 1,500 the calibrated buffer is cheaper, and at every penalty it is the only one of the two that leaves no unplanned failures at all. Two qualifications follow. The crossover on this benchmark lies above the $C_{\text{fail}} = 500$ we use as our headline point, so a reader who adopts that normalization should read the tuned constant buffer as the cheaper policy there; and the crossover is dataset-specific, exactly as Section 6.5 finds for the challenge fleet, where the HPT shop visit favours the calibrated buffer from 250 upward. What transfers is not a threshold but the shape of the trade-off: a calibrated buffer buys the elimination of failures at the price of forfeited life, and it is the right purchase when failures are expensive or when there is too little pre-deployment data to tune a constant.

The gate fares better than the buffer, and its behaviour is the same under both variants. It routes HPC to the prediction-free fallback, which removes all six unplanned failures on window E, at a realized cost of 13,220 (Variant I) or 12,980 (Variant II) against the naive policy's 3,000. On this component the choice is therefore between forfeiting a great deal of useful life and accepting failures, and the gate's contribution is to make that choice explicit rather than to conceal it inside a threshold rule. Whether the HPC conformal buffer fires at all is itself protocol-dependent: under Variant II its half-width of 997 cycles is wide enough to trigger and cuts failures from six to three, while under Variant I (493 cycles) it never triggers, exactly as in the secondary analysis. This is a further reason to route an uncalibrated component away from a constant threshold rather than to tune the threshold, and it is consistent with the instance-dependent triggers of Section 6.6, which reduce HPC failures under the secondary 60/20/20 calibration, the only setup in which they were evaluated, so their robustness across calibration protocols has not been demonstrated.

7. DISCUSSION

The three stages tell one story. The prediction pipeline is accurate in aggregate but unevenly reliable; conformal intervals expose that unevenness quantitatively and per component; and the decision layer shows that respecting it – buffering more where prediction is less calibrated – both saves cost and prevents failures, but only where the prediction carries usable signal at all. HPC is the boundary case that makes the argument: constant-margin policies are not robust across calibration protocols. The original and Nested Variant I margins leave all six failures, whereas the wider Variant II margin cuts failures to three at substantial early-service cost; instance-dependent triggers whose thresholds vary with the operating point also reduce failures under the secondary calibration (Section 6.6). Per-component reliability assessment must therefore precede, not follow, the decision.

7.1. Practical Guidance

Table 17 contrasts the proposed workflow with standard practice. The analysis yields three concrete recommendations for

Table 17. Qualitative comparison of the standard aggregate-accuracy PHM workflow with the proposed reliability-gated workflow, including the proposed approach's own weaknesses.

	Aggregate-accuracy PHM	Reliability-gated PHM (ours)
Model output	Point RUL	Point RUL + per-component achieved coverage
Trust model	Uniform across components	Per component, measured on a time-forward window
Decision input	Prediction only	Prediction + calibrated interval width
Failure mode	Silent: unreliable components indistinguishable	Explicit: low-coverage components routed to fallback
Weakness	Aggregate metric hides shift	Needs a labeled calibration window; marginal (not conditional) guarantee; buffer is a heuristic, not an optimum

deploying a learned RUL model in a maintenance pipeline. First, *calibrate per component on a time-forward window untouched by fitting and tuning, and inspect achieved coverage before trusting any single aggregate accuracy number*; a component whose interval coverage falls far below nominal (here HPC at 19–28% under the primary nested protocols) should be flagged as out of reliable range, and its predictions routed to a conservative fallback (scheduled servicing or manual inspection) rather than to an automated trigger. Second, *match the policy to the failure economics*: the cost sweep (Table 12) shows that an uncertainty-aware buffer pays off only when an unplanned failure is penalized by at least $\sim$250 forfeited cycles of remaining useful life under this dataset's normalization ($c_{\text{late}} = 2c_{\text{early}}$; the threshold is dataset-specific, not a universal constant); below that threshold a naive trigger is cheaper, so the buffer should be enabled only in genuinely safety-critical, high-C_{fail} regimes. That threshold should moreover be read per action rather than per fleet: giving each target its own penalty (Section 6.5) leaves the calibrated buffer cheapest at every penalty from 250 upward on the HPT shop visit, but never strictly cheapest on the on-wing water wash, where a tuned constant margin suffices. The buffer earns its place where the intervention is expensive relative to a forfeited cycle, and a practitioner should evaluate the crossover for each maintenance action separately. Third, *prefer the lightweight conformal buffer over a discretized sequential-decision model when data are scarce*: with few engines the conformal half-width is estimated far more stably than the transition and event probabilities a tabular MDP requires (Section 6.7), so the simpler policy is also the more robust one until substantially more run-to-event data are available.

7.2. Limitations

Three limitations bound these results and define the work we would prioritize next. First, the dataset has only four engines, and HPC in particular offers just six test maintenance cycles; the HPC findings are therefore qualitative, and the decision-layer advantages rest on 31 WW/HPT decision instances – the cluster bootstrap of Section 6 resolves the naive-relative gain but not the fixed-buffer-relative one. The comparison against a tuned fixed buffer is weaker still: it rests on two of the four units and changes sign with the calibration protocol (Table 10, Section 6.9), so we claim only the advantage over trusting the point prediction, which holds under every protocol we ran. Relatedly, the reliability diagnostic as demonstrated in Section 6 is retrospective: achieved coverage is computed against materialized test outcomes, which a fielded system observes only at the shop-visit cadence. Section 5.6 builds a prospective version from windows that precede deployment and finds it coarse: two independent gate statistics both route the uncalibrated HPC target to the fallback and both miss the marginal HPT target, whose coverage collapses abruptly at the deployment boundary rather than progressively. The gate is a screen for unusable components rather than a ranking, and the marginal case appears to be beyond prospective resolution at this fleet size. Operationally the diagnostic is therefore a *fleet-level, periodic* audit – recomputed as events materialize across engines – not a per-decision signal, and deploying it on a young fleet with no completed events requires borrowing calibration data from comparable fleets – an approach our cross-unit C-MAPSS experiment (Section 5.5) shows restores nominal coverage when such history exists. The coverage-collapse diagnosis itself, however, is corroborated externally: it replicates on all four C-MAPSS subsets with up to 260 units each (Section 5.5), so the small fleet bounds the decision-layer effect sizes, not the reliability phenomenon. Second, the decision layer uses a threshold-buffer policy rather than a formal sequential-decision optimum; casting the problem as a Markov decision process or partially observed MDP over the joint component state, with the conformal intervals supplying the belief width, is the natural next step and the one most likely to sharpen the contribution. Third, the conformal intervals assume exchangeability that the shift violates; the achieved coverage we report is exactly the consequence. Time-weighted non-exchangeable conformal prediction does not restore it (Table 5), and online adaptive schemes are ruled out by the label cadence, but richer covariate-shift weighting (Tibshirani et al., 2019) with an estimated likelihood ratio remains untested and is a natural extension.

8. CONCLUSION

We treated turbofan maintenance as a full prognostics-and-health-management loop rather than a regression problem. (1) Under the primary strictly nested analysis, a nominal 90% split-conformal interval achieves 90–93% deployment

coverage on water-wash but only 25–29% on HPT and 19–28% on HPC. The separation between the frequent event and the two shop visits is stable across both allocations of the limited pre-deployment history and is corroborated by alternative interval constructions, base-model families and all four C-MAPSS subsets. (2) Under the two nested protocols, a conformal safety buffer reduces cost by 23–41% relative to trusting the point prediction and cuts unplanned failures from 31 to 9–13. Its comparison with a tuned constant buffer changes from a 13% advantage to a 3% disadvantage as the pre-deployment data are apportioned, so this fleet does not order the two buffer policies. (3) The cost-sensitivity sweep places the relevant crossover near a penalty of 250 forfeited cycles for the expensive shop visits, while the conformal buffer is never strictly cheapest for the on-wing water wash. On HPC, constant-buffer policies are not robust: the original and Nested Variant I buffers leave all six failures, while the much wider Variant II buffer is costly and only partly effective. A prediction-free scheduled fallback and, under the secondary calibration, two instance-dependent triggers also reduce failures. The practical lesson is therefore specific: low per-component coverage argues against relying on automatic constant-margin scheduling, not against every possible use of the prediction.

DATA AND CODE AVAILABILITY

The dataset is publicly available from the PHM North America 2025 Data Challenge (PHM Society, 2025) at `https://data.phmsociety.org/phm-north-america-2025-conference-data-challenge/`. The complete experiment code – feature pipeline, stacking ensemble, conformal diagnostics, the nested-calibration and prospective-gate experiment, the causal-imputation sensitivity analysis, the predictive-distribution uncertainty experiment, the buffer-construction and target-specific cost experiments, the decision policies, and the C-MAPSS replication – is available at `https://github.com/Hiramdu/jetengmx`. The repository README maps each script to the table or figure of this paper that it produces, and indexes the revision experiments by the reviewer comment each one answers. It also provides separately pinned `requirements-main.txt` and `requirements-revision.txt` files and archives the tabular outputs behind the reported results.

ACKNOWLEDGMENT

The first author thanks Jacob Baratta and Jackson Beavers for their contributions to an early stage of this work. The PHM North America 2025 Data Challenge organizers are acknowledged for providing the dataset.

Table 18. Optuna-selected LightGBM hyperparameters per target (50 trials, validation competition score as the objective).

Hyperparameter	WW	HPT	HPC
Estimators	204	1,431	1,429
Max depth	9	3	9
Learning rate	0.014	0.189	0.026
Subsample	0.63	0.60	0.51
Column subsample	0.95	0.64	0.99
L_1 / L_2 reg.	1.2 / 2.4	1.6 / 0.01	0.6 / 7.8
Min child samples	59	31	64
Num. leaves	126	23	19

APPENDIX: REPRODUCIBILITY DETAILS

All experiments run on a single Linux workstation (16-core CPU); the full pipeline (feature extraction through decision evaluation) completes in under one hour, and no GPU is required. The archived 60/20/20 artifacts were generated with Python 3.12, NumPy 1.26.4, pandas 2.3.3, scikit-learn 1.6.1, LightGBM 4.6.0, SciPy 1.16.3, Optuna 4.9.0, statsmodels 0.14.6 and tsfresh 0.21.1. The revision analyses, including the causal-imputation rerun, used Python 3.11 with the same listed package versions except tsfresh 0.21.2; the predictive-density experiment additionally used PyTorch 2.6.0. Matplotlib 3.10.9 and Seaborn 0.13.2 were used only to render figures. Both environments are pinned separately in the public repository. Automated features use `MinimalFCParameters` on 10-cycle windows. Optuna uses 50 trials per target, optimizing the validation competition score. Table 18 lists the selected LightGBM hyperparameters; the remaining base learners use fixed settings (ridge penalty 10; gradient boosting: 100 estimators, depth 4, learning rate 0.1, subsample 0.8; random forest: 300 estimators, depth 20, min leaf 3; ridge meta-learner penalty 1). Seeds $\{0, 1, 2, 3, 42\}$ are used for the multi-seed repetition of Section 4.

REFERENCES

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2623–2631. doi: 10.1145/3292500.3330701

Angelopoulos, A. N., & Bates, S. (2023). Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, *16*(4), 494–591. doi: 10.1561/2200000101

Babu, G. S., Zhao, P., & Li, X.-L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International conference on database systems for advanced applications* (pp. 214–228). doi: 10.1007/978-3-319-32025-0_14

Barber, R. F., Candès, E. J., Ramdas, A., & Tibshirani, R. J.

(2023). Conformal prediction beyond exchangeability. *The Annals of Statistics*, *51*(2), 816–845. doi: 10.1214/23-AOS2276

Chen, Z., Wu, M., Zhao, R., Guretno, F., Yan, R., & Li, X. (2021). Machine remaining useful life prediction via an attention-based deep learning approach. *IEEE Transactions on Industrial Electronics*, *68*(3), 2521–2531. doi: 10.1109/TIE.2020.2972443

Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh). *Neurocomputing*, *307*, 72–77. doi: 10.1016/j.neucom.2018.03.067

Compare, M., Bellani, L., & Zio, E. (2017). Reliability model of a component equipped with phm capabilities. *Reliability Engineering and System Safety*, *168*, 4–11. doi: 10.1016/j.ress.2017.05.024

DeGroot, M. H., & Fienberg, S. E. (1983). The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D*, *32*(1-2), 12–22. doi: 10.2307/2987588

de Jonge, B., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*(3), 805–824. doi: 10.1016/j.ejor.2019.09.047

de Oliveira da Costa, P. R., Akcay, A., Zhang, Y., & Kaymak, U. (2020). Remaining useful lifetime prediction via deep domain adaptation. *Reliability Engineering and System Safety*, *195*, 106682. doi: 10.1016/j.ress.2019.106682

Ellefsen, A. L., Bjørlykhaug, E., Åsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering and System Safety*, *183*, 240–251. doi: 10.1016/j.ress.2018.11.027

Fink, O., Wang, Q., Svensén, M., Dersin, P., Lee, W.-J., & Ducoffe, M. (2020). Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, *92*, 103678. doi: 10.1016/j.engappai.2020.103678

Fink, W., & Al Hajhog, K. (2024). Post-clustering prioritization framework for autonomous decision making in the absence of ground truth via hypothetical probing. *International Journal of Prognostics and Health Management*, *15*(1). doi: 10.36001/ijphm.2024.v15i1.3206

Gibbs, I., & Candès, E. (2021). Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, *34*, 1660–1672.

Han, P., & Liang, Q. (2025). Estimating cycles to maintenance events for jet engines using engine-specific measurement residual. In *Annual conference of the phm society* (Vol. 17). doi: 10.36001/phmconf.2025.v17i1.4667

Ifeanyi, A. O., Zanotelli, M., & Coble, J. B. (2024). Data-driven remaining useful life estimation: Inference-based versus direct prediction. *Annual Conference of the PHM Society*. doi: 10.36001/phmconf.2024.v16i1.4071

Inoue, K., Wakimoto, K., Ozeki, K., Kuriyama, T., & Masuzaki, T. (2025). Extended abstract: Remaining cycle estimation based on a maintenance cycle model. In *Annual conference of the phm society* (Vol. 17). doi: 10.36001/phmconf.2025.v17i1.4669

Javanmardi, A., & Hüllermeier, E. (2023). Conformal prediction intervals for remaining useful lifetime estimation. *International Journal of Prognostics and Health Management*, *14*(2). doi: 10.36001/ijphm.2023.v14i2.3417

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*.

Khojandi, A., Maillart, L. M., & Prokopyev, O. A. (2014). Optimal planning of life-depleting maintenance activities. *IIE Transactions*, *46*(7), 636–652. doi: 10.1080/0740817X.2013.849835

Kong, H. B., Jo, S.-H., Jung, J. H., Ha, J. M., Shin, Y. C., & Yoon, H. (2023). A hybrid approach of data-driven and physics-based methods for estimating remaining useful life. *International Journal of Prognostics and Health Management*, *11*(1). doi: 10.36001/ijphm.2020.v11i1.2605

Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., & Siegel, D. (2014). Prognostics and health management design for rotary machinery systems – reviews, methodology and applications. *Mechanical Systems and Signal Processing*, *42*(1-2), 314–334. doi: 10.1016/j.ymssp.2013.06.004

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: 10.1016/j.ymssp.2017.11.016

Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, *172*, 1–11. doi: 10.1016/j.ress.2017.11.021

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

Olde Keizer, M. C. A., Flapper, S. D. P., & Teunter, R. H. (2017). Condition-based maintenance policies for systems with multiple dependent components: A review. *European Journal of Operational Research*, *261*(2), 405–420. doi: 10.1016/j.ejor.2017.02.044

Ong, K. S. H., Niyato, D., & Yuen, C. (2020). Predictive maintenance for edge-based sensor networks: A

deep reinforcement learning approach. In *Ieee world forum on internet of things* (pp. 1–6). doi: 10.1109/WF-IoT48130.2020.9221098

Pankaj, P., Joshi, S., Peng, X., Teramoto, R., & Hearn, T. (2025). Maintenance service events prediction modeling of aircraft gas turbine engines. In *Annual conference of the phm society* (Vol. 17). doi: 10.36001/phmconf.2025.v17i1.4668

PHM Society. (2025). *PHM north america 2025 conference data challenge.* https://data.phmsociety.org/phm-north-america-2025-conference-data-challenge/.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons. doi: 10.1002/9780470316887

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2009). *Dataset shift in machine learning*. MIT Press. doi: 10.7551/mitpress/9780262170055.001.0001

Ramasso, E., & Saxena, A. (2014). Performance benchmarking and analysis of prognostic methods for CMAPSS datasets. *International Journal of Prognostics and Health Management*, *5*(2), 1–15. doi: 10.36001/ijphm.2014.v5i2.2236

Romano, Y., Patterson, E., & Candès, E. (2019). Conformalized quantile regression. *Advances in Neural Information Processing Systems*, *32*.

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *International conference on prognostics and health management* (pp. 1–9). doi: 10.1109/PHM.2008.4711414

Si, X.-S., Wang, W., Hu, C.-H., & Zhou, D.-H. (2011). Remaining useful life estimation – a review on the statistical data driven approaches. *European Journal of Operational Research*, *213*(1), 1–14. doi: 10.1016/j.ejor.2010.11.018

Tibshirani, R. J., Foygel Barber, R., Candès, E., & Ramdas, A. (2019). Conformal prediction under covariate shift. *Advances in Neural Information Processing Systems*, *32*.

Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic learning in a random world*. Springer. doi: 10.1007/b106715

Wang, B., Lei, Y., Li, N., & Li, N. (2020). A hybrid prognostics approach for estimating remaining useful life of rolling element bearings. *IEEE Transactions on Reliability*, *69*(1), 401–412. doi: 10.1109/TR.2018.2882682

Wehrspohn, J., Pohya, A. A., & Wicke, K. (2021). An assessment of the economic viability of engine wash procedures on the lifecycle costs of an aircraft fleet. *PHM Society European Conference*, *6*(1), 14. doi: 10.36001/phme.2021.v6i1.2892

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259. doi: 10.1016/S0893-6080(05)80023-1

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In *Ieee international conference on prognostics and health management (icphm)* (pp. 88–95). doi: 10.1109/ICPHM.2017.7998311