Journal of Scientific Reports , 15(1), 415-434.
ISSN: 2708-7085 (Online) & 3079-9317 (Print)
Journal: ijsab.com/jsr

Hidden Faults Under Partial Sensor Acquisition: Revisiting the AI4I-PMDI Benchmark and the Value of Machine History

Christiana Lebbie [1,*], José da Conceição Ernesto Langa Júnior [1,2], Marvin Sinclair Mkaliainga [1,3], Chernor Illah Jalloh [1]

[1] School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China.
[2] Marshal Samora Machel Military Academy, Office of Study and Planning, Nampula, Mozambique.
[3] National AIDS Commission, ICT Department, Lilongwe, Malawi.
* Corresponding author: Christiana Lebbie (clebbie27@gmail.com)

Abstract

This study audits AI4I-PMDI, a public predictive maintenance benchmark with inspection-dependent missing values built on the AI4I 2020 data, and finds that it is easier than the same data with nothing missing. Its design assigns every failure record to an inspection type that measures that failure's causal sensors, so the published file contains no hidden faults, that is, faults recorded under an inspection that did not measure all the sensors their cause depends on. XGBoost reaches a PR-AUC (the area under the precision-recall curve, which summarises how well faults are ranked above normal records) of 0.878 on the published schedule but only 0.778 when no values are missing. The study separates random dropout, the usual way of simulating missing sensors, from schedule-dependent missingness, in which the inspection plan can leave out exactly the sensors that respond to a fault. To test the second case, the benchmark was rebuilt from the AI4I 2020 values so that a failure can fall under any inspection type. All models were evaluated with five-fold cross-validation grouped by machine, at fixed false-alarm rates (the share of normal records that trigger an alarm), and compared with a paired bootstrap over machines. Under the realistic schedule PR-AUC falls to 0.452, and at one false alarm per 100 normal records only 20% of hidden faults are detected; a model developed on the published schedule detects just 4%. Imputation, missingness indicators and separate models for each inspection type do not close this gap. Adding the last earlier reading of each sensor from the same machine raises PR-AUC by 0.130 (95% CI 0.088 to 0.165) and hidden-fault recall from 20% to 45%. The gain holds across five classifiers, and history features with XGBoost outperform GRU-D, a recurrent model designed for missing observations, on this benchmark. On C-MAPSS FD001 the gain is in the same direction; on MetroPT-3 history ranks windows better but detects leaks later; a combined rule recovers the leaks that history alone misses but does not outperform current readings alone. The benefit comes from the machine's history rather than from the exact way it is encoded. Benchmarks with schedule-dependent missing values should therefore also be evaluated under schedules that can hide faults, and monitoring systems should keep each sensor's last reading for every machine.

***Keywords**: Predictive maintenance, Partial sensor acquisition, Missing data, Hidden faults, Fault identification, Benchmark evaluation, Gradient boosting, AI4I-PMDI.*

ARTICLE INFO

Research paper
Received: 03 August 2026
Accepted: 20 November 2026
Published: 22 November 2026
DOI: 10.58970/JSR.1258

CITATION

Lebbie, C., Langa Júnior, J. da C. E., Mkaliainga, M. S., & Jalloh, C. I. (2026). Hidden Faults Under Partial Sensor Acquisition: Revisiting the AI4I-PMDI Benchmark and the Value of Machine History, Journal of Scientific Reports, 15(1), 415-434.

COPYRIGHT

Copyright © 2026 by author(s) Papers published by IJSAB International are licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Introduction

Condition monitoring in industry rarely reads every sensor at every visit. Route-based inspections, scheduled control procedures and IIoT gateways with limited bandwidth or battery all collect a subset of the available measurements, and the subset can change from one visit to the next (Tsallis et al., 2025; Azeta et al., 2026). A single inspection record therefore describes only part of the machine's state. Missing values are routine in IoT measurement data for this and other reasons, and handling them is a standing problem in practice (Dhungana et al., 2025). The missing part matters most when it contains the fault. If a machine is overheating but the inspection that day measured only torque and tool wear, the record holds little direct evidence of the problem. In this paper such a case is called a hidden fault: a fault recorded under an inspection that did not measure all the sensors its cause depends on. A visible fault, in contrast, is recorded under an inspection that measured all of them. Most recent work on missing sensor data in fault diagnosis studies a different situation. Methods such as MSIFFD (Lin et al., 2025) and MAEFF (Yan et al., 2026) remove channels at random from multichannel vibration signals and rebuild them from the channels observed at the same moment, and a mixture classifier for incomplete sensor variables likewise infers missing readings from the observed variables of the same sample (Liu, Lee, et al., 2025). This works when channels are redundant, but not when the inspection schedule leaves out exactly the sensors that respond to the fault. The distinction between random dropout and schedule-dependent missingness is central here. Under random dropout, missing values are spread evenly over normal and faulty records, and the question is how well gaps can be filled. Under schedule-dependent missingness, the plan decides which values are missing and can tie them to the fault, and the question is whether the evidence exists in the record at all.

The public benchmark closest to this setting is AI4I-PMDI (Autran et al., 2024). It extends the AI4I 2020 dataset (Matzka, 2020) with machine identities, irregular timestamps and three control types, each of which acquires two or three of the five process parameters. The control type of each record was drawn at random, "except for failure samples to keep the logic between measures and labels" (Autran et al., 2024, p. 1204). The authors state that this helps a model distinguish the correct failure, a reasonable aim, but it has a consequence for evaluation: the published file contains no hidden faults. A model tested on it is only ever asked to find faults whose evidence was measured. This study asks what happens when that condition is removed, and its primary contribution is the audit of AI4I-PMDI. The audit shows that every deterministic failure in the published file sits under the one control that measures its causal sensors, which makes the benchmark easier than the same data with no values missing. Three further advances follow. Firstly, the study builds a realistic version of the benchmark from the complete AI4I 2020 values, in which failures can fall under any control, with explicit rules for labelling hidden and visible faults. Secondly, it shows that standard missing-data handling (imputation, missingness indicators and one model per schedule) does not recover hidden faults, whereas each machine's earlier readings do, across five classifiers and on C-MAPSS FD001 and MetroPT-3. Thirdly, it compares history features with GRU-D and shows that the best form of history depends on whether faults appear suddenly or develop slowly. These are research advances rather than procedural steps: each answers an open question about what evaluations measure, which remedies work when evidence is absent from the record, and when they should be expected to fail.
Two evaluation terms are used throughout. Faults are rare (about 3.5% of AI4I records), so accuracy would reward a model that never raises an alarm. PR-AUC, the area under the precision-recall curve, instead summarises how well a model ranks faulty records above normal ones, combining precision (the share of alarms that are real faults) with recall (the share of real faults that raise an alarm) across all thresholds; it is more informative than the ROC curve for rare events (Saito & Rehmsmeier, 2015). In operation a plant must fix a threshold, so the false-alarm rate is fixed in advance, for example at one false alarm per 100 normal records, and recall is read at that threshold. The paper is organised as follows. The literature review identifies the gaps and develops five hypotheses, the research methods section describes the data, audit, schedules, representations and evaluation, and the results and discussion section tests each hypothesis, before applications, limitations, future directions and conclusions.

Literature Review

Missing sensor data in fault diagnosis

Several recent methods treat missing channels in rotating-machinery diagnosis. MSIFFD uses a learnable prompt to reconstruct missing features from the remaining signals and prior knowledge, and was evaluated with up to three sensor channels missing (Lin et al., 2025). MAEFF fuses missing-aware experts and trains with injected sensor dropout, temporal gaps and point-wise loss on two bearing datasets (Yan et al., 2026). Fu and Yan (2024) reconstruct each sensor from the others in a sliding window with a masked model. Outside rotating machinery, Pierros et al. (2024) study predictive maintenance when some sensors are absent, and Xiao et al. (2025) use medium- to long-term predictions to diagnose and recover faulty sensors in a workshop sensor network. Dhungana et al. (2025) review imputation choices for IoT measurements and give practical guidelines. These methods differ in how they handle the missing information. MSIFFD rebuilds missing features through a learnable prompt that combines the available signals with prior knowledge, and was tested with up to three sensor channels missing (Lin et al., 2025). MAEFF takes a different route: instead of reconstructing or imputing signals, it builds missingness awareness into the fusion step, combining reliability-guided multi-sensor fusion with relation-aware residual correction, and is trained on data with injected sensor dropout, temporal gaps and sporadic point-wise loss; it was tested on two public multi-sensor datasets (Yan et al., 2026). In these studies, missingness is simulated by removing channels or segments from multi-sensor recordings, and the observed signals are recorded at the same time as the missing ones, so the methods can draw on the other sensors for the same fault signature. That is a different situation from an inspection plan that leaves out exactly the sensors that respond to a fault, which is the situation studied here. Read together, these methods define the current standard for incomplete channels, and each rest on an assumption that the present setting breaks. MSIFFD reconstructs up to three channels through a learnable prompt (Lin et al., 2025) and MAEFF trains with injected sensor dropout and temporal gaps (Yan et al., 2026). In both the channels are multichannel vibration signals from rotating machinery, in which every channel views the same machine and neighbouring channels are strongly related, the condition under which reconstruction from the same moment is expected to work. A route-based inspection of a milling machine differs: temperatures, rotational speed, torque and tool wear are different physical quantities, several are never measured together, and a fixed plan, not chance, selects them. What these methods leave open is the case in which no observed channel is related to the missing one, so that the only remaining evidence lies at another point in time. These studies share two features. Missing information is recovered from what is observed at the same time, or forecast from a model of normal behaviour, and missingness is usually simulated at random. Random dropout spreads the missing values evenly over normal and faulty records, so a model rarely faces a fault whose decisive sensor is always absent. None of these studies tests faults placed under a schedule that leaves the fault's own sensors unobserved. Random deletion may also overstate performance: imputation was markedly more accurate when values were missing completely at random than under more realistic mechanisms (Toye et al., 2025).

Learning with missing values

For supervised learning, Josse et al. (2024) show that constant imputation combined with a powerful learner can be consistent, and that trees which route missing values directly are a sound default. Stempfle et al. (2025) train models that avoid relying on features likely to be missing at test time, and Le Morvan and Varoquaux (2025) find that imputation accuracy matters less when the learner is expressive and an indicator is supplied. For irregular time series, GRU-D decays each input towards its last observed value or the mean as time passes (Che et al., 2018), simple recurrent and patch-based representations remain competitive (Joshi & Hauskrecht, 2025; Liu, Qiu, et al., 2025), and learned time encodings can capture the pattern of missingness (Lee et al., 2025; Kumar & Gautam, 2026; Nozaki et al., 2026). The history features used in this study are close to last observation carried forward (LOCF) with indicators, a method that these works use as a baseline. They are used here to ask whether history can supply evidence that the current inspection lacks, not to propose a new imputation method.

Industrial IoT monitoring and tree-based models

Predictive maintenance systems increasingly sit on industrial IoT infrastructure that decides what is measured and when (Li et al., 2025; Tamizhamuthan et al., 2026; Kalpakam & Nisha, 2026). Bandwidth, battery and inspection effort limit how many sensors report at each visit, which is exactly the condition under which a fault's own evidence can go unrecorded. Maddali (2025) takes a reliability-engineering view, in which missed faults cost more than false alarms. Gradient-boosted trees are a common choice for tabular sensor data (Cureño Osornio et al., 2024) and a standard baseline on AI4I (Alnahhal et al., 2026). Tuned gradient-boosted trees remain competitive on broad tabular benchmarks (Erickson et al., 2025), although tabular foundation models now match or exceed them on many small datasets (Prior Labs Team, 2025; Qu et al., 2026). XGBoost has a further advantage here: it learns a default direction for missing values at each split, so it can use the pattern of missingness without imputation (Chen & Guestrin, 2016), which makes it a strong and fair baseline against which to measure whether history adds anything.

Predictive maintenance benchmarks

AI4I 2020 (Matzka, 2020) is widely used to test predictive maintenance models, including explainable approaches (Rajaoarisoa et al., 2025; Dereci & Tuzkaya, 2024) and methods for class imbalance (Alnahhal et al., 2026). C-MAPSS (Saxena et al., 2008) simulates turbofan engines run to failure and remains a standard test bed (Özcan, 2026). MetroPT-3 (Veloso et al., 2022) provides real air-compressor data from a metro train with recorded failure events. AI4I-PMDI (Autran et al., 2024) is newer. Studies that report results on it use the published schedule, and FusionNet excluded it from evaluation (Sharma et al., 2026). Reported scores also depend on evaluation: leakage-prone setups inflate AUC on AI4I 2020 by up to 8 to 9 percentage points (Qahtani, 2026), improper train-test separation flatters bearing-fault classifiers (Matania et al., 2024), and leakage-safe protocols have been proposed (Knap et al., 2026).

Gaps in existing literature

Recent work shows clear progress. Channel-recovery methods rebuild missing vibration channels from those observed at the same moment (Lin et al., 2025), missing-aware training tolerates sensor dropout (Yan et al., 2026), and work on supervised learning explains when simple imputation or native routing is enough (Josse et al., 2024). Together they show that missing sensor data can be handled well when the remaining data still carry the fault's signature. What remains uncertain is narrower and, for practice, more important. No study has asked what AI4I-PMDI's scores measure, given that its failure records are always placed under a control that sees their causes. It is not known whether standard missing-data methods help when a fault's decisive sensor is absent by design, nor whether the machine's own earlier readings can stand in for skipped sensors, and for which faults. Finally, it is unclear whether any benefit carries over to data with genuine time structure, and whether history features or sequence models suit it better. These questions lead to the hypotheses below.

Hypothesis development

The first hypothesis concerns the benchmark itself. Autran et al. (2024) state that failure records were given the control that measures their causes, and that adding the control type improves classification because each control acquires the parameters behind certain failures only. If the control type acts as a label for the failure mode, a model can use it to narrow the search and may do better with missing values than without them. Complete data contain no such hint. This leads to the first hypothesis:

H1. *A model trained and tested on the published AI4I-PMDI schedule achieves a higher PR-AUC than the same model on complete data with no missing values.*

The second hypothesis concerns standard missing-data methods, and its prediction follows from two lines of work. Josse et al. (2024) show that a powerful learner applied after constant imputation can be consistent, and that tree methods which route missing values directly are a sound default. If the

decisive sensor is never observed for a fault, that best prediction is itself limited, so a method can at most match native routing. Zhao et al. (2025) and Le Morvan and Varoquaux (2025) show that better imputation helps only modestly, and any imputation model learns a missing value from the other values of the same record, so it needs records in which the sensors were measured together. In AI4I-PMDI some sensor pairs are never measured together. Separate models per schedule face the same limit, since each sees only its own control's parameters. These results predict that none of the standard methods will improve on native routing when the decisive sensor is absent by design. This leads to the second hypothesis:

H2. *Under a realistic schedule, median imputation, imputation with missingness indicators, iterative imputation and per-schedule models do not improve PR-AUC over a gradient-boosted model that routes missing values natively.*

The third hypothesis concerns machine history. The theory that limits imputation also shows how to relax the limit: adding information that bears on the unmeasured sensor can raise the achievable performance (Josse et al., 2024). The machine's own earlier inspection supplies such information along the time axis, where in-record methods have none, and it is cheap, since the last reading of each sensor is already stored. GRU-D carries the last observed value forward and lets its influence fade with time (Che et al., 2018), and such simple representations are competitive (Joshi & Hauskrecht, 2025). If a fault's cause builds up over time, an earlier reading of the skipped sensor should carry part of the evidence the current inspection lacks. This leads to the third hypothesis:

H3. *Adding the last earlier reading of each sensor from the same machine increases recall on hidden faults at a fixed false-alarm rate, compared with current readings alone.*

The fourth hypothesis concerns where any gain should appear. An earlier reading can stand in for a current one only if the quantity changes slowly between inspections. In AI4I, temperatures and tool wear evolve gradually, whereas rotational speed and torque are drawn afresh for each record (Matzka, 2020). This leads to the fourth hypothesis:

H4. *The gain from history is concentrated in failure modes whose causal sensors change slowly (heat dissipation and overstrain failures) and is small or negative for power failures, whose causal sensors change quickly.*

The fifth hypothesis concerns generality and the form of history. AI4I's time structure is synthetic, so a result that holds only there would say little about real equipment. C-MAPSS provides simulated but physically motivated degradation (Saxena et al., 2008), and MetroPT-3 provides real compressor data (Veloso et al., 2022). When a fault is a threshold event on current values, as in AI4I, the most recent reading already holds the relevant information and a recurrent model has little to add. When degradation is slow and noisy, as in C-MAPSS, a single earlier reading contains noise that a model reading the whole sequence can average out, and recurrent models remain competitive on irregular data (Joshi & Hauskrecht, 2025). History should therefore help on both kinds of data, with a last reading best for threshold events and a sequence model best under slow degradation. This leads to the fifth hypothesis:

H5. *The benefit of history replicates on C-MAPSS FD001 and MetroPT-3, while the best form of history depends on how faults develop, with sequence models favoured under slow degradation.*

Research Methods

The study follows the pipeline in Figure 1. It starts from the public AI4I data, audits how the published schedule places failures, builds realistic schedules from the complete values, labels each fault as visible or hidden, and compares input representations and models under a common grouped evaluation. The main findings are then repeated on two further datasets.

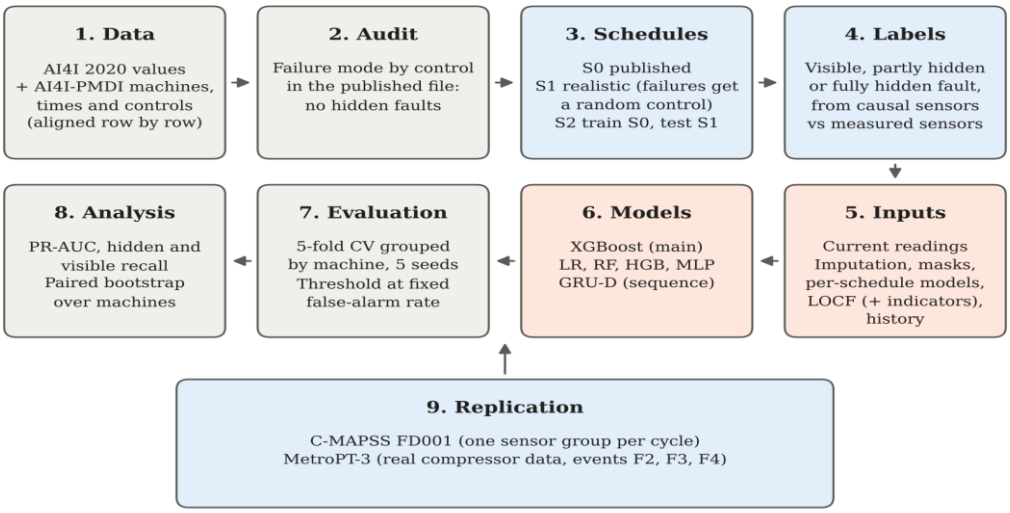


Figure 1. Study pipeline, from the data and the audit of the published schedule to the realistic schedules, fault labels, inputs, models, evaluation and replication.

AI4I 2020 and AI4I-PMDI

AI4I 2020 contains 10,000 records of a milling process with five parameters (air temperature, process temperature, rotational speed, torque and tool wear), a product quality type (L, M or H) and five failure modes: tool wear failure (TWF), heat dissipation failure (HDF), power failure (PWF), overstrain failure (OSF) and random failure (RNF) (Matzka, 2020). HDF, PWF and OSF are defined by thresholds on functions of the parameters, TWF by tool wear, and RNF occurs at random. AI4I-PMDI keeps the same records and adds a machine identifier (120 machines), a timestamp, a control type and a diagnostic label (Autran et al., 2024). Each control acquires a fixed subset of parameters (Table 1), and the other values are missing. The two files were aligned row by row on UDI and product ID, and every observed value in AI4I-PMDI matches AI4I 2020. The AI4I-PMDI diagnostic is used as the binary label (failure or not), giving 348 failures. The labels differ from AI4I 2020 in 27 records: AI4I-PMDI marks as failures the 18 random-failure rows that AI4I 2020 lists with no machine failure, and marks as normal the 9 rows that AI4I 2020 lists as failures with no failure mode. Twenty-four failure records carry more than one failure mode.

Table 1. Parameters acquired by each control type in AI4I-PMDI

Control	Air temp.	Process temp.	Rot. speed	Torque	Tool wear	Reveals in full
A	Yes	Yes	Yes	No	No	HDF
B	No	No	Yes	Yes	No	PWF
C	No	No	No	Yes	Yes	OSF, TWF

Source: Autran et al. (2024); last column from the failure definitions in Matzka (2020).

Problem formulation

Let machine m be inspected at times t = 1, ..., T_m. At each inspection the full parameter vector is $x_{m,t}$ = ($x_{m,t,1}$, ..., $x_{m,t,5}$), but only the parameters in the set O(c) acquired by the control $c_{m,t}$ are recorded. The observed record r is

$$r_{m,t,j} = \begin{cases} x_{m,t,j}, & j \in O(c_{m,t}) \\ \text{missing}, & \text{otherwise} \end{cases} \quad (1)$$

Each deterministic failure mode k depends on a set of causal parameters K(k): heat dissipation on air temperature, process temperature and rotational speed; power failure on rotational speed and torque; overstrain on torque and tool wear; and tool wear failure on tool wear. A failure of mode k recorded under control c is then classed as

$$\text{status} = \begin{cases} \text{visible}, & K(k) \subseteq O(c) \\ \text{fully hidden}, & K(k) \cap O(c) = \emptyset \\ \text{partly hidden}, & \text{otherwise} \end{cases} \quad (2)$$

The history feature for parameter j is its most recent earlier measured value on the same machine,

$$h_{m,t,j} = x_{m,s,j}, \quad s = \max\{u < t : j \in O(c_{m,u})\} \quad (3)$$

and it is missing when no earlier inspection measured j. The history model receives the current record and the history features side by side,

$$z_{m,t} = [r_{m,t},\ h_{m,t},\ \text{product type}] \quad (4)$$

so that the model can tell a current reading from an earlier one. The task is binary: predict whether record (m, t) is a failure.

Audit of the published schedule

Table 2 and the left panel of Figure 2 show failure modes by control in the published file. Every heat dissipation failure is recorded under control A, every power failure under B, and every overstrain and tool wear failure under C. In each case this is the only control that measures all the parameters the failure depends on. Only random failures, which have no causal parameters, appear under all three controls. On average a record has 2.34 of its 5 parameters observed, so 53.1% of values are missing, yet no deterministic failure is ever hidden. The control type therefore acts as a strong hint about which failure to look for.

Table 2. Failure modes by control type in the published AI4I-PMDI file

Failure mode	Control A	Control B	Control C
HDF	106	0	0
PWF	0	83	0
OSF	0	0	98
TWF	0	0	42
RNF	4	8	7
Failure rate	3.20%	2.81%	4.43%

Source: *Authors' analysis.*

Realistic schedules

Three evaluation settings were built from the aligned data. The first, S0 (published), uses the AI4I-PMDI controls and missing values as released. The second, S1 (realistic), gives each failure record a control drawn uniformly at random from A, B and C, while normal records keep their published control; values are then masked from the complete AI4I 2020 data using the mapping in Table 1, and a new draw is made for each random seed.

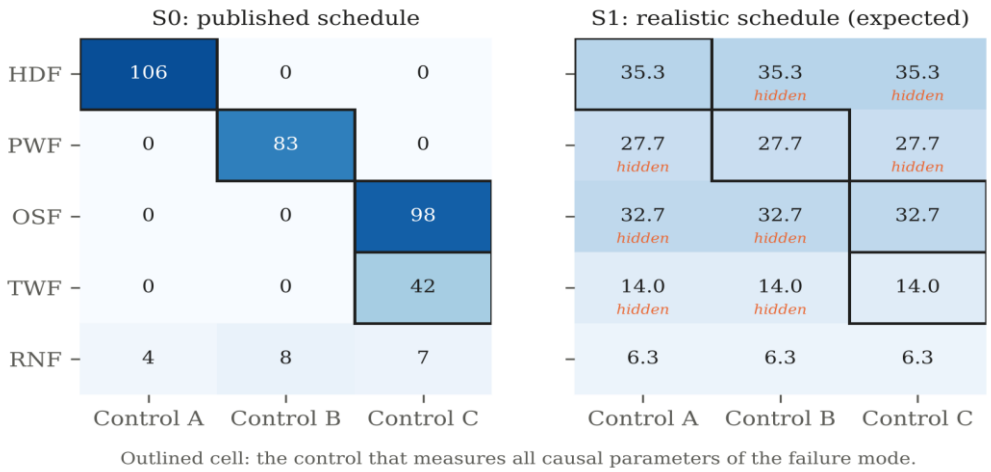


Figure 2. Failure records by mode and control under the published schedule S0 (left) and the expected counts under the realistic schedule S1 (right). Outlined cells mark the control that measures all causal parameters of each failure mode; other cells for deterministic modes hold hidden faults.

The third, S2 (transfer), trains models under S0 and tests them under S1, which mimics a model developed on the benchmark and then deployed where inspections do not follow the failure. As a reference, models were also trained and tested on the complete data with no values missing. Normal records keep the published schedule in S1 so that the only change is where failures fall. Figure 2 shows the difference between S0 and S1.

Hidden and visible faults

Under S1 each deterministic failure is labelled as follows. A fault is visible if it is recorded under the control that observes all its causal parameters (HDF under A, PWF under B, OSF and TWF under C). Otherwise it is hidden. A hidden fault is fully hidden if none of its causal parameters is observed and partly hidden if some are. RNF records are neither hidden nor visible and count only in overall results. A record's status follows the failure mode in its AI4I-PMDI diagnostic label. For the 24 records that carry more than one AI4I failure flag this is the labelled mode, and the single-mode check reported below removes all multi-mode records. Averaged over seeds, S1 contains 217.8 hidden faults (120.6 partly and 97.2 fully hidden), 111.2 visible faults and 19 random failures among 9,652 normal records. As a check, the main analysis was repeated on single-mode failures only.

Record order within machines

History features need the order of inspections for each machine. In AI4I-PMDI each machine occupies a contiguous block of UDI numbers, and timestamps within a machine follow UDI order closely (median Spearman rank correlation 0.998, minimum 0.80). Consecutive inspections of a machine are a median of 43 hours apart, and three pairs share a timestamp. About a third of consecutive pairs are swapped when sorted by timestamp rather than UDI. Records are sorted by timestamp, which is the order a deployed system would see, and UDI order is reported as a sensitivity check. The history in AI4I is partly synthetic. Within a machine, temperatures and tool wear are strongly autocorrelated (0.82 to 0.99), because the original data generates them as slowly changing series, while rotational speed and torque are close to independent from one record to the next. Tool wear drops (a tool change) at about a third of steps, and a drop is no more likely after a failure than after a normal record. The findings were therefore also tested on data whose time structure is genuine.

Replication datasets

C-MAPSS FD001 (Saxena et al., 2008) contains 100 simulated turbofan engines run to failure (20,631 cycles, one operating condition, one fault mode). A cycle is labelled positive if the engine is within 30 cycles of failure, which gives 3,100 positives (15%). This is current health-state identification, not prediction. The 14 informative sensors were kept and placed in three groups: temperatures and pressures (m2, m3, m4, m7, m11), shaft speeds (m8, m9, m13, m14) and flow-related channels (m12, m15, m17, m20, m21). Each cycle reports one group chosen at random.

Table 3. Datasets used in the study

Dataset	Source of data	Units	Records	Positives	Sensors	Partial acquisition
AI4I 2020 with AI4I-PMDI schedules	Synthetic milling process	120 machines	10,000	348 failures (3.5%)	5	Control A, B or C; 2 or 3 of 5 sensors per record
C-MAPSS FD001	Simulated turbofan engines	100 engines	20,631 cycles	3,100 cycles (15%)	14	One of three sensor groups per cycle, at random
MetroPT-3	Real air compressor, metro train	1 compressor, Feb to Sep 2020	25,256 ten-minute windows	4 air leaks (498 windows); F2, F3, F4 tested	3 groups	One of three signal groups per window, at random

Source: Matzka (2020), Autran et al. (2024), Saxena et al. (2008), Veloso et al. (2022); authors' set-up.

MetroPT-3 (Veloso et al., 2022) contains real sensor data from the air compressor of one metro train, recorded from February to September 2020, with four reported air leaks. The data were aggregated into 10-minute windows, giving 25,256 windows of which 498 fall inside a leak. Each window is treated as an inspection that reports one of three signal groups chosen at random: pressures (TP2, TP3, H1, DV pressure and reservoirs), motor current and oil temperature, or valve and switch states. Evaluation is forward in time. For each of the leaks F2, F3 and F4, the model is trained only on data recorded before the test period, which includes the earlier leaks, and tested on the period around that leak, so no future data enter training. The F2, F3 and F4 test periods contain 2,750, 3,070 and 7,750 windows, with 40, 286 and 28 leak windows respectively. Table 3 summarises the three datasets and how partial acquisition is applied to each.

Representations compared

Each record of a machine contains the parameters measured by its control, with the others missing, and all inputs include the product type. Five inputs correspond to standard ways of handling missing values. The current input leaves the measured parameters as they are and the unmeasured ones missing, and XGBoost learns a default branch for missing values at each split. Median imputation replaces missing values by the training median, and median plus mask adds one binary indicator per parameter marking imputed values. Iterative imputation applies chained-equation regression imputation (scikit-learn IterativeImputer; Pedregosa et al., 2011) fitted on training data. Per-schedule models train a separate model for each control type on records of that type. Three inputs use the machine's earlier readings. LOCF replaces each unmeasured parameter by its most recent earlier measured value from the same machine. LOCF plus indicators adds one binary indicator per parameter showing that the value was carried forward, and a variant also adds the age of the carried value in inspections. The history input leaves the current input as it is and adds, for every parameter, the last earlier measured value from the same machine as a separate feature. The history and LOCF features use only records strictly earlier than the current one for the same machine. An automated test confirmed this: features recomputed from an explicit list of earlier observations matched the vectorised implementation for 300 random records, and altering every same-time or later observation of the machine, and all data of other machines, left the history features of 40 records unchanged, on AI4I under the realistic schedule, with added sensor outages and on C-MAPSS.

Models

The main model is XGBoost (Chen & Guestrin, 2016) with 300 trees, maximum depth 4 and learning rate 0.05, without tuning, so that differences come from the inputs rather than from search effort. To check that the findings do not depend on one learner, logistic regression, random forest, histogram gradient boosting and a multilayer perceptron with layers of 64 and 32 units were also trained (scikit-learn; Pedregosa et al., 2011). Models without native missing-value handling receive median imputation with indicators. GRU-D (Che et al., 2018) reads each machine's full sequence of inspections. Its inputs are the readings, the missingness mask and, for each sensor, the time since it was last observed; missing inputs and the hidden state decay towards the last observation or the empirical mean as that time grows. It was trained with a fixed configuration (Table 4) and early stopping on 20% of the training machines, using the same folds and schedules as the other models. Its hyperparameters were not searched. It was left untuned so that all models receive the same treatment, because a search nested in every fold and seed would be costly for a recurrent model (3.9 minutes per seed on AI4I and 14.2 on FD001 on a laptop), and because the aim is to show which form of history suits which kind of fault, not the best achievable GRU-D score. A tuned GRU-D could close part of the gap on AI4I, so the comparison is read as a direction, as noted in the limitations.

Table 4. GRU-D settings used in all experiments

Setting	Value
Hidden units	64
Optimiser	Adam, learning rate 0.005, weight decay 0.0001
Batch size	8 machines per batch
Maximum epochs	150
Early stopping	Validation PR-AUC on 20% of training machines, patience 15 epochs
Loss	Binary cross-entropy with a positive-class weight

Source: *Authors' analysis.*

Evaluation protocol

All AI4I experiments use five-fold StratifiedGroupKFold with machines as groups, so no machine appears in both training and test data. Results are averaged over five random seeds; the seed controls the S1 control draw and the fold assignment. C-MAPSS folds are grouped by engine. PR-AUC (average precision), explained in the introduction, is the main ranking metric, because failures are about 3.5% of records and the precision-recall curve is more informative than the ROC curve for rare events (Saito & Rehmsmeier, 2015). For decisions a false-alarm rate is fixed. Within each outer fold, the alarm threshold is set from out-of-fold scores of training normal records only, at 0.5, 1 or 2 false alarms per 100 normal records. At that threshold, recall on hidden, visible and all faults is reported, together with precision. For MetroPT-3 the target is one false alarm per day, set on the latest 20% of training normal data, and detection delay is reported for each event. Detection delay is the time from the reported start of a leak to the end of the first alarming window inside it, averaged over the runs in which the leak was detected; false alarms are alarming windows outside the leak periods, expressed per day of the test period. With TP, FP and FN the numbers of true positives, false positives and false negatives, and N the number of normal records, the measures are

$$\text{Precision} = \frac{TP}{TP+FP}, \qquad \text{Recall} = \frac{TP}{TP+FN} \tag{5}$$

$$\text{FA}_{100} = 100 \times \frac{FP}{N} \tag{6}$$

PR-AUC is computed as average precision over the ranked scores,

$$\text{PR-AUC} = \sum_n (R_n - R_{n-1})\, P_n \tag{7}$$

where P_n and R_n are precision and recall at the n-th threshold. For a target of a false alarms per 100 normal records, the threshold is the $(1 - a/100)$ quantile of the out-of-fold scores of training normal records:

$$\tau = Q_{1-a/100}(\text{scores of training normal records}) \tag{8}$$

Recall on hidden faults uses only hidden faults as positives, and recall on visible faults only visible ones, both at the same threshold τ. Table 5 summarises the experimental design.

Table 5. Experimental design

Factor	Levels
Schedules	S0 published; S1 realistic; S2 trained on S0, tested on S1; complete data
Input representations	Current; median imputation; median plus mask; iterative imputation; per-schedule models; LOCF; LOCF plus indicators (with and without age); history
Classifiers	XGBoost (main); logistic regression; random forest; histogram gradient boosting; multilayer perceptron; GRU-D
Validation	5-fold StratifiedGroupKFold by machine (by engine for C-MAPSS)
Repetitions	5 seeds (3 seeds for the classifier comparison and GRU-D)
Decision thresholds	0.5, 1 and 2 false alarms per 100 normals; 1 per day for MetroPT-3
Statistics	Paired bootstrap over machines, 1,000 resamples, 95% percentile intervals

Source: *Authors' experimental design.*

Statistical comparison

Methods are compared with a paired bootstrap over machines (Efron & Tibshirani, 1993). Each of 1,000 resamples draws machines with replacement, and the difference between two methods is computed on the same resampled machines, averaged over the five seeds. Trained models, predictions, thresholds and schedules are held fixed; only the set of machines is resampled, so the intervals describe uncertainty over machines. The mean difference and the 95% percentile interval are reported. Analyses ran in Python 3.13 with XGBoost 3.4.1, scikit-learn 1.9.1 and pandas 3.0.6.

Results and Discussion

The published benchmark is easier than complete data

Table 6 compares the four settings on the AI4I-PMDI data (Autran et al., 2024). On the published schedule XGBoost reaches a PR-AUC of 0.878, well above the 0.778 obtained when every parameter is observed. Adding physics features (temperature difference, power and strain) to the complete data raises it to 0.86, still below the published schedule. A model with half the values missing does better than one with none missing because the control type tells it which kind of failure to expect. Under the realistic schedule S1, PR-AUC falls to 0.452. When a model trained on S0 is deployed under S1, PR-AUC is 0.401 and only 4% of hidden faults are detected. A model that looks strong on the published file would therefore miss almost every fault that an inspection fails to see.

Table 6. XGBoost on AI4I under each setting (PR-AUC, mean ± SD over five seeds)

Setting	PR-AUC	Hidden-fault recall
Complete data, nothing missing	0.778 ± 0.015	n/a
S0, published schedule	0.878 ± 0.008	n/a (no hidden faults)
S1, realistic schedule	0.452 ± 0.022	0.20
S2, trained on S0, tested on S1	0.401 ± 0.019	0.04

Source: *Authors' analysis.*

Standard missing-data handling does not help

Figure 3 compares the standard methods under S1. All fall between 0.42 and 0.45, within about one standard deviation of the current-reading baseline. Adding missingness indicators to XGBoost gave predictions identical to native missing-value routing. Iterative imputation placed about 38% of imputed test values outside the range seen in training, because some parameter pairs, such as air temperature and torque, are never measured together, so the imputation model has no joint data to learn from. This agrees with the theory that constant or model-based filling cannot add information the record lacks (Josse et al., 2024). Plain LOCF also gives no gain: when carried values are mixed in with current ones, the model cannot tell an old reading from a new one.

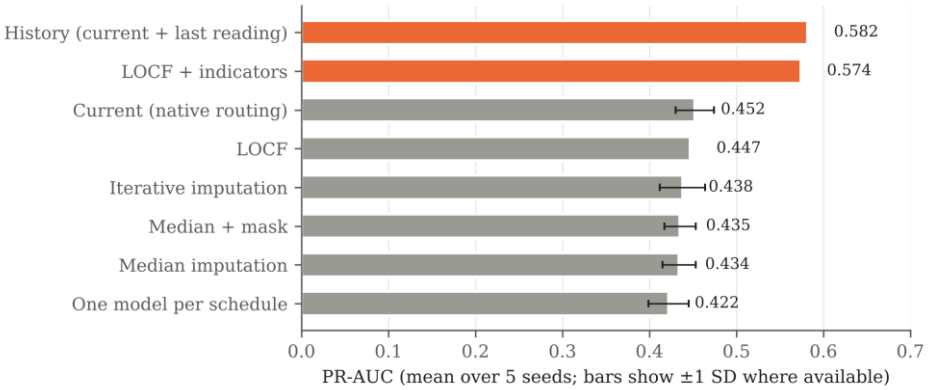


Figure 3. PR-AUC under the realistic schedule S1 for standard missing-data methods (grey) and history-based inputs (orange). XGBoost, mean over five seeds; error bars show ±1 SD where available.

Machine history recovers hidden faults

Adding the history features raises PR-AUC from 0.452 to 0.582, a difference of +0.130 (95% CI 0.088 to 0.165). LOCF with indicators scores 0.574. The difference between the history representation and

LOCF with indicators is +0.008 (95% CI -0.010 to 0.024), and none of the intervals for that comparison excludes zero, so history shows no clear advantage over LOCF with indicators; this does not show that the two are equivalent. The gain comes from giving the model earlier readings and marking them as earlier, not from the particular layout of the features. Table 7 shows recall at fixed false-alarm rates. At one false alarm per 100 normal records, hidden-fault recall rises from 19.9% to 44.9% (difference +25.0 points, 95% CI 18.7 to 31.5). Recall on visible faults falls from 81.0% to 68.2%, which is examined below. Restricted to hidden faults and normal records, PR-AUC doubles from 0.20 to 0.45. Hidden-fault recall is an aggregate over partly hidden and fully hidden faults and is not split between them.

Table 7. Recall (%) at fixed false-alarm rates under S1 (XGBoost, mean over five seeds)

False alarms per 100 normals	Method	Hidden	Visible	All faults	Precision
0.5	Current	14.8	72.5	32.5	0.71
0.5	LOCF + indicators	33.6	57.2	39.6	0.75
0.5	History	33.1	58.7	39.6	0.75
1	Current	19.9	81.0	38.6	0.60
1	LOCF + indicators	43.3	70.2	50.1	0.64
1	History	44.9	68.2	50.3	0.65
2	Current	28.0	86.1	45.4	0.46
2	LOCF + indicators	54.1	77.9	59.3	0.52
2	History	57.1	77.6	61.1	0.52

Source: *Authors' analysis.*

Two checks support the result. On single-mode failures only, hidden-fault recall at one false alarm per 100 is 18.1% (current), 41.6% (LOCF with indicators) and 43.5% (history), against 19.9%, 43.3% and 44.9% on all records. Sorting inspections by UDI instead of timestamp changes 46% of carried values but changes the history PR-AUC only from 0.582 to 0.595 and hidden recall from 44.9% to 47.0%.

Where the gains and losses come from

The gains are concentrated in faults whose causal parameters change slowly. Detection of heat dissipation failures recorded under control B rises from 10% to 66%, overstrain failures under B from 27% to 80%, and overstrain failures under A from 8% to 42%. Temperatures and tool wear carry over well between inspections, as their definition in AI4I implies (Matzka, 2020). Hidden power failures, which depend on rotational speed and torque, lose slightly (under A from 33% to 29%, under C from 67% to 62%), because those two parameters are close to independent from one record to the next. Summed over the five seeds, 307 hidden-fault detections were gained and 35 lost, 28 of them power failures. Visible recall drops from 81% to 68% probably because both models share one false-alarm budget: if history raises the scores of some normal records, the threshold moves up and a few visible faults fall below it. This explanation was not tested directly, because the two models are trained separately and their scores are on different scales. A simple alternative is to combine the two models and raise an alarm when either exceeds its own threshold, with the thresholds set jointly to keep the total false-alarm rate at one per 100. This combined rule detects 41.0% of hidden and 78.5% of visible faults (50.9% overall, precision 0.66) at 0.94 false alarms per 100 normals. On this benchmark the combined rule therefore trades hidden-fault recall (44.9% to 41.0%) for visible-fault recall (68.2% to 78.5%). Without recalibration it reaches 48.1% and 83.4%, but at 1.5 false alarms per 100.

Other classifiers

Figure 4 shows that history helps every classifier on hidden faults. Hidden-fault PR-AUC rises from 0.140 to 0.241 for logistic regression, 0.207 to 0.421 for random forest, 0.194 to 0.429 for histogram

gradient boosting, 0.233 to 0.334 for the multilayer perceptron and 0.209 to 0.446 for XGBoost. On all faults the multilayer perceptron is the exception (0.470 to 0.459). In the 20 model and dataset combinations tested, adding the age of each carried value gave no consistent benefit (judged on unrounded values, it improved none, made no difference in 14 and lowered the score in 6), so it is left out; this applies to these combinations only.

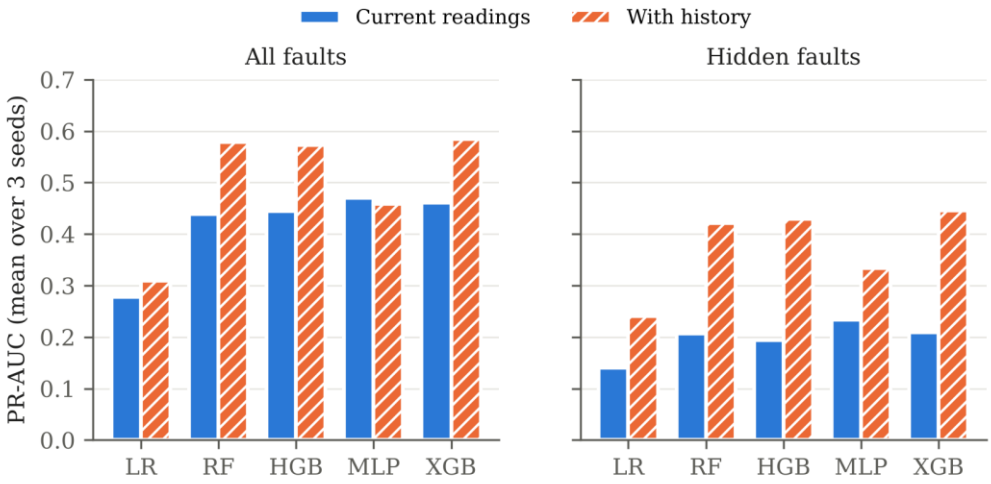


Figure 4. PR-AUC with current readings and with history for five classifiers under S1, on all faults (left) and hidden faults (right). Mean over three seeds. LR, logistic regression; RF, random forest; HGB, histogram gradient boosting; MLP, multilayer perceptron; XGB, XGBoost.

Replication on C-MAPSS and MetroPT-3

On C-MAPSS FD001 (Table 8; Saxena et al., 2008), partial acquisition costs less than on AI4I because every sensor group carries some degradation signal. Standard methods again give no gain, and history raises PR-AUC from 0.911 to 0.938 (difference +0.027, 95% CI 0.019 to 0.035), better in all five seeds. This recovers 74% of the gap between partial and complete data.

Table 8. C-MAPSS FD001 with one sensor group per cycle (XGBoost, mean ± SD over five seeds)

Method	PR-AUC	Recall	Precision	False alarms per 100 healthy
Complete data	0.947	0.85	0.85	2.7
Current	0.911 ± 0.002	0.84	0.78	4.2
Median imputation	0.910 ± 0.002	0.84	0.78	4.3
Missingness indicators	0.911 ± 0.002	0.84	0.78	4.2
One model per schedule	0.905 ± 0.002	0.84	0.77	4.4
History	0.938 ± 0.003	0.85	0.82	3.4

Source: *Authors' analysis.*

On MetroPT-3 (Table 9), history raises the average PR-AUC over the three events from 0.491 to 0.540, with the largest gain on F3; each event is evaluated under five seeds, giving 15 event-seed pairs. Detection timing tells a different story. At one false alarm per day, the current-reading model detects every event in every seed with a mean delay of 38 minutes. The history model detects 13 of 15 event-seed pairs with a mean delay of 125 minutes, because at the start of a leak the last earlier reading still shows normal operation. Thresholds set on calibration data gave 0.3 to 1.0 false alarms per day there, but 5 to 9 per day during the F2 test period, a sign of drift in the real data. The two results are consistent. PR-AUC ranks windows over the whole test period, including the middle of an event, where the carried reading has already moved into the abnormal range. Detection delay is set at the onset, when the carried reading still shows normal operation. History therefore improves the ranking but delays detection, and the combined rule uses the current-reading model for onset and the history model for the established fault, which detects all 15 pairs at a mean delay of 55 minutes. The

combined rule therefore improves on the history model's misses and delay but does not outperform the current-reading model, which also detects all 15 pairs, with a shorter mean delay (38 minutes) and fewer test false alarms per day (2.1 against 3.0). A higher PR-AUC with history does not by itself show better practical alarming.

Table 9. MetroPT-3 results per event (XGBoost, five seeds; threshold at one false alarm per day)

Method	PR-AUC F2	PR-AUC F3	PR-AUC F4	Average PR-AUC	Events detected (of 15)	Mean delay (min)	Test false alarms per day
Current	0.238	0.906	0.330	0.491	15	38	2.1
History	0.255	0.983	0.383	0.540	13	125	2.8
Combined rule	n/a	n/a	n/a	n/a	15	55	3.0

Source: *Authors' analysis.*

Comparison with GRU-D

GRU-D (Che et al., 2018) is well below the history features with XGBoost on AI4I, and even below current readings alone (Figure 5). On C-MAPSS FD001 it is the best method. The two datasets differ in how faults arise. AI4I failures are threshold events: a failure occurs when a function of the current parameters crosses a limit, so the most recent reading of each parameter is what matters. C-MAPSS engines degrade slowly and noisily, so a model that reads the whole sequence can average out the noise. XGBoost with history features trains in seconds, while GRU-D took 3.9 minutes per seed on AI4I and 14.2 minutes on FD001 on a laptop.

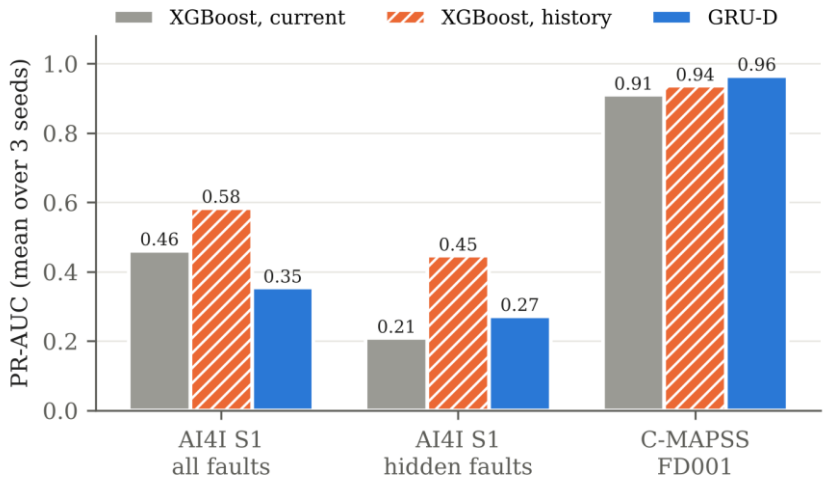


Figure 5. XGBoost with current readings or history compared with GRU-D (PR-AUC, mean over three seeds).

Hypothesis results

Table 10 summarises the outcome of each hypothesis; the evidence is given below. H1 (published schedule inflates performance). XGBoost reached a PR-AUC of 0.878 ± 0.008 on the published schedule and 0.778 ± 0.015 on complete data (Table 6). The gap of 0.10 is more than six times the larger seed standard deviation, and it remained when physics features were added to the complete data (0.86). Supported.

H2 (standard methods do not help). Under S1, median imputation (0.434), median plus mask (0.435), iterative imputation (0.438) and per-schedule models (0.422) all scored at or below native routing (0.452), and missingness indicators gave identical predictions to native routing (Figure 3). Supported.

H3 (history raises hidden-fault recall). At one false alarm per 100 normals, hidden-fault recall rose from 19.9% to 44.9% (+25.0 points, 95% CI 18.7 to 31.5), and the result held at 0.5 and 2 false alarms per 100 (Table 7) and on single-mode failures. LOCF with indicators showed no clear difference from history, so the support is for using history, not for one particular encoding. Supported.

H4 (gain follows slowly changing sensors). Detection of hidden heat dissipation failures under control B rose from 10% to 66% and of overstrain failures under B from 27% to 80%, while hidden power failures fell slightly (from 33% to 29% under A and from 67% to 62% under C). Twenty-eight of the 35 lost detections were power failures. Supported.

H5 (replication and the form of history). History improved PR-AUC on C-MAPSS FD001 (+0.027, 95% CI 0.019 to 0.035) and on average over the MetroPT-3 events (0.491 to 0.540). On MetroPT-3, however, it detected events later and missed 2 of 15 event-seed pairs, and GRU-D outperformed history with XGBoost on FD001 (0.965 against 0.936, three-seed means as in Figure 5; Table 8 gives the five-seed mean of 0.938 for history) but not on AI4I (0.354 against 0.584). Partly supported: the ranking benefit replicates, the timing benefit does not, and the best form of history depends on fault dynamics.

Table 10. Hypothesis outcomes

Hypothesis	Prediction	Key evidence	Decision	Strength
H1	Published schedule scores above complete data	PR-AUC 0.878 vs 0.778; gap > 6 SD	Supported	Strong
H2	Standard missing-data methods give no gain	All four at or below 0.452; indicators identical to native	Supported	Strong
H3	History raises hidden-fault recall	+25.0 points (95% CI 18.7 to 31.5) at 1 FA per 100	Supported	Strong; not specific to one encoding
H4	Gain follows slowly changing sensors	HDF-B 10% to 66%; PWF slightly worse	Supported	Moderate
H5	Benefit replicates; best form depends on dynamics	FD001 +0.027; MetroPT-3 PR-AUC up but later detection; GRU-D best on FD001	Partly supported	Bounded

Note: *support is uneven. H1 to H3 rest on large, consistent differences; H4 rests on per-mode rates; H5 holds for ranking but not for detection timing on real data.*

Implications for benchmark design

The main lesson concerns benchmark design. AI4I-PMDI was built to add the irregularities of real plants, and it does, but its rule for failure records guarantees that each fault's evidence is measured. Scores on the published file therefore describe the easiest version of the problem, and they exceed scores on complete data. The point is that results on it say little about the case in which an inspection misses the fault, and that this case is easy to construct from the same values. Benchmarks with schedule-dependent missingness should also report results under a schedule that can hide faults, together with hidden- and visible-fault recall at a fixed false-alarm rate.

Why history works, and what it costs

Hidden faults are not a missing-data problem in the usual sense. Imputation and indicators try to fill in or flag what is absent from the current record, but when the decisive parameter was not measured, nothing in the current record can recover it. The machine's own earlier inspections can, provided the parameter changes slowly. The gains here follow that pattern: large for temperatures and tool wear, negative for rotational speed and torque. The claim about the method is narrow. The specific history representation is not significantly better than LOCF with indicators, and both rest on known building blocks. What matters is supplying earlier readings and marking them as earlier. A gateway that caches each sensor's last reading per machine can provide this at almost no cost. History also has costs. Under a shared false-alarm budget it lowers recall on visible faults, and on real data it reacts later to faults that begin suddenly, because the carried reading predates the fault. Combining a current-reading model with a history model reduces both costs on AI4I-PMDI, where it recovers visible-fault recall at some cost in hidden-fault recall; on MetroPT-3 it recovers the leaks the history model missed but does not outperform the current-reading model. GRU-D shows that slow

degradation rewards models that read the whole sequence, so the representation of history should follow how faults develop on the equipment in question.

Applications and Implications

Theoretical implications. The results separate hidden faults from missing data in the usual sense, since no in-record imputation can recover evidence that the record does not contain (Tables 6 and 7, Figure 3), and they tie the value of history to the rate of change of the causal quantity (H4), a property of the fault and not of the model.

Implications for benchmark builders and researchers. A benchmark that places failures where their evidence is always measured rewards models for reading the schedule, not the machine. Builders can release a version in which failures follow the same schedule as normal records, and researchers can rebuild it from AI4I 2020 values, as described in the research methods section, and should report both versions.

Practical implications for maintenance teams. Where inspections follow fixed routes, the cheapest improvement is to pass each machine's last measured value of every sensor, with its date, to the model alongside the current readings, and to run a current-reading model beside the history model and alarm on either (Li et al., 2025; Tamizhamuthan et al., 2026). Table 11 sets out these steps.

Table 11. Workflow for maintenance teams and the results that support each step

Step	Action	Supporting result	Practical note	Caution
1	Keep each sensor's last value per machine, dated and marked as earlier	Hidden-fault recall 19.9% to 44.9% (Table 7); plain LOCF no gain (Figure 3)	Store in gateway or database	Encoding matters little
2	Trust slowly changing sensors	Heat dissipation 10% to 66%; power failures slightly worse (H4)	Check on own equipment	Speed and torque did not benefit
3	Run current and history models together	41.0% hidden, 78.5% visible recall; all 15 MetroPT-3 pairs (Table 9)	Set thresholds jointly	History alone is later (125 against 38 minutes)

Source: *Authors' analysis.*

Limitations

Several limits apply to these findings. The time structure of AI4I is synthetic: temperatures and tool wear follow generated series, and the elapsed time between inspections carries no physical meaning. This is why the findings were replicated on C-MAPSS and MetroPT-3, but C-MAPSS is itself simulated, and MetroPT-3 provides only three test events, so its timing results are indicative. On MetroPT-3, false alarms per day rose well above the calibration level in the F2 test period, a drift that this study did not address. The realistic schedule uses one masking rule and a uniform random draw of controls for failure records; real inspection plans may cluster around suspected faults or follow seasonal patterns. The audit covers one public benchmark, so the finding that it is easier than complete data should not be assumed for other benchmarks without the same check. Models were not tuned, GRU-D included, so the comparison with GRU-D shows a direction rather than a final ranking. The analysis treats fault identification as a binary task; separating failure modes under hidden-fault conditions is left open. Finally, the decision rules assume that every alarm is checked, and the cost of false alarms is not modelled.

Future Directions

Several directions follow from these results. The first is to study when historical readings remain reliable evidence. This includes how fast the value of an earlier reading decays as time passes since it was taken, how it changes when operating conditions shift between inspections, and what happens when a sensor goes unread for a long run of inspections. A simple test that tells a maintenance team which carried values to trust, and which to treat with caution, would make history features safer to

use in practice. The second direction is to design inspection schedules with hidden faults in mind. Rather than rotating controls on a fixed plan, the next control for a machine could be chosen according to which failure modes its recent history makes likely, so that the sensors most needed are the ones measured. Such a policy could be compared with fixed and random schedules under the same inspection budget. The third direction concerns the model itself. The results on C-MAPSS FD001 suggest that slow degradation rewards models that read the whole sequence, so combining the last-reading features used here with tuned sequence models such as GRU-D, or with summaries of longer history windows, may give the benefits of both. The decision rules could also be made cost-sensitive, so that the cost of a missed fault and the cost of checking a false alarm are set by the plant rather than by a fixed false-alarm rate. The fourth direction is to carry the audit to other public benchmarks with inspection-dependent or schedule-dependent missing values, and to publish, with each benchmark, a version in which failures can fall under any inspection so that hidden faults can be measured. Finally, the evaluation should be extended to multi-class identification of failure modes and to further real datasets with recorded inspection plans, ideally from plants where the inspection schedule and the failure log are kept together, so that the bounds found here can be tightened.

Conclusion

This study examined what happens to fault identification when an inspection does not measure the sensors that reveal a fault. An audit showed that the published AI4I-PMDI schedule places every heat dissipation failure under control A, every power failure under B and every overstrain and tool wear failure under C, the only controls that measure their causes, so the file contains no hidden faults and XGBoost scores higher on it (PR-AUC 0.878) than on the same data with nothing missing (0.778), supporting H1. When failures could fall under any control, PR-AUC fell to 0.452 and only 19.9% of hidden faults were detected at one false alarm per 100 normal records; a model developed on the published file detected 4%. Standard methods stayed between 0.42 and 0.45, supporting H2. Adding the last earlier reading of each sensor raised PR-AUC by 0.130 (95% CI 0.088 to 0.165) and hidden-fault recall to 44.9%, supporting H3, mainly in slowly changing causes (heat dissipation, overstrain), supporting H4. The benefit replicated on C-MAPSS FD001 (0.911 to 0.938) and in ranking on MetroPT-3 (0.491 to 0.540), but history detected leaks later on the real compressor and GRU-D did better on slow engine degradation, so H5 is only partly supported. The combined rule recovered the leaks that history missed but did not outperform the current-reading model on the real compressor. The theoretical contribution is a bounded one and is separate from the empirical findings. The conceptual advance is two distinctions. The first is between random dropout and schedule-dependent missingness: under the first, missing evidence can be recovered from what remains, while under the second it may exist nowhere in the current record. The second is between filling a gap and supplying evidence: earlier readings, marked as earlier, can supply evidence the current inspection lacks, but only for quantities that persist between inspections, so the value of history is tied to the rate of change of the fault's causal quantity and not to the model. These conclusions remain limited by the synthetic time structure of AI4I, a single masking rule, untuned models and three real test events. Looking forward, the audit points to a broader change in how predictive maintenance models are evaluated. Leakage between training and test data (Matania et al., 2024; Qahtani, 2026), differing protocols (Knap et al., 2026) and random deletion of values (Toye et al., 2025) can each raise scores without improving a model in operation. Studies with incomplete data should report results under the convenient condition and under one in which the inspection plan can hide faults, group evaluation by machine, and report hidden- and visible-fault recall at a fixed false-alarm rate.

Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors. The work was carried out as part of the first author's master's degree studies in Information and Communication Engineering at Huazhong University of Science and Technology, Wuhan, China.

Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The authors have no connection with the creators or owners of the AI4I 2020, AI4I-PMDI, C-MAPSS or MetroPT-3 datasets, all of which are publicly available and were used under their published terms.

Data Availability

AI4I 2020 and AI4I-PMDI are publicly available (Matzka, 2020; Autran et al., 2024). C-MAPSS (Saxena et al., 2008) is available from the NASA Prognostics Data Repository, and MetroPT-3 (Veloso et al., 2022) from the UCI Machine Learning Repository. Code to build the realistic schedules and reproduce all results is available at https://github.com/ChristianaLebbie/hidden-faults-history.

Declaration of AI Use

During the preparation of this work the authors used Claude (Anthropic) to help draft and edit the text and to format the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the paper.

References

Alnahhal, M., Tabash, M. I., Safi, S. K., Al-Absy, M. S. M., & Mamadiyarov, Z. (2026). A comparative study of imbalance-handling methods in multiclass predictive maintenance. *Computation, 14*(4), 88. https://doi.org/10.3390/computation14040088

Autran, J.-V., Kuhn, V., Diguet, J.-P., Dubois, M., & Buche, C. (2024). AI4I-PMDI: Predictive maintenance datasets with complex industrial settings' irregularities. *Procedia Computer Science, 246*, 1201-1209. https://doi.org/10.1016/j.procs.2024.09.546

Azeta, J., Omeche, T. T., Daniyan, I., Abiola, J. O., Daniyan, L., Phuluwa, H. S., & Muvunzi, R. (2026). Artificial intelligence and robotics in predictive maintenance: A comprehensive review. *Frontiers in Mechanical Engineering, 11*. https://doi.org/10.3389/fmech.2025.1722114

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports, 8*, 6085. https://doi.org/10.1038/s41598-018-24271-9

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

Cureño Osornio, J., Díaz-Saldaña, G., Zamudio-Ramírez, I., Osornio-Rios, R. A., & Antonino-Daviu, J. A. (2024). Automatic mechanical fault diagnosis in induction motors through thermography and gradient boosting. In *2024 IEEE 33rd International Symposium on Industrial Electronics (ISIE)* (pp. 1-6). IEEE. https://doi.org/10.1109/ISIE54533.2024.10595734

Dereci, U., & Tuzkaya, G. (2024). An explainable artificial intelligence model for predictive maintenance and spare parts optimization. *Supply Chain Analytics, 8*, 100078.

Dhungana, P., Dhungana, H., Khadka, S. K., & Baskota, S. (2025). Data imputation in IoT measurements: Challenges and guidelines. In *Lecture Notes in Electrical Engineering* (Vol. 1369, pp. 429-438). Springer. https://doi.org/10.1007/978-3-031-84100–2_51

Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. Chapman & Hall.

Erickson, N., Purucker, L., Tschalzev, A., Holzmüller, D., Desai, P. M., Salinas, D., & Hutter, F. (2025). TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems (NeurIPS 2025), Datasets and Benchmarks Track. https://arxiv.org/abs/2506.16791*

Fu, Y., & Yan, W. (2024). *One masked model is all you need for sensor fault detection, isolation and accommodation*. arXiv. https://arxiv.org/abs/2403.16153.

Joshi, A., & Hauskrecht, M. (2025). *Still competitive: Revisiting recurrent models for irregular time series prediction*. arXiv. https://arxiv.org/abs/2510.16161.

Josse, J., Chen, J. M., Prost, N., Varoquaux, G., & Scornet, E. (2024). On the consistency of supervised learning with missing values. *Statistical Papers*. https://doi.org/10.1007/s00362-024-01550-4

Kalpakam, T. H., & Nisha, K. L. (2026). Predictive maintenance system using ensemble machine learning: A novel approach with real-time dashboard integration. In *2026 IEEE ICITSIF*. IEEE. https://doi.org/10.1109/ICITSIF69060.2026.11608913

Knap, P., Jachymczyk, U., & Lalik, K. (2026). Leakage-safe, reproducible benchmarking for vibration-based fault diagnosis. *PHM Society European Conference, 9*(1). https://doi.org/10.36001/phme.2026.v9i1.4924

Kumar, R., & Gautam, A. (2026). Provenance-aware transformer for robust PM2.5 forecasting under missing and irregular sensor data. In *Proceedings of the IEEE AIC 2026*. https://doi.org/10.1109/AIC70896.2026.11717366

Lee, S., Min, K., Son, Y., & Do, H. (2025). Adaptive time encoding for irregular multivariate time-series classification. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025). https://doi.org/10.52202/085713-3407*

Le Morvan, M., & Varoquaux, G. (2025). *Imputation for prediction: Beware of diminishing returns*. arXiv. https://arxiv.org/abs/2407.19804

Li, Y., Yan, H., Zhao, C., Chen, T., & Li, J. (2025). Dual-mode industrial IoT predictive maintenance. In *2025 IEEE CAIBDA*. IEEE. https://doi.org/10.1109/CAIBDA65784.2025.11183224

Lin, T., Ren, Z., Huang, K., Zhu, Y., & Karimi, H. R. (2025). A novel multi-sensor information fusion method for fault diagnosis of rotating machinery with missing signals. *Advanced Engineering Informatics, 68*, 103595. https://doi.org/10.1016/j.aei.2025.103595

Liu, X., Lee, C. K. M., Zhang, H., Chen, P., Huang, J., & Wong, C. N. (2025). Fault diagnosis based on incomplete sensor variables with a hierarchical semi-supervised Gaussian mixture classifier. *Applied Mathematical Modelling, 138*, Article 115764. https://doi.org/10.1016/j.apm.2024.115764

Liu, X., Qiu, X., Wu, X., Li, Z., Guo, C., Hu, J., & Yang, B. (2025). *Rethinking irregular time series forecasting: A simple yet effective baseline*. arXiv. https://arxiv.org/abs/2505.11250.

Maddali, G. (2025). AI-powered fault detection predictive maintenance for industrial machinery: A reliability engineering perspective. In *2025 IEEE ETNCC*. IEEE. https://doi.org/10.1109/ETNCC66224.2025.11299803

Matania, O., Cohen, R., Bechhoefer, E., & Bortman, J. (2024). Test-training leakage in evaluation of machine learning algorithms for condition-based maintenance. *PHM Society European Conference, 8*(1). https://doi.org/10.36001/phme.2024.v8i1.4125

Matzka, S. (2020). Explainable artificial intelligence for predictive maintenance applications. In *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*. IEEE. https://doi.org/10.1109/AI4I49448.2020.00023

Nozaki, Y., Ohnishi, T., Alqassem, I., & Temmei, K. (2026). Concept-based missingness-aware recurrent encoding for prolonged ICU stay prediction. In *Proceedings of the IEEE ICHI 2026*. https://doi.org/10.1109/ICHI69079.2026.00058

Özcan, H. (2026). Predictive maintenance in aircraft engine maintenance using the C-MAPSS dataset. *AI EDAM, 40*, e4. https://doi.org/10.1017/S0890060426100249

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Pierros, I., Kochliaridis, V., Apostolidou, E., Delimpasi, E., Zygouris, V., & Vlahavas, I. (2024). Predictive maintenance under absence of sensor data. In *Artificial Intelligence Applications and Innovations (AIAI 2024)* (pp. 279-292). Springer. https://doi.org/10.1007/978-3-031–63215–0_21

Prior Labs Team. (2025). *TabPFN-2.5: Advancing the state of the art in tabular foundation models*. arXiv. https://arxiv.org/abs/2511.08667.

Qahtani, A. M. (2026). A leakage-aware and reproducible evaluation framework for predictive maintenance classification. *International Journal of Advanced Computer Science and Applications, 17(3), 942-949.*

Qu, J., Holzmüller, D., Varoquaux, G., & Le Morvan, M. (2026). *TabICLv2: A better, faster, scalable, and open tabular foundation model*. arXiv. https://arxiv.org/abs/2602.11139.

Rajaoarisoa, L., Randrianandraina, R., Nalepa, G. J., & Gama, J. (2025). Decision-making systems improvement based on explainable AI approaches for predictive maintenance. *Engineering Applications of Artificial Intelligence, 139, 109601. https://doi.org/10.1016/j.engappai.2024.109601*

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE, 10*(3), Article e0118432. https://doi.org/10.1371/journal.pone.0118432

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *Proceedings of the International Conference on Prognostics and Health Management*. IEEE. https://doi.org/10.1109/PHM.2008.4711414

Sharma, A., Sim, K. Y., & Chandrasekaran, S. (2026). FusionNet: Intelligent sequence fusion for predictive maintenance in edge-enabled industrial IoT systems. *Cluster Computing, 29*, 615. https://doi.org/10.1007/s10586-026-06422-0

Stempfle, L., Matsson, A., Mwai, N., & Johansson, F. D. (2025). Prediction models that learn to avoid missing values. In *Proceedings of the 42nd International Conference on Machine Learning*. arXiv:2505.03393

Tamizhamuthan, R. V., Yogesh, K., & Ahila Devi, E. (2026). Predictive maintenance in leveraging supervised machine learning for wireless sensor networks. In *2026 IEEE ICVADV*. IEEE. https://doi.org/10.1109/ICVADV67766.2026.11469842

Toye, A. A., Celik, A., & Kleinberg, S. (2025). Benchmarking missing data imputation methods for time series using real-world test cases. In *Proceedings of the Sixth Conference on Health, Inference, and Learning (Proceedings of Machine Learning Research, 287, pp. 480*–501).

Tsallis, C., Papageorgas, P., Piromalis, D., & Munteanu, R. A. (2025). Application-wise review of machine learning-based predictive maintenance: Trends, challenges, and future directions. *Applied Sciences, 15*(9), 4898. https://doi.org/10.3390/app15094898

Veloso, B., Ribeiro, R. P., Gama, J., & Pereira, P. M. (2022). The MetroPT dataset for predictive maintenance. *Scientific Data, 9*, 764.

Xiao, S., Ye, F., Zhang, X., Tan, M., & Zhang, C. (2025). The diagnosis and recovery of faults in the workshop environmental control system sensor network based on medium-to-long-term predictions. *Machines, 13*(11), 975. https://doi.org/10.3390/machines13110975

Yan, L., Du, S., Wang, S., Luo, H., Li, S., & Chen, X. (2026). MAEFF: Missing-aware adaptive expert fusion framework for multi-sensor bearing fault diagnosis. *Annals of Operations Research*. https://doi.org/10.1007/s10479-026-07381-w

Zhao, J., Li, Y., & Mo, H. (2025). Handling missing data using the XGBoost-based multiple imputation by chained equations regression method. *Frontiers in Artificial Intelligence, 8*, 1553220. https://doi.org/10.3389/frai.2025.1553220

Published by

IJSAB International