IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

Drift-Aware Online Ensemble Learning for Imbalanced Industrial Failure Detection: A Comparative Study

Janani M[1], Dr.M.Daniel Naraeshkumar[2]

M.Tech 2nd Year, Department of Computer Science and Engineering, SRMIST, Vadapalani[1]

Assistant Professor, Department of Computer Science and Engineering, SRMIST, Vadapalani[2]

Abstract: Predictive-maintenance models are usually reported on fixed, shuffled train-test splits. This hides two practical weaknesses: class imbalance, which makes accuracy misleading, and sensor drift, which static models never meet during evaluation. This paper presents a comparative study on the public AI4I 2020 dataset (10,000 records, 3.39% failures) that addresses both. First, we repeat the 80/20 evaluation protocol of a recent benchmark study. Gradient boosting with physics-informed features (machine power, temperature difference and wear-torque load) reaches a failure-class F1-score of 0.852 (precision 0.979, recall 0.754, ROC-AUC 0.979), against 0.823 published for the benchmark's tuned XGBoost model. The bootstrap 95% interval [0.774, 0.917] contains the published value, so the gain is indicative rather than conclusive. Second, we simulate sensor drift at five severities and compare static models with online learners that retrain on a sliding window, triggered by the Drift Detection Method (DDM), by a timer, or by both. The static physics-informed model falls from F1 0.768 without drift to 0.396 at severity 0.5 and 0.047 at severity 2. From severity 0.5 the best online variant has the higher point estimate, but the 95% intervals separate only from severity 1.5, where F1 is 0.313 against 0.144. Without drift the static model is ahead. Online adaptation therefore mitigates drift without curing it, because it needs timely labels and enough failure examples in each window. The study contributes a reproducible drift-aware evaluation protocol, an ablation of feature engineering, and a candid account of where adaptive learning helps and where it does not.

Index Terms: predictive maintenance, concept drift, online learning, class imbalance, gradient boosting, industrial IoT, fault detection, AI4I 2020.

I. INTRODUCTION

A. Background and Motivation

Machines on a modern shop floor report their state continuously through temperature, speed, torque and wear sensors. Repairing equipment only after it breaks causes unplanned downtime, while servicing it on a fixed calendar replaces parts that still work. Predictive maintenance (PdM) tries to avoid both costs by forecasting failures from sensor data [15]. Machine learning is now the standard tool for this forecast, and gradient-boosted trees are a common choice for tabular sensor records [13], [14], [16].

B. Problem Statement

Most published PdM studies evaluate a model once, on a random split of a fixed benchmark. The AI4I 2020 dataset [1], [2] is the most common choice, and many papers report accuracy near 99% on it. Two problems limit what such a number tells an engineer. First, only 3.39% of the records are failures, so a classifier that never predicts a failure already scores about 96.6% accuracy; failure-class precision, recall and F1 are the informative metrics. Second, a random split mixes all operating conditions into both the training and the test set. A plant does not behave this way: sensors are recalibrated or age, and the data seen after deployment differs from the data used for training. This change is called concept drift [12].

C. Research Gap and Contributions

Adaptive learners that detect drift and retrain have been proposed for PdM [4], and drift detectors such as DDM, EDDM and ADWIN are well established [9]–[11]. However, we found no study that places a drift-aware learner and a strong static learner on the same public benchmark, with the same failure-class metrics and uncertainty estimates, while also reproducing the protocol of a published static baseline. This paper fills that gap. Its contributions are:

1) a protocol-matched comparison with the static benchmark of Nadarajan et al. [3], with an ablation of physics-informed features and threshold selection;

2) a five-level drift-severity experiment comparing static and online learners, with bootstrap confidence intervals;

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

3) an explicit account of where online adaptation helps, where it costs accuracy, and which assumptions (immediate labels, synthetic data) limit the conclusions.

D. Organisation of the Paper

Section II reviews related work. Section III describes the dataset, features, models and the drift protocol. Section IV presents and discusses the results and their limitations. Section V concludes.

II. LITERATURE STUDY

A. Related Studies

Static studies compare classifiers on fixed splits of the AI4I data. Nadarajan et al. [3] tune an XGBoost model after smoothing and permutation-importance selection and report an F1-score of 82.25% for the failure class, against 70.00% for an AdaBoost method [5] and 54.95–66.67% for an SVM, a decision tree and a random forest. Other studies summarised in [3] report accuracies of 96–97% with bagged nearest neighbours [6], XGBoost with outlier detection [7] and neural networks [8]. These studies describe how well a model fits one random split; they do not test it after the data changes.

Concept drift is the change of the relation between inputs and labels over time [12]. The Drift Detection Method [9] watches the error rate of a classifier and signals drift when it rises above its best recorded level by three standard deviations; EDDM [10] watches the distance between errors, and ADWIN [11] compares the means of two sub-windows. In PdM, Subashree et al. [4] combine such detectors with sliding-window retraining and model selection and report an advantage over non-adaptive models, but on turbofan and bearing data rather than on the shared AI4I benchmark. Table I lists the fourteen studies that shape this work.

TABLE I
LITERATURE STUDY: TECHNIQUES, MATHEMATICAL IDEAS AND GAPS

No.	Paper / Author	Year	Core Technique	Key Mathematical Idea	Reason for Selection	Limitation / Gap Finding
1	S. Matzka [1], [2]	2020	Synthetic milling-machine dataset with five rule-based failure modes; explainable-AI use case	Failure modes are threshold rules on derived quantities, e.g. power $\tau\omega$ outside 3,500–9,000 W; $\Delta T < 8.6$ K with $\omega <$ 1,380 rpm; wear×τ above 11,000/12,000/13,000 by product type	Source of the benchmark used in every comparison	Static data, no drift; labels follow explicit rules, so models can learn the rules rather than real physics
2	Nadarajan et al. [3]	2025	3-point rolling-mean smoothing, permutation-importance selection and grid-searched XGBoost (E.XGB)	Regularised boosting objective $Obj = \Sigma L(y, \hat{y}) + \Sigma\Omega(f_k)$	Direct base paper: same dataset, published failure-class metrics (F1 = 82.25%)	Random shuffled 80/20 split, so no drift evaluation; recall 73.3%; benefit of smoothing not reproducible (Section IV-A)
3	Subashree et al. [4]	2026	Adaptive ensemble with DDM/EDDM drift detection, sliding-window retraining and dynamic model selection	Model chosen by window F1: $M^*_t = \text{argmax_i } F1(M_i, W_t)$	Closest adaptive-learning study; reports adaptive > non-adaptive models	Uses NASA turbofan and PRONOSTIA data, not AI4I 2020; data available only on request
4	Elkateb et al. [5]	2024	AdaBoost for IoT-based machine-stop classification (the E.AB benchmark in [3])	Re-weighting of misclassified samples at each boosting round	Boosting baseline replicated in [3]	High processing latency reported in [3]; single dataset

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.151088

No.	Paper / Author	Year	Core Technique	Key Mathematical Idea	Reason for Selection	Limitation / Gap Finding
5	Arisani and Muljono [6]	2024	Bagged nearest-neighbour classifier with under-sampling	Majority vote of KNN learners on resampled subsets	Resampling approach to imbalance (97.1% accuracy, per [3])	Accuracy-based reporting; under-sampling can discard information
6	Ghadekar et al. [7]	2024	XGBoost combined with Local Outlier Factor and explainable AI	Local density ratio used as outlier score	Supervised plus anomaly view (96% and 91% accuracy, per [3])	SMOTE noise and scalability concerns noted in [3]
7	Baldovino et al. [8]	2024	Logistic regression, KNN and ANN comparison	Sigmoid and neural-network classification	Classical comparison on failure data (ANN 96.85%, per [3])	ANN produced false positives; accuracy-centred
8	Gama et al. [9]	2004	Drift Detection Method (DDM)	Error rate p_t and $s_t = \sqrt{(p_t(1-p_t)/t)}$; drift when $p_t + s_t \geq p_{min} + 3 \cdot s_{min}$	Detector used in our online branch	Designed for error streams; unstable when errors are rare
9	Baena-García et al. [10]	2006	Early Drift Detection Method (EDDM)	Monitors the distance between consecutive classification errors	Alternative detector for gradual drift	Needs many errors to estimate the distance; not tested here
10	Bifet and Gavaldà [11]	2007	ADWIN adaptive windowing	Hoeffding bound on sub-window means: $\lvert\mu 0 - \mu 1\rvert > \varepsilon_{cut}$	Reference for window-based adaptation	Needs a stream-processing framework; not used here
11	Gama et al. [12]	2014	Survey of concept-drift adaptation	Taxonomy of detectors, windows and ensembles	Frames the design choices of this study	General survey; no predictive-maintenance benchmark
12	Chen and Guestrin [13]	2016	XGBoost	Second-order Taylor expansion of the loss with tree regularisation	Boosting method behind the published figures in [3]	Not run here (library unavailable); compared via published figures
13	Ke et al. [14]	2017	LightGBM histogram-based boosting	Histogram binning, gradient-based one-side sampling, feature bundling	Basis of the histogram boosting model used here	General-purpose; no drift handling
14	Dalzochio et al. [15]	2020	Review of ML and reasoning for predictive maintenance	Taxonomy of tasks and techniques	Context on open challenges	Review only; no experiments

B. Final Observation of the Literature Review

1) Key trends.

Two lines of work run in parallel. Static studies push accuracy to 97–99% through boosting, tuning and resampling [3], [5]–[8]. Adaptive studies add drift detectors and retraining, but are tested on other datasets [4] or only describe the detectors [9]–[12]. The static line reports failure-class metrics only occasionally and never under drift; the adaptive line reports drift robustness but not on the shared benchmark.

2) Overall research gap.

No reviewed study answers a practical question: when sensor drift occurs, how much of a strong static model's failure-detection ability survives, and does sliding-window retraining recover it? Answering it needs one benchmark, one set of failure-class metrics, uncertainty estimates and a protocol that can be matched against a published static result. The rest of this paper does that.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

III. METHODOLOGY

3. *Dataset*

The AI4I 2020 dataset [1], [2] holds 10,000 synthetic records of a milling machine with five sensor readings, a product type and a binary failure label (Table II). It has 339 failures spread over five failure modes (Fig. 1). The failure modes are defined by explicit rules on derived quantities [1], which matters for the interpretation of the results (Section IV-D). The file was used as downloaded and has no missing values.

TABLE II
VARIABLES OF THE AI4I 2020 DATASET

Variable	Unit	Description	Range
Product type	–	Quality variant L/M/H (6,000/2,997/1,003 records)	L, M, H
Air temperature	K	Ambient temperature	295.3–304.5
Process temp.	K	Machining process temperature	305.7–313.8
Rotational speed	rpm	Spindle speed	1168–2886
Torque	Nm	Spindle torque	3.8–76.6
Tool wear	min	Cumulative tool usage time	0–253
Machine failure	0/1	Target; 339 failures (3.39%)	binary

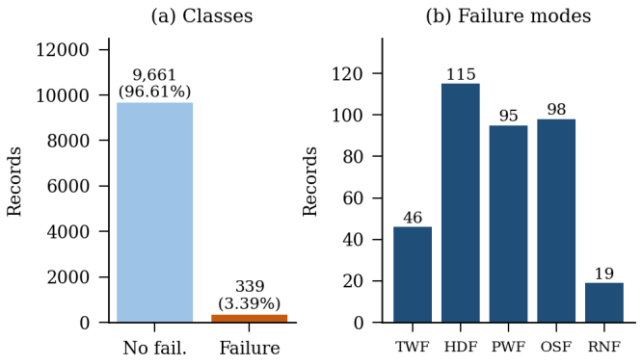


Fig. 1. (a) Class distribution and (b) failure modes.

B. Pre-processing and Physics-Informed Features

The baseline feature set is the five numeric readings. The proposed set adds the product type (coded 0, 1, 2) and three quantities that follow from the machining physics: the mechanical power, the temperature difference and the wear-torque load,

$$P = \tau \cdot \omega \cdot 2\pi/60, \quad \Delta T = T_{process} - T_{air},$$

$$L = wear \cdot \tau \quad (1)$$

where τ is the torque in N·m and ω the speed in rpm. These are the quantities the dataset documentation uses to define power, heat-dissipation and overstrain failures [1]. They are products and differences of raw inputs, which axis-aligned trees can only approximate with many splits. No smoothing is applied to the proposed features.

C. Static Models and Decision Threshold

The benchmark models are a decision tree, a random forest [17], an RBF support vector machine and histogram-based gradient boosting [14], [16], all from scikit-learn [18] with default settings except where Table III states otherwise. The

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

proposed static model is histogram gradient boosting on the physics-informed features. Its decision threshold is chosen to maximise the F1-score of five-fold out-of-fold predictions on the training set only,

$$\theta^* = argmax_\theta\, F1(\, y_train\, ,\, 1[\, p_{oof} \geq \theta\,]\,) \quad (2)$$

so no test information enters the choice. Imbalance is handled by this threshold; no synthetic oversampling is used because it can add noise [3], [19].

D. Online Drift-Adaptive Framework

Fig. 2 shows the design and Algorithm 1 the procedure. The static branch is trained once on the first 60% of the records, in record order, and never updated. In the online branch the test stream arrives in batches of 150 records. For each batch the current model predicts first and the labels are revealed afterwards (test-then-train). The errors feed the Drift Detection Method [9]; for t instances with error rate p_t it tracks

$$s_t = \sqrt{(\, p_t\, (1 - p_t)\, /\, t\,)},$$

$$drift\ if\ \ p_t + s_t \geq p_{min} + 3 \cdot s_{min} \quad (3)$$

where p_{min} and s_{min} belong to the lowest $p_t + s_t$ seen so far. The detector is reset after each signal. Three online variants are compared: periodic retraining every five batches, retraining on DDM signals only, and both.

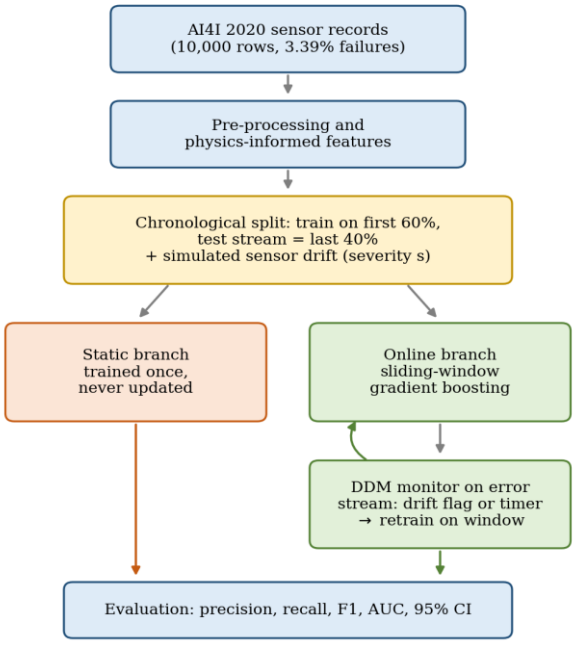


Fig. 2. Experimental framework: a static and an online branch evaluated on the same drifted test stream.

Algorithm 1. Online drift-adaptive detection

Input: train set D0, test batches B1..Bn,
window w, period k, mode: drift/periodic/both

```
M <- fit(last w records of D0); H <- D0
for t = 1..n do
  y_hat <- M.predict(B_t)      // test first
  feed e_i = 1[y_hat_i != y_i] to DDM
  H <- H + B_t                 // labels known
  if (DDM drift and mode != periodic) or
     (t mod k = 0 and mode != drift) then
    if last w of H has both classes then
      M <- fit(last w records of H)
return all y_hat
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

E. Drift Simulation

The dataset has no natural drift, so drift is injected into the test stream only. A severity $s \geq 0$ shifts the readings of every test record as a sensor miscalibration would,

$$T_{air} \leftarrow T_{air} + 3.5s,\ T_{proc} \leftarrow T_{proc} + 2.0s,\ \omega \leftarrow \omega(1 - 0.07s),$$

$$\tau \leftarrow \tau(1 + 0.12s),\ wear \leftarrow wear(1 + 0.15s) + 5s \quad (4)$$

while the labels stay unchanged. The simulation therefore represents sensors that report shifted values although the machine state is the same: an abrupt drift that breaks the relation between features and labels. Five severities (0, 0.5, 1, 1.5, 2) are used. This is a stylised scenario, not a measurement of real plant drift.

F. Evaluation Protocol and Metrics

Two protocols are used (Table III). The static protocol repeats the 80/20 random split of [3] (random_state = 42) so the results can be set against the published numbers. The drift protocol splits the records chronologically. Failure-class precision, recall and F1, accuracy and ROC-AUC are reported, with

$$F1 = 2 \cdot Precision \cdot Recall / (Precision + Recall) \quad (5)$$

and a 95% bootstrap interval for F1. The static test set holds only 61 failures, so the intervals are wide and are given with every headline comparison.

TABLE III
EXPERIMENTAL SETUP

Item	Setting
Static protocol	Random 80/20 split, random_state = 42, not stratified (2,000 test records, 61 failures)
Drift protocol	Chronological split by record ID: first 6,000 records train, last 4,000 are the test stream
Classifier	Histogram gradient boosting (scikit-learn 1.8.0): 60 iterations, learning rate 0.15, random_state = 42
Online branch	Window 3,000; batches of 150; periodic retraining every 5 batches; labels assumed to arrive after each batch
DDM	Instance-level error stream; minimum 100 instances; warning 2σ, drift 3σ; reset after each drift signal
Threshold	0.5 in the drift protocol; chosen on 5-fold out-of-fold training predictions for the static proposed model
Uncertainty	300 bootstrap resamples per 95% interval; three random splits for static robustness
Environment	Python 3, scikit-learn 1.8.0, single CPU core; XGBoost not installed, hence not re-run

IV. RESULTS AND DISCUSSION

A. Static Protocol: Comparison with the Published Benchmark

Table IV sets our results against the figures published in [3]. We first checked whether the published baselines can be reproduced. With raw sensor readings our decision tree (F1 68.70%) and random forest (68.57%) are close to the published 66.67% and 60.61%. A 3-point rolling mean applied in record order, as the description in [3] suggests, collapsed the same models (Table V, row A: F1 0.173), because consecutive records are unrelated machines and averaging them blends independent observations. We could not determine how smoothing was implemented in [3], so the reproduction uses raw features and makes no claim about their code.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

The proposed model reaches F1 85.19% with precision 97.87% and recall 75.41% (Fig. 5: 46 of 61 failures found, 1 false alarm). Its point estimate exceeds the published E.XGB on every reported metric, and over three random splits F1 is 0.856 ± 0.007. However, the bootstrap interval [0.774, 0.917] contains the published 0.8225, XGBoost was not available here, and the published values come from a different implementation. The improvement is therefore indicative, not established.

TABLE IV
STATIC PROTOCOL: FAILURE-CLASS RESULTS (%) AGAINST THE PUBLISHED BENCHMARK

Model	Src.	Acc.	Prec.	Rec.	F1	AUC
E.XGB (published)	[3]	99.05	93.62	73.33	82.25	97.10
E.AB (published)	[3]	98.50	89.74	57.38	70.00	94.64
Dec. tree (published)	[3]	98.00	67.80	65.57	66.67	87.26
Rand. forest (published)	[3]	98.05	78.95	49.18	60.61	86.63
SVM (published)	[3]	97.95	83.33	40.98	54.95	85.35
Decision tree	ours	97.95	64.29	73.77	68.70	86.24
Random forest	ours	98.35	81.82	59.02	68.57	96.64
SVM (RBF)	ours	97.65	81.82	29.51	43.37	95.53
Hist. GB, raw	ours	98.65	86.96	65.57	74.77	97.48
Proposed (physics GB)	**ours**	**99.20**	**97.87**	**75.41**	**85.19**	**97.92**

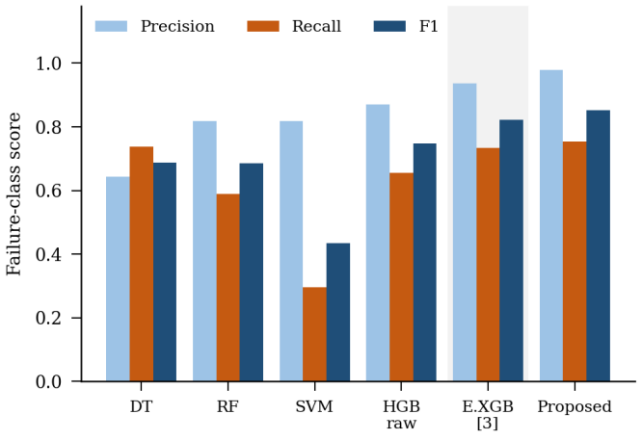


Fig. 3. Failure-class precision, recall and F1 under the static protocol; the shaded bars show the published E.XGB [3].

B. Ablation: What Produces the Gain

Table V and Fig. 4 add one component at a time to histogram gradient boosting. Moving from smoothed to raw features raises F1 from 0.173 to 0.748; adding the product type alone does not help; the physics-informed features give the largest step, to 0.852. Threshold selection ($\theta^* = 0.560$) leaves the predictions on this split unchanged and changes the three-split mean F1 by less than 0.01, so it is kept for protocol hygiene, not for accuracy.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

TABLE V
ABLATION OF THE STATIC MODEL

Variant	Prec.	Rec.	F1	95% CI	3 splits
A) smoothed 5 features	0.350	0.115	0.173	[0.063, 0.287]	0.156±0.039
B) raw 5 features	0.870	0.656	0.748	[0.656, 0.837]	0.732±0.021
C) + product type	0.875	0.574	0.693	[0.570, 0.788]	0.735±0.038
D) + physics features	0.979	0.754	0.852	[0.774, 0.917]	0.861±0.013
E) + tuned threshold	0.979	0.754	0.852	[0.774, 0.917]	0.856±0.007

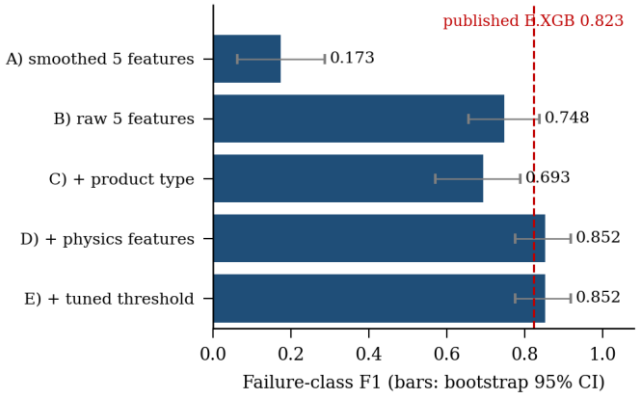


Fig. 4. Ablation of the static model; bars show bootstrap 95% intervals, the dashed line the published E.XGB F1.

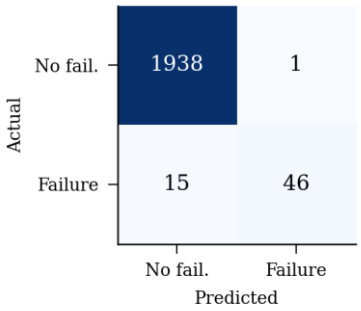


Fig. 5. Confusion matrix of the proposed static model.

TABLE VI
FAILURE-CLASS F1 (95% BOOTSTRAP CI) UNDER SIMULATED DRIFT; LAST COLUMN: DDM SIGNALS AT S = 0 / 0.5 / 1 / 1.5 / 2

Method	s = 0	s = 0.5	s = 1.0	s = 1.5	s = 2.0	DDM signals
static (base features)	0.571 [0.46, 0.67]	0.358 [0.29, 0.41]	0.226 [0.18, 0.27]	0.144 [0.11, 0.17]	0.132 [0.10, 0.16]	–
static (physics features)	0.768 [0.68, 0.85]	0.396 [0.33, 0.46]	0.164 [0.13, 0.19]	0.086 [0.07, 0.10]	0.047 [0.04, 0.06]	–
online periodic (base)	0.296 [0.18, 0.39]	0.287 [0.21, 0.36]	0.198 [0.14, 0.25]	0.255 [0.19, 0.33]	0.168 [0.11, 0.22]	0/0/0/0/0

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

Method	s = 0	s = 0.5	s = 1.0	s = 1.5	s = 2.0	DDM signals
online periodic (physics)	0.628 [0.53, 0.73]	0.391 [0.31, 0.47]	0.195 [0.14, 0.24]	0.168 [0.12, 0.21]	0.133 [0.10, 0.17]	0/0/0/0/0
online DDM-only (base)	0.388 [0.27, 0.49]	0.296 [0.22, 0.37]	0.236 [0.17, 0.29]	0.239 [0.17, 0.29]	0.234 [0.16, 0.30]	1/2/3/2/3
online DDM-only (physics)	0.550 [0.44, 0.65]	0.402 [0.32, 0.48]	0.222 [0.16, 0.27]	0.083 [0.06, 0.10]	0.047 [0.04, 0.06]	1/1/3/0/0
online hybrid (base)	0.304 [0.19, 0.40]	0.316 [0.23, 0.40]	0.222 [0.15, 0.29]	0.313 [0.22, 0.40]	0.230 [0.16, 0.30]	1/2/2/1/1
online hybrid (physics)	0.628 [0.53, 0.73]	0.422 [0.34, 0.50]	0.262 [0.19, 0.32]	0.168 [0.12, 0.21]	0.133 [0.10, 0.17]	1/1/2/0/0
Best static	**0.768**	**0.396**	**0.226**	**0.144**	**0.132**	
Best online	**0.628**	**0.422**	**0.262**	**0.313**	**0.234**	
Intervals disjoint, online ahead	**no**	**no**	**no**	**yes**	**yes**	

C. Drift Protocol: Static versus Online Learning

Table VI and Fig. 6 give the main drift result. Without drift (s = 0) the static physics-informed model has the higher F1 (0.768 against 0.628 for the best online variant); the intervals overlap slightly, so this is suggestive, not conclusive. An online model sees only a 3,000-record window with about one hundred failures. Drift hits the static models at once: F1 of the physics-informed static model falls to 0.396 at s = 0.5 and 0.164 at s = 1 (base features: 0.358 and 0.226).

From s = 0.5 the best online variant has the higher point estimate (0.422 against 0.396 at s = 0.5), but the 95% intervals separate only from s = 1.5: 0.313 [0.216, 0.401] for online hybrid (base) against 0.144 [0.114, 0.172] for static (base features). At s = 2 the gap is 0.234 against 0.132. No single online variant dominates (best online variant: online periodic (physics) at s = 0; online hybrid (physics) at s = 0.5; online hybrid (physics) at s = 1.0; online hybrid (base) at s = 1.5; online DDM-only (base) at s = 2.0). The "best" rows of Table VI choose the maximum of four online and two static variants, which favours the online group, so the individual rows should be read as well.

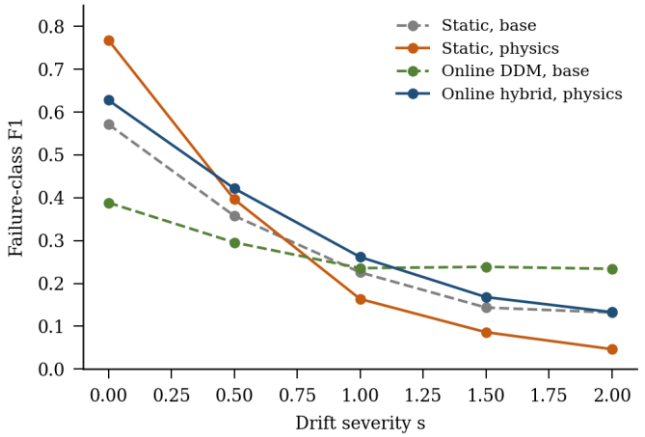


Fig. 6. Failure-class F1 against drift severity for selected static and online methods.

Fig. 7 shows the rolling error rate at s = 1. DDM raised 2 signals in the hybrid physics variant (at stream positions 150 and 1200). One signal also appeared without drift (Table VI), so DDM with rare errors can raise false alarms. Table VII shows that window length changes the s = 1 result by less than the confidence intervals, so no window size is recommended from these data.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151088

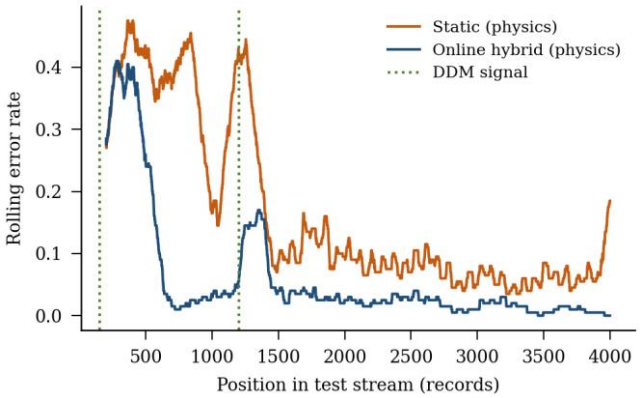


Fig. 7. Rolling error rate (window of 200 records) at s = 1; dotted lines are DDM signals.

TABLE VII
WINDOW LENGTH: F1 OF THE HYBRID ONLINE MODEL AT S = 1

Feature set	2,000	3,000	4,000
Base features	0.283	0.222	0.255
Physics features	0.243	0.262	0.282

D. Discussion

Three observations follow. First, domain features help because the labels of this dataset are generated from rules on exactly those derived quantities [1]. The gain shows that explicit physics lets simple tree models recover the rules; it does not show that the same gain would appear on real machines, where failure mechanisms are not given as formulas. Second, adaptation is a trade. The online models give up peak performance when nothing changes and recover only part of it when sensors drift, because each window holds about one hundred failures and because the labels needed for retraining must exist, which in a plant may arrive days later. Third, DDM is not free of false alarms when errors are rare, and in this protocol it had no pre-drift reference period (Section IV-E), so a deployment would need a tuned or ensemble detector that is started before drift occurs.

E. Limitations

The conclusions are limited by the synthetic, rule-generated dataset; a single stylised abrupt drift applied to all sensors; the assumption that true labels arrive right after each batch; the detector starting at the beginning of the already drifted test stream, so it has no pre-drift reference period and reacts to changes in the error rate after drift began rather than to its onset; a static test set with only 61 failures and wide intervals; one dataset; no hyperparameter search for the online models; the selection effect of the best-of rows; and the comparison with published numbers instead of a re-run of XGBoost. The results are a controlled comparison under stated assumptions.

V. CONCLUSION AND FUTURE WORK

Under the evaluation protocol of a published benchmark, gradient boosting with physics-informed features reached an F1-score of 0.852 on the AI4I 2020 failure class, higher than the published 0.823 in point estimate but not separable from it statistically. Under simulated sensor drift the same static model lost most of its failure-detection ability. Sliding-window retraining with drift detection kept part of it, with a statistically clear advantage only from severity 1.5 and a lower point estimate when no drift occurs. Online adaptation therefore helps but does not replace data on how real drift behaves. Future work will use datasets with natural drift, delayed and partial labels, ADWIN and ensemble detectors, cost-sensitive thresholds, and XGBoost and LightGBM in the comparison.

DECLARATION ON THE USE OF AI ASSISTANCE

[Authors to confirm and edit according to the target journal and institutional policy.] This manuscript was drafted with the help of an AI language model (Claude, Anthropic). The experiments were run in a sandboxed environment with the scripts supplied with the manuscript. The authors are responsible for checking the results, the references and the final text.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026
DOI: 10.17148/IJARCCE.2026.151088

DATA AND CODE AVAILABILITY

The AI4I 2020 dataset is public [2]. The scripts that produce every table and figure are provided with this manuscript.

REFERENCES

[1]. S. Matzka, "Explainable artificial intelligence for predictive maintenance applications," in Proc. 3rd Int. Conf. Artif. Intell. for Industries (AI4I), 2020, pp. 69–74, doi: 10.1109/AI4I49448.2020.00023.
[2]. S. Matzka, "AI4I 2020 predictive maintenance dataset," UCI Machine Learning Repository, 2020, doi: 10.24432/C5HS5C.
[3]. A. Nadarajan, I. Ishak, N. Manshor, R. Mohamed, and M. Y. Yusof, "Enhancing predictive maintenance method using machine learning to improve IoT-embedded machinery efficiency and performance," Int. J. Adv. Comput. Sci. Appl., vol. 16, no. 10, 2025.
[4]. S. Subashree, M. Rajakumaran, G. Pushpa, and R. Manivannan, "Adaptive machine learning models for predictive maintenance in industrial internet of things (IIoT) systems," Sci. Rep., vol. 16, Art. no. 12451, 2026, doi: 10.1038/s41598-026-42666-x.
[5]. S. Elkateb, A. Métwalli, A. Shendy, and A. E. Abu-Elanien, "Machine learning and IoT-based predictive maintenance approach for industrial applications," Alexandria Eng. J., vol. 88, pp. 298–309, 2024, doi: 10.1016/j.aej.2023.12.065.
[6]. M. I. Arisani and M. Muljono, "Bagging nearest neighbor and its enhancement for machinery predictive maintenance," J. Appl. Informat. Comput., vol. 8, no. 2, pp. 248–256, 2024, doi: 10.30871/jaic.v8i2.8158.
[7]. P. Ghadekar, A. Manakshe, S. Madhikar, S. Patil, M. Mukadam, and T. Gambhir, "Predictive maintenance for industrial equipment: Using XGBoost and local outlier factor with explainable AI for analysis," in Proc. Confluence, 2024, pp. 25–30, doi: 10.1109/confluence60223.2024.10463280.
[8]. R. G. Baldovino et al., "Comparative analysis of machine learning models for predictive analysis of machine failures," in Proc. 9th Int. Conf. Mechatronics Eng. (ICOM), 2024, pp. 288–293, doi: 10.1109/icom61675.2024.10652325.
[9]. J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in Advances in Artificial Intelligence – SBIA 2004, Lecture Notes in Computer Science, vol. 3171. Berlin, Germany: Springer, 2004, pp. 286–295.
[10]. M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early drift detection method," in Proc. 4th Int. Workshop Knowledge Discovery from Data Streams, 2006, pp. 77–86.
[11]. A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in Proc. SIAM Int. Conf. Data Mining, 2007, pp. 443–448.
[12]. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Comput. Surv., vol. 46, no. 4, Art. no. 44, 2014, doi: 10.1145/2523813.
[13]. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
[14]. G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 3146–3154.
[15]. J. Dalzochio et al., "Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges," Comput. Ind., vol. 123, Art. no. 103298, 2020, doi: 10.1016/j.compind.2020.103298.
[16]. J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Ann. Statist., vol. 29, no. 5, pp. 1189–1232, 2001.
[17]. L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.
[18]. F. Pedregosa et al., "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[19]. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

 This work is licensed under a Creative Commons Attribution 4.0 International License