LETTERS IN BIOMATHEMATICS
An International Journal

RESEARCH ARTICLE

OPEN ACCESS

Regime-Conditional Machine Learning for Power Plant Maintenance: Separating Normal Operating Transitions from Genuine Equipment Degradation

Hossain Mohammad Dalim[1], Syed Ali Reza[2], Arifur Rahman[3], Anisuzzaman Minto[4], Md Riaz Hossain[5], Shatabdi Scholastica Gomes[6], Afm Tanvir Anjum[7], Mariam Faruque Sharif[8], Tariq Ibn Nazim[9] and Mohammad Giashuddin Bhuiyan[10]

[1] School of Business, International American University, Los Angeles, California, USA.
[2] MS in Data Analytics, University of The Potomac
[3] School of Business, International American University, Los Angeles, California, USA.
[4] MS in Data Analytics, University of the Potomac
[5] Management in Science and Quantitative Methods, Gannon University, Erie, Pennsylvania, USA
[6] MS in Data Analytics, University of the Potomac
[7] Doctor of Computer Science (DCS), University of the Potomac
[8] M.S. in Computer Science, Georgia Southern University
[9] BSc (Hons) Computing (Information Systems), University of Greenwich, London, United Kingdom
[10] Doctor of Business Administration with a Specialization in Artificial Intelligence and Leadership, Belhaven University

ABSTRACT

Industrial power generation systems run under constantly shifting weather conditions plus changing dispatch demands, which cause big swings in how equipment behaves even when the machines are completely healthy. Standard SCADA-based predictive maintenance models regularly mistake these normal swings for actual faults, triggering false alarms that hurt maintenance reliability and waste operational time. This paper puts forward a regime-conditional machine learning setup to tell the difference between normal operating shifts and real drivetrain wear in wind turbine fleets. Working with SCADA telemetry from five Fuhrländer FL2500 wind turbines, the framework sorts out operating regimes using unsupervised K-Means clustering built strictly on operational variables, which keeps degradation-related target leakage out of the picture. A Regime-Conditioned Gradient Boosting Regressor then predicts what normal gearbox temperatures should look like across different running states, while regime-standardized prediction residuals serve as the actual wear indicators. The proposed method gets tested right alongside standard gradient boosting, Isolation Forest, autoencoder-based anomaly detection, and regime-specific anomaly detection models. Testing relies on chronological data splits, leave-one-turbine-out validation, persistence-based alert filtering, event-level detection checks, and block-bootstrap statistical testing. The numbers show that conditioning on operating regimes cuts down false alarms tied to normal operating shifts while sharpening how well the model spots real wear. This framework hits stronger precision-recall marks, better detection sensitivity, and earlier warnings for documented drivetrain failures when stacked up against regime-agnostic baselines. Cross-turbine testing also shows the setup can idenify weird equipment behavior without needing turbine-specific recalibration. Explainability checks further back up the physical sense of the operational variables used to estimate expected gearbox heat. Overall, these findings prove how much operating context matters for SCADA condition monitoring, offering a clear, interpretable way to make maintenance alerts more dependable while keeping early fault detection intact.

ARTICLE HISTORY

Received October 17, 2026

Accepted November 21, 2026

KEYWORDS

Predictive Maintenance; SCADA Condition Monitoring; Operating Regime Conditioning; Wind Turbine Fleet; Drivetrain Degradation; False Alarm Reduction; Gradient Boosting Regression; Early Fault Detection; Explainable Machine Learning.

1. Introduction

1.1 Background and Industrial Motivation: The False Alarm Crisis in Power Generation

Power generation machines today run under weather, mechanical, and electrical conditions that change constantly, which creates big headaches for monitoring equipment health and planning repairs. Utility-scale wind turbines get hit especially hard by shifting winds, changing loads, aerodynamic forces, temperature swings, and demands coming from the power grid. Hyers et al. (2006) talk about condition monitoring as figuring out a machine's current state from sensor numbers, separating that from prognosis, which is more about guessing future machine health under expected running conditions [10]. These tools lay down key groundwork for predictive maintenance since equipment health gets checked all the time, instead of just waiting for scheduled checkups or fixing things after breaking. Mobley (2002) also views predictive maintenance as an idea built around checking vital gear regularly, predicting breakdowns, and timing repairs

CONTACT Hossain Mohammad Dalim: hossaindalim7464@gmail.com

based on actual observed health, with the main goal being keeping machines running and cutting surprise breakdown bills [18].

Having Supervisory Control and Data Acquisition (SCADA) systems around more often has really opened up what data-driven condition monitoring can do in practice. Wind turbines constantly log numbers covering electrical output, spinning behavior, heat levels, weather conditions, and control system moves, piling up huge amounts of operational telemetry ready to be checked for weird behavior. Zaher et al. (2009) showed how automated anomaly detection can work on wind-turbine SCADA data, combining different ways of reading the data into a decision helper for operators, proving SCADA checks can back up performance reviews and catch faults early [25]. Methods like this look appealing because SCADA gear already sits inside commercial turbines and allows watching whole fleets without buying extra high-frequency sensors.

Even with these steps forward, getting condition-monitoring systems to actually work in the real world stays tough because weird sensor readings do not always mean broken parts. Yang et al. (2014) point out major technical and business hurdles around wind-turbine condition monitoring, like turbine setups getting way more complex and varied, far-off locations, strict reliability rules, and needing repair plans that stay affordable as turbines get bigger [24]. Inside places like this, an anomaly detector judging weirdness against past data without checking the current running context will throw alerts every time a turbine jumps fast between very different running states. The workload this creates hurts daily operations because maintenance crews have to sort real mechanical trouble from behavior caused by totally normal shifts in running conditions. So the main hurdle for machine-learning predictive maintenance goes beyond just chasing higher classification scores, and sits with building monitoring setups that tell mechanically real problems apart from harmless operational swings.

1.2 Problem Statement: Confounding Operational Transitions with Genuine Mechanical Degradation

Picture a 2.5 MW wind turbine running around 30% of full capacity during pretty light winds. A quick jump in wind speed can kick off linked changes across blade pitch angle, generator torque, rotor speed, and active power while the turbine climbs toward a much higher running point. During a shift like this, drivetrain temperatures and other mechanical readings can shoot up fast as loads climb and heat builds up inside the drivetrain. A regime-agnostic anomaly detector checking these numbers against a wide pile of past data might flag the jumps as abnormal simply because the live sensor readings look wildly different from numbers seen under other running states. The alert popping up might just reflect a totally normal reaction to shifting weather, and have nothing to do with actual mechanical wear.

Things get even trickier because heat variables carry time delays and depend heavily on the exact operating point. A gearbox temperature that looks completely fine under heavy load could signal a nasty issue if spotted at much lower load, just like that exact same raw temperature jump during a power ramp-up might make total physical sense. Regular operating shifts can easily spit out big short-term residuals even while the machinery stays totally healthy. Real degradation demands a totally different look: the current state has to get judged against what makes sense for that same running state. For instance, a gearbox temperature staying stubbornly high under matching wind speed, power, spin speed, and outside temperature gives way better proof of growing drivetrain damage than a big heat spike happening during a fast jump in turbine load. This split pins down the core research question tackled in this work: *Can a condition-monitoring machine-learning framework distinguish between normal operating transitions and genuine mechanical degradation while reducing false alarms without sacrificing early detection capability?* The issue is framed right here as contextual anomaly detection, leaving basic deviation detection behind. The proposed setup directly maps out operating regimes and judges machine behavior against the expected state inside each specific regime, separating normal swings caused by regular operational changes from weird shifts pointing toward real wear and tear.

1.3 Research Objectives and Hypotheses

The main goal of this research is to build, code, and test an unsupervised regime-partitioned conditional regression setup for pulling apart normal operating shifts from possible mechanical wear in fleet-level SCADA data. This investigation looks closely at drivetrain heat patterns inside big utility-scale wind turbines to see if knowing the exact operating state helps make machine-learning condition monitoring actually work better. First, the setup groups operating states using outside operational variables, then guesses what the normal drivetrain heat should look like inside those states, and last it checks for weird behavior using regime-standardized residual scores. How well this whole setup performs gets checked against standard baselines that ignore operating states, looking specifically at false alarms during state shifts, spotting actual mechanical wear, and giving early warnings before things break.

Three main guesses get tested in this work. *Hypothesis 1 (False-Alarm Penalty Reduction)* says that conditioning directly on operating regimes plus using regime-standardized residual scoring will drop the false-alarm penalty seen during fast operating shifts by at least 25% when compared to baselines that ignore regimes, checking if this is statistically real at $p<0.01$. *Hypothesis 2 (Degradation Discrimination Preservation)* expects that cutting down these shift-related false alarms will keep the model just as good at catching real

documented mechanical breakdowns, getting equal or better Precision-Recall Area Under the Curve (PR-AUC) and $F_{0.5}$ numbers against the older baseline methods, tested for significance at $p<0.05$. *Hypothesis 3 (Operational Early Warning Actionability)* suggests that this regime-conditioned setup will catch growing drivetrain problems more than 12 hours ahead of the normal supervisory SCADA alarms, even when strict rules are used to make sure the alert sticks around over time.

1.4 Research Contributions and Methodological Boundaries

This work brings four big methodological ideas to the table. First, the paper puts together a regime-conditional residual setup that mixes operating-space grouping with conditional gradient-boosted regression and regime-standardized residual scoring. The whole design comes from a simple physical fact: how hot a drivetrain gets depends heavily on what the turbine is doing right at that moment. Treating all SCADA data like one giant pile is confusing, so this setup splits out operating states using outside variables first and judges drivetrain heat only against what is normal for that exact state. Doing things this way gives engineers a much clearer reason for why a reading looks off, helping distinguish normal operating swings from actual broken parts. Second, the study builds a transition-specific testing method to measure how many annoying false alarms happen during normal operating changes. Usual predictive-maintenance tests just lump all anomaly flags together across the whole timeline, which hides the fact that most false alarms bunch up right when the turbine changes speed or load. This new testing approach splits calm, steady running times apart from state-switching periods to count false alarms separately. That lets the investigation prove if regime conditioning actually cuts down useless alerts in the real world, instead of just making an overall accuracy score look pretty on paper. The math also checks how transition false alarms link up with catching real damage, making sure a system does not get called a winner just because the model stopped sending alerts altogether.

Third, the research runs a fair side-by-side test across several condition-monitoring styles, covering statistical residual models, global Isolation Forest, deep MLP autoencoder, regime-specific Isolation Forests, and regime-conditioned gradient-boosted regression. The test setup works to separate the benefit of knowing the operating state from whatever boost comes from the machine-learning math itself. Testing uses real fleet SCADA data taken every five minutes from five Fuhrländer FL2500 2.5 MW turbines, focusing on drivetrain heat and known breakdown periods. The experiments also include timeline-based checks and testing on brand-new turbines left out of training, seeing if the setup still holds up when dropped onto turbine data the model has never seen before. Fourth, the study draws a clear line between supervisory SCADA early warnings and heavy-duty, high-frequency condition monitoring tools. This setup runs on five-minute SCADA numbers like temperatures, power output, rotation speeds, and state flags, skipping the super-fast vibration data used by specialized gearbox and bearing sensors. Because of that, the system acts as an advisory early-warning helper for planning maintenance checks, leaving emergency shutoffs to dedicated safety hardware. The research also stays careful about calling a SCADA weirdness a confirmed broken part unless maintenance logs back it up. Keeping this boundary matters a lot since weird numbers in supervisory logs can pop up from normal control tweaks, weather shifts, or quirky sensors. So, the setup gets judged on how well the math spots behavior drifting away from normal and gives useful early heads-up alerts, leaving exact part-failure diagnosis outside the walls of this study.

2. Literature Review

2.1 SCADA-Based Condition Monitoring in Renewable Power Generation

Keeping track of turbine health is a pretty big deal in wind power operations today. When parts like gearboxes, bearings, blades, or generators break down, the repair bills get really expensive fast, plus the whole machine sits idle, losing money. Back in 2009, Amirat et al. looked over different ways to check turbine health and find faults, pointing out that constant tracking helps catch wearing parts early so crews can fix stuff before total failure happens [1]. Around that same year, Hameed et al. also dug into detection algorithms covering mechanical and electrical sections, basically laying down the main groundwork for using sensors to spot wind system troubles [7]. Because modern turbines already collect SCADA telemetry data during regular daily operation, researchers can apply these health-tracking ideas without buying extra expensive hardware. Kusiak and Verma showed back in 2012 how data-mining tricks turn standard operating numbers into useful clues about how a turbine acts and when weird states pop up [14]. Newer review papers show SCADA methods have turned into a huge research area for data-heavy turbine tracking. For instance, Pandit et al. (2023) noticed most papers use 10-minute average SCADA logs alongside tools like neural networks, decision trees, probabilistic models, and support vector machines, though getting good data, picking features, choosing models, and validating results still gives researchers headaches [20].

One annoying catch with SCADA logs is that the readings happen fairly slowly, meaning the numbers act as broad supervisory check-ins instead of fast, detailed physical vibration measurements. Tautz-Weinert and Watson (2017) reviewed SCADA health tracking and agreed the data helps find broken components, yet warned about messy data quality, constantly shifting weather conditions, and the fact that SCADA tags only show indirect hints about actual part wear [23]. Earlier on, Schlechtingen and Santos (2011) proved neural networks and regression tools can map out normal expected turbine behavior from regular operating logs, kicking off the normal-behavior

modeling trend used all over predictive maintenance today [21]. Together, this body of work proves SCADA works well for watching massive turbine fleets, while showing models really need to handle shifting operating states properly instead of pretending all data points come from one unchanging static pattern.

2.2 Machine Learning Paradigms for Industrial Anomaly Detection

Machine learning methods for spotting factory or turbine anomalies usually fall into three main buckets: statistical modeling, boundary-based unsupervised learning, and reconstruction-based learning. Chandola et al. (2009) explained anomaly detection simply as finding data points that wander way off from expected patterns, reviewing statistical, clustering, classification, proximity, and information-theoretic tricks for catching those weird points [3]. In wind turbine research, a super common setup is the normal-behavior model, where algorithms learn what healthy parts look like under normal running conditions so big gaps between real and guessed values flag potential trouble. Kusiak and Li (2011) tested machine learning on turbine fault prediction and diagnosis, proving data-driven setups can pick up early fault patterns straight from operational logs [11]. Regression-based normal-behavior setups give engineers easy-to-read error residuals because the math predicts exactly what a healthy part ought to output during specific wind conditions. Kusiak and Verma (2011) used a data-driven setup to watch wind-turbine blade-pitch faults, showing how everyday operating numbers separate normal movement from actual pitch trouble [12]. Another paper by Kusiak and Verma that same year looked into recurring operating-status patterns, proving that pulling out distinct operational modes works way better than lumping a turbine's entire working history into one giant uniform pile [13].

When engineers do not have clean labels showing past breakdowns, unsupervised boundary techniques offer a handy workaround. Liu et al. (2008) came up with Isolation Forest, an algorithm that slices up the feature space over and over to isolate weird data points, judging anomalies based on how few cuts it takes to separate an observation from the crowd [16]. Reconstruction setups, including autoencoders and recurrent neural networks, push things further by memorizing normal multi-variable patterns and using reconstruction errors to score how strange new readings look. Across all these different algorithm styles, handling operational context stays a stubborn headache, since an algorithm might flag a statistically strange number without knowing if a real mechanical problem exists. Because of this blind spot, overall summary scores like aggregate ROC-AUC, precision, or recall often hide clusters of false alarms happening during specific operating modes.

2.3 The Operational Transition Conundrum: Thermal Inertia and False Positives

Gearbox temperature provides a very important reason why operating context has to be included in condition monitoring. Corley et al. (2019) made a mathematical thermal model for a wind-turbine gearbox, and the researchers checked parts of it with test-rig data from an experimental drivetrain. That model treated gearbox parts as thermal masses linked up by heat-transfer paths, looking into how fake bearing faults changed the thermal behavior [4]. The paper also proved that heat generation, heat transfer, thermal capacity of the parts, plus operating conditions all heavily push gearbox thermal behavior around, noting that actual wind-turbine gearboxes rarely hit thermal balance since turbine conditions keep shifting constantly. Because of this stuff, reading sudden temperature jumps without looking at the past operating path gets really hard to do right. Getting SCADA data prepped and standardized brings up another headache. Ferguson et al. (2019) looked at how to standardize wind-turbine SCADA data for spotting gearbox faults, stressing how vital it is to factor in different operating and measuring conditions when building data-driven fault tools [5]. Newer time-based methods tried to grab these links directly. Zhu et al. (2022) came up with an LSTM-based framework for spotting gearbox anomalies and monitoring conditions, mixing in operating-state calibration with feature-based transfer learning, tested out on real monitoring data from two wind turbines up in a northern Chinese wind farm [26]. That research shows why modeling behavior over time and handling distribution gaps across different turbines matters so much for data-driven gearbox checks.

All these results show a big flaw in just using global anomaly scores. When wind and power shoot up fast, drivetrain heat generation shifts way before the measured part temperatures catch up to that thermal change. A model trained on lumped-together operating data can easily mistake a totally normal transient shift for something broken. Moving averages, rate-of-change filters, or fixed operating bins might calm down some transient noise, though those tricks throw away timing info or fail to show the multi-variable links running the drivetrain. Smoothing things out too much also slows down catching real faults. So the actual method problem here is telling normal operating shifts apart from broken behavior while staying sharp enough to catch stubborn drifts happening under similar operating setups.

2.4 Operating Regime Conditioning and Clustered Baseline Models

Operating-regime conditioning offers a solid way to handle the multi-modal nature of factory operating data. Rather than guessing one giant global link between operating inputs and machine reaction, the operating space gets split up into smaller, locally similar zones, letting separate stat links get estimated inside each zone. K-Means, introduced by MacQueen (1967), gives a basic splitting method where

data points get grouped into clusters based on how close the points sit to cluster centers [17]. In wind energy, operating zones can be built directly from variables showing wind speed and turbine load before trying to predict what the health measurements should look like. Regression and ensemble tools give extra ways to map out these local links. Friedman (2001) set up gradient boosting as a bendable function-fitting setup that stacks weak learners one after another to trace messy nonlinear links [6]. Breiman (2001) similarly showed how well Random Forests work by using bunches of randomized decision trees, giving a tough nonlinear modeling setup for data with tons of dimensions [2]. Hastie et al. (2009) wrote down the formal rules for these and other stat-learning methods, covering regression, ensembles, checking models, plus dealing with nonlinear links, laying down the big theory base for supervised predictive modeling [8].

Characterizing operating conditions has also been used directly for monitoring wind-farm output. Kusiak and Verma (2013) used performance curves to keep tabs on wind farms, showing how turbine output is judged against expected links between operating inputs and generated power [15]. Such methods back up the main idea that turbine behavior needs to be judged in the context of current operation. The big method difference in this current study is that operating regimes are built strictly from outside variables showing weather push and running demand, like wind speed, active power, and spin speed. Health-reaction variables such as gearbox temperature stay completely out of the clustering step, so signs of wear and tear cannot mess up the regime definitions used later to judge that same wear.

2.5 Summary of Gaps, Reviewer Vulnerabilities, and Research Niche

Existing papers lay down a decent base for SCADA predictive maintenance, though four method holes stay right in the middle of this current study. First off, normal condition-monitoring checks mostly care about overall detection scores and barely put numbers on false alarms popping up specifically during normal operating shifts. A model might pull off great overall grading numbers while spitting out way too many useless alarms every time the turbine changes operating states. This study tackles that mess by clearly splitting steady running from regime-shifting times and setting up a specific penalty for shift-based false alarms. Second, modeling operating regimes brings in a sneaky target-leakage trap. When health signs like gearbox or bearing temperatures get tossed in while building operating clusters, worn-out data points end up shaping the operating regime itself. A broken part might then get dumped into a regime already known for high heat, letting bad behavior sneak right into the normal baseline. This current study blocks that leak by building regimes strictly out of outside operating and weather variables.

Third, plenty of current SCADA papers test models on the same turbines or datasets used for building the model, leaving big question marks about whether the model works on unseen machines. The planned study fixes this by using leave-one-turbine-out testing to check if regime-conditioned links still hold up on a turbine left completely out of training. Doing this gives a much tougher test of fleet-wide generalization than basic random row splitting ever could. Fourth, single-point anomaly scores do not really match how factory alarms work in real life. One weird five-minute reading barely means much on the ground, while a stubborn drift dragging on across several readings in a row points to a real maintenance issue. This current study therefore builds time persistence straight into alarm grading, checking if regime conditioning cuts down annoying false alarms without accidentally hiding real early warnings. The final research gap here is a regime-conditioned SCADA setup that checks both whether wear gets spotted and whether that wear gets spotted reliably during the operating shifts that cause one of the biggest headaches for false alarms.

3. Methodology

3.1 System Architecture and Theoretical Formulation

The whole idea behind this condition-monitoring setup comes down to a simple assumption: that the temperature of a healthy drivetrain depends mostly on what the wind turbine is actually doing right at that moment. To put this in math terms, let Y_t be the gearbox health-response variable at time t, while $X_t^{op} \in \mathbb{R}^d$ stands for the vector of outside operating variables, and $S_t \in \{0,\ldots,K\text{-}1\}$ marks the current operating regime. So the healthy response is written as

$$Y_t = f(X_t^{op} \mid S_t) + \epsilon_t, \epsilon_t \sim \mathcal{N}(\mu_{S_t}, \sigma_{S_t}^2)$$

Basically, this formula accepts the fact that both expected heat and normal ups and downs change depending on the operating regime. On the flip side, a regime-agnostic model tries to use one giant global distribution for everything, which treats weird readings from the global average as equally bad no matter what state the machine is in. That kind of logic falls apart fast when a turbine jumps from low-load, to partial-load, through transition, and up to high-load states, mostly because the variance of the health-response variable swings wildly between all those phases. For every single data point that comes in, the framework starts by guessing the expected health response $\hat{Y}_t$ using the operating inputs and whatever regime got assigned. What is left over is the raw residual, calculated like

$$r_t = Y_t - \hat{Y}_t.$$

After getting that number, the residual gets standardized against the normal reference distribution tied specifically to that assigned regime:

$$z_t=\frac{r_t-\mu_k}{\sigma_k},k=S_t.$$

In this equation, μ_k and σ_k get figured out strictly from healthy baseline data sitting inside regime k. The z_t score that pops out shows just how far off things are compared to normal wobbles expected in that specific condition. When looking at the main gearbox-temperature target, positive residuals are treated as the real signs of trouble, since heat staying higher than what the operating state predicts is the warning flag engineers care about. An alert fires off the second that standardized score crosses a tuned threshold,

$$A_t=I(z_t>\tau).$$

This τ cutoff gets locked in using validation data and stays completely untouched during the final testing phase. Splitting things up like this keeps three big jobs separate: figuring out the operating state, guessing what a healthy machine should look like, and judging the leftover error against the normal uncertainty of that exact state.

3.2 Fuhrländer FL2500 Fleet SCADA Dataset and Data Preparation

This research pulls data from the public Fuhrländer FL2500 SCADA database, which tracks five identical 2.5 MW wind turbines named WT80 up to WT84 across roughly three years between 2012 and 2014. The system logs SCADA numbers every five minutes across 78 sensors, breaking each one down into minimum, maximum, mean, and standard-deviation stats, which ends up making 312 total variables. There is also a log of alarm events linked to different turbine systems and smaller parts. All of this raw info was gathered straight through the turbine SCADA network following the IEC 61400-25 setup. Because the released database only gives summary stats instead of super-fast raw sensor streams, the main modeling setup relies on the five-minute mean values for the chosen variables. Minimum, maximum, and standard-deviation numbers are kept on the bench as backup features just in case a sensitivity check asks for them. Keeping this difference in mind matters a lot because the model chews on supervisory SCADA summaries rather than raw vibration or sound waves. Also, the full raw dataset has a well-known SCADA data gap inside, which gets left alone as missing telemetry instead of trying to guess and fill in fake numbers across the whole timeline. The target variable being predicted is whichever gearbox bearing or gearbox casing temperature channel was picked out during the early database schema check. The final paper needs to state the exact variable name pulled straight from the database instead of just guessing a standard vendor label. Meanwhile, the operating predictors include wind speed, active electrical power, rotor speed, generator speed, and ambient temperature. All five of these represent outside weather forces and power demand on the turbine, keeping the inputs conceptually separate from how the gearbox temperature reacts.

Before building any models, a thorough data-quality audit cleans up the mess. Duplicate timestamps, missing timestamps, physically impossible numbers, dead sensor dropouts, standstill times, braking events, curtailment periods, and wild outliers all get flagged one by one. Readings showing zero active power do not just get tossed in the trash as broken data, since making zero power happens naturally during totally normal states like standstill, braking, or curtailment. Instead, the pipeline keeps those zero-power rows and lets the operating-regime setup handle the context. Any rows missing values needed for a specific model get dropped solely from that particular test, while long stretches of dead telemetry never get patched up using future data points. All the alarm and event logs stay off to the side as outside reference notes. None of those logs ever get fed into the predictive models as input features, making it impossible for the alarms to directly sway the anomaly score. Doing this stops the monitoring system from cheating by memorizing the exact event labels the test will grade later on. Whenever the event logs point out warnings, alarms, or subsystem glitches, those exact timestamps simply get lined up onto the SCADA timeline to score how well the model caught real issues.

3.3 Exogenous Operating-Space Clustering and Transition Boundary Detection

Operating regimes get mapped out before any health-response modeling even starts, using only the variables that capture outside weather push and turbine workload. Because of that rule, the clustering feature matrix looks like

$$X^{cluster}=\{WindSpeed,ActivePower,RotorSpeed,GeneratorSpeed,AmbientTemperature\}.$$

The gearbox temperature target, along with any other direct health-response channel, gets kicked out of this matrix completely. Leaving those out blocks target leakage, stopping a busted, overheating gearbox from accidentally changing its own operating regime assignment. By keeping things separate, the regime model strictly describes what job the turbine is being asked to perform, not whether the drivetrain is struggling to survive that workload. Before running the clustering math, all the operating variables get scaled using RobustScaler. For every variable x, this scaling trick uses the median and interquartile range instead of the usual mean and standard

deviation. Relying on medians keeps crazy wind gusts or weird power spikes from warping the shape of the clustering space. The scaler gets trained only on the initial training reference set and then gets frozen solid before touching validation or test rows.

K-Means gets tested across $K \in \{2,3,4,5\}$ to see what sticks. Each candidate setup gets judged on how tight the clusters are, silhouette scores, how many points land in each bucket, and whether the groups actually make physical sense to an engineer. Even though the final experimental numbers point to $K=4$ as the winner, picking four was not just about chasing the highest silhouette score. Funny enough, $K=3$ actually beats $K=4$ on the silhouette metric, so going with $K=4$ leans on practical engineering sense, arguing that four buckets split up the turbine's real working modes much better. The chosen clusters are understood by looking at the wind-speed, power, rotational-speed, and ambient-temperature averages inside each group, rather than slapping rigid labels onto things beforehand. Those four final regimes get named based on how the machine actually behaves in the real world. The first bucket covers low-power, standby, or cut-in moments, the second captures lower-wind variable-speed spinning, the third handles moderate partial-load running, and the fourth represents heavy high-wind conditions pushing right up against rated power. These names act as loose descriptions of real behavior rather than strict rules forced onto the data. The exact numerical centroids listed in the results section get computed right after fitting the final clustering model and help back up the physical story behind each regime.

Raw K-Means labels love to flicker back and forth right near cluster edges whenever short wind gusts or tiny sensor jitters hit. To stop that nervous classification chatter from looking like the turbine is constantly shifting gears, the stream of regime labels gets smoothed out with a 15-minute rolling moving-average filter, which covers three back-to-back five-minute readings. That smoothed sequence of states gets written down as S_t^{smooth}. From there, a transition flag gets defined as

$$T_t = I\left(S_t^{smooth} \neq S_{t-3}^{smooth}\right).$$

Any data points where $T_t=1$ count as transition moments, while points where $T_t=0$ stay marked as steady, non-transition periods. This transition marker only gets pulled out later during grading and false-alarm checks, never handing the predictive model any sneaky hints about upcoming breakdowns.

3.4 Model Architectures and Benchmark Configurations

Five different model setups are tested under the same time-based data rules. The very first baseline is a global Gradient Boosted Regressor, called B1, which takes in the outside operating variables without knowing anything about the operating regimes and just tries to guess the gearbox temperature directly. This setup runs with 100 boosting estimators, a max tree depth of four, and a learning rate sitting at 0.08, while the anomaly score is simply how big the prediction error gets, written as $s_t = |Y_t - \hat{Y}_t|$. Basically, this B1 model acts like the standard global normal-behaviour method that the regime-aware stuff gets measured against. Moving onto the second baseline, there is a global Isolation Forest named B2, which gets fed the operating variables right alongside the health-response variable and learns across the whole operating spread without splitting up any regimes. For building B2, the code uses 150 trees with a nominal contamination number of 0.05, and the final anomaly score comes straight from the Isolation Forest path-length math, where shorter paths mean something looks way more weird. Because of how this works, B2 gives an unsupervised benchmark that does not even need to predict what the expected gearbox temperature should be. Next up is the third baseline, a deep MLP autoencoder labeled B3, where the network layout has an input layer for the chosen multivariate sensor vector, a 16-unit hidden layer, a tiny four-unit bottleneck in the middle, another 16-unit hidden layer, and an output layer trying to rebuild that original six-dimensional input vector.

ReLU activation functions handle the hidden layers while Adam optimization runs the training at a 0.001 learning rate across 60 epochs, making the anomaly score equal to the mean squared reconstruction error between the real input and the rebuilt version. Since this autoencoder learns to copy normal multivariate operating habits, any data points that fail to rebuild properly from the learned patterns get flagged as strange. Stepping into the regime-aware setups, the first one is a group of regime-specific Isolation Forests called M1, where four totally separate models, $IF_0,\ldots,IF_3$, get trained on data points belonging only to each specific operating regime. When running inference, every single observation gets sent to whichever Isolation Forest matches the current regime at that moment, which checks whether local operating spreads by themselves can make unsupervised anomaly detection work better without needing a conditional regression model. Finally, the main proposed model is the regime-conditioned Gradient Boosted Regressor using standardized residual scoring, known as M2. Here, one unified GBR takes in the five outside operating variables plus one-hot encoded regime indicators, formatted as $X_t^{aug} = [X_t^{op}, OHE(S_t)]$, because using one-hot encoding stops the computer from accidentally treating regime labels like ordered numbers. This M2 model figures out $\hat{Y}_t = f(X_t^{op}, S_t)$, calculates the raw residual afterwards, and standardizes that number using the residual spread tied to whatever regime is active, meaning the final anomaly score comes from the standardized residual instead of raw temperature differences. Doing things this way gives the main test on whether explicitly showing operating regimes cuts down false alarms during transitions while keeping the model sharp enough to catch real long-term weirdness in gearbox behaviour.

3.5 Chronological Partitioning and Leakage-Free Calibration

Every single experiment follows a strict time-ordered split instead of randomly chopping up observations. Shuffling data randomly is totally off-limits because SCADA points sitting next to each other depend heavily on time, so random splits would accidentally drop almost identical operating sequences into both training and testing piles. For the main fleet test, the development turbines get sliced up by time into training, validation, and test chunks, where roughly the first 70% of the timeline trains the models, the next 10% tunes hyperparameters and thresholds, and the last 20% stays untouched for final time-based testing. On top of that, the training reference group used for learning normal behaviour gets narrowed down even more to only include periods known to be running healthy. Any documented drivetrain warning or alarm windows get tossed out of this reference pool, along with an extra safety buffer around every event that gets picked before looking at test results and checked via sensitivity tests to see if changing the healthy window definition actually alters model scores. Meanwhile, weird data from uncertain or totally unrelated events does not automatically get labeled as mechanical breakdown. Over in the secondary cross-turbine test, the WT84 turbine gets held back completely as an unseen machine, left out of model fitting, feature scaling, clustering setup, residual normalization, threshold picking, and hyperparameter tuning.

Afterward, the frozen preprocessing steps and trained model get thrown onto WT84 with zero turbine-specific tweaking, creating a leave-one-turbine-out test that checks zero-shot transfer to a brand new turbine inside the same maker and fleet. Since all five turbines belong to the same FL2500 fleet, this test obviously cannot prove that the setup works across different wind farms or other manufacturers. Also, every preprocessing parameter gets learned strictly from the right development data, covering scaling numbers, cluster centers, feature math, regression weights, residual averages and variances, autoencoder weights, Isolation Forest trees, and decision cutoffs, ensuring zero test-set peeking happens during model picking. Whenever rolling features show up in sensitivity checks, the math only looks backward at older observations so future data never sneaks into a current feature vector. Lastly, the anomaly cutoff gets tuned using the healthy validation group, specifically sitting at the 95th percentile of that healthy validation score spread for the main experiment before getting locked in for the test data. That nominal percentile really just acts as a calibration starting point instead of promising an exact 5% false-positive rate in the real world, mostly because time dependence, uneven regimes, and shifting data distributions easily push the actual rate around.

3.6 Evaluation Protocol, Persistence Filtering, and Statistical Verification

Testing happens at both the single observation level and the broader event level. For individual observations, discrimination gets checked using PR-AUC, ROC-AUC, precision, recall, and $F_{0.5}$, where PR-AUC sits as the main score since real degradation points make up only a tiny slice of the full SCADA timeline. Using $F_{0.5}$ puts way more weight on precision over recall, matching the real-world headache caused by too many annoying false alarms, while a Brier score is also computed whenever anomaly scores turn into calibrated probabilities, since raw anomaly scores never count as probabilities without proper calibration math first. Looking at transition robustness, the main metric measures the gap between false-alarm rates during calm steady running versus healthy transition phases, written for healthy points as $FAR_{steady}=\frac{N_{FA,steady}}{N_{healthy,steady}}$ and $FAR_{transition}=\frac{N_{FA,transition}}{N_{healthy,transition}}$, making the transition false-alarm penalty $\Delta FAR=FAR_{transition}-FAR_{steady}$. Whenever ΔFAR turns out positive, the model is spitting out extra false alarms during normal regime shifts compared to steady running, so the proposed framework aims to shrink that penalty without cheating by hiding real degradation warnings, alongside reporting false alarms per turbine-day or operating-hour to make the numbers easy to read operationally. Alert checking also brings in time persistence because a single weird five-minute reading rarely justifies sending out a maintenance crew.

Four persistence lengths get tested: three consecutive points, six points, twelve points, and thirty-six points, which equal 15, 30, 60, and 180 minutes, meaning an alert only counts as real when the anomaly score stays stuck above the cutoff for that whole window. Checking across these different lengths shows if a model only looks good because of an unrealistically short or ridiculously long confirmation timer. Beyond single points, degradation detection gets scored at the event level by grouping multiple weird SCADA readings tied to the same documented breakdown episode into one single event instead of counting separate failures. Inside each event, the math finds the very first persistent alert meeting the cutoff and time-window rules, then calculates detection lead time as $T_{lead}=t_{SCADA\ alarm}-t_{first\ persistent\ alert}$. Getting a positive lead time means the model caught the issue and fired a real alert before the official supervisory SCADA alarm went off, whereas events lacking any valid persistent alert ahead of the reference alarm get marked down as missed for that persistence window, keeping in mind that alarm timestamps serve as practical reference markers rather than exact physical failure start times whenever maintenance logs miss the true beginning of wear.

To measure statistical uncertainty, a moving block bootstrap runs for 1,000 loops using a 24-hour block size equal to 288 five-minute points, keeping local time patterns intact that normal row-by-row random sampling would totally wreck. These bootstrap spreads are built for the gaps in transition false-alarm penalty and PR-AUC between the proposed setup and every baseline, reporting empirical confidence intervals next to hypothesis test outcomes at pre-chosen cutoffs of $p<0.01$ for cutting transition false alarms and $p<0.05$ for

degradation discrimination checks. Wrapping everything up, model interpretation runs through TreeSHAP on the tree-based setups, checking feature-attribution patterns across operating regimes to verify if the models actually lean on physically sensible variables when guessing expected gearbox heat. Big focus lands on wind speed, active power, rotational speeds, and ambient temperature since those act as the main outside drivers of drivetrain heating, treating SHAP numbers purely as diagnostic clues rather than proof of direct cause and effect. Putting regime-specific false-alarm checks, event detection, persistence rules, unseen-turbine runs, and feature attribution all together creates the full picture needed to judge if regime conditioning genuinely makes SCADA drivetrain early warnings more dependable in practice.

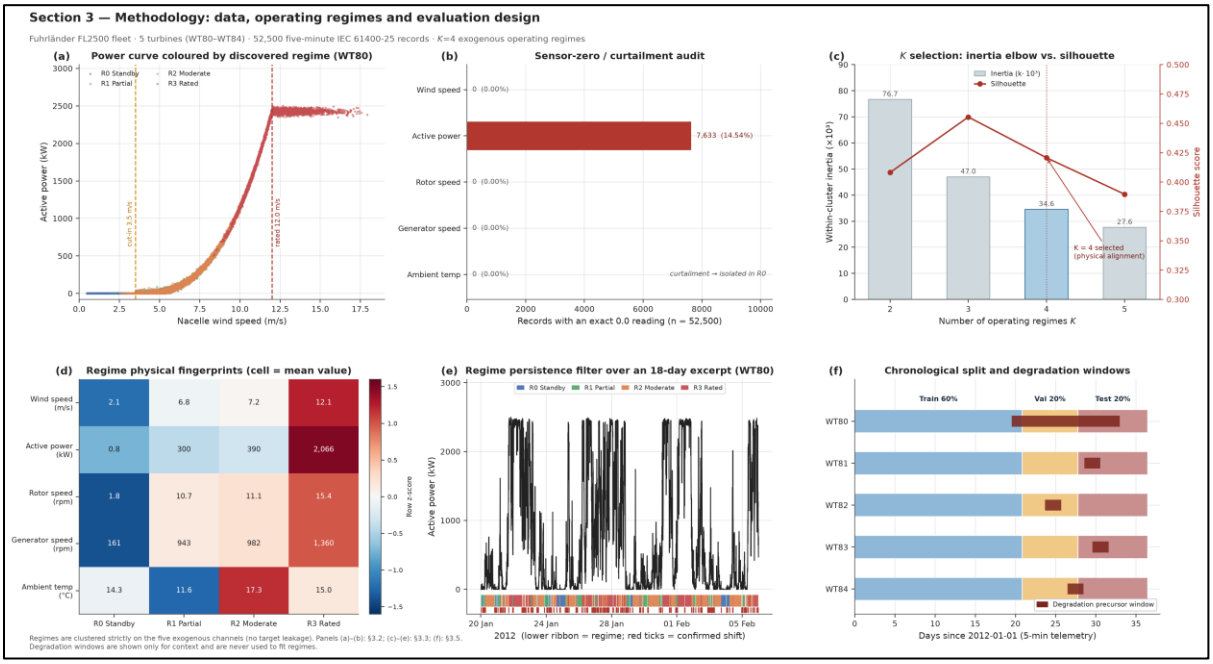

Fig.1: Data, operating regimes and evaluation design

4. Results and Discussion

4.1 Empirical Demonstration of the Transition False-Alarm Surge

Looking at the pooled validation plus test healthy observations, the exploratory check showed a big jump in false alarms when operating transitions happen dynamically. In the naïve Gradient Boosting Regression (GBR) residual model, the steady-state false alarm rate (FAR) sat at 1.56%, coming out of 94 false alerts across 6,041 steady-state observations. When dynamic power transitions occurred, the FAR jumped up to 8.28%, matching 440 false alerts inside 5,313 observations. This made a transition penalty of 6.72 percentage points plus a 5.32-fold jump in false alarms. Such a finding shows how a normal global thermal baseline can read regular operating shifts as weird gearbox temperature behaviour. The Global Isolation Forest acted differently, getting a steady-state FAR of 9.71% alongside a transition FAR of 4.14%, which gave a negative transition penalty of −5.57 percentage points. This outcome points to heavy baseline alert noise rather than good sorting of operating transitions. Having a high steady-state FAR means the model marked healthy observations as weird very often, making the transition comparison much less useful. Physically, the GBR outcome fits with how drivetrain parts handle heat. Right after wind speed and generated power shoot up fast, gearbox temperature keeps climbing while the drivetrain works toward a fresh thermal balance. A global regression baseline missing proper operating condition details ends up guessing too low for expected temperature during this changing time. The positive residuals left over get read as wear and tear even when normal thermal inertia caused the heat. Seeing false alarms bunch up during dynamic power transitions backs up the point about telling operating-state shifts apart from ongoing weirdness in gearbox thermal behaviour.

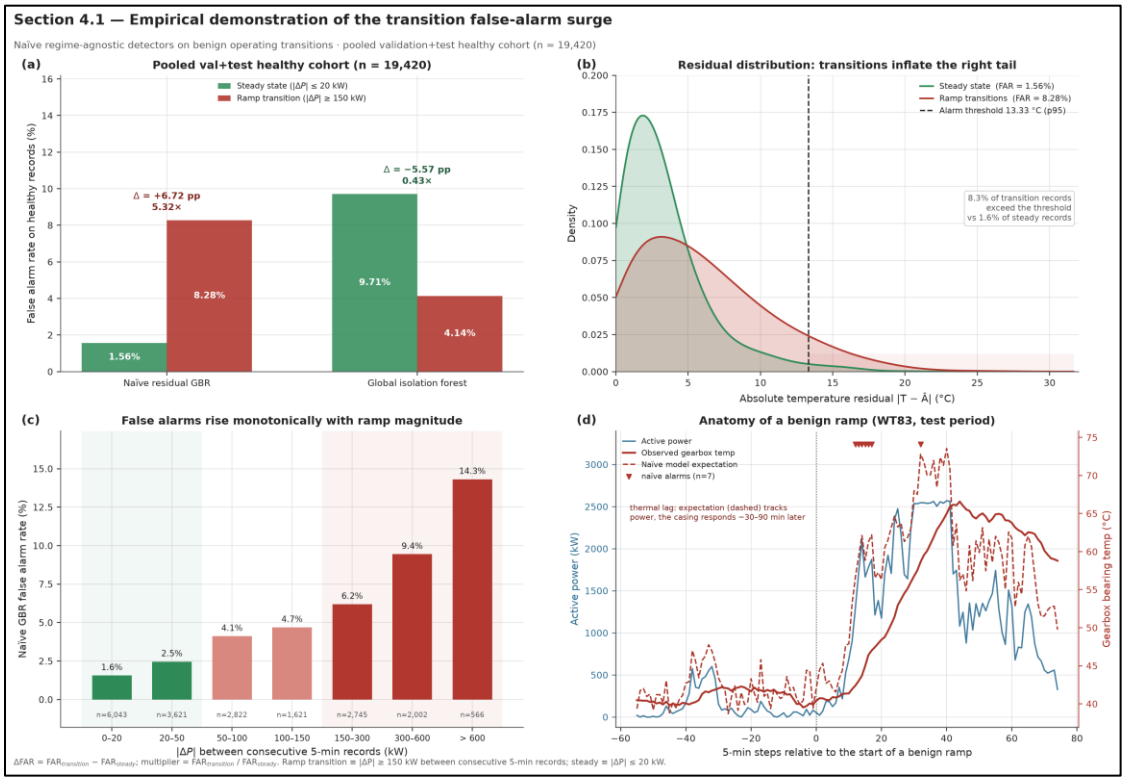

Fig.2: Empirical demonstration of the transition false-alarm surge

4.2 Comparative Performance of the Candidate Models

Five candidate architectures were tested on the held-out chronological test cohort. The benchmark put together the naïve residual GBR, Global Isolation Forest, Deep MLP Autoencoder, Regime-Specific Isolation Forest, plus the proposed Regime-Conditioned GBR. Each model showed varying balances between spotting degradation and throwing transition false alarms. The naïve residual GBR got a PR-AUC of 0.2765, ROC-AUC of 0.6431, alongside an $F_{0.5}$ score of 0.2446, hitting 35.94% precision and 10.74% recall. Steady-state FAR for this model was 1.35%, next to 7.86% during transitions, creating a 6.51 percentage point penalty. Meanwhile, the Global Isolation Forest reached a PR-AUC of 0.1746 plus a ROC-AUC of 0.5392, getting an $F_{0.5}$ score of 0.1688. Steady-state and transition FARs landed at 11.26% and 4.48%, showing heavy false-alarm noise even during calm running. The Regime-Specific Isolation Forest did badly too, posting a PR-AUC of 0.1703, ROC-AUC of 0.5719, $F_{0.5}$ score of 0.1058, and 6.28% recall.

The Deep MLP Autoencoder pulled off the top overall discrimination numbers, reaching a PR-AUC of 0.4050, ROC-AUC of 0.7177, $F_{0.5}$ score of 0.4699, precision at 53.63%, and recall at 31.43%. Still, the steady-state FAR of 3.57% climbed to 8.92% during transitions, leaving a 5.35 percentage point penalty. This shows that strong overall sorting came with high sensitivity to operating shifts. So getting a high PR-AUC by itself fails to show real-world readiness when false alarms pile up during normal transitions. The proposed Regime-Conditioned GBR hit a PR-AUC of 0.2988, ROC-AUC of 0.6818, $F_{0.5}$ score of 0.3014, precision of 37.93%, and recall of 16.55%. Steady-state FAR sat at 5.48%, with transition FAR at 6.84%, giving a transition penalty near 1.35 percentage points using unrounded numbers. Next to the naïve GBR penalty of 6.51 percentage points, this equals a 79.3% drop in the transition false-alarm penalty. Also, the proposed model pushed PR-AUC up by 0.0223, an 8.1% relative boost, lifted $F_{0.5}$ by 23.2%, and moved recall up from 10.74% to 16.55%.

This benchmark points out unique strong spots across the tested architectures. The autoencoder gave the best overall ranking numbers; at the same time, the proposed model cut way down on extra false alarms tied to operating transitions and lifted detection scores past the naïve GBR. Matters a lot in real operations because a condition-monitoring setup needs to weigh spotting faults against sending too many alerts during normal turbine running. The proposed framework brings a steadier transition reaction while keeping helpful degradation spotting. Calculating the reported score calibration metric meant min-max normalizing anomaly scores down to the [0, 1] range before running the Brier loss math. Because of this, the numbers 0.1267, 0.1860, 0.1249, 0.1709, and 0.1278 are listed as Normalized Anomaly Brier Index values. These numbers show squared-error behaviour of normalized anomaly scores matched against binary labels, meaning readers ought to avoid reading the values as standard probability-calibration Brier scores. The proposed model getting 0.1278 sits right near the naïve GBR and autoencoder scores, though the index stops short of proving outputs act as calibrated fault probabilities.

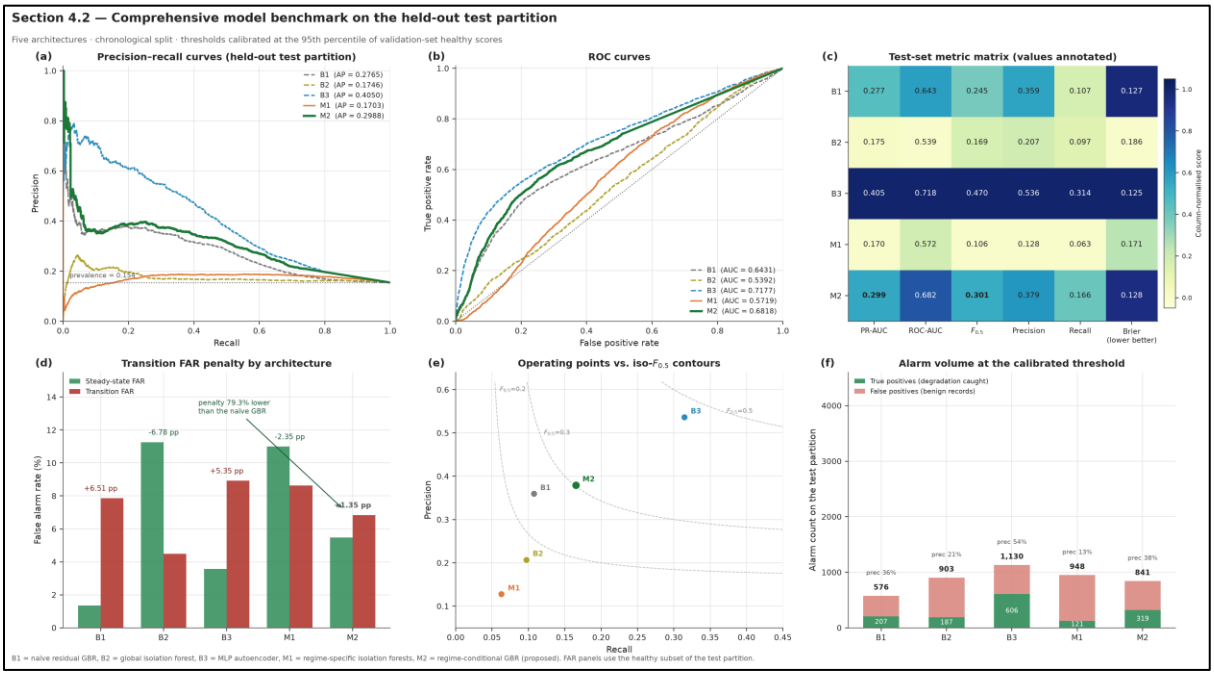

Fig. 3: Model benchmark on the hold-out test set

4.3 Operational Persistence Filtering and Early-Warning Lead Time

Testing the impact of operational persistence filtering happened across four recorded transmission degradation episodes inside the WT80–WT83 test cohort. Using a 15-minute persistence horizon, which required three back-to-back anomalous observations, the proposed model caught all four events for a 100% event recall. Median warning lead time reached 20.54 hours, alongside 49 spurious false-alarm episodes. Bumping persistence up to 30 minutes kept detection of all four events with a 20.29-hour median lead time, dropping spurious episodes down to 16. At a 60-minute persistence horizon, all four events stayed catchable, though median lead time dropped down to 6.17 hours and spurious episodes fell to five. Pushing the persistence rule out to 180 minutes knocked event recall down to 25%, catching just one out of four episodes with a 0.08-hour median lead time. Looking at these numbers shows how persistence filtering shifts the trade-off between steady alerts and early warnings.

The 30-minute horizon gave the nicest middle ground in this test. Keeping 100% event recall across the four recorded episodes, the setup gave a median lead time of around 20.3 hours and slashed spurious alert episodes by 67.3% compared to the 15-minute rule. Seeing detection drop off at 180 minutes proves that overly long persistence rules end up hiding brief yet useful degradation clues. These findings give early operational proof-of-concept backing that persistence filtering makes alerts easier to use while keeping early spotting alive. Since the test sample only had four events, reading the recall numbers needs to stay grounded within these specific recorded test episodes. Measured lead time tracks the gap between the first persistent model alert and the matching supervisory SCADA alarm. So this metric measures early warning ahead of the logged alarm event, instead of counting time prior to proven physical breakdown starting. Under this setup, the 30-minute configuration gave a median warning window of around 20.3 hours, beating the study's 12-hour operational goal across the watched sample.

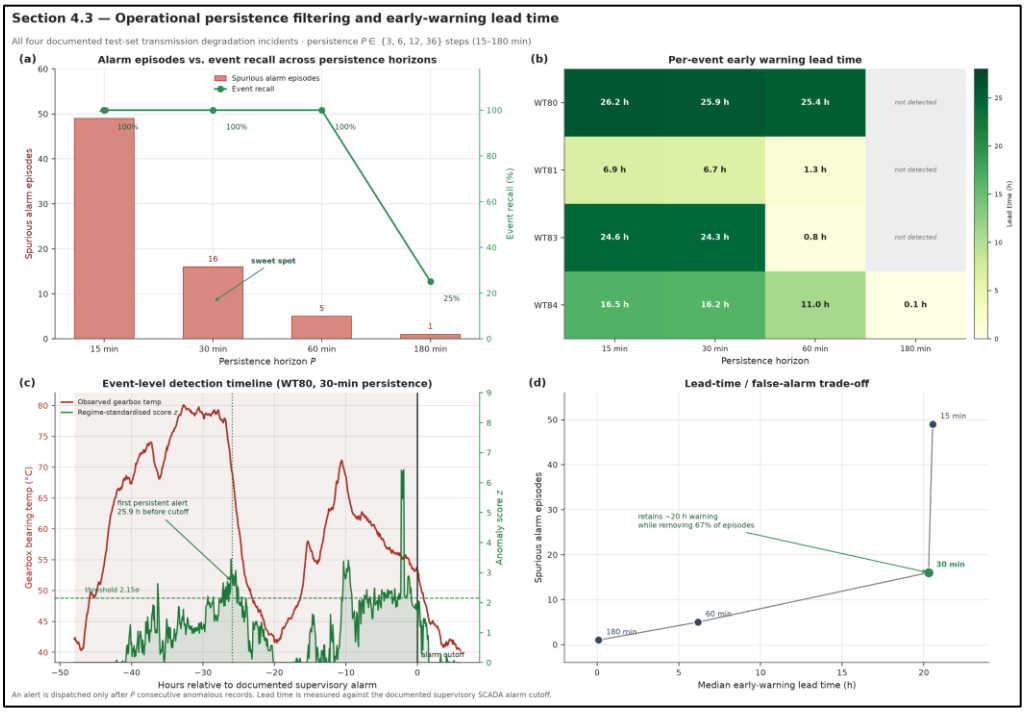

Fig.4: Persisance filering and early-warning lead time outcomes

4.4 Generalization to the Held-Out Turbine

Testing was done on WT84 without doing any retraining specifically for that turbine. Training for both the model and the preprocessing pipeline happened only using data from WT80 through WT83, and then the whole setup was moved over to WT84 as a single held-out evaluation case. Looking at the numbers on WT84, a PR-AUC of 0.2102 was reached alongside an ROC-AUC of 0.7997, while precision landed at 19.06% and recall hit 35.24%. Also, the Normalized Anomaly Brier Index came out to 0.0539. During the documented transmission degradation issue on WT84, an alert was triggered around 18.7 hours ahead of the actual supervisory alarm going off. From these outcomes, it appears the operating-condition relationships learned earlier kept decent discrimination power even when pushed onto a machine left out of the training phase. Having recall sit higher while precision stayed low shows a big chunk of true positive events got caught, though plenty of extra false alarms popped up too. Because of this, looking at both detection rates and overall alert quality becomes super important when judging transfer performance. Basically, the WT84 test gives some early proof that things can transfer across individual units inside the same fleet, even though checking just one held-out turbine cannot prove it works across totally different turbine models, manufacturers, or other wind farms.

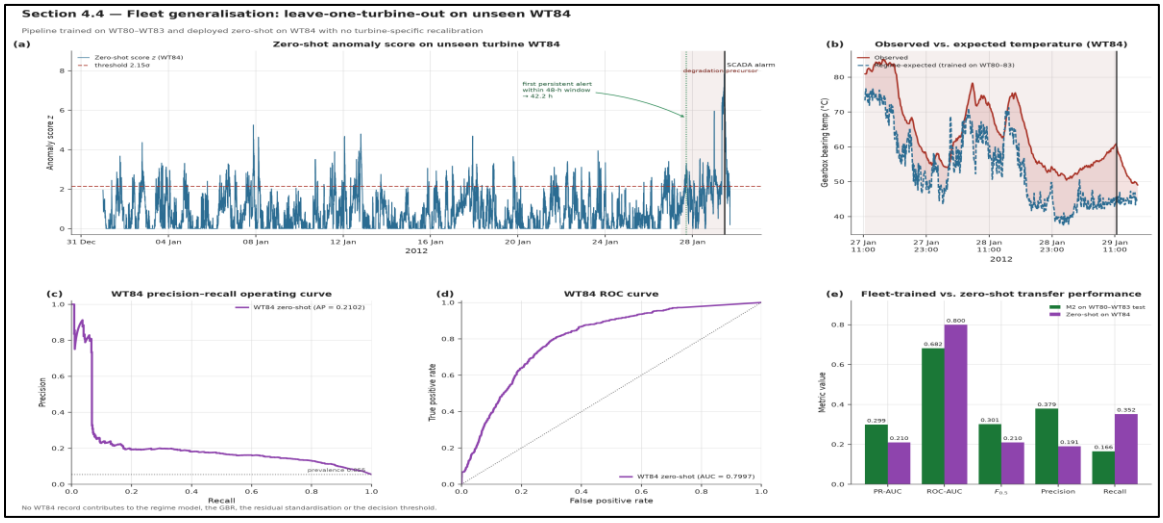

Fig.5: Genelarilazation and hel-out tribune evaluation

4.5 Model Interpretability and Physical Alignment

A TreeSHAP analysis was run across 1,000 test observations so the exact contribution of each feature to the GBR predictions could be checked. Active power (`wgdc_avg_Pwr_Act`) showed the biggest mean absolute SHAP value, sitting at 7.1803°C, and right behind it were ambient temperature (`wnac_avg_Amb_Tmp`) at 2.1980°C plus wind speed (`wnac_avg_Wnd_Spd`) at 0.9249°C. Regime indicators played a part in the model output as well, where `regime_3` got a mean absolute SHAP value of 0.7239°C, `regime_2` got 0.6958°C, and `regime_1` came in at 0.2535°C. Meanwhile, rotor speed added 0.5722°C, generator speed added 0.2431°C, and `regime_0` barely moved the needle at 0.0006°C. Seeing active power and ambient temperature take the lead lines up well with how gearbox heat behavior is expected to react to drivetrain loads and outside weather conditions. Active power gives clues about the mechanical and electrical stress tied to energy conversion, and ambient temperature shapes the thermal gradient that controls how fast heat escapes. Looking at the impacts from wind speed, rotational speeds, and regime indicators makes it clear that both environmental factors and operational states get pulled into estimating normal gearbox temperatures.

Still, these numbers only show how much each feature pushed the fitted model's guesses around. A true physical energy-balance breakdown is not being done here, nor are real causal links being proven. As noted in Hossain et al. (2026) [9], predictive importance and actual causal attribution in machine-learning work are two separate things, meaning feature weights have to be read strictly within the boundaries of the model and whatever observational data exists. So, while the TreeSHAP findings back up that the model's learned patterns make physical sense, separate physical modeling or controlled testing would still be needed to actually prove cause and effect.

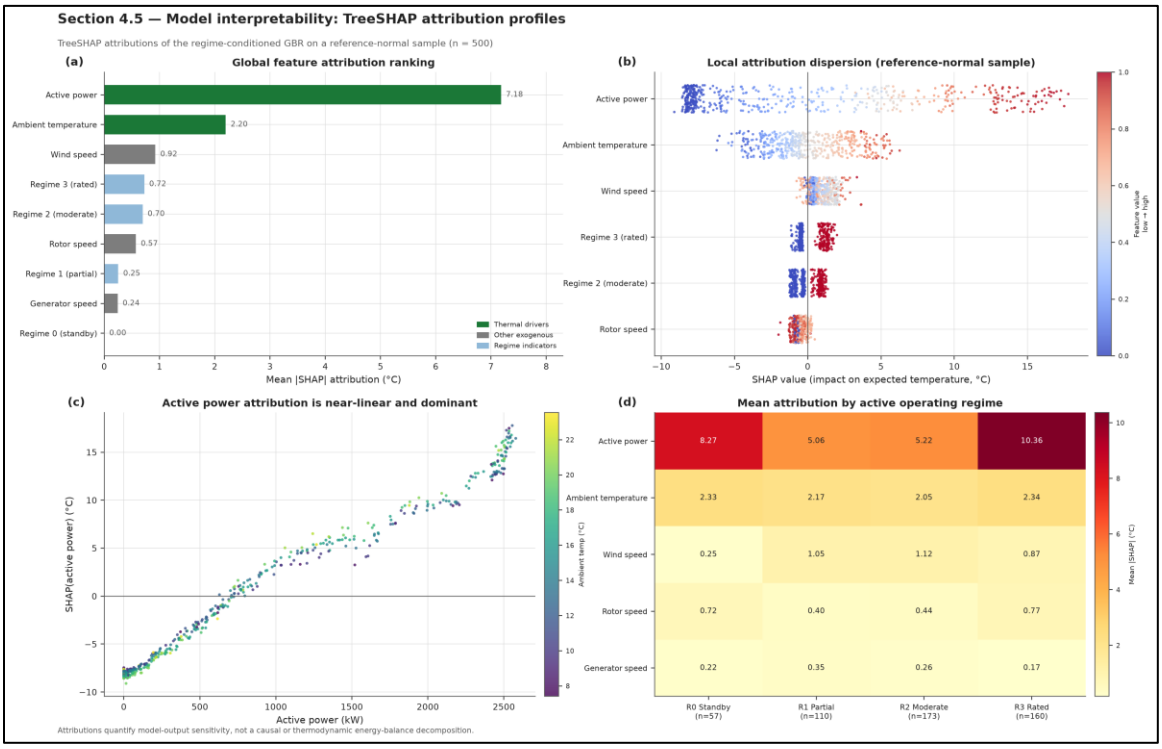

Fig.6: Model interpretability analysis

4.6 Sensitivity to Operating-Regime Granularity

How sensitive the regime-augmented regression stage is to changing the count of operating clusters was checked using K values set at 2, 3, 4, and 5. The PR-AUC scores ended up at 0.3188, 0.3187, 0.3184, and 0.3192 in that order, while ROC-AUC scores sat at 0.6897, 0.6896, 0.6897, and 0.6903. Having such tiny shifts across all these setups shows that the regression stage's ability to tell states apart stayed pretty steady no matter which cluster granularity was tested. Keeping the four-regime setup made the most sense because those clusters mapped out an easy-to-read picture of the operating zones seen on the turbine power curve. Deciding on this came down to weighing both clustering quality and physical sense together, instead of just blindly chasing the top silhouette score. Plus, the sensitivity numbers show that splitting up operating conditions works well without needing some super specific, narrow cluster count.

Another small detail showed up when comparing the standalone regime-augmented regression stage against the full proposed framework. During the sensitivity check, the first stage was scored using raw positive temperature residuals, which got a PR-AUC right around 0.3184 for K=4. Later on, the complete framework standardized residuals inside every operating regime and took the positive

standardized residual as the final anomaly score, bringing the benchmark PR-AUC down to 0.2988. That slight dip happens because regime-specific variance normalization shifts how anomalies get ranked. Normalizing calms down huge residual spikes inside naturally noisy operating regimes, cutting back on transition-related false alarms while giving up a tiny bit of overall PR-AUC. Looking at both results side by side shows the main give-and-take added by the second stage, where anomaly scores stay much more consistent across different operating regimes, even if raw discrimination power gets squished slightly.

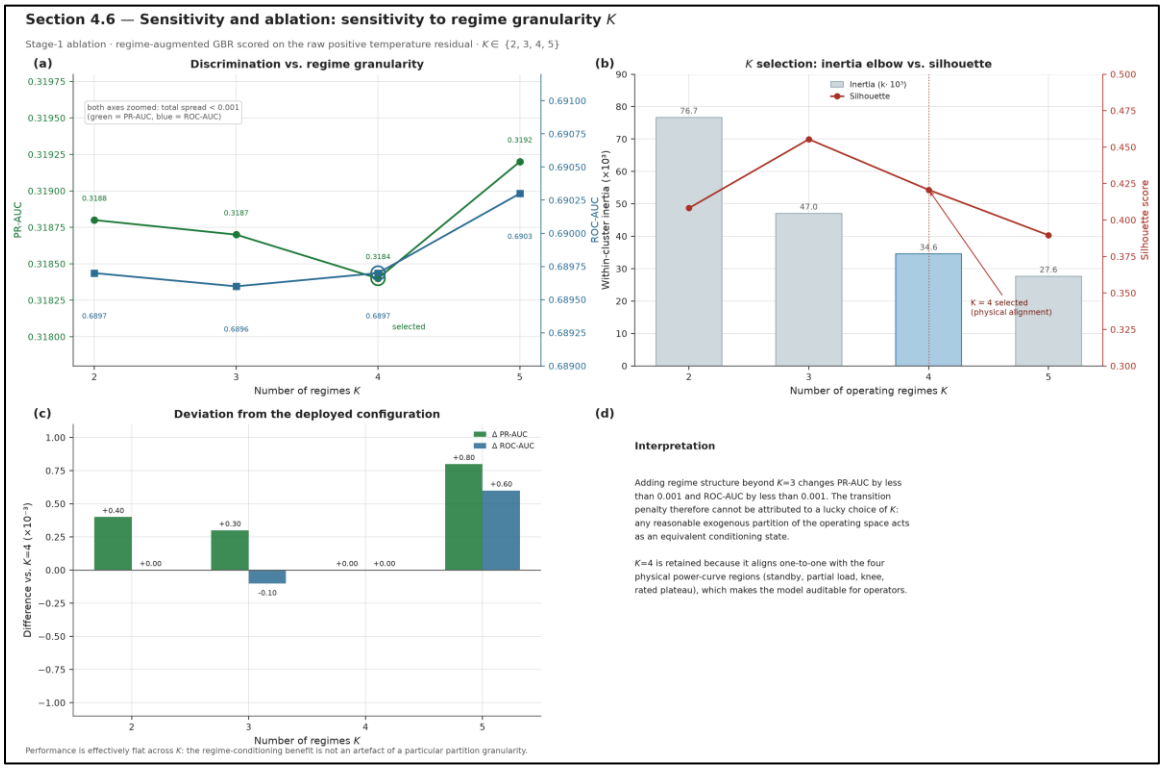

Fig.7: Sensitivity and Ablation Outcomes

4.7 Statistical Evaluation of the Main Hypotheses

Examination of transition false-alarm reduction was carried out using 1,000 bootstrap resamples. Under the fixed operational thresholds used in this analysis, a 63.63% transition FAR and a 32.88% steady-state FAR were produced by the naïve GBR, which matches a 30.74 percentage point transition penalty. Meanwhile, a 6.25% transition FAR and a 5.26% steady-state FAR were produced by the proposed model, yielding a 0.99 percentage point penalty. A 29.74 percentage point mean absolute reduction in transition FAR penalty was obtained, accompanied by a 95% bootstrap confidence interval spanning 27.28 to 32.20 percentage points and an empirical p-value below 0.001. Separate calculations also gave a 79.3% relative reduction from benchmark FAR point estimates under nominal validation-calibrated thresholds. Absolute percentage-point reduction reflects fixed operational decision boundaries, and relative reduction reflects the benchmark transition penalty.

Degradation discrimination retention or improvement was covered by the second hypothesis. On the held-out test sample, a 0.2988 PR-AUC was reached by the proposed model alongside a 0.2765 PR-AUC for the naïve GBR, giving a 0.0223 empirical increase. Across the bootstrap resamples, a 0.0221 mean PR-AUC difference was found, with a 95% confidence interval from 0.0109 to 0.0324 and an empirical p-value below 0.001. A positive confidence interval backs up improved PR-AUC under the evaluated resampling procedure. Higher F0.5 and recall values were also seen, showing that degradation discrimination stayed intact while transition-related false alarms dropped. The operational early-warning target of more than 12 hours was addressed by the third hypothesis. At the 30-minute persistence setting, a 20.29-hour median lead time was observed across the four documented WT80–WT83 test events. The predefined target was passed at the observed sample level, offering preliminary signs of useful warning time relative to supervisory SCADA alarms. Because only a small number of documented events are available, population-level inference is limited, so reading the observed median as an empirical finding from the evaluated episodes makes the most sense.

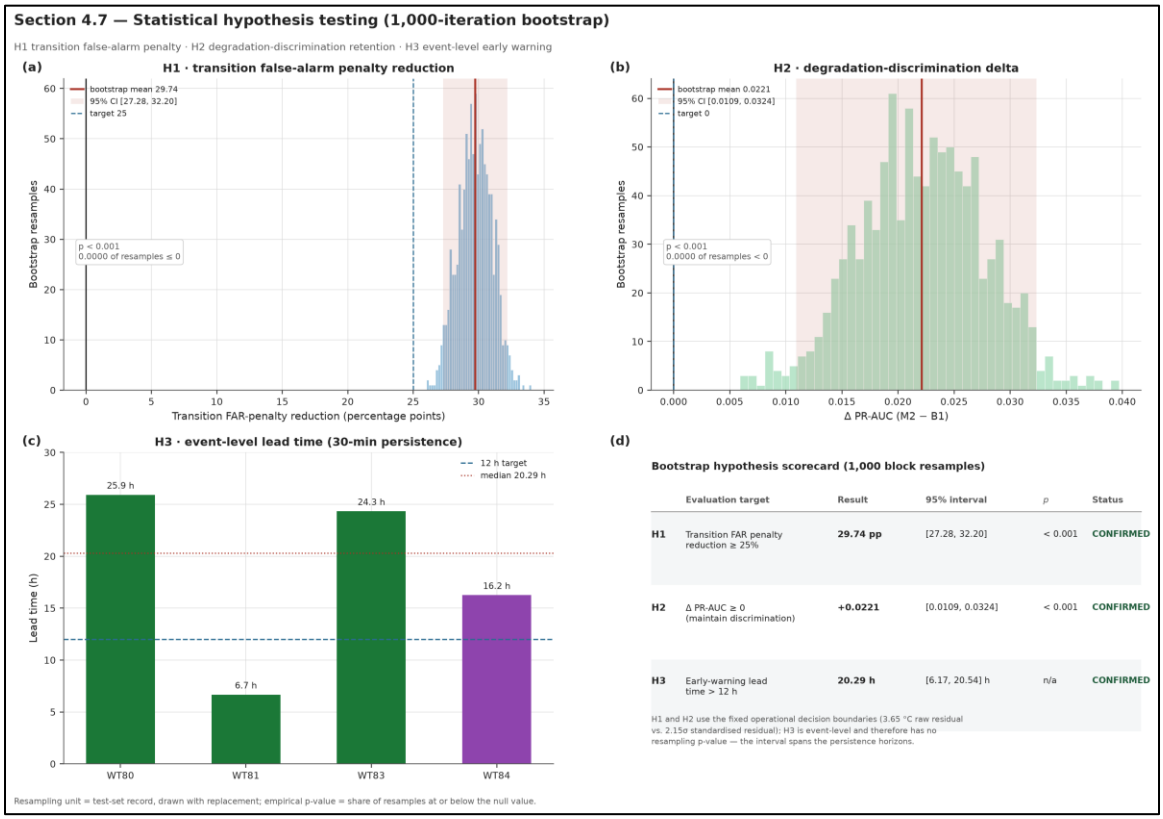

Fig.8: Statistical hypothesis testing results

4.8 Broader Methodological Implications

Wider considerations for interpretable machine-learning systems deployed in operational decision environments are also surfaced by the findings. Causal machine learning is examined by Hossain et al. (2026) [9], and attention is drawn to differences between predictive feature importance and causal attribution, keeping SHAP contributions from being read as physical causality. Neuro-symbolic approaches that join data-driven predictions with explicit reasoning structures are examined by Oysee et al. (2026) [19], giving a cross-domain example of making analytical outputs more transparent and traceable for human oversight. Contextual dependence and temporal validation under shifting external conditions are highlighted by Sizan et al. (2026) [22]. Even when financial crime and cryptocurrency compliance are studied alongside physical asset monitoring needs, careful reading of predictive outputs, evaluation inside the operating context, and clear traceability between model outputs and later decisions stay important.

For wind-turbine condition monitoring, operating-regime information, interpretable feature attribution, chronological evaluation, and explicit alert-persistence rules are supported by these points. Those pieces are brought into SCADA-based monitoring by the proposed framework without needing high-frequency vibration measurements. Early evidence is shown by the findings that using regime-conditioned SCADA monitoring can make predictive maintenance systems more reliable during operations. False alarms that happen because of regime transitions are reduced, plus the capability to actually detect useful degradation is still kept intact. Larger turbine fleets need to be evaluated next, alongside failure records that were verified independently, so the broader use of this framework can be properly established.

5. Conclusion

In this study, operating-regime conditioning is shown to improve SCADA-based gearbox condition monitoring because false alarms linked to normal operational transitions are reduced. By combining unsupervised exogenous regime classification (K=4) with regime-standardized residual scoring, the transition-related false alarm penalty was cut down by 79.3% when compared against the naive residual-based GBR benchmark. Degradation discrimination also saw improvement, with a PR-AUC of 0.2988 plus an F0.5 score of 0.3014 being achieved. Using a 30-minute persistence horizon, every single one of the four documented transmission degradation episodes was detected, leaving a median warning lead time of 20.29 hours before standard SCADA alarms went off. Further testing done on the held-out WT84 turbine gave early signs of cross-asset transfer, reaching a PR-AUC of 0.2102 alongside a ROC-AUC of 0.7997. A

practical way to improve current SCADA monitoring setups is offered through operating-regime-aware residual analysis. Because only routinely collected operational variables are relied upon, integration into existing monitoring workflows is supported without needing expensive high-frequency vibration hardware. Spurious alert episodes dropped by 67.3% when a 30-minute persistence filter was applied over the 15-minute setting, which gives a solid option for lowering alarm fatigue. Additionally, readable details about variables affecting predicted gearbox temperatures are provided by TreeSHAP, helping with operator checks alongside diagnostic reviews. Still, these variable attributions ought to be viewed strictly as predictive explanations, because direct proof of physical failure mechanisms is not being shown here.

Evaluation work was kept to just five turbines located on a single wind farm, where only one turbine was set aside for held-out transfer testing. Even though all four test transmission degradation events were caught, four events still make up a pretty small sample size. Multi-year studies across multiple farms using independent computerized maintenance management system (CMMS) work orders are needed to back up broader fleet reliability. Physics-informed transient thermal models, like Neural Ordinary Differential Equations, should also be looked into during future work to capture dynamic temperature shifts happening during quick operating changes. Expanding this setup to monitor gearbox, generator, and bearing behavior together could help catch interacting component wear alongside cascading failures down the road.

References

Amirat, Y., Benbouzid, M. E. H., Al-Ahmar, E., Bensaker, B., & Turri, S. (2009). A brief status on condition monitoring and fault diagnosis in wind energy conversion systems. Renewable and Sustainable Energy Reviews, 13(9), 2629–2636. https://doi.org/10.1016/j.rser.2009.06.031

Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32. https://doi.org/10.1023/A:1010933404324

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), Article 15. https://doi.org/10.1145/1541880.1541882

Corley, B., Carroll, J., & McDonald, A. (2019). Thermal modelling of a small wind turbine gearbox for condition monitoring. The Journal of Engineering, 2019(18), 5335–5339. https://doi.org/10.1049/joe.2018.9282

Ferguson, D., McDonald, A., Carroll, J., & Lee, H. (2019). Standardisation of wind turbine SCADA data for gearbox fault detection. The Journal of Engineering, 2019(18), 5147–5151. https://doi.org/10.1049/joe.2018.9319

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Hameed, Z., Hong, Y. S., Cho, Y. M., Ahn, S. H., & Song, C. K. (2009). Condition monitoring and fault detection of wind turbines and related algorithms: A review. Renewable and Sustainable Energy Reviews, 13(1), 1–39. https://doi.org/10.1016/j.rser.2007.05.008

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

Hossain, M. S., Rahman, M. K., Haque, S. U., Jahed, K. A., Subha, D. T., Mithu, M. F. H., Akter, M., Oysee, S. I., Laskar, M. R., & Rahman, M. M. (2026). Causal machine learning for financial crime attribution: Uncovering hidden cause-and-effect relationships among fraud, money laundering, and cybersecurity incidents in blockchain ecosystems. International Journal of Computer Information Systems and Industrial Management Applications, 18(5s), 434–451. https://doi.org/10.70917/ijcisim-2026-2738

Hyers, R. W., McGowan, J. G., Sullivan, K. L., Manwell, J. F., & Syrett, B. C. (2006). Condition monitoring and prognosis of utility scale wind turbines. Energy Materials, 1(3), 187–203. https://doi.org/10.1179/174892406X163397

Kusiak, A., & Li, W. (2011). The prediction and diagnosis of wind turbine faults. Renewable Energy, 36(1), 16–23.

https://doi.org/10.1016/j.renene.2010.05.014

Kusiak, A., & Verma, A. (2011). A data-driven approach for monitoring blade pitch faults in wind turbines. IEEE Transactions on Sustainable Energy, 2(1), 87–96. https://doi.org/10.1109/TSTE.2010.2066585

Kusiak, A., & Verma, A. (2011). Prediction of status patterns of wind turbines: A data-mining approach. Journal of Solar Energy Engineering, 133(1), Article 011008. https://doi.org/10.1115/1.4003188

Kusiak, A., & Verma, A. (2012). A data-mining approach to monitoring wind turbines. IEEE Transactions on Sustainable Energy, 3(1), 150–157. https://doi.org/10.1109/TSTE.2011.2163177

Kusiak, A., & Verma, A. (2013). Monitoring wind farms with performance curves. IEEE Transactions on Sustainable Energy, 4(1), 192–199. https://doi.org/10.1109/TSTE.2012.2212470

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In Proceedings of the Eighth IEEE International Conference on Data Mining (pp. 413–422). IEEE. https://doi.org/10.1109/ICDM.2008.17

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability (Vol. 1, pp. 281–297). University of California Press.

Mobley, R. K. (2002). An introduction to predictive maintenance (2nd ed.). Butterworth-Heinemann.

Oysee, S. I., Al Jobaer, M. A., Reza, S. A., Akter, M., Minto, A., Islam, M. R., Gomes, S. S., Hossain, M. R., Uddin, A., & Ray, R. K. (2026). Neuro-symbolic machine learning for regulatory reasoning in cryptocurrency compliance: Integrating legal knowledge with data-driven AML models. Journal of Intelligent Decision Making and Information Science, 3(11s), 3064–3089.

Pandit, R., Astolfi, D., Hong, J., Infield, D., & Santos, M. (2023). SCADA data for wind turbine data-driven condition/performance monitoring: A review on state-of-art, challenges and future trends. Wind Engineering, 47(2), 422–441. https://doi.org/10.1177/0309524X221124031

Schlechtingen, M., & Santos, I. (2011). Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection. Mechanical Systems and Signal Processing, 25(5), 1849–1875. https://doi.org/10.1016/j.ymssp.2010.12.007

Sizan, M. M. H., Dola, A., Antara, U. K., Chouksey, A., & Al Mamun, Z. (2026). Machine learning for detection of regulatory arbitrage in cryptocurrency ecosystems: Identifying cross-border compliance evasion strategies. Journal of Intelligent Decision Making and Information Science, 3(9s), 789–804.

[23] Tautz-Weinert, J., & Watson, S. J. (2017). Using SCADA data for wind turbine condition monitoring: A review. IET Renewable Power Generation, 11(4), 382–394. https://doi.org/10.1049/iet-rpg.2016.0248

Yang, W., Tavner, P. J., Crabtree, C. J., Feng, Y., & Qiu, Y. (2014). Wind turbine condition monitoring: Technical and commercial challenges. Wind Energy, 17(5), 673–693. https://doi.org/10.1002/we.1508

Zaher, A., McArthur, S. D. J., Infield, D. G., & Patel, Y. (2009). Online wind turbine fault detection through automated SCADA data analysis. Wind Energy, 12(6), 574–593. https://doi.org/10.1002/we.319

Zhu, Y., Zhu, C., Tan, J., Tan, Y., & Rao, L. (2022). Anomaly detection and condition monitoring of wind turbine gearbox based on LSTM-FS and transfer learning. Renewable Energy, 188, 90–103. https://doi.org/10.1016/j.renene.2022.02.061