Predictive Maintenance for Reduced Downtime A Model Survey and Recommendation

Jason Im Sonith
School of Computing
University of South Alabama
Mobile, Alabama
jis2123@jagmail.southalabama.edu

Steve Nguyen
School of Computing
University of South Alabama
Mobile, Alabama
tn1423@jagmail.southalabama.edu

***Abstract*—Unexpected machine downtime is expensive, especially for small and mid-size plants with messy sensor data, few labeled failures, and limited computing power. This project will test and compare practical ways to spot problems early while keeping the system simple to set up and run. We will evaluate simple rule-based thresholds on health signals, anomaly detectors such as Isolation Forest [5] and k-Nearest Neighbors, and small autoencoders. Our evaluation focuses on what matters in day-to-day operations: how early a warning appears, how often it is a false alarm, how effectively it identifies rare failures (precision and recall), how fast it detects issues, and how much CPU power and memory it needs on the device. We will also study how understandable the alerts are by using feature attributions (for example, SHAP [8]) so operators can see what caused a warning and review incidents. The expected output is a clear-cut decision manual that bridges the plant limitations, such as sensor coverage, labeling effort, and hardware limitations, to proposed models and configurations. We shall offer a lightweight pipeline for data cleaning, model training, threshold calibration, and alerting. In line with prior research, we expect lightweight anomaly detectors with robust feature engineering to compete with or surpass heavier models in terms of early warning and stability but incur lower operational costs. Using explanations in thresholding also reduces false alarms at a fixed recall.**

Index Terms - Predictive maintenance, anomaly detection, autoencoder, isolation forest, k -NN, SHAP, class imbalance, edge deployment, downtime reduction.

I. Introduction

Unplanned equipment downtime can bring entire production lines to a halt, causing major financial losses and safety risks for operations of any size. For smaller plants, where sensor data are noisy, recorded failures are rare, and on-site computing is limited, heavy deep learning models are often impractical. Under these constraints, teams need methods that deliver reliable early warnings while remaining lightweight, interpretable, and maintainable for a small staff [10].

We study a range of predictive maintenance approaches. These include simple thresholds on engineered health indicators, classic anomaly detectors such as Isolation Forest and k-Nearest Neighbors [1], [5], and compact representation learning models such as shallow autoencoders. Our focus is practical. Although fixed thresholds often fail to adapt to noisy and changing sensor data and deep learning models are too heavy for smaller plants, lightweight anomaly detection offers a middle ground: balance accuracy, efficiency, and ease of use. We evaluate not only detection quality but also day-to-day operational needs. Key criteria are early-warning lead time, false-alarm burden, stability when failures are rare, and CPU and memory use for laptop or edge deployment [10].

A second focus is operator trust and post-incident review. We use feature-level explanation tools, for example SHAP, to show why an alert fired and to guide threshold tuning [8]. This helps reduce alarm fatigue and creates a repeatable process for model review and improvement [7].

The outcomes are twofold. First, we provide an evidence-based recommendation that maps common plant constraints, such as sensor availability, labeling effort, and hardware limits, to a short list of model families and settings. Second, we deliver a minimal, reproducible pipeline for preprocessing, training, thresholding, and alerting that runs efficiently on standard CPUs. By grounding model selection in real operating conditions, the project aims to reduce unplanned downtime with methods that are clear, fast, and easy to adopt.

Our contributions are:

1) A comparative study of lightweight anomaly detection and compact representation learning under limited labeled data and limited computing resources [1], [5].
2) An evaluation protocol centered on lead time, false-alarm rate, and precision and recall under class imbalance, plus resource usage [6].
3) An explanation-guided thresholding procedure using SHAP to improve precision at a fixed recall [8].
4) A minimal, deployable pipeline for data preprocessing, model training, threshold calibration, and alerting, designed to run efficiently on standard CPU hardware and accompanied by a concise runbook for adoption.

By choosing models that fit real-world limits, this project aims to cut surprise downtime with methods that are clear, fast, and easy to maintain.

II. Background

Predictive maintenance relies on analyzing sensor data to detect early signs of equipment failure before breakdowns occur. Over the years, several model families have been proposed to tackle this challenge, ranging from simple statistical thresholds to machine learning methods.

Early work introduced methods such as Local Outlier Factor (LOF) [1] and One-Class Support Vector Machines (SVMs) [2], which learn patterns of "normal" operation and flag unusual behavior as anomalies. Later, approaches like Isolation Forest [5] improved scalability by directly isolating abnormal points instead of modeling the full data distribution. These methods are lightweight and well-suited for smaller datasets, but their performance depends heavily on how features are chosen and scaled.

In recent years, researchers have also paid attention to model interpretability. Techniques such as SHAP (SHapley Additive Explanations) [8] allow users to understand which sensor features most influenced an anomaly score. This is important in industrial settings, where operators need to know why a model activated an alert so that they can have confidence in and act upon it.

At the same time, there is growing interest in edge computing and lightweight artificial intelligence for machine condition monitoring [10]. Surveys show that small and mid-size plants often cannot afford the infrastructure needed for large deep learning models, highlighting the need for methods that balance accuracy, efficiency, and ease of deployment.

Taken together, this background shows a clear progression: from early statistical and density-based anomaly detection, to scalable algorithms like Isolation Forest, to interpretable and edge-friendly models. These insights guide our focus on lightweight anomaly detection and shallow representation learning as practical solutions for smaller plants.

III. Related Work

Research in predictive maintenance has explored multiple options dealing with anomaly detection approaches, interpretability tools, and deployment strategies. In this section, we present related work on three main fronts: traditional anomaly detection methods, model interpretability, and lightweight deployment for industrial settings.

A. Anomaly Detection Methods

Early work in anomaly detection for predictive maintenance focused on density- and boundary-based techniques. Local Outlier Factor (LOF) [1] detects unusual points by comparing local density, while One-Class SVM [3] builds a boundary around normal data and flags anything outside as anomalous. More scalable methods such as Isolation Forest [5] improve efficiency by randomly isolating points in feature space, making them effective for high-dimensional or noisy sensor data. These methods remain widely used because they perform well even with limited labeled failures.

B. Interpretability and Trust

Model accuracy alone is not enough in industrial environments—operators must also understand why an alert was raised. Black-box models can create resistance to adoption if their predictions are opaque. To address this, interpretability frameworks such as SHAP (SHapley Additive Explanations) [7] provide feature-level attributions that explain anomaly scores. This not only improves trust but also helps teams tune thresholds and conduct post-incident reviews, reducing alarm fatigue in practice.

C. Deployment Strategies

Another important challenge is how predictive maintenance systems are deployed. Large deep learning models often require GPUs and dedicated infrastructure, which are impractical for small and mid-size plants. Surveys on edge AI and machine condition monitoring [8] highlight the importance of lightweight, resource-efficient methods that can run on standard CPU hardware while still providing timely and reliable alerts. These approaches emphasize accessibility, simplicity, and minimal upkeep.

D. Synthesis

Taken together, prior research highlights three key themes: effective anomaly detection under limited labeled failures, interpretability for operator trust, and deployment feasibility in constrained environments. While each area has seen progress, few studies integrate all three. This motivates our proposal: a lightweight, interpretable pipeline designed for small and mid-size plants with noisy data and limited computing resources.

IV. Methodology

A. Datasets and setup

We use four public datasets that together cover both failure progression and labeled fault states. **Run-to-failure (progression):** NASA C-MAPSS turbofan degradation with RUL targets [4] and the IMS (PHM 2009) bearing dataset collected until physical failure [12]. These two provide explicit failure evidence and support early-warning and RUL experiments. **Labeled fault states (supervised baselines):** CWRU bearing data with healthy vs. seeded faults on a 2 HP motor [11] and the AI4I 2020 tabular dataset with machine-failure labels [9]. These provide clear positive/negative labels for fault detection/classification.

We run two setups. **Unsupervised** models train on healthy data and flag abnormal shifts (mainly IMS/CWRU vibration streams). **Supervised** models train with labels when available (AI4I, CWRU; and C-MAPSS when mapped to binary "at risk"). We split by unit/run (e.g., turbofan engine ID or bearing run) to avoid leakage, keep a validation split for tuning, and reserve a final test split that remains untouched.

B. Feature extraction and scaling

For vibration data (IMS, CWRU) we compute standard indicators: RMS, peak, kurtosis, crest factor, band-limited energy, and short rolling stats (mean, std). For C-MAPSS we keep operational settings and engineered health metrics; for binary framing, we threshold RUL to mark "at risk." For AI4I we retain numeric fields and one-hot encode categories. All features are standardized using statistics from the training split only.

C. Models

We compare simple, reproducible models. (1) **Indicator rules:** z-scores and EWMA with per-asset thresholds. (2) **Anomaly detectors:** Isolation Forest, K-Nearest-Neighbors (KNN), and One-Class SVM for unsupervised vibration. (3) **Compact autoencoder:** trained on healthy data; score by reconstruction error. (4) **Supervised baselines:** Logistic Regression and a shallow Gradient Boosting/XGBoost model for AI4I, CWRU, and the C-MAPSS binary setup. Hyperparameters are tuned by grid search on validation.

D. Thresholding and imbalance

Unsupervised scores use validation-based thresholds targeting a fixed false-alarm rate. Supervised models use class weights and probability calibration on validation. We aggregate alerts within short windows so one spike does not yield many duplicates.

E. Evaluation

We report metrics matched to each dataset's role. **Operational metrics** (from progression datasets IMS/C-MAPSS): lead time (time between first alert and failure onset when available) and false alarms per asset-week. **Classification metrics** (AI4I, CWRU; C-MAPSS binary): precision–recall curves and average precision on imbalanced data. **RUL metrics** (C-MAPSS with continuous targets): RMSE and MAE. We also track **runtime and memory** on a CPU-only laptop. All thresholds and hyperparameters are set on validation before testing. We average over multiple non-overlapping splits and report mean$\pm$std.

F. Explainability

For top alerts in supervised models, we compute SHAP values on engineered features and list the main drivers (e.g., "high-band RMS increased"). If a feature repeatedly drives false positives, we adjust its window or threshold and revalidate.

V. Potential Challenges

Limited labeled failures and sensor noise may cause thresholds to get shaky. To keep them stable, we use robust scaling, median filtering, and calibration across similar assets. Data can also drift over time, so we track alert rates and score distributions and plan regular refreshes of thresholds and models. Finally, hardware is limited, so we keep models small, choose CPU-friendly algorithms, and set firm caps on model size.

VI. Ease of Use

A key challenge for predictive maintenance systems is not only model accuracy but also operator adoption. Many small and mid-size plants lack the resources for specialized infrastructure or advanced ML engineering support. Therefore, our proposed pipeline emphasizes simplicity, reproducibility, and low overhead.

Hardware and Software Footprint: The pipeline is designed to run on standard CPU hardware (e.g., laptops or low-power edge boxes) with modest memory requirements. Models such as Isolation Forest, k-NN, and shallow autoencoders can be executed efficiently using common open-source libraries (scikit-learn, PyTorch).

Configuration and Workflow: System operation is streamlined to reduce technical barriers. Data preprocessing, feature extraction, and threshold calibration are controlled by a single configuration file, enabling quick adaptation to different assets without code changes. Operator workflows follow a simple review–decide–feedback loop: review alerts and contributing features, decide whether maintenance action is needed, and log feedback for threshold or model refinement.

Model Transparency and Trust: Each alert is accompanied by feature-level explanations (via SHAP), ensuring that operators can understand the cause of warnings. This reduces alarm fatigue, supports post-incident review, and encourages iterative trust-building.

Maintenance and Drift Handling: To remain robust under changing operating conditions, the system includes procedures for periodic threshold recalibration, drift monitoring, and lightweight retraining. These measures ensure stable operation over time without requiring constant expert intervention.

By focusing on ease of deployment, interpretability, and minimal upkeep, the proposed system provides not only technical performance but also practical feasibility for smaller plants operating under limited resources.

Conclusion

The goal of this project is to contrast light predictive maintenance methods that are viable for resource-constrained plants. Our goal is to implement these methods on public vibration and temperature data, contrast their performances, and quantify trade-offs between speed, accuracy, and interpretability.

Its broader impact is to provide smaller organizations with a short, evidence-based guide to model choice that reduces downtime without requiring special infrastructure investment. By bridging technical analysis and user usability, this research aims to close the gap between machine learning research and implementation in real-world use in resource-constrained settings.

Acknowledgment

The authors thank Dr. Jesse Ables and the School of Computing at the University of South Alabama for guidance and feedback on the project design. We also thank classmates in CSC Data Mining for formative discussions, and acknowledge the open-source tools used in this work (Python, scikit-learn, and optional PyTorch). This project uses publicly available predictive maintenance datasets; we thank the dataset maintainers for making these resources accessible to the community.

References

[1] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000, pp. 93–104. DOI: 10.1145/335191.335388. [Online]. Available: https://dl.acm.org/doi/10.1145/335191.335388.

[2] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. Müller, Eds., MIT Press, 2000, pp. 582–588. [Online]. Available: https://papers.nips.cc/paper_files/paper/1999/file/8725fb777f25776ffa9076e44fcfd776-Paper.pdf.

[3] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001. DOI: 10.1162/089976601750264965.

[4] A. Saxena and K. Goebel, *Turbofan engine degradation simulation data set (c-mapss)*, Dataset, NASA Prognostics Center of Excellence, Run-to-failure trajectories with remaining useful life labels, 2008. [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

[5] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, 3:1–3:39, 2012. DOI: 10.1145/2133360.2133363. [Online]. Available: https://dl.acm.org/doi/10.1145/2133360.2133363.

[6] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PLOS ONE*, vol. 10, no. 3, e0118432, 2015. DOI: 10.1371/journal.pone.0118432. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118432.

[7] International Society of Automation (ISA), "Isa-18.2 alarm management standard updated," *InTech Magazine*, 2016. [Online]. Available: https://www.isa.org/intech-home/2016/may-june/departments/isa18-alarm-management-standard-updated (visited on 09/20/2025).

[8] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, 2017. arXiv: 1705.07874. [Online]. Available: https://arxiv.org/abs/1705.07874.

[9] *Ai4i 2020 predictive maintenance dataset*, CC BY 4.0, UCI Machine Learning Repository, 2020. DOI: 10.24432/C5HS5C. [Online]. Available: https://archive.ics.uci.edu/dataset/601/ai4i+2020+predictive+maintenance+dataset.

[10] A. Bala, R. Singh, and S. Kaur, "Artificial intelligence and edge computing for machine condition monitoring: A survey," *Artificial Intelligence Review*, 2024. [Online]. Available: https://link.springer.com/article/10.1007/s10462-024-10748-9.

[11] Case Western Reserve University Bearing Data Center, *Bearing data center*, Dataset, 2025. [Online]. Available: https://engineering.case.edu/bearingdatacenter (visited on 09/20/2025).

[12] NASA, *Ims bearings*, Dataset, 2025. [Online]. Available: https://data.nasa.gov/dataset/ims-bearings (visited on 09/20/2025).