OpsPilot

Anomaly Detection — User Manual

Pattern-Based Monitoring Beyond Thresholds · AI Engineering Co-Pilot

What this guide covers — what anomaly detection adds beyond threshold monitoring, how the OpsPilot module designs it, what to have ready, and the output you receive.

1. What is anomaly detection?

Anomaly detection finds problems that simple thresholds miss — patterns across multiple variables that signal something is wrong before any single reading crosses an alarm. Threshold monitoring asks “is this one number too high?”; anomaly detection asks “is this combination of readings behaving unusually for these conditions?” A bearing temperature, a vibration level and a load that are each individually normal can together form a pattern that isn't — and that's what this catches.

It follows the ISO progression that distinguishes *detection (something is wrong), diagnosis (what / where / why) and prognosis (when will it fail — Remaining Useful Life)*, per ISO 13379, ISO 13381 and ISO 17359, with IEEE 2418.6 for trusted machine learning. It is the pattern-based counterpart to the threshold-based Condition Triggers module.

> **The deployment trap.** Most industrial ML programs fail not in the lab but in deployment — data drift, false-positive rates at scale, and operators who stop trusting the system. OpsPilot designs for those realities, not just for accuracy on a curated dataset.

2. What the OpsPilot module does

Role	Responsibility
🎯 **AI Coach — Reliability Engineer with ML specialisation (OpsPilot)**	Designs a defensible anomaly-detection approach — multi-variable pattern detection, ML-augmented where it earns its place, RUL estimation — and, critically, designs for deployment reality: false-positive management, data drift, and operator adoption.
👷 **Reliability Engineer / Data Scientist (you)**	Provide the equipment, the available data, the failure history and the operational context — and judge whether the approach is realistic for your data and your operators.

3. Detection → Diagnosis → Prognosis

Stage	The question it answers
Detection	Something is wrong — the pattern is abnormal for these conditions.
Diagnosis	What, where and why — which failure mode the pattern points to.
Prognosis	When will it fail — an estimate of Remaining Useful Life.

4. What you will be asked — have this ready

- The equipment and the data you actually have (the variables, the history, the quality).
- The failure history — what's gone wrong before, and what the lead-up looked like.
- The operational context — the conditions that make readings vary legitimately.
- An honest view of false-positive tolerance and operator readiness.

5. What you receive — the output

A defensible Anomaly Detection design (Word): the detection approach (multi-variable, ML-augmented where justified), the diagnosis and prognosis (RUL) elements, and — as a first-class part of the design — the deployment plan covering false-positive management, data-drift handling and operator adoption.

6. Worked example (illustrative)

A compressor where threshold alarms have never given useful warning — failures arrive with the first alarm already too late. Anomaly detection looks across discharge temperature, vibration, flow and ambient conditions together, learning what normal correlations look like across the operating envelope. A developing fault shows as a subtle drift in the relationship between vibration and load — each variable still within its own limits, but the pattern abnormal — flagged days before any threshold trips (detection). The pattern points to a specific failure mode (diagnosis), and the trend supports a Remaining-Useful-Life estimate (prognosis). But the design spends as much effort on the deployment reality: tuning the false-positive rate so operators don't get cried-wolf into ignoring it, and a plan to retrain as the plant's normal drifts — because that, not the algorithm, is what makes or breaks it in service.

7. Getting the best result

- **Use it where thresholds fail.** Multi-variable patterns are its strength; for a single clear limit, use Condition Triggers.
- **Design for false positives.** A system that cries wolf gets switched off — manage the false-positive rate deliberately.
- **Plan for data drift.** Normal changes over time; the model must be retrained or it degrades.
- **Win operator trust.** Adoption, not accuracy, is where most industrial ML quietly dies.