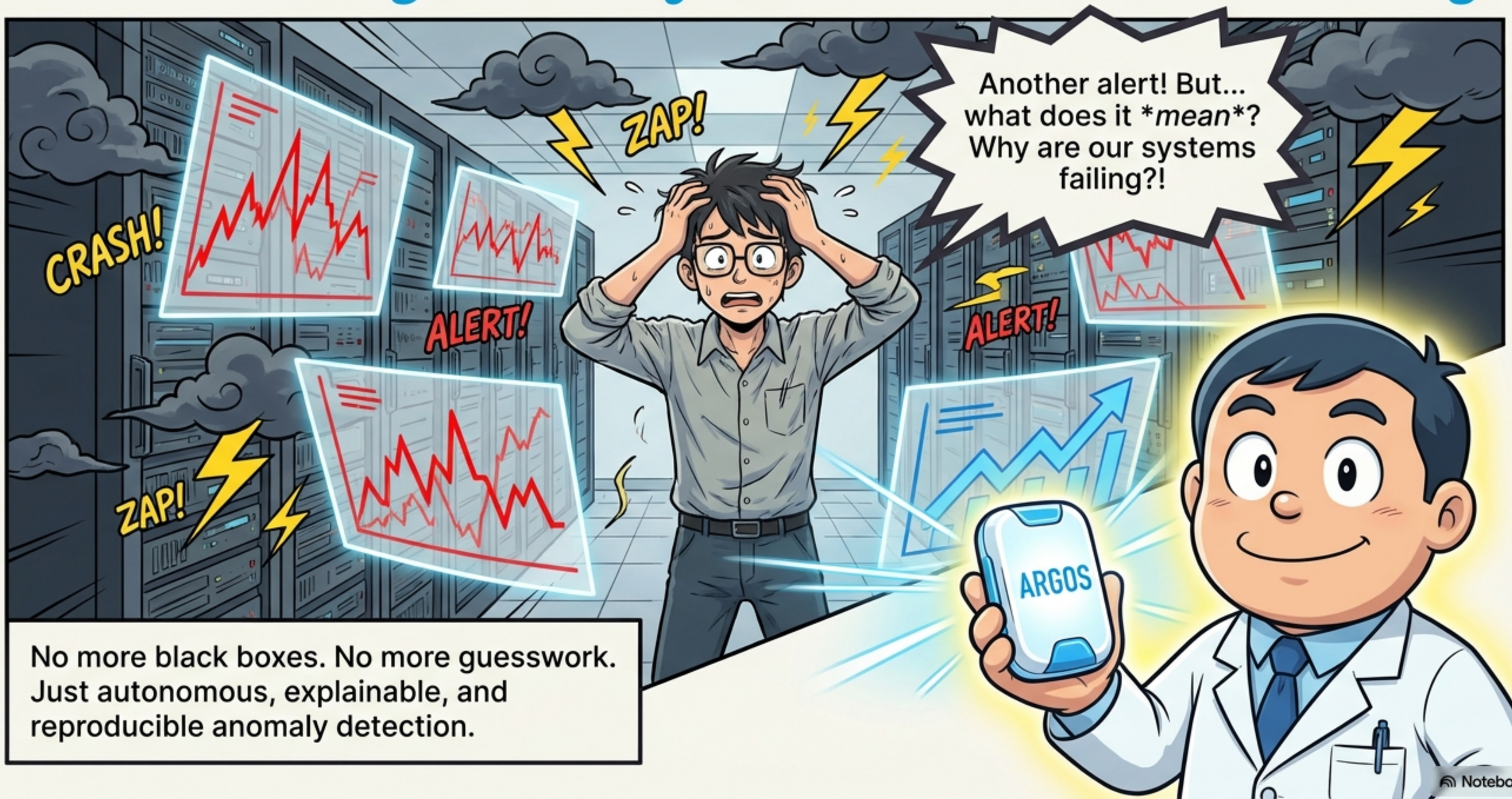
ARGOS: A Gadget for Crystal-Clear Cloud Monitoring!

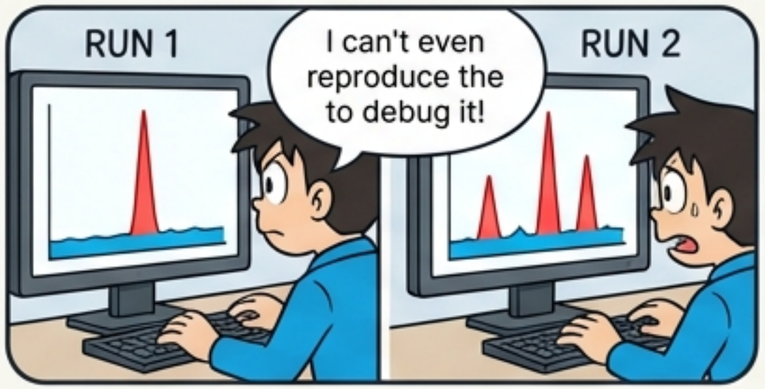


THE IMPOSSIBLE CHOICE IN ANOMALY DETECTION

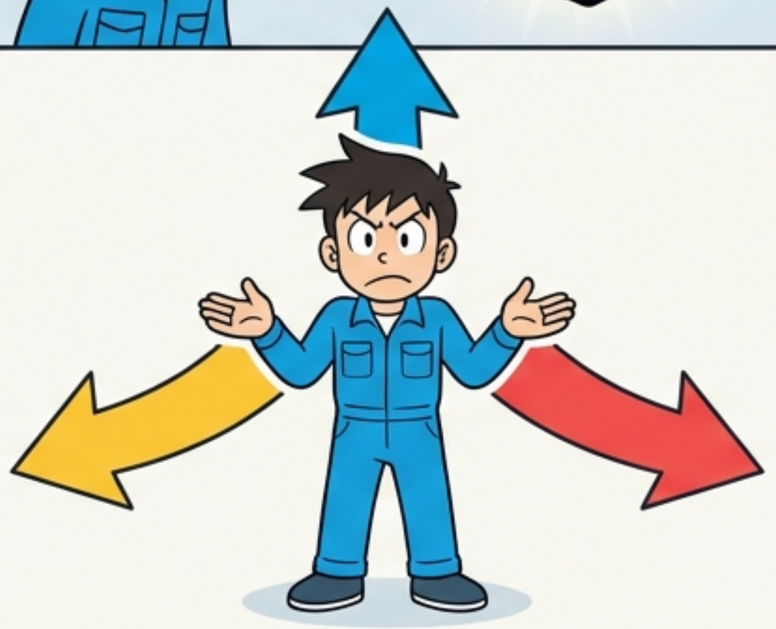
EXPLAINABILITY

When an alarm fires, OCEs must understand *why*. Without this, we can't trust the system or fix inaccuracies.

REPRODUCIBILITY

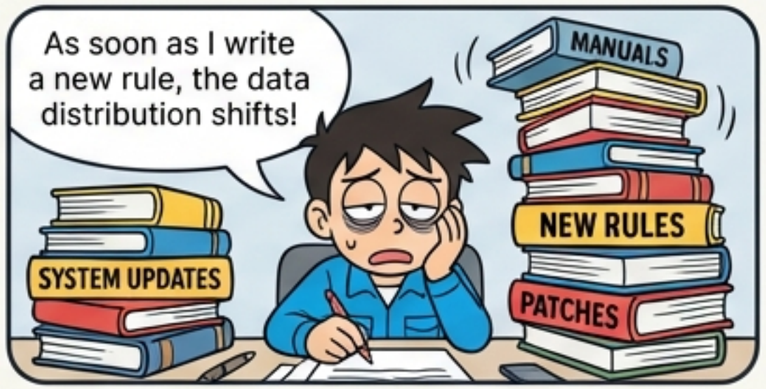


Results must be consistent. Non-deterministic alarms waste engineering effort on phantom issues.



Existing systems force a trade-off.
You can’t have all three.
Until now.

AUTONOMY



Cloud systems constantly change. The detection system must adapt without constant manual intervention.

Unboxing the ARGOS Gadget: A Step-by-Step Guide

1

The Problem We're Solving

A deep dive into why current tools—from Deep Learning to manual rules—fall short.

2

Introducing ARGOS

How ARGOS combines the best of all worlds by autonomously generating explainable rules.

3

How the Magic Works

A look inside the agentic pipeline, the feedback loop, and the model fusion safety net.

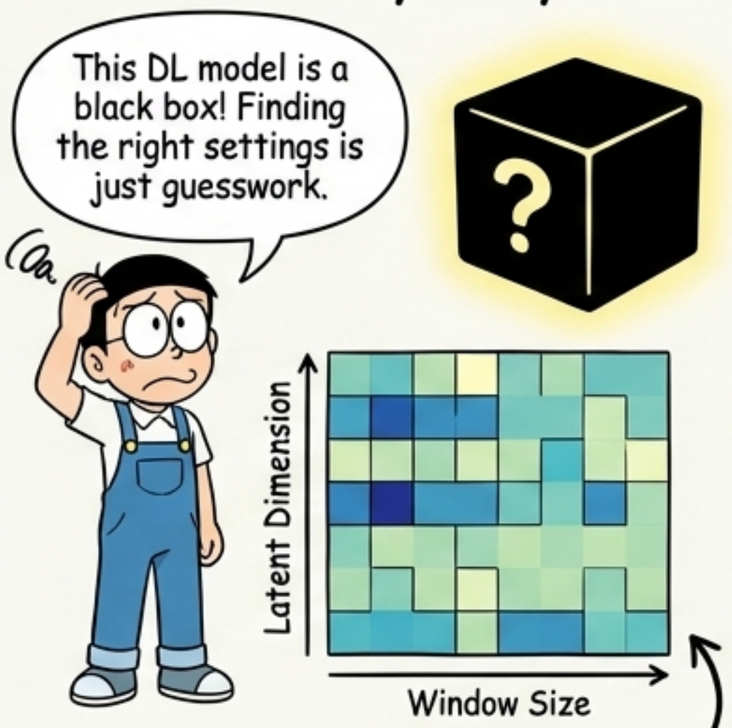
4

The Proof is in the Pudding

Head-to-head results, real-world case studies, and performance benchmarks.

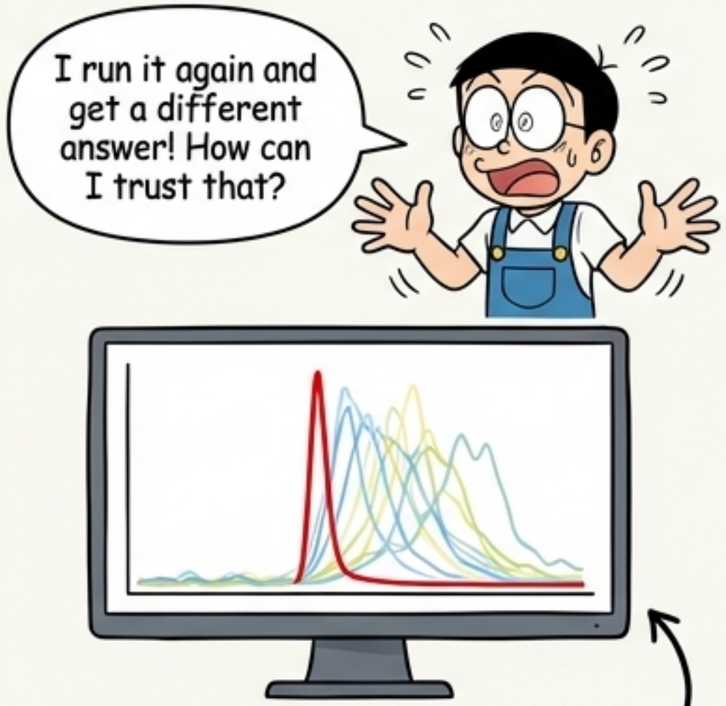
Nobi's Old Tools Just Don't Work!

Deep Learning Models are a Mystery



DL models lack explainability. Performance is highly sensitive to hyperparameters (like window size and latent dimension) with no clear intuition.

LLM-based Detectors are Unreliable



Using LLMs for runtime detection sacrifices reproducibility. The inherent non-determinism leads to inconsistent results.

Manual Rules are Brittle and Slow

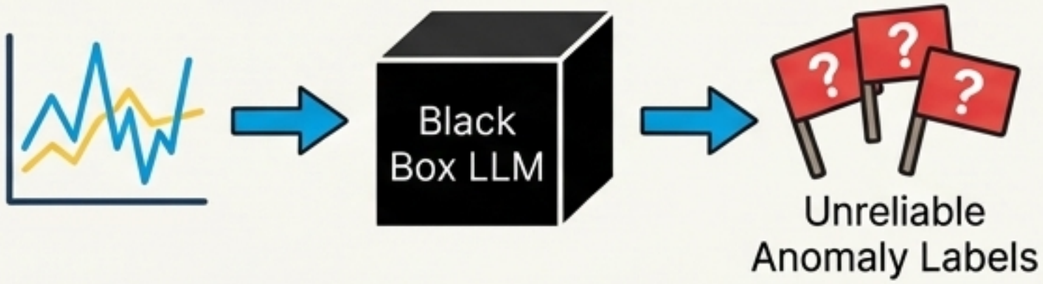
Rule-based methods lack autonomy. They require manual effort to create and update, often after an incident has already occurred.

Doraemon's Big Idea: Use the LLM to **Write the Rules!**

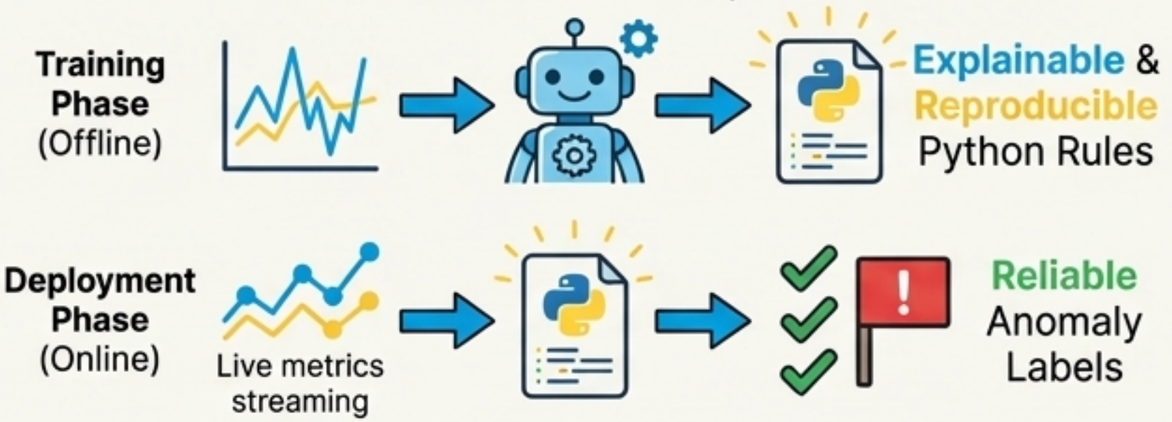
What if we don't use the LLM to ***find** the anomalies at runtime? What if we use its intelligence during a ***training phase*** to autonomously write perfect, human-readable detection rules?

The ARGOS Shift

Old Way

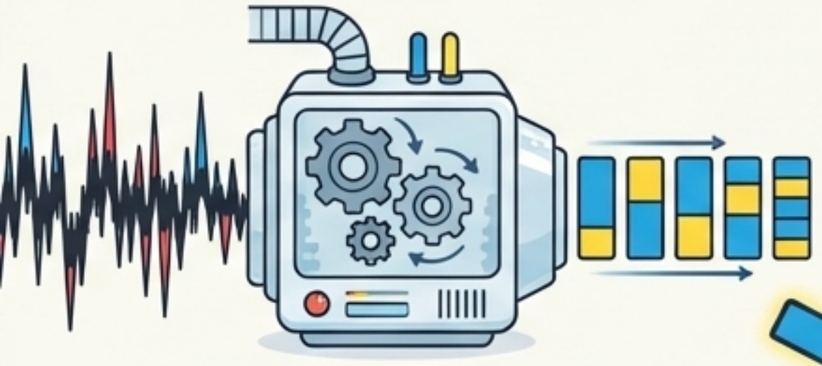


ARGOS Way



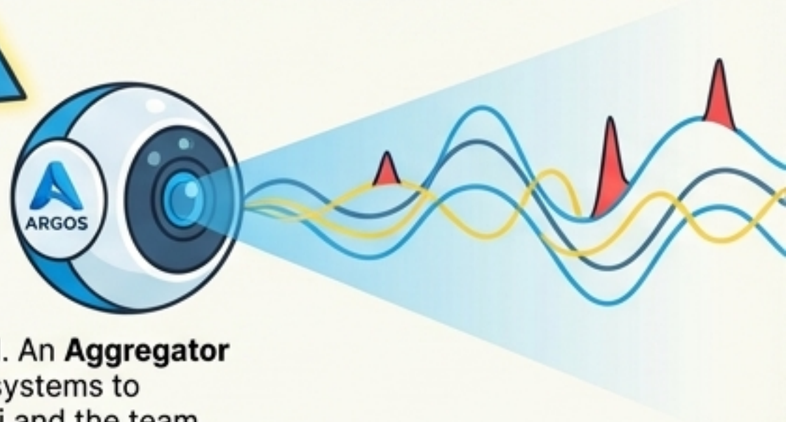
ARGOS bridges the gap. We get the **autonomy** of LLMs to generate rules, and the **explainability** & **reproducibility** of simple, executable code for detection.

A Look Inside the Gadget: The ARGOS Workflow



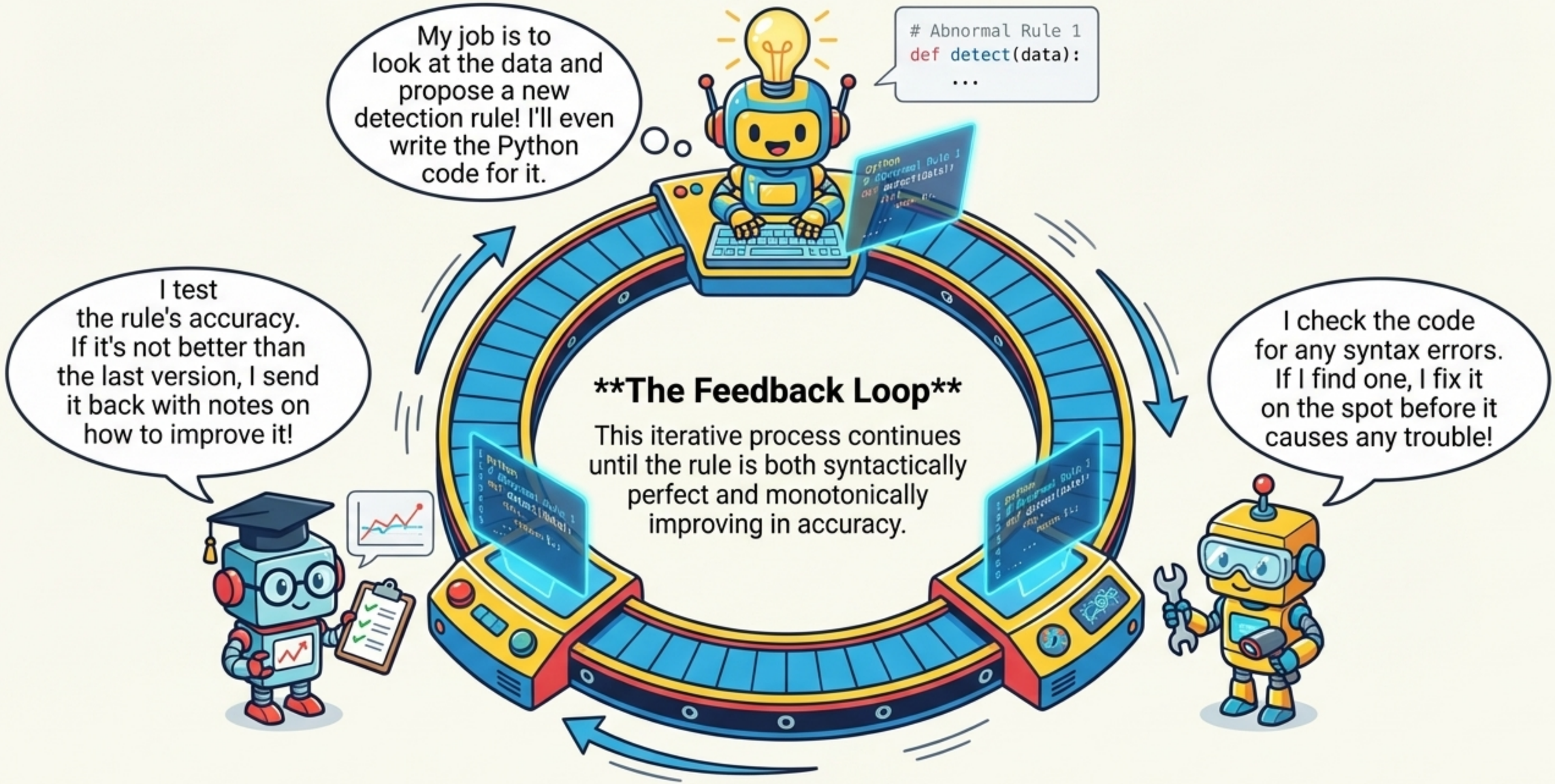
Raw time-series data is received. The **Data Preprocessor** chunks, scales, and indexes it, making it ready for the AI agents.

Preprocessed data enters the **Training Engine**. Here, our collaborative agents iteratively propose, fix, and refine anomaly detection rules.



The trained, error-free rules are deployed. An **Aggregator** combines their predictions with existing systems to catch anomalies in live data, alerting Nobi and the team.

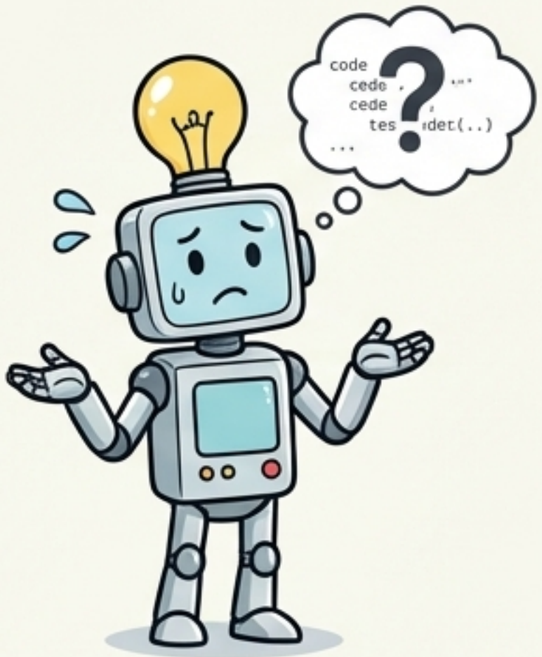
Meet the Rule-Making Team: A Trio of Collaborative Agents

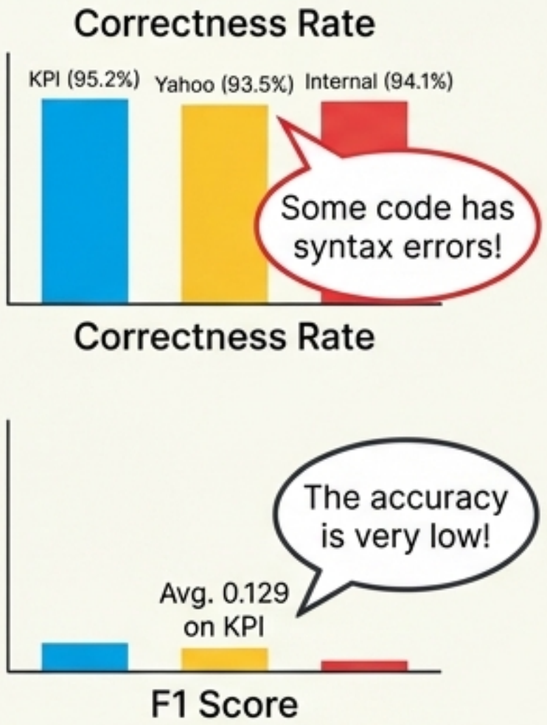


The Team Makes Perfect: How Feedback Loops Ensure Quality

Detection Agent Only

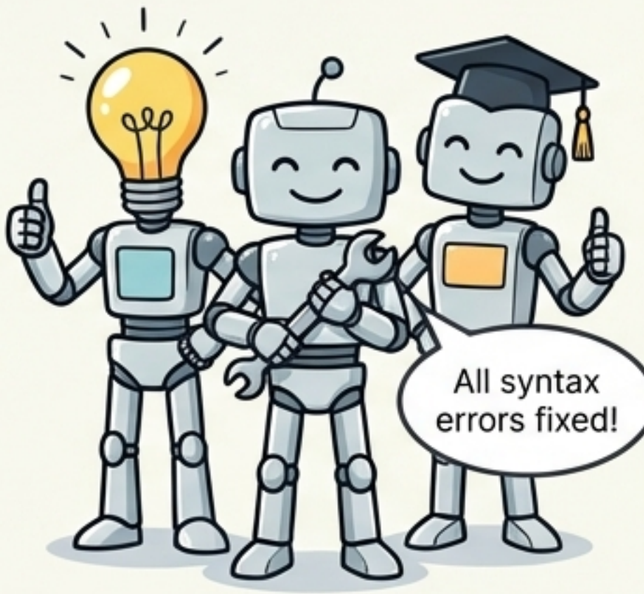
Good Idea, But Flawed

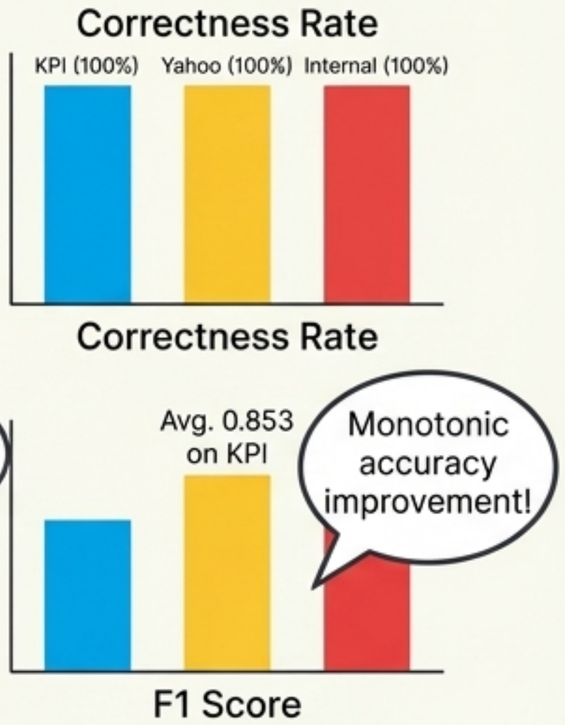




Full Agent Team

Flawless and Accurate

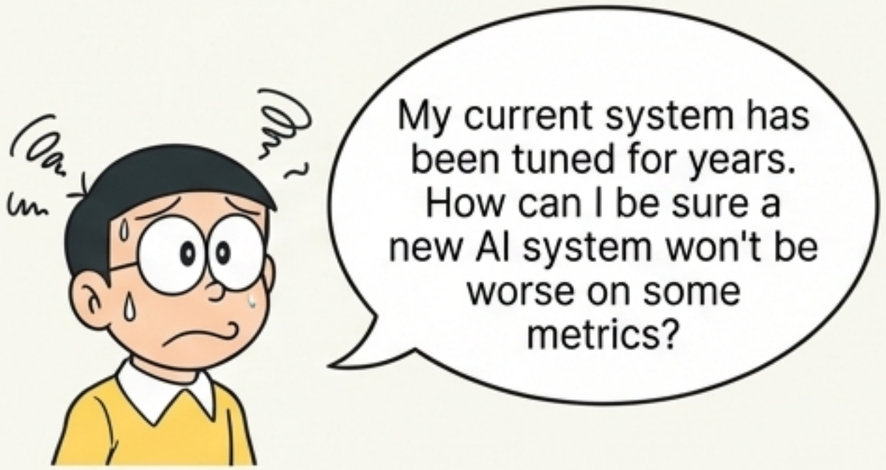




The full agentic pipeline with Repair and Review Agents achieves **100% code correctness** and **improves** the average F1 score by up to **11.3x** (on the Yahoo dataset).

A Safety Net for Accuracy: Improving, Never Regressing

The Challenge



The Solution: Model Fusion

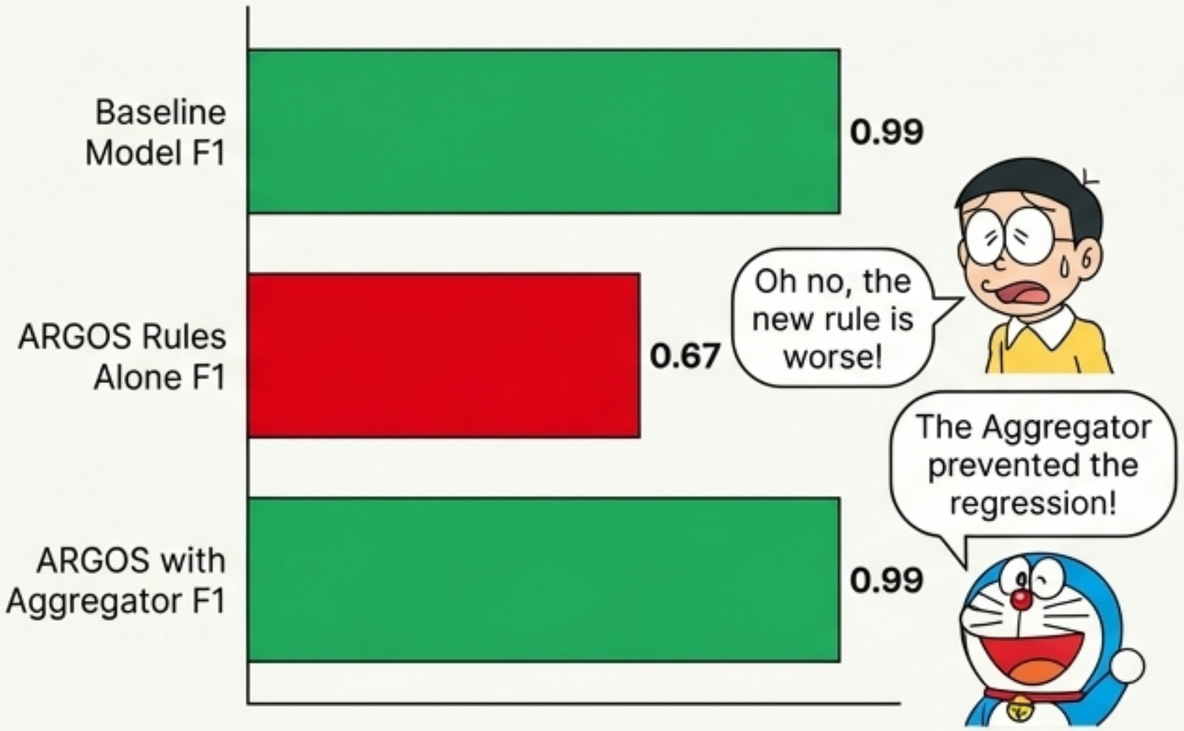
ARGOS doesn't replace your system; it supercharges it. It specifically trains rules to catch the anomalies your base detector **misses** (False Negatives) and to ignore the false alarms it **creates** (False Positives).

How It Works

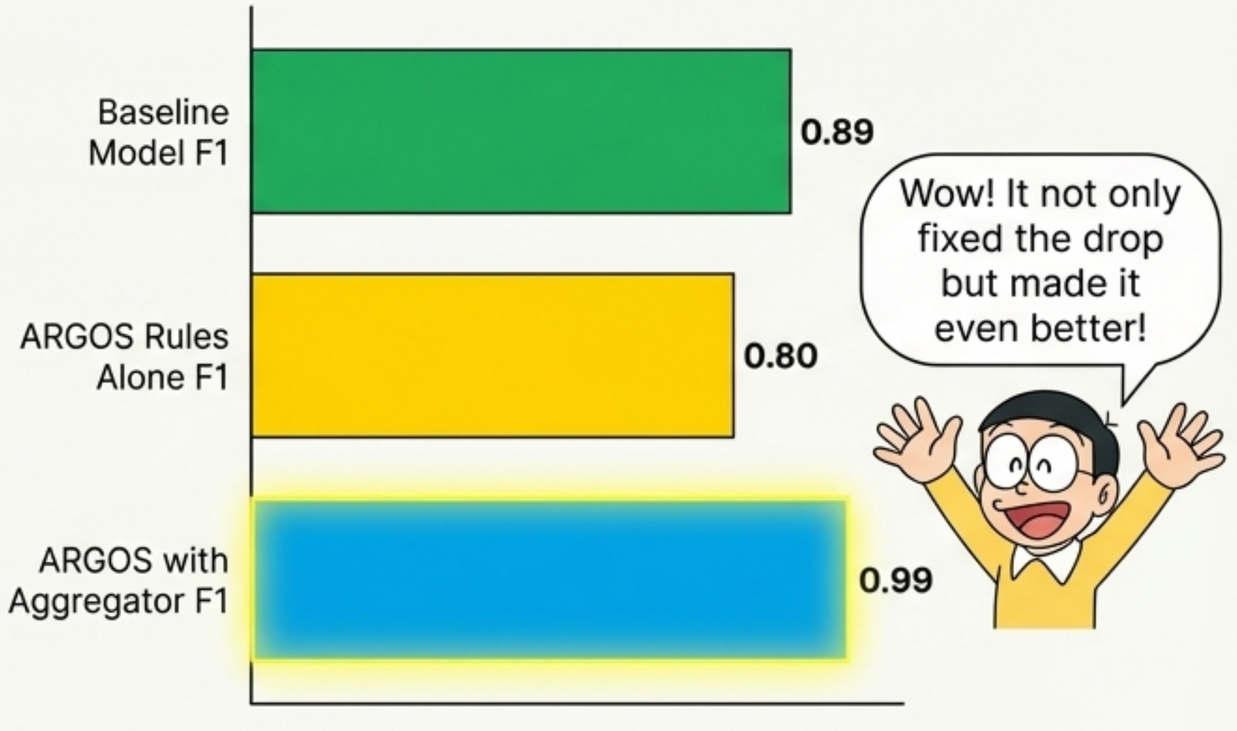
The final prediction is guaranteed to be at least as good as the original, and often much better.

Proof of the Safety Net

Case 1: Metric KPI-07927



Case 2: Metric 1c35d



Across all datasets, ARGOS with the Aggregator had **zero accuracy regressions** compared to the baseline, with improvements of up to **+0.10** on challenging metrics.

Working Faster and Smarter with Top-k Selection

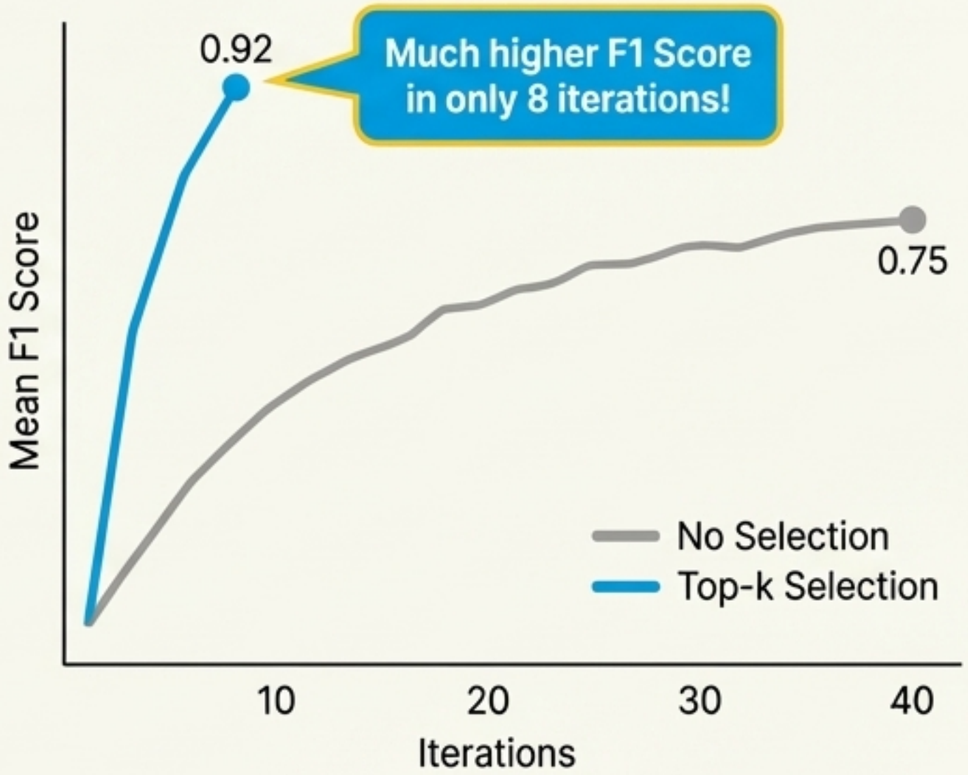
LLM generation is slow and variable. How can we find the best rule without endless trials?

A Rule-Making Brainstorm

Rule Selector

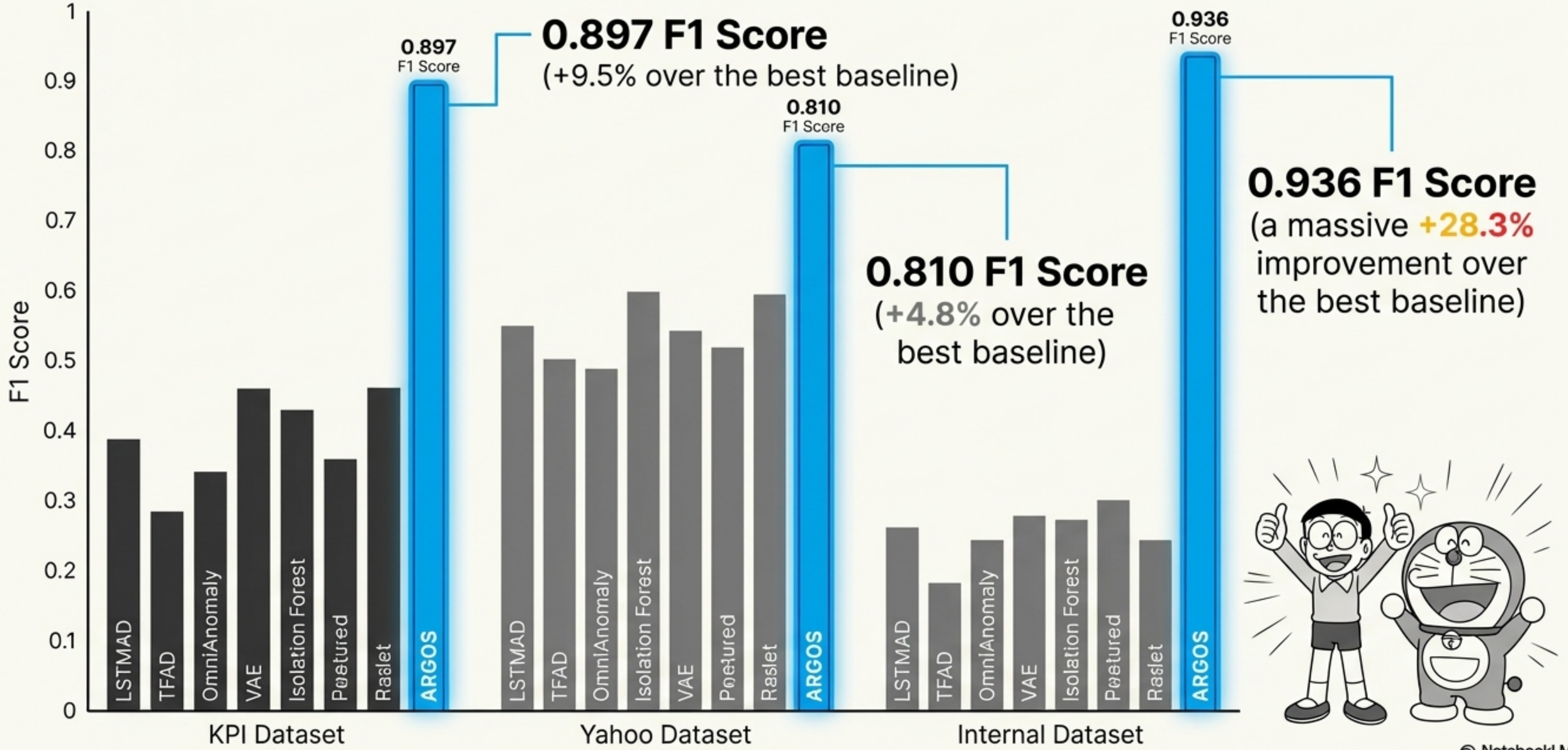
1. **Propose**: The Detection Agent generates `n` (e.g., 5) rule candidates at once.
2. **Validate**: The Repair and Review agents check all 5 rules.
3. **Select**: The **Rule Selector** picks the top-`k` (e.g., the single best) rule based on validation accuracy.
4. **Iterate**: The next "brainstorm" session starts with the best rule from the previous round.

Better Rules, Faster



Top-k selection not only accelerates training but also helps escape local optima to find more accurate final rules.

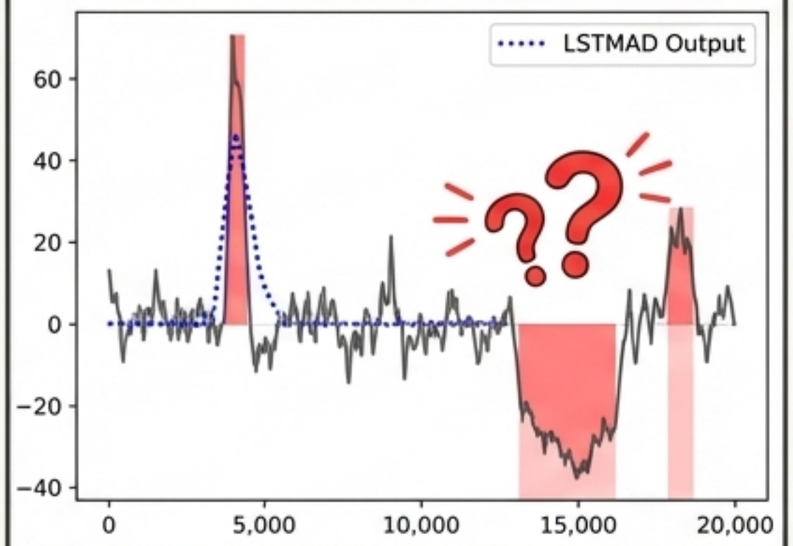
The Ultimate Test: ARGOS vs. The World

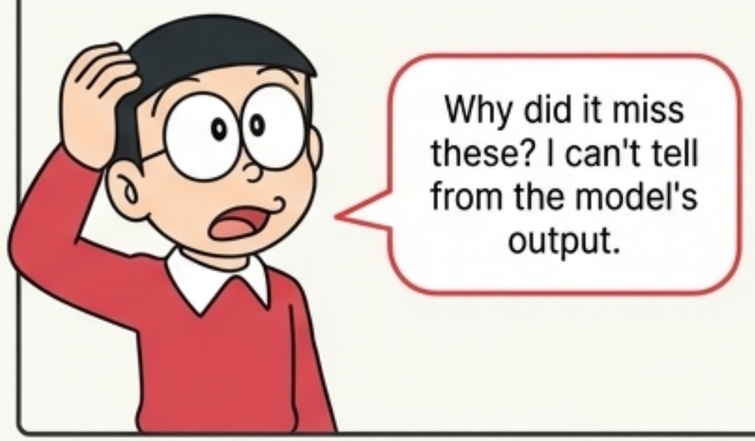


A Real-World Rescue Mission: Catching What Others Miss

On metric `e0770` from the KPI dataset, the best baseline model (LSTMAD) struggled, only catching 1 of 3 anomalies.

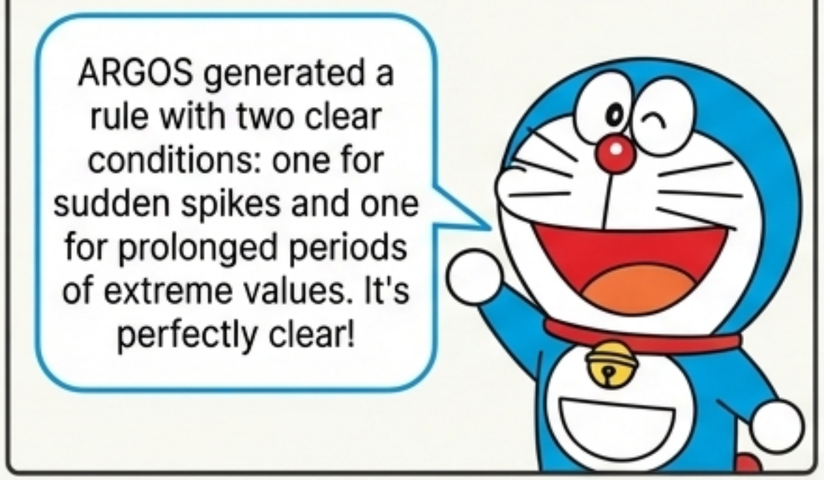
Panel 1: The Missed Anomaly



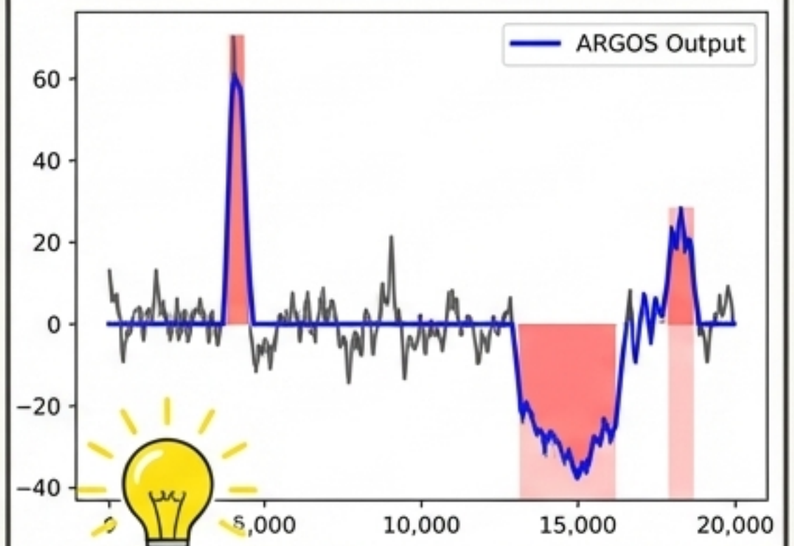


Panel 2: ARGOS's Explainable Rule

```
def is_anomaly(data_point, spikes_or_drops, extreme_values):
    # Rule generated by ARGOS
    if spikes_or_drops > 4000: # Sudden Spikes
        return True
    if np.all(extreme_values[i:i+10]): # Prolonged Extremes
        return True
    return False
```



Panel 3: The Successful Detection

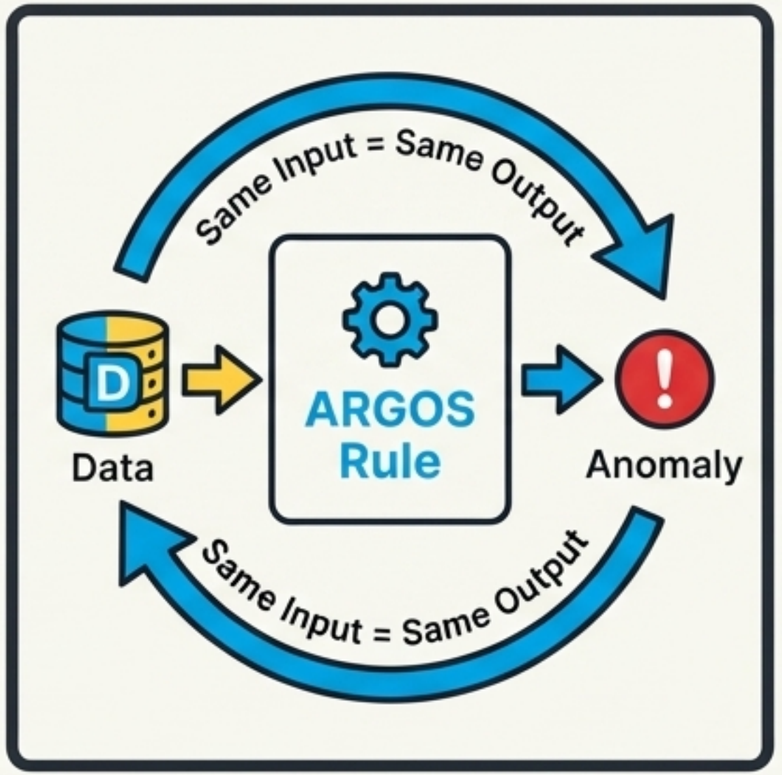


The ARGOS Promise: Delivered.

Explainable

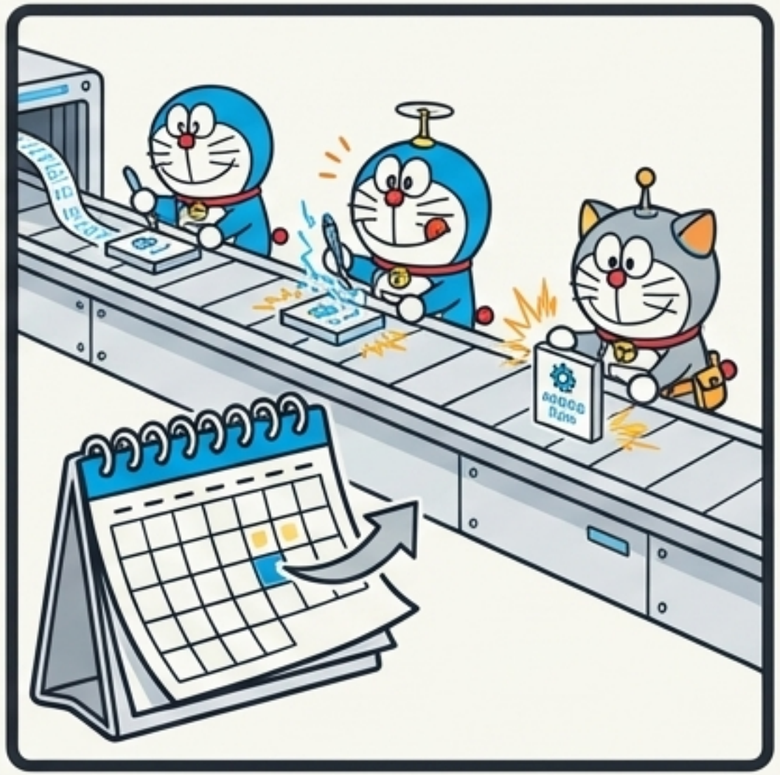
"I can read the rule. I understand the logic. I can trust the alert. No more black boxes."

Reproducible



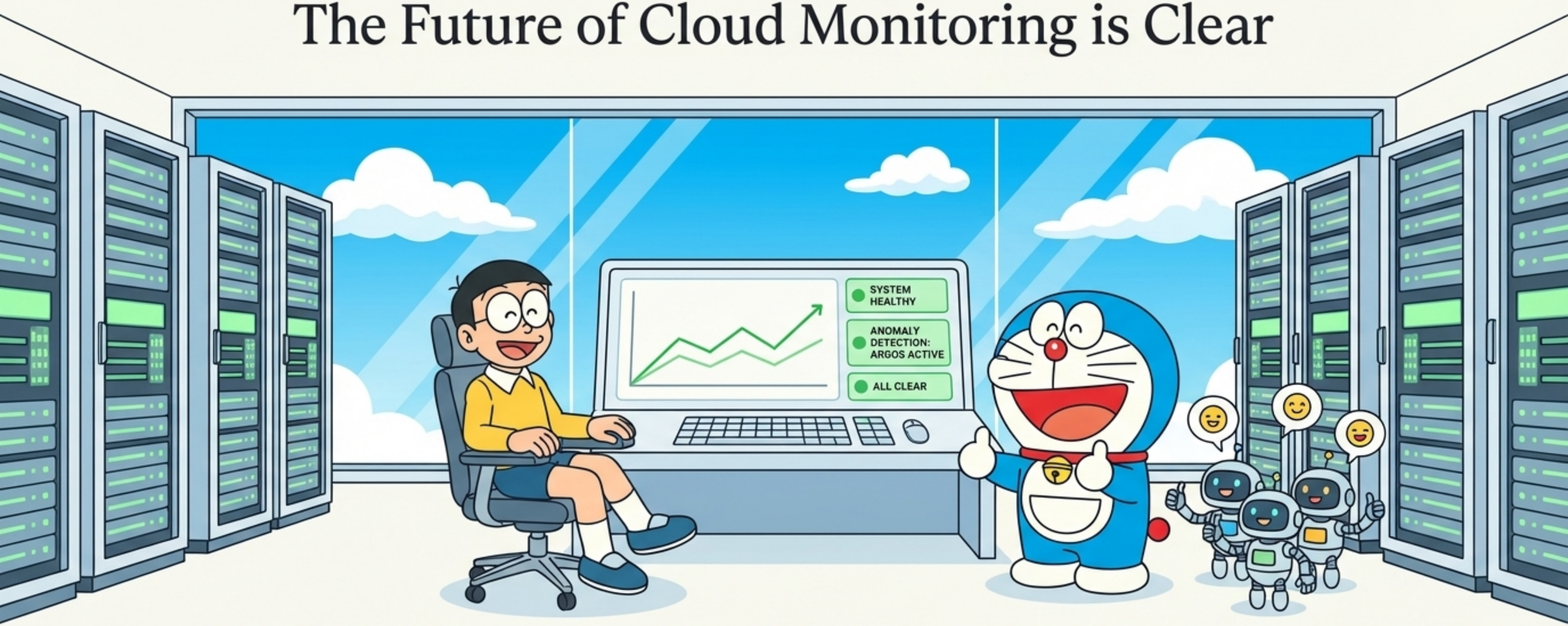
"The rules are deterministic code. The same input will always produce the same output. Every time."

Autonomous



"The agentic pipeline continuously trains and improves rules, adapting to new data without manual intervention."

The Future of Cloud Monitoring is Clear



With ARGOS, anomaly detection is no longer about choosing between trade-offs. It’s a new era of clear, reliable, and intelligent observability for the systems that power our world.