

## Part 4 – Optimization & Benchmarking

To evaluate the improved detection logic, we benchmarked Version 1 (V1) and Version 2 (V2) on the same sample dataset.

### Dataset description:

The sample dataset contains 60 consecutive data points (one hour of readings). Most of the data is flat and stable, with two sustained rises in temperature and smoke:

- First rise between minutes 10 - 20
- Second rise between minutes 40 - 50

In addition, the dataset includes two sharp, isolated spikes (around minutes 33 and 37) that represent unrealistic single-point sensor noise.

### Behavior comparison:

- **V1 (original algorithm)** relies on smoothed signals and **variance-aware statistical anomaly scoring with dynamic damping**, which reduces sensitivity in low-variance environments and often mitigates isolated noise. However, in this dataset, two sharp single-point spikes were still strong enough to influence the statistics and trigger alerts. As a result, V1 produced alerts during both sustained rises and also reacted to these spikes, leading to alerts across three episodes instead of the expected two.
- **V2 (improved algorithm)** introduces **single-point spike suppression** during preprocessing and **alert hysteresis** to emit only one alert per sustained high-risk incident. This allows V2 to ignore spike-induced anomalies and generate exactly one alert for each true sustained rise, matching the underlying behavior of the data.

### Benchmark results:

V1 runtime: 0.003484 seconds      V1 events: 16

V2 runtime: 0.003652 seconds      V2 events: 2

### Quality metric:

The primary quality metric is the number of detected events. V2 reduces alert count from 16 to 2, significantly lowering false positives and alert noise while preserving detection of genuine incidents.

### Performance impact:

The runtime difference between V1 and V2 is negligible, demonstrating that the improvements reduce false positives without affecting performance.

### Running the benchmark test:

From the project root, run: `python benchmarks/benchmark_detection.py`