

Isolation Forest for sensor anomalies: how it works and where it misleads

Read flagged rows in a sensor table by their isolation score, understand what contamination does in the model, and see where the method misleads you.

In short

- Isolation Forest separates rare, unusual points with fewer random splits, and it needs no distance calculation.
- A score near 1 means a point was isolated quickly, while scores around 0.5 show no clear separation from normal data.
- Contamination is an assumed share that places the cut-off; it is not a measured failure rate of your equipment.

01 / Why anomalies are easy to isolate

Isolation Forest is an ensemble method for anomaly detection. It starts from a simple observation: anomalies are rare and differ from normal points. Each isolation tree splits the data by choosing a random sensor and a random cut value within that sensor's range. Splitting repeats until a point is alone in its own region. The number of splits needed is the point's path length.

A normal point sits in a dense region and needs many splits to be separated. An unusual point is separated after only a few. The method computes no distances or densities. Because each split uses the range of a single variable, differences in units do not change where the splits fall, so scaling is usually unnecessary. This simplicity is not a guarantee of accuracy. The method ranks how easily each point was isolated, but it does not explain why.

02 / Path length and the score

The model builds many trees, each trained on a small random subsample; 256 points is a common default that should be checked on your own data. For a point x , the average path length over all trees is $E[h(x)]$. The anomaly score is:

$$s(x) = 2^{-(E[h(x)] / c(n))}$$

Here $c(n)$ is the average path length of an unsuccessful search in a binary search tree built on n points. The score lies between 0 and 1: scores near 1 mean very quick isolation, and scores near 0.5 mean no clear separation.

Hypothetical check: if a point's average path length equals $c(n)$, the score is $2^{-1} = 0.5$. If it is half of $c(n)$, the score is $2^{-0.5} \approx 0.71$, so the point was separated clearly faster than an average point.

03 / Worked example: a pump after a seal change

The numbers below are hypothetical. Picture a pump table with five sensors: vibration, bearing temperature, flow, pressure and motor current. The model was trained on 4 000 normal rows. After a seal change, 6 of the next 200 rows are flagged. The flagged share is $6 / 200 = 0.03$, or 3 %. If contamination was set to 0.01 for training, the cut-off sits so that about 1 % of the training rows, roughly $4\,000 \times 0.01 = 40$ rows, lie beyond it by construction. A 3 % rate on new data is therefore three times the design share.

Operators reviewed the six flagged rows. Four were judged to be a real change in pump behaviour after the seal work. Two coincided with start-up transients. The model appears to have caught a genuine change, but not every flag is a fault. Per-sensor explanations for the four real rows mostly pointed to flow and motor current.

04 / Trees, subsamples and contamination

Three settings shape the result. More trees make the averaged scores more stable, at the cost of more computation. The subsample size keeps each tree small and varied; 256 is a common starting point, not a universal rule.

Contamination sets where the cut-off falls. It should be read as a design choice, the share of the training data the team is willing to flag, and not as a measured failure rate for the plant. Choose it together with the number of alerts per shift that the maintenance team can realistically handle. Record how the number of flagged rows changes as you move the threshold. A single threshold value hides how sensitive the result is.

05 / Where the method misleads

Isolation Forest treats rows as independent. It has no notion of time order or seasonality, so a slow drift that looks normal at each reading can go unnoticed for weeks. Points that are normal in every variable on its own but unusual in combination are also hard to catch reliably. For those cases, multivariate monitoring with Hotelling T^2 and SPE is more direct, and CUSUM or EWMA suits slow shifts.

Regime changes are a common problem. A new product recipe or a different speed changes what normal means, so the model has to be refitted. If the training window already contains a fault, the model may learn that fault as normal behaviour.

06 / How to start on your plant

Start with a training window of known normal operation: stable conditions, no maintenance intervention and no recorded fault. Fit the model on that window, then evaluate it on a later period that was not used for fitting. Review every flagged row with the operators who know the machine, and record the reason for each decision.

Count false alarms per week separately. A model with good offline numbers that produces a flood of alerts will not survive daily use. If the weekly count exceeds what the team can handle, revisit the threshold or the sensor set. Finally, check whether each alert can be explained in terms of named sensors. An alert nobody can explain loses trust quickly.

Try it

In Sentinel's sample sensor table, open one flagged row, read its per-sensor explanations, and write one sentence naming the two sensors that contribute most.

References and online guide

Open the online guide

scikit-learn: IsolationForest

Isolation forest (Wikipedia)

Isolation Forest (Liu, Ting, Zhou, ICDM 2008)

scikit-learn: novelty and outlier detection

Proposed pilot steps do not claim field accuracy. Synthetic examples are not measurements from a production site.