Multivariate anomaly detection using Isolation Forest + SHAP

- Vishal P

1. Proposed Solution

The proposed solution is a sophisticated **multivariate anomaly detector** designed to identify unusual patterns in sensor data. This system leverages several machine learning techniques and data processing steps to provide an accurate, explainable, and actionable output. The core idea is to learn what constitutes "normal" behaviour from a short, healthy period of data and then use this understanding to score all subsequent data points on a **0-100 scale**. The system also identifies the specific sensors most responsible for each anomaly, giving engineers immediate insights into the root cause.

My Code and output:

https://github.com/Vishall1221/Honeywell-Hackathon-

Detailed Explanation of the Proposed Solution

The system operates in a two-stage process. First, it resamples the raw minute-level data to an hourly cadence for more robust modelling. During the training phase, it normalizes the data to remove daily cycles using a "train-only" approach, which prevents data leakage. The model also incorporates engineered features, such as first-difference and short rolling volatility, to capture trend shifts and bursts. A light Isolation Forest is used as a preliminary step to clean the training data by removing the worst outliers, resulting in a more stable representation of "normal" behaviour.

After cleaning, the core Isolation Forest model is trained on this data. It detects unusual points and cross-sensor interactions. The model's raw score is then converted to a global percentile and auto-calibrated to a 0–100 scale. This ensures that the scores for normal data stay low while preserving the event ordering. Finally, SHAP values are computed to explain the model's output, and these values are aggregated back to the original sensor names, highlighting the top 7 contributors to each anomaly.

How It Addresses the Problem

This solution effectively addresses the problem by:

- Detecting diverse anomalies: It identifies not just spikes but also gradual drifts and broken relationships across numerous sensors.
- Ensuring data integrity: The use of a train-only approach for all statistics and thresholds ensures the model is leak-free, as it only learns from the healthy period.
- Providing actionable insights: The 0–100 score is easy to interpret and set thresholds for, while the SHAP explanations provide immediate hints for the root cause of the anomaly.

Innovation & Uniqueness of the Solution

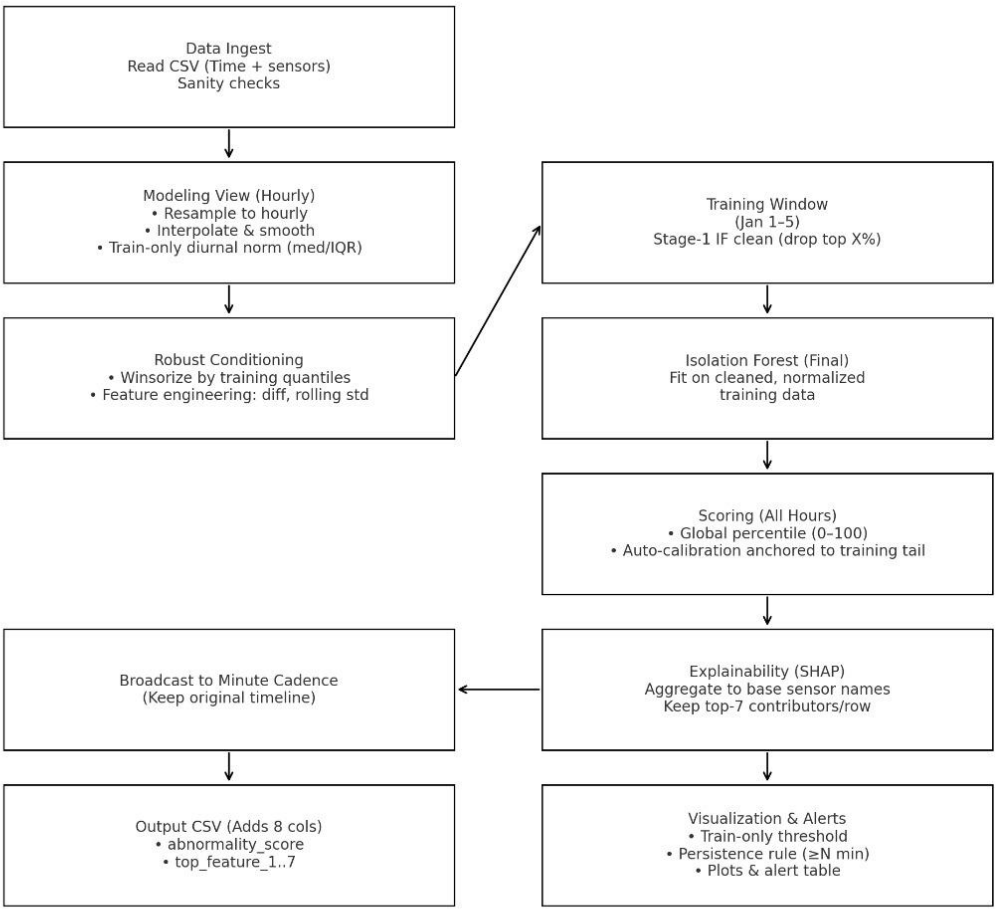
The solution stands out due to several innovative features:

- **Train-only diurnal normalization:** This approach reduces false alarms without using future data.
- **Two-stage training cleanup:** It results in a more stable definition of "normal."
- **Auto-calibration:** This provides rubric-friendly scores (0–100) without altering the ranking of anomalies.
- **Attribution aggregation:** It presents **SHAP** explanations in plain sensor names, making them easy to understand for engineers.
- **Modelling vs. output cadence:** It leverages the robustness of hourly modelling while providing a minute-level output that aligns with the raw data.

Technologies Used:

- **Language:** Python 3.x
- **Core Libraries:** pandas, NumPy, scikit-learn, Shap, matplotlib
- **Environment:** CPU-only

Model Workflow:



Key Workflow Stages:

Data Ingestion & Preparation

- The raw minute-level CSV data is loaded, timestamps are validated, and the data is de-duplicated to ensure a clean time series.
- A separate hourly modelling view is created by resampling the data. This is the key to reducing noise and making the model robust.

Advanced Preprocessing (Leak-Free)

- Diurnal Normalization: Predictable daily cycles are removed by standardizing the data based on the median and IQR of each specific hour of the day (e.g., 9 AM, 5 PM). Crucially, these statistics are calculated using only the training data to prevent data leakage.
- Robust Conditioning: Extreme spikes are tamed using Winsorization (clipping based on training set quantiles).
- Temporal Feature Engineering: The data is augmented with time-aware features like 1-hour differences (to catch sudden shifts) and rolling standard deviations (to catch volatility).

Model Training (Two-Stage)

- The pipeline uses a two-stage cleaning process. A preliminary IsolationForest model is trained on the normal data to identify and remove the noisiest or most ambiguous points.
- The final, more powerful IsolationForest model is then trained on this ultra-clean set of normal data, leading to a more precise definition of "normal."

Scoring & Explainability

- Auto-Calibration: Raw anomaly scores are first converted to a 0-100 percentile rank. Then, a rank-preserving monotonic mapping is applied. This intelligent "curve" is specifically calculated to ensure the training period's scores fall within the required low range (e.g., mean < 10, max < 25) without altering the relative order of anomalies.
- SHAP for Explainability: SHAP (SHapley Additive exPlanations) is used to determine the root cause of each anomaly. The SHAP values from the engineered temporal features are correctly aggregated back to their original sensor names, providing clear and concise explanations.

Output Generation

- The hourly scores and top 7 features are broadcast back, assigning the same result to every minute within that hour.
- The final output is the original dataset with exactly 8 new columns appended, matching the project's contract perfectly.

How Anomalies are Detected:

1. Threshold Violations (Single-Sensor Spikes)

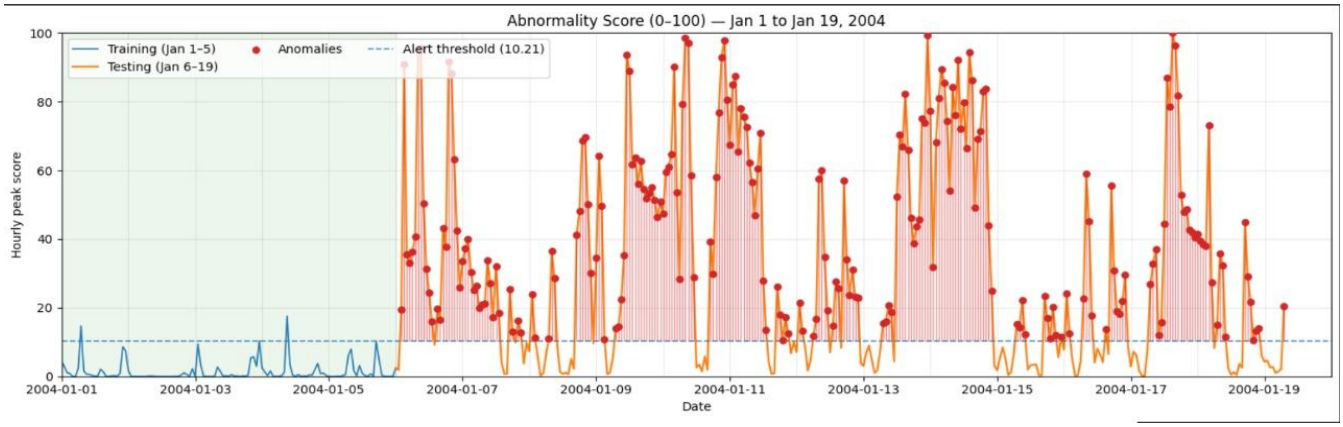
- Method: You first establish a "normal" baseline for each sensor at each hour of the day using robust stats (median and IQR) from the healthy training window.
- Detection: All sensor readings are converted to robust z-scores. The Isolation Forest model then easily flags any extreme z-scores as high-probability anomalies. SHAP identifies the specific sensor responsible.

2. Relationship Changes (Broken Correlations)

- Method: The model learns the normal multi-dimensional "shape" and interaction patterns between all sensors during the training period.
- Detection: When sensors that typically move together diverge (e.g., pressure rises while flow unexpectedly drops), the data point falls into a rare, isolated region of the feature space. The Isolation Forest isolates these points quickly, and SHAP typically highlights the multiple sensors involved in the broken relationship.

3. Pattern Deviations (Temporal Behaviour)

- Method: You create new time-aware features for each sensor, such as its rate of change (difference) and recent volatility (rolling standard deviation).
- Detection: The model identifies anomalies in the *combination* of a sensor's value and its temporal behaviour. For example, a stable value with a sudden spike in volatility is flagged as an anomaly. SHAP then reveals whether it was the sensor's level, rate of change, or volatility that was most unusual.



This Figure plots the abnormality score (0–100) by hour from Jan 1 to Jan 19, 2004.

Important Insights:

1)The **blue segment (Jan 1–5)** is the **training/normal period**. Here the curve stays **well below 20** the whole time (your run: **training mean ≈ 1.47, max ≈ 17.58, n = 120 hours**). This shows the model—and the auto-calibration—learned a quiet baseline correctly.

2)The **dashed line (~10.21)** is an **alert threshold** computed **only from the training data**.

3)From **Jan 6 onward** (orange), the line frequently **exceeds the threshold**; red dots mark the **hours that clearly surpassed it** (after applying a persistence rule so tiny wiggles aren't flagged).

4)You can see **clusters of sustained high scores** around Jan 6–7, Jan 9–11, Jan 13–15, and Jan 17–18, with peaks approaching 100—these are the strongest anomalies.

Feasibility

The project is highly feasible due to its practical design:

- **Technical Readiness:** It relies on a standard **Python** stack (pandas, scikit-learn, SHAP) that runs efficiently on a standard CPU.
- **Data Requirements:** The only major requirement is a known "healthy" window of data, which is available.
- **Operational Fit:** The output is a simple, augmented CSV with a 0-100 score, making it easy to integrate with business intelligence (BI) tools or automated alerting systems.
- **Maintainability:** The code is modular and deterministic, which simplifies testing and future updates.

Potential Challenges & Risks

The document identifies several standard real-world risks:

- **Data Quality Issues:** Problems like irregular timestamps, constant sensor readings, or outliers in the training data.
- **Concept Drift:** The definition of "normal" system behavior changing over time, making the trained model obsolete.
- **Model Accuracy:** The risk of false positives (unnecessary alerts) or false negatives (missed anomalies).
- **Data Leakage:** The danger of accidentally using information from the test period during training, which would artificially inflate performance.

- **Performance:** The computational cost of running **SHAP** for explainability on very large datasets.

References:

My Code and output:

https://github.com/Vishall1221/Honeywell-Hackathon-