IOT Sensor Anomaly Detection – Project Summary Report

Problem Understanding and Approach

Modern manufacturing facilities use IoT sensors to continuously monitor the condition of machines and equipment. These sensors generate large amounts of time series data representing physical parameters such as temperature, vibration, or pressure.

The client's goal is to identify abnormal sensor readings that could signal potential equipment failure or maintenance requirements before costly breakdowns occur.

To address this, we developed an end-to-end anomaly detection system capable of learning from sensor data and flagging unusual behavior. The chosen dataset (art_daily_jumpsup.csv) from the Numenta Anomaly Benchmark (NAB) simulates real sensor data with sudden jumps that represent anomalies.

The overall workflow:

Data Preparation & Exploration – Load, clean, and explore the time series data.

Feature Engineering – Create meaningful time-dependent features to enhance anomaly detection.

Modeling – Implement and compare two different anomaly detection approaches:

- Isolation Forest (unsupervised, statistical)
- Autoencoder Neural Network (deep learning)

Evaluation & Insights – Compare models, visualize anomalies, and extract business insights.

Feature Engineering Rationale

Raw time series data (timestamp, value) was enriched with derived features to capture temporal patterns and behavioral trends of the sensor:

Feature	Description	Purpose
rolling_mean	Mean of last 50 readings	Captures short-term trend
rolling_std	Standard deviation over 50 readings	Detects volatility or irregular fluctuations
roll_min, roll_max	Minimum & maximum in window	Measures local range of operation
diff_1, diff_2	First and second differences	Capture rate of change and acceleration

All features were normalized using StandardScaler to bring values into comparable ranges, improving model performance and convergence.

Model Selection and Comparison

Two distinct modeling approaches were implemented to detect anomalies:

Approach 1 – Isolation Forest

- A tree-based unsupervised model that isolates anomalies based on their statistical rarity.
- Works by recursively partitioning data; points that require fewer splits to isolate are likely anomalies.
- Advantages: Fast, interpretable, and does not need labeled data.
- Key Hyperparameters: n_estimators = 100 , contamination = 0.03

Results:

- Detected 120 anomalies in total.
- Captured both sharp spikes and moderate deviations, though with some false positives.

Approach 2 – Autoencoder (Deep Learning)

A neural network trained to reconstruct normal sensor behavior.

It compresses input data (encoding) and then reconstructs it (decoding).

Anomalies are identified when reconstruction error is high (model fails to predict normal pattern).

Results:

- Detected 40 anomalies in total.
- Produced fewer but more confident anomaly points, minimizing false alarms.

Model Comparison Summary

Criterion	Isolation Forest	Autoencoder
Algorithm Type	Unsupervised Statistical	Deep Learning (Neural Network)
Total Anomalies	120	40
Sensitivity	High	Moderate
Precision	Medium	High
Interpretability	Easy	Moderate
Computational Cost	Low	Moderate
Best Use Case	Real-time monitoring	High-precision anomaly verification

Key Findings and Business Insights

- The dataset exhibited stable sensor behavior with rare but sharp “jump” anomalies — typical of potential sensor faults or equipment malfunctions.
- Outliers were successfully detected without labeled data using unsupervised and deep learning models.
- Visual inspection confirmed that detected anomalies align with spikes in the time series, validating the models’ behavior.
- The system can be deployed to monitor IoT data streams in real-time, automatically flagging anomalies for operator review.

Limitations and Future Improvements

Current Limitations:

- The dataset used is small and synthetic; real-world data would have more noise and complex patterns.
- No labeled ground truth for anomalies — evaluation relied on visual validation and domain reasoning.
- The Autoencoder model used a basic architecture; performance can be improved with tuning.

Future Improvements:

- Apply the approach to real sensor datasets with labeled events.
- Experiment with LSTM Autoencoders to capture temporal dependencies more effectively.
- Incorporate online learning or streaming data pipelines for real-time detection.
- Introduce ensemble anomaly detection systems that combine multiple models’ strengths.
- Develop interactive dashboards for anomaly visualization and alert monitoring.