Smart Factory Anomaly Detection

AI-Powered Industrial Sensor Analysis Pipeline

Assignment Report & Results

By Ng Yee Chian

Made with GAMMA

Project Overview

Objective

Build and evaluate a comprehensive anomaly detection pipeline for Smart Factory sensor data to identify equipment failures before they occur

Methodology

Leverage AI-powered tools for rapid development, from dataset generation to model deployment and documentation

Deliverables

- Production-ready dataset
- Isolation Forest model
- Anomaly detection agent
- Complete documentation

Made with GAMMA

AI Development Toolkit

AI Assistant (ChatGPT)

Strategic guidance for workflow design, prompt engineering, and technical research. Provided insights into anomaly detection algorithms and Isolation Forest methodology.

Code Generator (ChatGPT)

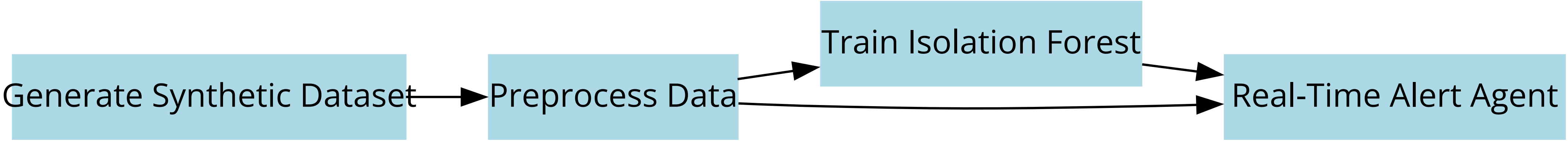
Automated creation of Python scripts for dataset generation, preprocessing pipelines, model training, and evaluation frameworks. Significantly reduced development time.

Gamma Platform

Professional presentation generation with sophisticated layouts and visual storytelling capabilities for technical reporting and stakeholder communication.

Made with GAMMA

Pipeline Architecture



Made with GAMMA

Model Configuration

Algorithm: Isolation Forest

Unsupervised learning approach using sklearn.ensemble.IsolationForest for robust anomaly detection in high-dimensional sensor data.

Key Features: Temperature, pressure, vibration measurements from industrial equipment sensors.

Parameter	Value
n_estimators	300
max_samples	auto
contamination	0.15
random_state	42
n_jobs	-1

Made with GAMMA

Performance Metrics

0.85

Overall Accuracy

0.89

Average Precision

0.79

Average Recall

0.91

ROC_AUC

0.95

Abnormal Precision

High confidence in anomaly detection

0.59

Abnormal Recall

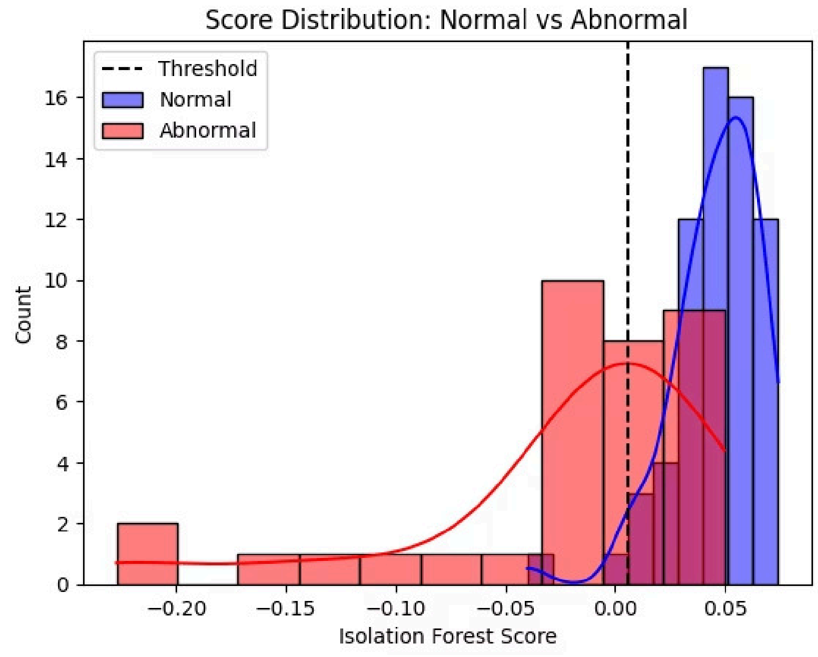
Target for further optimization

Made with GAMMA

Key Findings

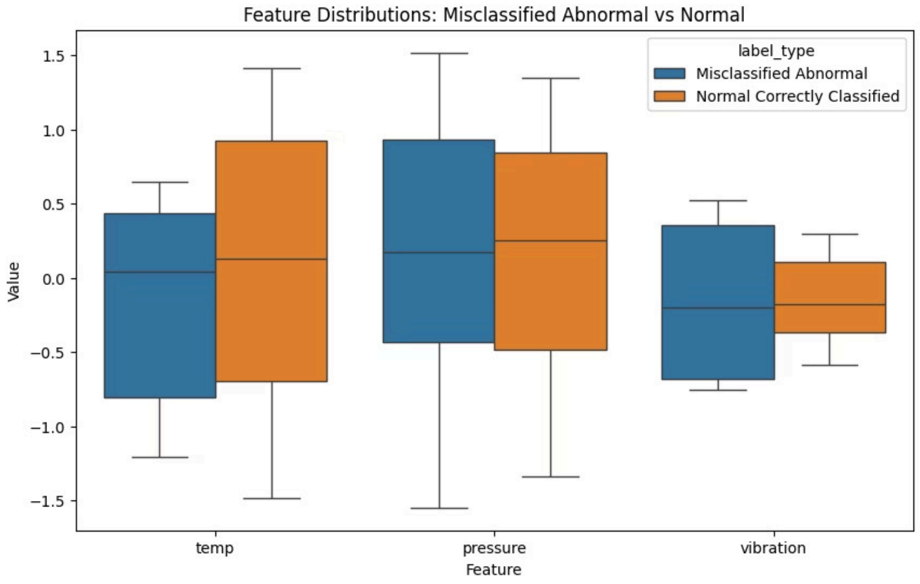
1 Score Distribution of IF

Score Distributions show overlap between normal and abnormal samples



2 Feature distribution

Misclassified abnormal samples tend to have feature values similar to normal samples.



Made with GAMMA

Threshold Optimization Strategy

The Problem: Over-alarming

The original Isolation Forest model often flags a high volume of false positives, leading to "alarm fatigue" and potentially obscuring truly critical anomalies.

The Solution: Thresholding

We propose adjusting the detection threshold to identify only the top 2% of data points with the highest anomaly scores as truly anomalous.

Expected Benefits

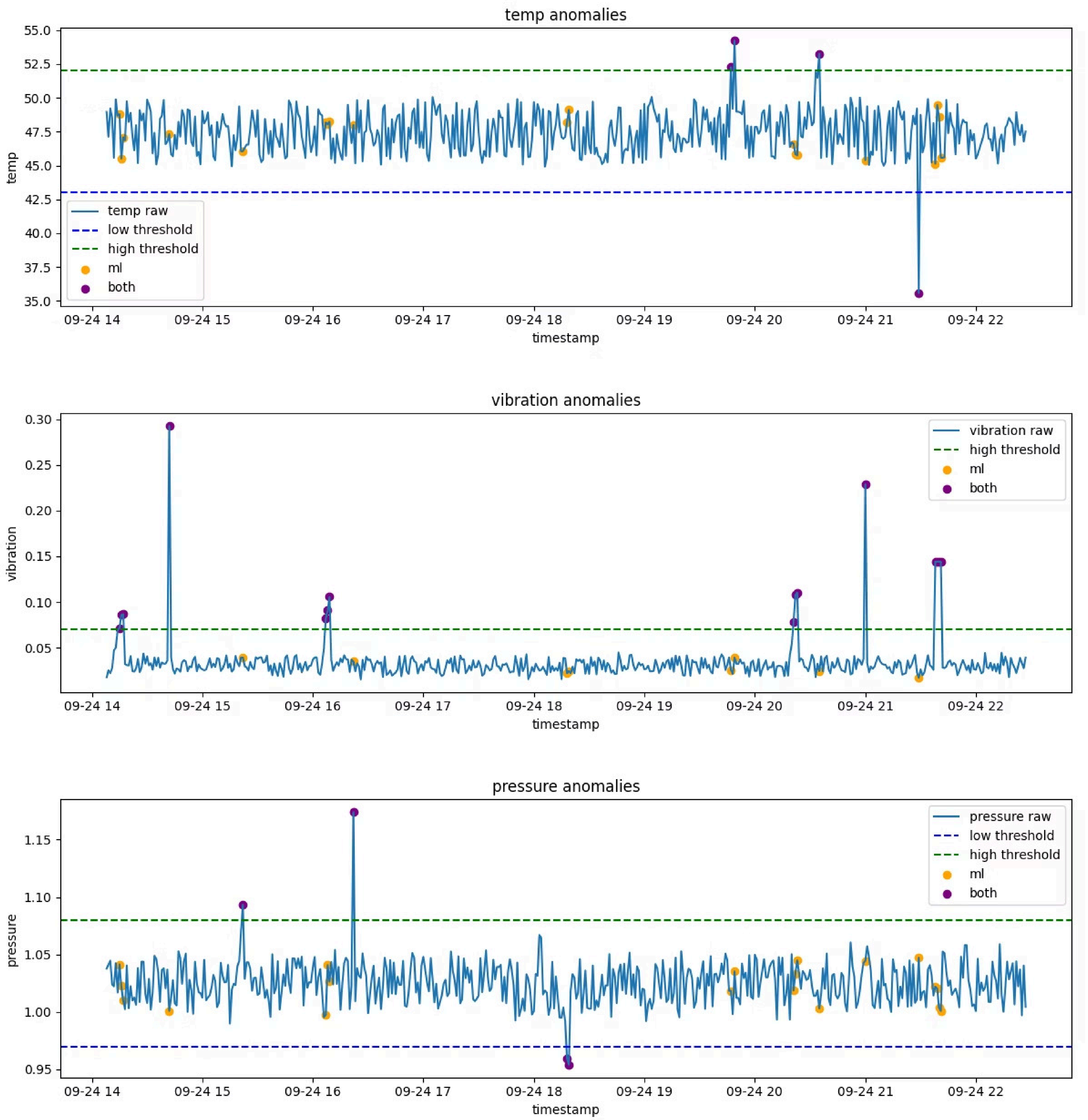
This strategy significantly reduces alarm fatigue, allowing teams to focus on actionable insights while ensuring critical anomalies are still detected.

Implementation

The new threshold will be percentile-based, dynamically setting the anomaly score cutoff at the 98th percentile, rather than relying on a fixed contamination parameter.

Made with GAMMA

Agent Detection



Made with GAMMA

Repository Structure

Source Code

Complete Python implementation including dataset generation, preprocessing pipelines, model training, and anomaly detection agent

Data

Generated datasets with realistic sensor patterns, anomaly labels, and validation sets for comprehensive testing

Model Outputs

Trained models, performance metrics, evaluation reports, and visualization assets for stakeholder review

Documentation

Comprehensive README, workflow diagrams, technical specifications, and deployment guidance

GitHub Repository: [GitHub - Nameyc99/Factory_Agent]

Made with GAMMA

Impact & Next Steps

AI-Accelerated Development

- **AI Assistant:** Strategic planning and technical research
- **Code Generator:** Rapid prototyping and automation
- **Gamma Platform:** Professional documentation

Delivered a complete end-to-end anomaly detection pipeline in accelerated timeframe.

Future Enhancements

- Hyperparameter optimization with expanded search space
- Time-series models for temporal dependency capture

Made with GAMMA

Example Use Case of AI tools

User: I want you to be my assistant to the assistant, give me prompts that helps a LLM assistant to complete task that I specify.

ChatGPT: If you want, next I can produce ready-to-send prompts for an LLM assistant to: (a) generate the dataset script, (b) implement the anomaly detector (rule-based + simple ML baseline), (c) create README and report sections, and (d) prepare demo screenshots and CLI outputs.

User: Yes, give me prompt (a)

ChatGPT:
You are a helpful Python developer. Produce a single, well-documented, runnable Python script named `generate_smartfactory_dataset.py` that creates a synthetic CSV dataset for the Smart Factory Alert Agent (Task 3). Follow these exact requirements:

1) CSV format & header
- Header columns (in this order): `timestamp,temp,pressure,vibration,label`
- Timestamp format: `YYYY-MM-DD HH:MM:SS` (24h)
- Save the CSV default path: `./data/smartfactory_dataset.csv`

.
.
.

Made with GAMMA

Example Use case of AI tools

User: Create content for gammas to generate. Presentation must include below: introduction to the AI tools used: AI Assistant: Generating Prompt for Code Generator and Drafting technical report and README.md Code Generator: Generating Python code templates for dataset generation, preprocessing, and model training. Debugging and suggesting improvements for edge-case handling. a complete report of the assignment results and all relevant screenshots or source code on Github

ChatGPT:
Slide 1 – Title

Smart Factory Anomaly Detection with AI-Powered Tools

Assignment Report & Results

By [Your Name]

.
.
.

Made with GAMMA

Example Use case of AI tools

User: (source code)
Give me prompt to ask you to write the whole code again according above, with modifying according to below 1) Give average precision and recall 2) Save metrics to a file

ChatGPT:
Prompt:

Rewrite the entire train_isolation_forest.py script with the following specifications:

Keep all existing functionality (loading data, preprocessing, Isolation Forest training, grid search, threshold selection, saving model).

Modify the evaluation section to additionally compute average precision and average recall across both classes.

Save all evaluation metrics (accuracy, precision/recall/F1 per class, average precision/recall, ROC-AUC, anomaly fraction) to a JSON file in the output directory.

Ensure logging still prints all metrics to console in a clear tabular format.

Maintain compatibility with command-line arguments (--train, --test, --scaler, --output_model, --grid_search, --out_dir, --report).

Made with GAMMA