Process Fault Detection in the Tennessee Eastman Process Using Statistical and Wavelet Features

Rishabh Goenka

EE 344 — Data-Driven Modeling And Machine Learning, Winter 2026

March 2026

1 Introduction and Problem Description

Modern chemical manufacturing facilities rely on dozens of sensors monitoring temperature, pressure, flow rates, and chemical compositions in real time. When process faults occur—valve sticking, feed composition shifts, slow drift in operating conditions—the plant may continue trusting degraded readings, leading to poor control decisions, safety hazards, and costly unplanned downtime. Detecting these faults quickly and accurately is therefore a critical objective in industrial process monitoring [4].

Simple threshold-based alarm systems, which flag individual sensor readings that exceed fixed limits, are insufficient for this task. The sensors in a typical chemical plant are highly correlated with nonlinear dynamics: a temperature spike may be normal under one set of operating conditions and dangerous under another. Machine learning offers the ability to learn complex multivariate patterns across all sensors simultaneously, capturing fault signatures that are invisible to single-sensor monitoring rules.

This project addresses fault detection in the **Tennessee Eastman Process** (TEP), a widely used benchmark that simulates a realistic chemical plant originally developed by Eastman Chemical Company [1]. The TEP has been extensively adopted in the process monitoring community for evaluating fault detection and diagnosis methods [3, 9], making it a well-characterized testbed with known baselines for comparison.

Two complementary machine learning tasks are formulated:

1. **Binary anomaly detection:** Is the plant operating normally, or is a fault present? (Classification with $y \in \{0, 1\}$.)
2. **Multi-class fault classification:** If a fault is present, which of 20 specific fault types is it? (Classification with $y \in \{0, 1, \ldots, 20\}$.)

The TEP benchmark includes 20 distinct fault types representing process-level disturbances: step changes in feed composition, random variation in process parameters, valve sticking, slow drift in operating conditions, and changes in reactor or condenser cooling water temperature, among others.

Methodological note. The original project proposal framed this work as "sensor degradation detection via signal quality analysis." After detailed analysis, the framing was refined: TEP faults are process-level disturbances, not sensor-specific degradation events. Signal-quality features (variance structure, wavelet-based frequency content) are included as complementary descriptors, but the primary detection mechanism relies on changes in process variable behavior. Results throughout this report should therefore be interpreted as *process fault classification performance*.

The central methodological question is whether wavelet-based frequency-domain features improve fault detection beyond what simple statistical summaries (mean, standard deviation,

skewness, kurtosis) already capture. Three parallel feature extraction pipelines—Statistical, Wavelet, and Combined—are evaluated across four supervised classifiers and two one-class anomaly detectors, providing a systematic comparison of feature representations and modeling paradigms.

2 Dataset Description

This study uses the Tennessee Eastman Process (TEP) simulation dataset published by Rieth et al. [2], which is derived from the benchmark chemical plant simulator originally developed by Downs and Vogel [1]. The dataset is publicly available through the Harvard Dataverse and is mirrored on Kaggle.

The TEP simulates a continuous chemical process comprising a reactor, condenser, compressor, vapor–liquid separator, and stripper (see Figure 1). The process converts four gaseous reactants (A, C, D, E) and one inert (B) into two liquid products (G, H) with one byproduct (F). The simulation generates readings from 52 process variables: 41 measured sensor outputs (`xmeas_1` through `xmeas_41`) and 11 manipulated variables or controller outputs (`xmv_1` through `xmv_11`). All features are continuous and numerical, sampled at regular 3-minute intervals.

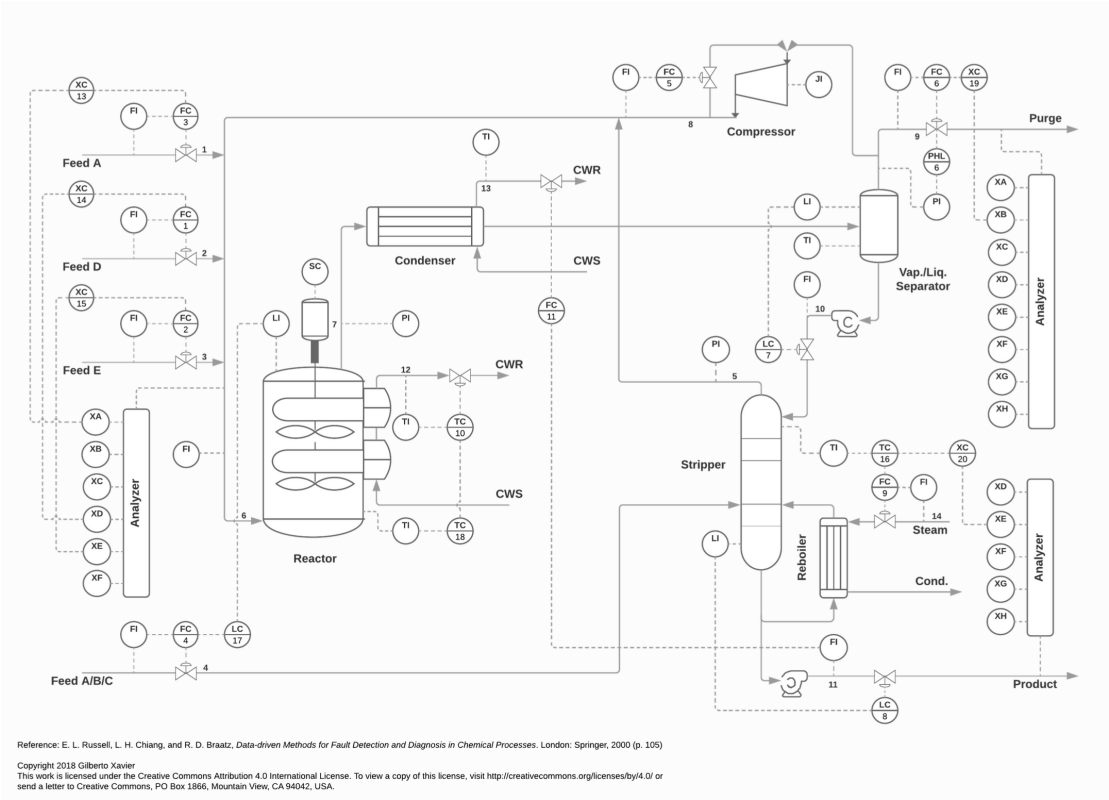

Figure 1: Process flow diagram of the Tennessee Eastman Process, showing the reactor, condenser, compressor, separator, and stripper with associated instrumentation. Adapted from Russell, Chiang, and Braatz [3]; diagram by Xavier (2018), licensed under CC BY 4.0.

The raw data are distributed across four Parquet files, summarized in Table 1.

Each record represents a single 3-minute snapshot of all process variable readings during one simulation run. The label `faultNumber` takes values in $\{0, 1, \ldots, 20\}$, where 0 denotes normal operation and 1–20 correspond to specific process disturbances. Faults are injected at sample 20 in training runs and sample 160 in testing runs. Fault types include step changes in feed composition, random variation in process parameters, valve sticking, slow drift in operating conditions, and changes in cooling water temperature, among others.

Table 1: Structure of the TEP simulation dataset.

File	Description	Runs	Samples/Run	Total Samples
FaultFree Training	Normal operation	500	500	250 000
FaultFree Testing	Normal operation	500	960	480 000
Faulty Training	20 fault types	500 each	500	10 000 000
Faulty Testing	20 fault types	500 each	960	19 200 000

To maintain tractable computation, the dataset was subsampled to 50 of the available 500 runs per fault type using a fixed random seed of 42. After subsampling, the training set contains 525 000 rows × 55 columns and the testing set contains 1 008 000 rows × 55 columns (52 sensor features plus three metadata columns: `faultNumber`, `simulationRun`, and `sample`).

A data audit confirmed zero missing values, zero infinite values, and zero duplicate rows in both splits. All 21 fault classes have exactly 50 runs and 25 000 samples each in training, confirming uniform representation at the multi-class level.

Limitations. Three properties of the dataset merit discussion. First, the data are *simulated*: generated from a mathematical model, they may not fully capture the noise characteristics of real industrial sensors. Second, the *binary class balance* becomes heavily skewed after windowing (Section 4), posing challenges for binary anomaly detection. Third, certain fault types are well documented in the TEP literature as extremely difficult to detect [3, 9], a property confirmed by our results (Section 7).

3 Exploratory Data Analysis

Exploratory analysis focused on understanding sensor relationships, assessing the separability of fault types in reduced-dimensional space, and identifying data properties that would inform modeling decisions.

3.1 Sensor Correlation Structure

Figure 2 shows the pairwise Pearson correlation matrix of all 52 process variables during normal (fault-free) operation. The matrix reveals pronounced block structure: groups of measured sensors (e.g., `xmeas_1`–`xmeas_5`, `xmeas_17`–`xmeas_20`, `xmeas_22`–`xmeas_30`) are strongly correlated with one another, reflecting the underlying physical coupling between process units. Several off-diagonal blocks show correlations exceeding $|r| > 0.7$, while the manipulated variables (`xmv_1` through `xmv_11`) in the lower-right corner are comparatively isolated, consistent with controller outputs being partially decoupled from raw sensor readings.

This high degree of inter-sensor redundancy confirms that single-sensor threshold monitoring is fundamentally limited and motivates the use of multivariate feature extraction (Section 4).

3.2 PCA Visualization of Fault Separability

To assess whether fault types form distinguishable clusters, principal component analysis (PCA) was applied to the raw sensor data and projected onto the first two principal components. Figure 3 presents two views of this projection.

The first two components capture a combined 44.6% of total variance. Several fault types separate clearly—for example, the cyan cluster at the upper right and the red cluster at the lower right form well-isolated groups—while a dense central mass contains considerable overlap. The right panel zooms into faults 3, 9, and 15, which overlap almost entirely with the normal

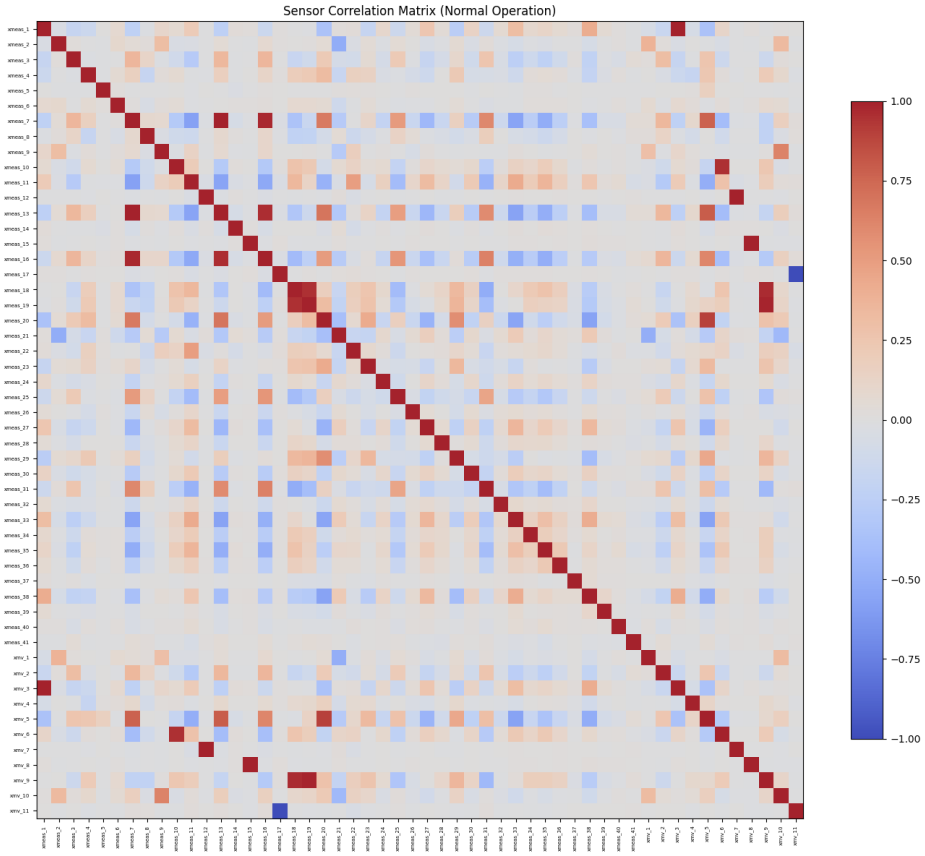

Figure 2: Pairwise Pearson correlation matrix of the 52 process variables during normal operation. Strong block structure is visible among measured sensors, indicating substantial redundancy. Manipulated variables (lower right) are comparatively uncorrelated with sensor readings.

data cloud. This sets an expectation that any classifier will struggle with these specific fault types, a prediction borne out in Section 7.

3.3 EDA Insights That Informed Modeling

Three observations directly influenced subsequent modeling decisions:

1. Strong inter-sensor correlations justified multivariate feature extraction over single-sensor statistics.
2. PCA overlap for faults 3, 9, and 15 set realistic expectations—perfect classification across all 21 classes is unlikely with any method.
3. The clean data (zero missing values, zero duplicates) meant that all differences in model performance can be attributed to feature representation and model capacity rather than data quality artifacts.

4 Data Preprocessing and Feature Engineering

4.1 Data Cleaning

No cleaning was required: the audit described in Section 2 found zero missing values, infinite values, or duplicates. All 52 process variables are continuous and numerical, so no categorical encoding was needed.

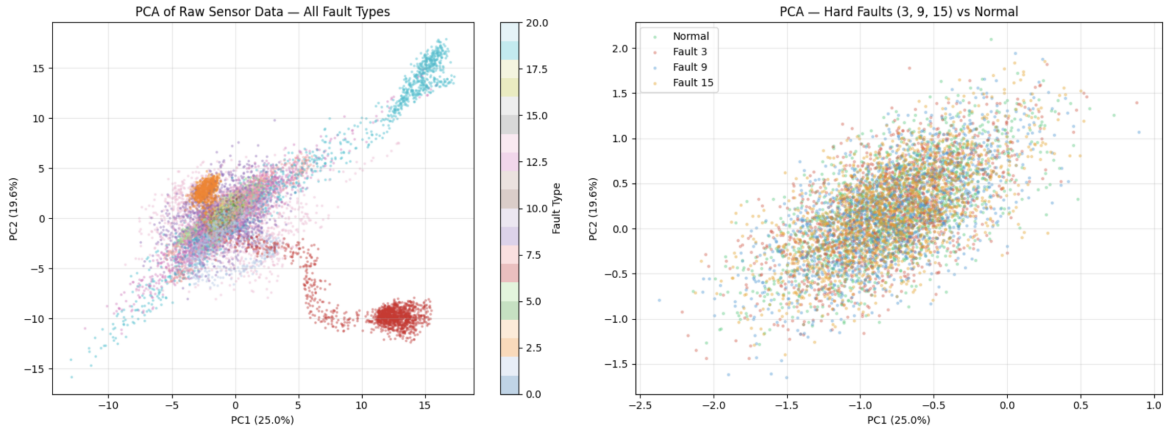

Figure 3: **Left:** PCA projection of raw sensor data colored by fault type (0–20). PC1 captures 25.0% and PC2 captures 19.6% of total variance. Several fault types form distinct clusters, while many overlap in the central mass. **Right:** Zoomed view of the hard faults (3, 9, 15) versus normal operation, showing near-complete overlap—these faults produce minimal displacement in the leading principal components.

4.2 Sliding Window Segmentation

Raw time-series data were segmented into overlapping windows to capture local temporal patterns. Each window spans **30 consecutive samples** (90 minutes at the 3-minute sampling interval) with a **stride of 10 samples** (30 minutes), producing overlapping segments that share 20 of 30 samples with their neighbors. This overlap is important for cross-validation design (Section 6).

Two constraints governed window construction:

- **Run boundaries:** Windows never span across different simulation runs, preventing cross-run information leakage.
- **Label assignment:** A window from a faulty run is labeled as faulty only if more than 50% of its samples fall after the known fault injection point; otherwise, it is labeled as normal (fault 0).

Each unique (`faultNumber`, `simulationRun`) pair was assigned a group identifier (computed as faultNumber $\times$ 1000 + simulationRun) for use in grouped cross-validation (Section 6).

4.3 Feature Extraction Pipelines

All features are computed per sensor, per window across the 52 process variables. Three parallel pipelines were implemented:

Pipeline 1: Statistical Features (312 features). Six summary statistics are extracted from each sensor's 30-sample window: mean, standard deviation, skewness, kurtosis, minimum, and maximum. With 52 sensors, this yields $6 \times 52 = 312$ features per window. These capture the level, spread, and shape of each sensor's distribution within the window.

Pipeline 2: Wavelet Features (624 features). A three-level discrete wavelet transform (DWT) using the Daubechies-4 (`db4`) mother wavelet [8] is applied to each sensor's window via PyWavelets [12]. The decomposition produces four sub-bands (three detail levels plus one approximation). From each sub-band, three descriptors are extracted: energy (sum of squared coefficients), Shannon entropy (of normalized absolute coefficients), and mean magnitude (mean

of absolute coefficients). This yields $4 \times 3 \times 52 = 624$ features per window, capturing frequency content—how signals oscillate at different time scales.

Pipeline 3: Combined Features (936 features). The union of the Statistical and Wavelet pipelines: $312 + 624 = 936$ features per window.

Table 2 summarizes the resulting feature matrices.

Table 2: Dimensions of the feature matrices after windowing and feature extraction.

Feature Set	Training Shape	Testing Shape
Statistical (312)	50 400 × 312	98 700 × 312
Wavelet (624)	50 400 × 624	98 700 × 624
Combined (936)	50 400 × 936	98 700 × 936

After windowing, the training label distribution is imbalanced at the binary level: 3 400 normal windows (6.75%) versus 47 000 faulty windows (93.25%). At the multi-class level, each of the 20 fault types contributes 2 350 windows (4.66% each). This severe binary imbalance motivated the use of `class_weight="balanced"` in all supervised models and `scale_pos_weight` in XGBoost for the binary task.

4.4 Scaling

All features were standardized to zero mean and unit variance using `StandardScaler`, fit on training data only within each cross-validation fold and applied to validation and test partitions via `transform()`. This ensures no information from held-out data leaks into the scaling parameters.

4.5 Dimensionality Reduction

Principal component analysis (PCA) retaining 95% of explained variance was tested in a separate experimental branch, applied to the Wavelet and Combined feature sets after scaling. PCA was fit within each cross-validation fold on the fold's training portion only. This experiment produced a *negative finding*: PCA did not clearly improve model performance on any feature–model combination, and in some cases slightly degraded it.

5 Methods and Models

Two categories of models were evaluated: supervised classifiers (trained on labeled examples of both normal and faulty operation) and one-class anomaly detectors (trained exclusively on normal data). The supervised models address both the binary and multi-class tasks; the one-class models address binary anomaly detection only, representing the realistic industrial scenario in which labeled fault examples are scarce or unavailable.

5.1 Supervised Models

Four classifiers spanning two model families—linear and tree-based—were selected to test whether fault boundaries are linearly separable and to compare ensemble strategies.

Logistic Regression (linear baseline). A multinomial logistic regression classifier with `max_iter = 2000` and `class_weight = "balanced"` was used as the primary linear baseline, establishing the performance achievable under a purely linear decision boundary.

Random Forest (independent tree ensemble). A Random Forest [6] with 200 trees, `class_weight = "balanced"`, and `n_jobs = -1` was trained as a tree-based ensemble baseline. Random Forest constructs each tree independently on a bootstrap sample and averages their predictions, producing robust but potentially less precise estimates than sequential methods.

Linear SVM (margin-based linear baseline). A linear support vector machine (`LinearSVC`) with `class_weight = "balanced"` was included as a second linear baseline. Unlike Logistic Regression, which optimizes log-likelihood, LinearSVC maximizes the margin between classes, offering a complementary view of linear separability.

XGBoost (gradient-boosted trees). XGBoost [5] was selected as the primary high-capacity model. Default configuration used `n_estimators = 200`, `tree_method = "hist"`, and `eval_metric = "logloss"` (binary) or `"mlogloss"` (multi-class). For the binary task, `scale_pos_weight` was set to the ratio of negative to positive training samples. XGBoost builds trees sequentially, with each tree correcting the residual errors of its predecessors, which makes it particularly effective for tabular data with complex, non-linear decision boundaries.

Table 3 summarizes the supervised model configurations.

Table 3: Supervised model configurations.

Model	Role	Key Settings
Logistic Regression	Linear baseline	`max_iter=2000`, `class_weight="balanced"`, `random_state=42`
Random Forest	Tree ensemble	`n_estimators=200`, `class_weight="balanced"`, `random_state=42`, `n_jobs=-1`
Linear SVM	Margin-based linear	`class_weight="balanced"`
XGBoost	Gradient-boosted trees	`n_estimators=200`, `tree_method="hist"`, `random_state=42`, `scale_pos_weight` (binary)

5.2 One-Class / Anomaly Detection Models

Two one-class models were evaluated, each representing a distinct anomaly detection paradigm.

Isolation Forest. Isolation Forest [7] detects anomalies by randomly partitioning the feature space and measuring how easily each sample can be isolated. Anomalous points require fewer random splits and thus receive higher anomaly scores. Configuration used `contamination = "auto"`, `random_state = 42`, and `n_jobs = -1`.

Autoencoder. A fully connected autoencoder implemented in PyTorch [11] learns a compressed representation of normal operating data and reconstructs its input. The reconstruction error (mean squared error) on test data serves as the anomaly score. The architecture consists of a three-layer encoder (input $\to$ 64 $\to$ 32 $\to$ 16) and a symmetric decoder (16 $\to$ 32 $\to$ 64 $\to$ input). Training used the Adam optimizer with a learning rate of 10^{-3}, batch size of 256, MSE loss, and 50 epochs.

One-class threshold. For both models, the anomaly threshold was set at the 95th percentile of anomaly scores computed on fault-free training windows. No test data were used in threshold selection.

5.3 Hyperparameter Tuning

Hyperparameter tuning was conducted only for XGBoost on the multi-class task with Statistical features, as it produced the strongest default results. A grid search evaluated the parameter space shown in Table 4, using 5-fold GroupKFold.

Table 4: XGBoost hyperparameter tuning grid.

Parameter	Values Tested
`n_estimators`	{200, 300}
`max_depth`	{4, 6, 8}
`learning_rate`	{0.05, 0.1, 0.2}

The best configuration was `n_estimators = 300`, `max_depth = 6`, `learning_rate = 0.1`, achieving a cross-validation macro F1 of 0.8744. The tuned test F1 was 0.8720, essentially identical to the default test F1 of 0.8721, indicating that the default hyperparameters were already near-optimal for this dataset.

Tuning was not extended to the remaining supervised models. For Linear SVM, this was a practical necessity: a single multi-class configuration with Combined features required approximately 3 hours of computation, making a grid search infeasible. For Logistic Regression and Random Forest, default parameters produced competitive results, and the XGBoost tuning outcome suggested that the marginal benefit would be small.

6 Experimental Setup and Evaluation Methodology

6.1 Train/Test Split

The TEP dataset provides pre-defined training and testing files generated from non-overlapping random seeds [2]. No additional random splitting was performed; the original TEP train/test partition was used throughout.

6.2 Cross-Validation

Model selection and performance estimation on training data used **5-fold GroupKFold** cross-validation, with groups defined by the unique simulation-run identifier described in Section 4. This design ensures that all windows derived from the same simulation run remain in the same fold. Because adjacent sliding windows share 20 of 30 samples, a naive random split would allow the model to train and validate on highly overlapping windows, artificially inflating scores. GroupKFold eliminates this temporal leakage. Scaling was performed inside each fold as described in Section 4.

6.3 Evaluation Metrics

Primary metric: Macro F1-Score. The macro-averaged F1-score computes the F1 for each class independently and then averages equally across all classes. This prevents a model from achieving a deceptively high score by detecting only the easy fault types while failing on the hard ones. Plain accuracy would mask such failures.

Supporting metrics. Accuracy is reported for completeness. Per-fault F1 scores decompose performance across individual fault types. Confusion matrices reveal systematic inter-fault confusion. For one-class models, ROC-AUC measures ranking quality across all possible decision thresholds. Train versus test F1 comparisons serve as an overfitting diagnostic.

6.4 Supervised Evaluation Protocol

For each of the four supervised models (Section 5), performance was evaluated on all three feature sets under two task formulations: binary classification (normal versus any fault) and multi-class classification (all 21 classes). This produces $4 \times 3 \times 2 = 24$ supervised configurations. After cross-validation, the final model for each configuration was retrained on the full training set and evaluated on the held-out test set.

6.5 One-Class Evaluation Protocol

One-class models (Isolation Forest, Autoencoder) were trained exclusively on fault-free training windows. The anomaly threshold was set at the 95th percentile of anomaly scores on fault-free training data; no test data were used in threshold selection. Evaluation was performed on the full mixed test set under the binary formulation, using both F1-score and ROC-AUC.

Implementation note. The one-class results reported in Section 7 are drawn from the optimized notebook variant (see Section 6.6), which used 30 runs per fault type with non-overlapping training windows. Both Isolation Forest and Autoencoder results come from this single notebook, ensuring a fair comparison on identical data.

6.6 Computational Note

The primary experimental notebook executed successfully for all supervised models and for Isolation Forest. However, the Autoencoder training routine caused a kernel crash due to memory exhaustion when operating on the full-dimensional feature matrices (up to $50\,400 \times 936$ for Combined features). A second, optimized notebook was created with two modifications: (i) runs per fault type reduced from 50 to 30, and (ii) non-overlapping training windows. These changes reduced memory requirements sufficiently for the Autoencoder to train to completion. The optimized notebook also tested PCA-based dimensionality reduction, yielding the negative finding noted in Section 4. Supervised results are drawn from the primary notebook; one-class results from the optimized notebook.

7 Results

7.1 Binary Classification

Table 5 presents all 12 supervised configurations for the binary task, ranked by test F1.

The most striking result is that **linear models generalize substantially better than tree-based models** for binary detection. Linear SVM with Combined features achieves the best test F1 of 0.821, while Random Forest—despite near-perfect training F1 of 0.998—collapses to test F1 $\approx$ 0.50, no better than chance. XGBoost occupies a middle ground (test F1 $\approx$ 0.68–0.70) but still exhibits a large train–test gap. The linear models show the opposite pattern: their test F1 *exceeds* their cross-validation F1, suggesting that the linear decision boundary generalizes well. This behavior is explained in Section 8.

Table 5: Binary classification results (normal vs. fault), ranked by test macro F1. Bold indicates the best test F1.

Rank	Features	Model	CV F1	CV Std	Train F1	Test F1	Test Acc
1	Combined	Linear SVM	0.679	0.037	0.708	**0.821**	0.866
2	Wavelet	Linear SVM	0.659	0.038	0.674	0.811	0.854
3	Statistical	Linear SVM	0.655	0.037	0.667	0.810	0.852
4	Combined	Logistic Reg	0.672	0.037	0.702	0.809	0.858
5	Statistical	Logistic Reg	0.653	0.035	0.666	0.807	0.851
6	Wavelet	Logistic Reg	0.655	0.036	0.675	0.804	0.850
7	Statistical	XGBoost	0.711	0.014	0.981	0.701	0.847
8	Wavelet	XGBoost	0.713	0.012	0.967	0.683	0.841
9	Combined	XGBoost	0.722	0.018	0.985	0.679	0.842
10	Statistical	Random Forest	0.618	0.038	0.998	0.507	0.812
11	Combined	Random Forest	0.681	0.039	0.998	0.500	0.811
12	Wavelet	Random Forest	0.681	0.039	0.998	0.497	0.811

Table 6: Multi-class classification results (21 classes), ranked by test macro F1. Bold indicates the best test F1. Runtime is wall-clock time for one full cross-validation run.

Rank	Features	Model	CV F1	CV Std	Test F1	Runtime
1	Statistical	XGBoost	0.873	0.005	**0.872**	345 s
2	Combined	XGBoost	0.875	0.004	0.871	1 441 s
3	Wavelet	XGBoost	0.863	0.004	0.856	1 231 s
4	Statistical	Random Forest	0.846	0.006	0.849	225 s
5	Combined	Random Forest	0.846	0.008	0.848	634 s
6	Wavelet	Random Forest	0.834	0.007	0.836	400 s
7	Combined	Linear SVM	0.838	0.005	0.820	10 819 s
8	Statistical	Logistic Reg	0.835	0.003	0.816	24 s
9	Statistical	Linear SVM	0.830	0.003	0.814	599 s
10	Wavelet	Linear SVM	0.832	0.006	0.812	4 146 s
11	Combined	Logistic Reg	0.831	0.003	0.811	204 s
12	Wavelet	Logistic Reg	0.822	0.004	0.801	79 s

7.2 Multi-class Classification

Table 6 presents all 12 supervised configurations for the 21-class task, ranked by test F1.

XGBoost dominates: the top three configurations are all XGBoost. The best result—XGBoost with Statistical features—achieves test macro F1 of **0.872** in 345 seconds, outperforming the same model with Combined features (0.871, 1 441 s) at roughly one quarter the runtime. Statistical features alone are sufficient; wavelet features do not add meaningful discriminative information.

7.3 Per-Fault Performance of the Best Model

Table 7 presents the full classification report for the best model (XGBoost, Statistical features, test macro F1 = 0.872, test accuracy = 0.815).

Performance is bimodal. Seventeen of the 20 fault types achieve F1 $>$ 0.93, with seven exceeding 0.99. Three faults are dramatically worse: fault 9 (F1 = 0.190), fault 15 (F1 = 0.280), and fault 3 (F1 = 0.591). The normal class also underperforms at F1 = 0.596. Figure 4 visualizes this pattern.

Table 7: Per-fault classification report for XGBoost with Statistical features on the 21-class test set. Hard faults (3, 9, 15) and the normal class (0) are highlighted.

Fault	Precision	Recall	F1-Score	Support
0 (normal)	**0.741**	**0.498**	**0.596**	19 700
1	0.996	0.999	0.997	3 950
2	1.000	0.998	0.999	3 950
3	**0.523**	**0.678**	**0.591**	3 950
4	1.000	1.000	1.000	3 950
5	0.999	0.999	0.999	3 950
6	1.000	1.000	1.000	3 950
7	1.000	0.999	0.999	3 950
8	0.977	0.975	0.976	3 950
9	**0.149**	**0.260**	**0.190**	3 950
10	0.942	0.910	0.926	3 950
11	0.994	0.999	0.996	3 950
12	0.986	0.968	0.977	3 950
13	0.990	0.893	0.939	3 950
14	1.000	0.999	1.000	3 950
15	**0.216**	**0.396**	**0.280**	3 950
16	0.970	0.948	0.959	3 950
17	0.999	0.972	0.986	3 950
18	0.930	0.945	0.937	3 950
19	0.996	1.000	0.998	3 950
20	0.997	0.947	0.971	3 950
Macro avg	**0.876**	**0.875**	**0.872**	98 700
Weighted avg	0.855	0.815	0.828	98 700

7.4 Confusion Matrix Analysis

Figure 5 shows the normalized confusion matrix for the best model.

The off-diagonal errors concentrate in a **four-class confusion cluster** involving faults 0 (normal), 3, 9, and 15:

- **Fault 9:** 35% misclassified as normal, 16% as fault 3, 22% as fault 15, with only 26% correct.
- **Fault 15:** 31% as normal, 21% as fault 9, 40% correct.
- **Fault 3:** 12% as normal, 10% as fault 9, 10% as fault 15, 68% correct.
- **Normal (0):** 50% correct, with 22% leaking into fault 9 and 21% into fault 15.

These four classes are confused primarily with *each other*, not with the other 17 fault types—consistent with the PCA overlap observed in Section 3.

7.5 Hyperparameter Tuning

Table 8 compares the default and tuned XGBoost configurations.

Table 8: XGBoost default vs. tuned performance (multi-class, Statistical features).

	Default	Tuned
Configuration	$n=200$, depth=None, lr=default	$n=300$, depth=6, lr=0.1
CV Macro F1	0.873	0.874
Test Macro F1	0.872	0.872
Test Accuracy	—	0.813

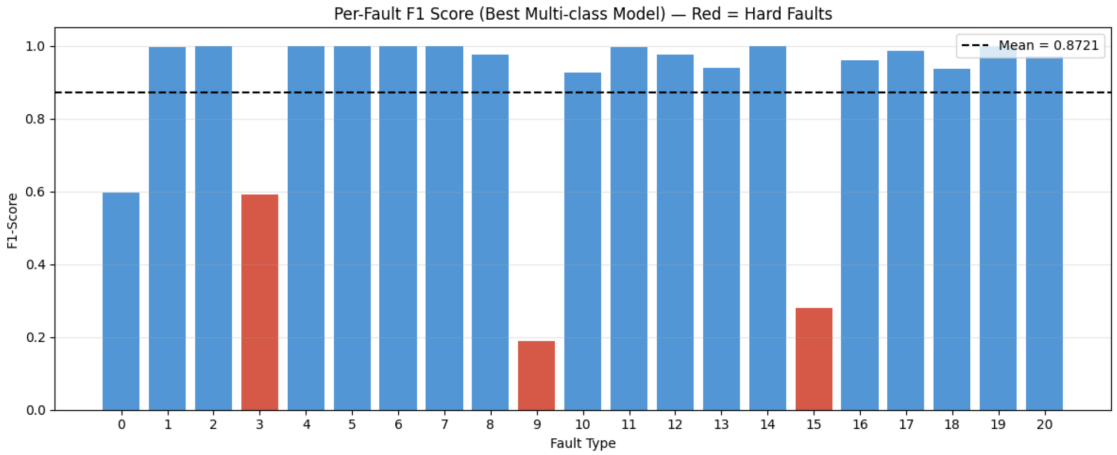

Figure 4: Per-fault F1 scores for the best multi-class model (XGBoost, Statistical features). Red bars indicate faults 3, 9, and 15. The dashed line marks the macro-average F1 of 0.872.

The tuned test F1 is essentially identical to the default (0.8720 vs. 0.8721), confirming that the performance ceiling is set by the inherent difficulty of the hard faults, not by hyperparameter choices.

7.6 One-Class Anomaly Detection

Table 9 presents the one-class results from the optimized notebook (Section 6.6).

Table 9: One-class anomaly detection results (binary: normal vs. fault). All results from the optimized notebook (30 runs, non-overlapping windows). Bold indicates best F1 and best ROC-AUC.

Model	Features	F1	ROC-AUC
Autoencoder	Statistical	**0.731**	0.880
Autoencoder	Combined	0.730	**0.882**
Autoencoder	Wavelet	0.715	0.877
Isolation Forest	Statistical	0.604	0.850
Isolation Forest	Wavelet	0.606	0.833
Isolation Forest	Combined	0.601	0.834

The Autoencoder consistently outperforms Isolation Forest by approximately 13 F1 points (best: 0.731 vs. 0.606), suggesting that learned reconstruction captures richer normal-state structure than random partitioning. Comparing the best one-class model (Autoencoder, F1 = 0.731) against the best supervised binary model (Linear SVM, F1 = 0.821), the gap is approximately **9 F1 points**, quantifying the value of labeled fault data.

7.7 Feature Set Comparison

Table 10 summarizes average test F1 across models for each feature set on the multi-class task.

Statistical features alone match or slightly outperform both alternatives. Adding 624 wavelet features did not consistently improve performance for any model.

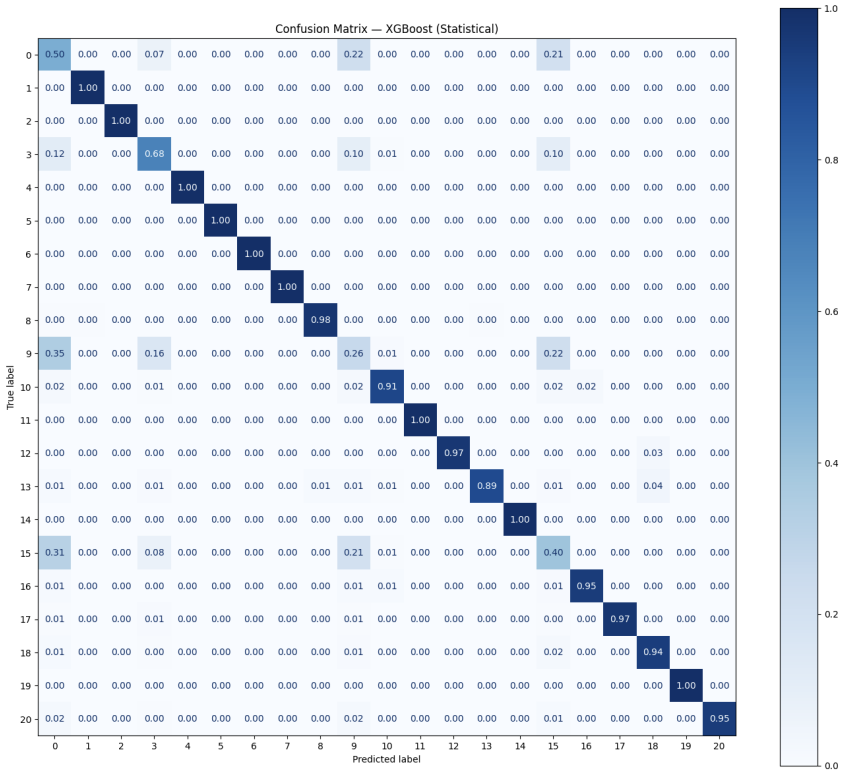

Figure 5: Normalized confusion matrix for XGBoost with Statistical features on the 21-class test set. Off-diagonal mass concentrates in a 4-class confusion cluster involving faults 0, 3, 9, and 15.

Table 10: Average test macro F1 across all supervised models, by feature set (multi-class task).

Feature Set	Avg Test F1
Statistical	∼0.84
Combined	∼0.83
Wavelet	∼0.83

7.8 Overfitting Analysis

Binary task. Tree-based models exhibited severe overfitting. Random Forest achieved training F1 $\approx$ 0.998 but test F1 $\approx$ 0.50—a gap of roughly 50 points. XGBoost showed a gap of approximately 28 points. The linear models showed small or even *negative* gaps: training F1 $\approx$ 0.67–0.71 with test F1 $\approx$ 0.80–0.82.

Multi-class task. Overfitting was substantially less severe across all models. The contrast is explained by the binary class imbalance (Section 4): tree models memorize the sparse minority class rather than learning a generalizable boundary. In the multi-class setting, more uniform class representation reduces this failure mode.

8 Discussion and Interpretation

8.1 Why Simple Features Won

The detectable TEP faults tend to manifest as *level shifts* or *variance changes* in sensor readings—precisely the patterns captured by mean and standard deviation. The wavelet de-

composition adds frequency-resolution information (energy and entropy at different scales), but if fault signatures are primarily amplitude-based rather than frequency-based, this information is redundant at best and noisy at worst. The 936-dimension Combined set may also suffer from a mild curse of dimensionality: additional features increase the volume of the feature space without adding discriminative signal. The practical implication is that, for TEP-style process monitoring, simple window-level statistics are sufficient.

8.2 The Hard-Fault Confusion Cluster

The confusion matrix (Figure 5) reveals that faults 3, 9, and 15 form a closed cluster with the normal class: they are confused with each other and with normal operation, but not with the other 17 faults. Combined with the PCA overlap (Figure 3), this indicates that these faults produce process disturbances so subtle they lie within the noise floor of normal operating variability. Detecting them likely requires either richer temporal representations (e.g., sequence models on raw time series) or domain-specific features tailored to the known physical mechanisms of these disturbances. This difficulty is consistent with the published TEP literature [3, 9].

8.3 Why Linear Models Win for Binary Detection

The binary training set is severely imbalanced (Section 4). Tree models with sufficient depth can memorize the sparse normal-class examples via narrow, sample-specific splits that do not transfer to unseen data. Linear models, constrained to a single hyperplane, cannot memorize in this way; they are forced to find the best globally-linear separator, which generalizes better. This reversal does not occur in the multi-class setting, where class representation is more uniform and tree models can learn meaningful splits.

8.4 The Value of Labeled Data

The 9-point gap between the best supervised binary model (F1 = 0.821) and the best one-class model (Autoencoder, F1 = 0.731) is meaningful but moderate. One-class detection is a viable starting point for industrial deployment—it requires no historical fault data—but performance improves substantially when even modest labeled fault data become available. Within the one-class paradigm, the Autoencoder's 13-point advantage over Isolation Forest demonstrates the benefit of learned representations over random partitioning.

8.5 Comparison to Published Results

Our best multi-class macro F1 of 0.872 is broadly consistent with recent TEP benchmarks. Hu et al. [9] report an overall fault diagnosis rate of 91% using an XGBoost-based feature selection pipeline with a KELM classifier, though their methodology differs (per-fault detection rate rather than macro F1). Direct comparison is complicated by differences in preprocessing and evaluation metrics, but the qualitative finding—most TEP faults are highly detectable while faults 3, 9, and 15 remain stubborn—is universal across the literature.

8.6 Limitations

- Simulated data may not capture real industrial sensor noise and degradation.
- Only 50 of 500 available runs were used (30 for one-class), increasing variance in performance estimates.
- Hyperparameter tuning was limited to XGBoost; broader tuning could narrow gaps, though the XGBoost outcome suggests marginal benefit.
- Evaluation is at the window level; run-level aggregation and fault detection delay analysis were not performed.

- TEP faults are process-level disturbances, not sensor-specific degradation; the original project framing was adjusted accordingly.

9 Conclusion and Future Work

9.1 Summary

This project applied three feature extraction pipelines (Statistical, Wavelet, Combined) and six models (Logistic Regression, Random Forest, Linear SVM, XGBoost, Isolation Forest, Autoencoder) to the Tennessee Eastman Process benchmark for binary anomaly detection and 21-class fault classification. A total of 24 supervised and 6 one-class configurations were evaluated using grouped cross-validation designed to prevent temporal leakage.

9.2 Main Findings

1. **Simple statistical features are sufficient.** The 312-feature statistical set matched or outperformed the 624-feature wavelet and 936-feature combined sets across all models and tasks.
2. **XGBoost is the strongest multi-class model** (test macro F1 = 0.872), while linear models generalize better for binary detection (Linear SVM, F1 = 0.821) due to severe class imbalance.
3. **Faults 3, 9, and 15 are near-undetectable** (F1 = 0.591, 0.190, 0.280), forming a closed confusion cluster with normal operation—consistent with published TEP literature.
4. **Labeled data provides a moderate advantage:** 9 F1 points over the best one-class model (Autoencoder, F1 = 0.731).
5. **Hyperparameter tuning yields negligible returns,** confirming that the performance ceiling is set by the hard faults.

9.3 Future Work

- **Deep learning on raw time series:** 1D-CNN or LSTM models could learn temporal patterns that hand-crafted features miss.
- **Additional wavelet families** (Symlets, Coiflets, Morlet) or adaptive wavelet selection.
- **Supervised + one-class ensembles** combining high accuracy on known faults with novelty detection for unseen faults.
- **Validation on real industrial data** to test transferability beyond simulation.
- **Fault detection delay analysis:** measuring time from fault onset to reliable detection.
- **Targeted hard-fault investigation:** identifying which sensors carry the most subtle signal for faults 3, 9, and 15.

10 Team Contributions

This was an individual project; all work was performed solely by the author.

References

[1] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245–255, 1993.

[2] C. A. Rieth, B. D. Amsel, R. Tran, and M. B. Cook, "Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation," Harvard Dataverse, 2017. https://doi.org/10.7910/DVN/6C3JR1.

[3] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault Detection and Diagnosis in Industrial Systems*. London: Springer, 2001.

[4] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis—Part III: Process history based methods," *Computers & Chemical Engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

[6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, pp. 413–422, 2008.

[8] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed. Academic Press, 1999.

[9] M. Hu, X. Hu, Z. Deng, and B. Tu, "Fault diagnosis of Tennessee Eastman Process with XGB-AVSSA-KELM algorithm," *Energies*, vol. 15, no. 9, p. 3198, 2022.

[10] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 8024–8035, 2019.

[12] G. R. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. O'Leary, "PyWavelets: A Python package for wavelet analysis," *Journal of Open Source Software*, vol. 4, no. 36, p. 1237, 2019.