**International Journal of Artificial Intelligence, Data Science, and Machine Learning**
Grace Horizon Publication | Volume 7, Issue 3, 52-57, 2026
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V7I3P107

*Original Article*

# Automated Incident Detection in Refinery Operations Using Machine Learning

Jasvitha Buggana
Independent Researcher, USA.

***Received On: 28/05/2026*** ***Revised On: 26/06/2026*** ***Accepted On: 04/07/2026*** ***Published On: 13/07/2026***

***Abstract -*** *Refinery operations run continuously and involve complex, tightly coupled processes in which a single equipment failure or chemical imbalance can escalate rapidly into a major safety incident. Most refineries still rely on human operators to detect and classify incidents manually from alarm panels and sensor dashboards a process that is slow, cognitively demanding, and error-prone. This paper proposes an automated incident-detection system based on machine learning that monitors real-time process-sensor streams, identifies anomalies, and classifies them into specific incident types without waiting for human recognition. The system couples two unsupervised anomaly-detection techniques (an autoencoder neural network and an isolation forest) with two supervised classifiers (a random forest and a long short-term memory network). The paper states the problem, describes the proposed architecture, explains each machine-learning technique and the rationale for its selection, and presents a step-by-step methodology for building and deploying the system in a production refinery environment.*

***Keywords -*** *Automated Incident Detection, Refinery Operations, Anomaly Detection, Machine Learning, Process Safety, Sensor Data, Event Classification, Real-Time Monitoring.*

## 1. Introduction

A refinery is among the most complex industrial environments in the world. At any moment, thousands of sensors measure temperature, pressure, flow rate, chemical composition, equipment vibration, and electrical load across hundreds of interconnected units distillation columns, heat exchangers, reactors, compressors, storage tanks, and pipelines. These processes must operate within tightly defined safe limits around the clock, every day of the year. When a fault begins a pump overheating, a relief valve lifting unexpectedly, a catalyst bed degrading the change first appears as a subtle shift in one or more sensor readings. Caught early, the corrective action is simple; missed, the disturbance grows, propagates to neighbouring equipment, triggers alarms, and can escalate within minutes into an unplanned shutdown, a fire, a hazardous release, or catastrophic failure. The detection and diagnosis of such abnormal situations has therefore been a central concern of process-safety engineering for decades [1].

### *1.1. The Limits of Manual Monitoring*

Today, a control-room operator watches a screen showing hundreds of process values. When an incident begins, the early warning signs are subtle a reading drifting slightly out of range, a vibration frequency shifting by a few hertz while the operator is simultaneously managing routine operations, acknowledging alarms on other units, and coordinating with field technicians. By the time a developing incident is obvious enough for a human to recognize and classify, valuable response time has already been lost. In high-risk scenarios such as hydrogen leaks, overheating furnaces, or runaway reactions, the difference between detecting an incident two minutes early and eight minutes late can be the difference between a controlled shutdown and an emergency evacuation. Detecting deviations from normal behaviour before they cross alarm thresholds is fundamentally an anomaly-detection problem [2]. The incident types targeted by this system include equipment failures (pump cavitation, bearing degradation, compressor surge, heat-exchanger fouling), process upsets (column flooding, reactor temperature excursions, pressure-relief events), chemical imbalances (incorrect feed ratios, catalyst deactivation), and abnormal states (unexpected shutdowns, cooling-water loss, steam-system failures).

### *1.2. Why Machine Learning Is Suited to This Problem*

Machine learning is well matched to this task for three reasons. First, sensor data contains patterns that precede incidents: the transition from normal to abnormal is gradual and has a characteristic shape that algorithms can learn to recognise. Second, a refinery generates far more data than any operator can monitor in real time, whereas learned models do not tire or lose attention. Third, historical incident logs provide labelled examples of the sensor behaviour before, during, and after each incident type precisely the data that supervised classifiers require [1], [2].

## 2. Proposed System Overview

The proposed Automated Incident Detection System sits between the raw sensor-data layer and the operator alarm console, continuously analysing incoming process data and emitting structured incident alerts. Its end-to-end architecture is shown in Fig. 1. The system performs two complementary functions that operate in sequence: an anomaly-detection layer that provides early warning, and an incident-classification layer that names and localises the developing event.

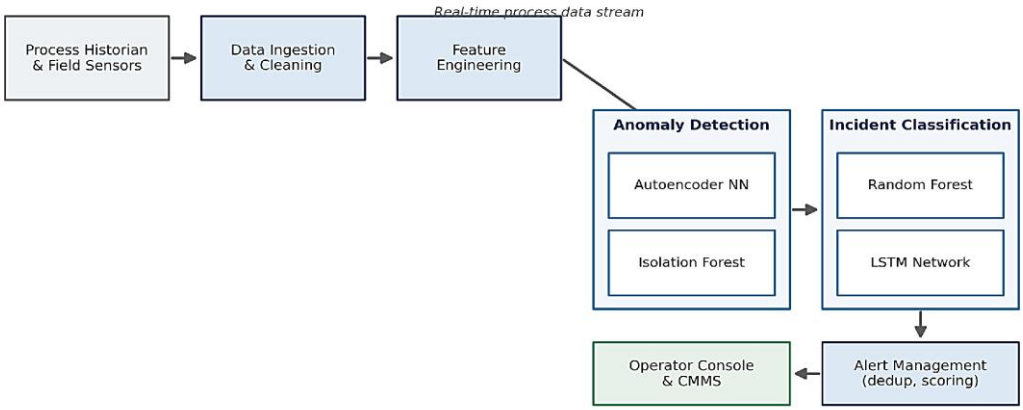


**Fig 1: End-to-End Architecture of the Automated Incident-Detection System.**

### *2.1. Anomaly-Detection Layer*

The anomaly-detection layer learns what normal operation looks like across all sensors and flags when current readings deviate significantly from that learned baseline. It is the early-warning stage: it does not state what is wrong, only that something is wrong.

### *2.2. Incident-Classification Layer*

Once an anomaly is flagged, the classification layer assigns it to a specific incident type based on the pattern of sensor deviations. This tells the operator not only that something is wrong, but what is wrong and where. Table I illustrates the two layers acting together on a developing pump-bearing fault: the system raises a classified, actionable alert minutes before the condition would have reached a conventional high-temperature alarm

**Table 1: Illustrative Bearing-Fault Detection Sequence**

| Time | Sensor Observation | System / Operator Response |
|---|---|---|
| Normal | Pump P-201: 1,450 RPM, 12 bar discharge, 48 A, bearing 62 °C; all values within learned normal range. | No action required; operation nominal. |
| T = 0 min | Bearing temperature rising (63 → 64 → 66 °C), still within alarm limits. | Human operator notices nothing unusual. |
| T = 2 min | Rising bearing temperature, vibration-frequency shift, and marginal current increase jointly deviate from the learned pattern. | ML anomaly detection flags an anomaly. |
| T = 2 min | Pattern matched to historical bearing-degradation examples. | Classifier output: probable bearing fault on P-201 (87% confidence); reduce load, inspect within 4 h. |
| Without ML | Operator is typically unaware until the high-temperature alarm trips. | Detection 15–20 min later; the bearing may already have failed. |

## 3. Data Requirements

The system depends on four types of data, each playing a distinct role, summarized in Table 2. Understanding the contribution of each source is essential before implementation begins.

**Table 2: Required Data Sources and Their Roles**

| Data Source | Role and Significance |
|---|---|
| Process Sensor Streams | Real-time measurements from all instruments (temperature, pressure, flow, level, vibration, current, chemical analysers). This is the primary model input. A plant with 2,000 sensors produces billions of data points per year and describes what is happening at the present moment. |
| Historical Incident Logs | Records of every past incident type, onset time, equipment involved, resolution, and confirmed root cause. These provide the labelled data used to train the incident-classification model; without them the system cannot distinguish one incident type from another. |
| Event Logs | Time-stamped operator actions, equipment state changes, valve positions, controller-mode changes, and maintenance activities. They provide context that pure sensor data cannot, distinguishing a deliberate adjustment from an unexpected fault. |
| Equipment State Data | Design operating range, age, maintenance history, and run-hours since last service for each asset. This lets the model judge whether a reading is abnormal for the equipment's current condition; an asset near end-of-service has a different normal profile than a newly serviced one. |

### *3.1. Data Preparation*

Raw refinery sensor data is imperfect: instruments drift, go offline for calibration, and produce communication gaps, while spikes, frozen values, and out-of-range readings must be detected and handled. Data cleaning is therefore the first implementation step. Before any model is trained, records are quality-filtered, timestamps are aligned to a common grid, and incident periods are separated from normal operation so that each model is trained on appropriate data.

## 4. Machine Learning Techniques

The system uses two categories of machine learning, each selected for the task it performs. This section explains each technique and the reasoning behind its inclusion.

### *4.1. Anomaly Detection*

Anomaly detection learns what normal looks like and identifies anything that deviates from it [2]. In a refinery, normal is not a single fixed state: it varies with operating mode, feed type, ambient temperature, and production rate. The detection model must learn this full range of normal variation and distinguish genuine deviations from routine fluctuation.

#### *4.1.1. Autoencoder Neural Network*

An autoencoder is a neural network trained to compress sensor data into a compact internal representation and reconstruct it back to its original form [3]. Trained on thousands of examples of normal operation, it learns to reconstruct normal data accurately. When an anomaly occurs, the readings no longer match the learned patterns and the reconstruction error the difference between the actual and reconstructed values rises sharply, as illustrated in Fig. 2. The system raises an anomaly flag when this error exceeds a threshold derived from the training data. A key advantage is that only normal operating data is required for training; no labelled incidents are needed [4]. Deep autoencoders have been shown to outperform conventional statistical monitoring on benchmark chemical-process data [12].

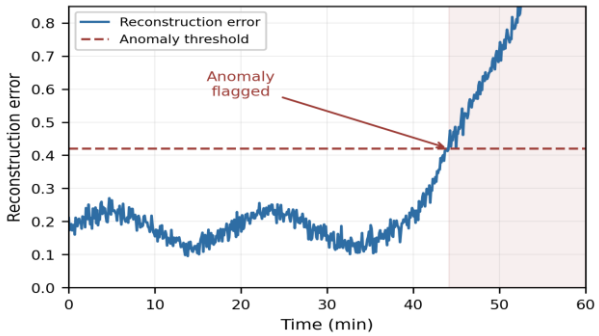


**Fig 2: Autoencoder Reconstruction Error Rises Above the Learned Threshold as an Incident Develops, Producing the Anomaly Signal**

#### *4.1.2. Isolation Forest*

An isolation forest is a faster, simpler algorithm that randomly partitions the data space [5]. Normal points, which cluster together, require many partitions to isolate; anomalous points, which sit far from the cluster, are isolated quickly. Each point receives an anomaly score based on how readily it is isolated. The isolation forest runs as a complementary detector alongside the autoencoder: when both models agree, alert confidence increases; when they disagree, the event is flagged for investigation rather than triggering an immediate alert, which reduces false positives.

### *4.2. Incident Classification*

Once an anomaly is detected, the classification layer assigns it to one of a predefined set of incident types. This is a supervised learning problem: the models are trained on historical examples of each incident type, labelled with the confirmed category.

#### *4.2.1. Random Forest Classifier*

A random forest is an ensemble of decision trees, each trained on a random subset of the data; the final class is decided by majority vote [6]. It suits this task because it handles the mixed feature types present (continuous sensor values, categorical equipment states, integer event counts), tolerates noisy or missing readings, and yields a natural confidence score from the fraction of trees voting for each class.

#### *4.2.2. Long Short-Term Memory Network*

Many incidents do not appear as a single abnormal reading but develop over minutes or hours as a sequence of changes across multiple sensors. A classifier that sees only the current snapshot cannot capture this. A long short-term memory (LSTM) network is a recurrent neural network designed to learn from sequences, maintaining a memory of recent inputs to inform its current output [7].

For classification, the LSTM receives a sliding window of recent sensor data 30 minutes by default and classifies the developing pattern, allowing it to recognise, for example, the joint signature of rising feed temperature, falling product quality, and increasing differential pressure as catalyst deactivation before any single sensor crosses an alarm threshold. LSTM networks have been applied successfully to multivariate sensor anomaly and fault detection [8].

**Table 3: Complementary Roles of the Two Classifiers**

| Aspect | Random Forest | LSTM Network |
|---|---|---|
| Principal strength | Fast and accurate for clear, point-in-time sensor deviations. | Captures incidents that develop gradually over minutes or hours. |
| Best-suited incidents | Pump trips, pressure-relief-valve lifts. | Catalyst deactivation, progressive fouling. |
| Model input | Flat feature vector (single snapshot). | 30-minute sliding window of sensor sequences. |
| Confidence measure | Fraction of trees voting for each class. | Softmax probability per class. |
| Role on disagreement | Raises confidence when it agrees with the LSTM. | Disagreement flags the event for operator review. |

### *4.3. Combining the Classifiers*

The random forest and LSTM are complementary, as summarised in Table 3. Both run in parallel for every flagged anomaly. If they agree on the incident type and their combined confidence is high, a high-confidence alert is issued; if they agree at lower confidence, a medium-confidence alert is issued; if they disagree, a review-required alert is issued showing both classifications. This decision logic is shown in Fig. 3. The dual-model approach consistently outperforms either model alone.

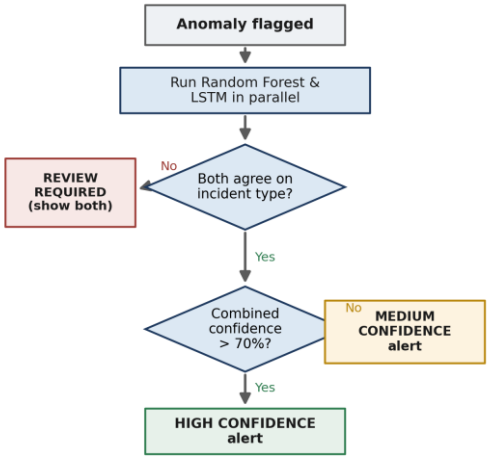


**Fig 3: Decision Logic Combining the Two Classifiers into a Single Graded Alert**

## 5. Implementation Methodology

This section provides a practical implementation guide. It assumes access to a process historian (OSIsoft PI, Honeywell Uniformance, or similar), historical incident records, and a Python-based data-science environment.

### *5.1. Step 1: Data Collection and Cleaning*

Extract at least 24 months of sensor data for all relevant tags, preferring one-minute averaged data as the base resolution and reserving raw one-second data for high-frequency vibration analysis. Remove bad-quality data readings flagged by the historian, values outside the instrument's physical range (e.g., a temperature of –999 indicating a communication failure), and periods of planned shutdown or turnaround. Identify frozen values (a sensor reporting an identical value for more than 30 consecutive minutes is likely stuck) and exclude them. Align all sensors to a common one-minute grid, using forward-fill for occasional missing samples and mean imputation for gaps shorter than ten minutes; treat longer gaps as outages and exclude them from sequence models. Finally, associate each incident record type, start and end times, equipment tags, and confirmed root cause with the sensor readings from 60 minutes before onset to the incident end. The pre-incident window is especially valuable because it contains the early-warning patterns the models must learn. A minimum of roughly 80 labelled incidents, with at least ten examples per type, is recommended for reliable generalisation.

### *5.2. Step 2: Feature Engineering*

Raw readings alone are not the most effective model input. Rolling statistics (mean, standard deviation, minimum, and maximum over 5-, 15-, and 30-minute windows) reveal developing trends more clearly than instantaneous values. Rate-of-change (first-derivative) features are often more diagnostic than absolute values. Cross-sensor ratios for example, the compressor discharge-to-suction pressure ratio capture process relationships that single sensors cannot, and are best identified with process-engineering domain knowledge. Equipment run-hours since last maintenance and a categorical operating-mode indicator (normal, reduced rate, startup, shutdown) complete the feature set.

### *5.3. Step 3: Training the Anomaly-Detection Models*

Split the clean normal-operation data 80/20 into training and validation sets using a time-based split never random, which would leak information in time-series data. Train the

autoencoder with an encoder that compresses the sensor vector to a 10–20-dimensional bottleneck and a decoder that reconstructs it, minimising mean-squared reconstruction error on normal data only, and set the anomaly threshold at the 99th percentile of the validation-error distribution [4]. Train the isolation forest on the same data, tuning the contamination rate (starting at 1%) against the observed false-positive rate. Evaluate both on a held-out test set containing normal and labelled-incident periods, targeting a detection rate above 90% and a false-positive rate below 5%.

### *5.4. Step 4: Training the Incident-Classification Models*

Represent each labelled incident window as a flat feature vector for the random forest and as a 30-step sequence for the LSTM. Split the dataset 70/15/15 (training/validation/test) with time-based splitting, ensuring every incident type appears in all splits. Because some incident types are far rarer than others, apply Synthetic Minority Over-sampling (SMOTE) to the training set to balance class representation [9]. Train the random forest with 200 trees, tuning maximum depth and minimum samples per leaf by cross-validation and evaluating with the weighted F1-score. Train the LSTM with two recurrent layers (64 and 32 units), a dropout layer (rate 0.3) to limit overfitting [11], and a softmax output, using categorical cross-entropy loss with the Adam optimiser [10] and early stopping when validation loss stops improving. For each flagged anomaly, run both classifiers and combine their outputs using the confidence logic of Fig. 3.

### *5.5. Step 5: Real-Time Deployment*

Once validated, the models are connected to live data through five services. A data-ingestion layer subscribes to the historian's real-time interface (PI Web API, REST, or OPC-UA [13]) and pulls the latest values every 60 seconds. A preprocessing service applies exactly the same cleaning, feature engineering, and scaling parameters used in training. An inference service runs the autoencoder, isolation forest, and classifiers on each batch; it should be containerised and deployable on a plant data centre or an industrial edge node, with inference latency below five seconds per cycle. An alert-management service deduplicates repeated flags and formats alerts, and an operator-console integration delivers each alert incident type, equipment tag, confidence level, triggering sensors, and recommended action through the plant's existing alarm interface.

### *5.6. Step 6: Validation and Continuous Improvement*

Before going live, run the system in shadow mode for about four weeks: it processes live data and generates alerts that are not shown to operators, allowing detection rate, false-positive rate, and average lead time to be measured against actual incidents. After go-live, collect operator feedback on every alert and log it systematically. Retrain the models quarterly using accumulated incident data, since operating conditions, equipment, and failure patterns evolve. Monitor performance continuously, and trigger an immediate retraining cycle if the weighted F1-score falls more than five percentage points below baseline.

## 6. Expected Results and Benefits

A fully deployed system, trained on adequate historical data, is expected to deliver the practical improvements summarised in Table 4.

**Table 4: Expected Operational and Safety Benefits**

| Benefit | Explanation |
|---|---|
| Earlier detection | Detects developing incidents 10–25 minutes before they reach alarm thresholds, giving operators more time to intervene before equipment damage or process upset occurs. |
| Fewer missed incidents | Monitors all sensors continuously and simultaneously, without attention gaps, catching subtle faults that manual monitoring misses during busy periods or shift changes. |
| Faster classification | Delivers the incident type immediately with supporting evidence, replacing 5–10 minutes of manual diagnosis a critical saving during fast-moving events. |
| Reduced unplanned shutdowns | Enables planned corrective maintenance before failure occurs, converting unplanned shutdowns into scheduled maintenance windows. |
| Sustained operator focus | Lets operators manage the plant rather than continuously scan dashboards for anomalies. |
| Continuous learning | Each new incident enriches the training set, so accuracy improves over time for the incident types specific to each plant. |

## 7. Challenges and Mitigations

Several obstacles commonly arise when deploying ML-based incident detection in refineries. Data quality is often poor in older plants; the recommended approach is to begin with the highest-quality sensors and best-documented incidents and to expand scope as quality improves. Insufficient incident history some incident types may have occurred only two or three times can be mitigated with physics-based process simulation to generate synthetic sensor profiles that supplement scarce real examples. Operator trust must be earned through transparency (showing exactly which sensors triggered each flag) and an initial advisory-only deployment period. Changing operating conditions, such as a new crude type, produce data outside the training distribution and more false positives; flagging known mode changes to the system and temporarily widening anomaly thresholds during transitions addresses this, while robust anomaly-detection design limits spurious alerts [2]. Finally, integration with legacy distributed control systems is best handled by deploying lightweight collection agents at the historian level and using standard industrial protocols such as OPC-UA rather than attempting direct control-system integration [13].

## 8. Conclusion

This paper has presented a complete framework for automated incident detection in refinery operations using machine learning. Manual monitoring cannot match the speed, continuity, or sensitivity required to detect developing incidents before they escalate in a high-risk, high-complexity environment. The proposed system addresses this by using data that refineries already generate sensor streams, incident logs, equipment records, and event logs to detect anomalies early and classify them accurately. It combines two anomaly-detection techniques (autoencoder and isolation forest) with two classification techniques (random forest and LSTM), each chosen for a specific role: the autoencoder learns complex multi-sensor normal patterns; the isolation forest provides a fast, robust complementary signal; the random forest handles point-in-time classification reliably; and the LSTM captures incidents that unfold over time. The implementation roadmap of Section V provides a practical path from raw-data extraction to live operator-console integration, and the expected outcomes earlier detection, fewer missed incidents, faster classification, and reduced unplanned shutdowns—represent meaningful gains in both safety and operational continuity. The path forward is to gather and clean the data, train on a well-labelled historical dataset, validate thoroughly before deployment, and commit to continuous improvement as new incidents are recorded.

## References


[1] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods," Comput. Chem. Eng., vol. 27, no. 3, pp. 293–311, 2003.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, Jul. 2009.

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[4] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in Proc. MLSDA 2nd Workshop Mach. Learn. Sensory Data Anal., Gold Coast, Australia, 2014, pp. 4–11.

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

[6] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[8] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in Proc. 23rd Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn. (ESANN), Bruges, Belgium, 2015, pp. 89–94.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. 3rd Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, 2015.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, 2014.

[12] Z. Xiao, A. Kordon, and S. Sen, "Fault detection and diagnosis in Tennessee Eastman process with deep autoencoder," in Proc. Annu. Conf. PHM Soc., vol. 15, no. 1, 2023.

[13] OPC Foundation, OPC Unified Architecture, IEC 62541, Int. Electrotech. Comm., Geneva, Switzerland, 2020.