

4th International Conference on Electronics, Engineering, Computer Science and Applied Development (EESD 2026)

Article

Intelligent Control and Collaboration Techniques for Industrial Robots

Bosi Zhao^{1,*}

¹ Chengdu Jincheng College, Chengdu, China

* Correspondence: Bosi Zhao, Chengdu Jincheng College, Chengdu, China

Abstract: Industrial robot collaboration requires precise decisions about whether an operation has completed normally before another robot continues the sequential process. This paper investigates a novel execution supervisor that combines force and torque classification with an explicit permission condition for a shared transfer zone. Thirty statistical signal features and five task indicators support a robust ensemble classifier, while a training-only threshold procedure dynamically adjusts the balance between abnormal release and unnecessary holding. The evaluation utilizes 463 publicly available robot execution records containing 251 distinct sensor trajectories. Identical trajectories are systematically grouped across task files throughout repeated outer evaluation and inner threshold selection. The statistical random forest model achieves a mean balanced accuracy of 0.9669 and a macro F1 score of 0.9687 across five grouped evaluation repetitions. Furthermore, cost-aware threshold selection effectively reduces the abnormal-release rate from 1.50% to 1.14%, while increasing the normal-hold rate from 5.12% to 9.61%. The resulting mean weighted penalty changes only slightly, from 0.0683 to 0.0678. Tests on excluded task types yield balanced accuracy between 0.5250 and 0.9776, demonstrating limited transferability of the default threshold. A Boolean interlock is checked independently of the learned decision. These results strongly support a bounded design for post-observation execution supervision and sequential robot coordination, though physical contact safety and production throughput remain outside the current experimental scope.

Keywords: industrial robots; intelligent control; robot collaboration; execution monitoring; force sensing

Received: 04 August 2026

Revised: 18 September 2026

Accepted: 28 September 2026

Published: 03 October 2026



Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Industrial robots often operate in a sequence of connected tasks. One manipulator acquires or positions a component, and another performs transfer, inspection, or manufacturing-related handling. A successful motion command does not necessarily establish that the component is ready for the next operation. Contact with a fixture, an obstructed approach, or a displaced part can change the condition of the workpiece while the robot program continues. Collaboration therefore depends on how observations become permission to proceed. Research on industrial human–robot collaboration identifies safety, interfaces, and application integration as connected concerns. The broader collaboration literature also distinguishes physical interaction from coordination and shared task execution [1].

Intelligent handling combines perception with decisions about action [2]. Sensing and computation influence the ability to grasp and handle objects in less structured environments. Reinforcement learning offers tools for adapting robot behavior, although representations, prior knowledge, and sample requirements affect practical application.

Many industrial systems also retain established motion controllers. Impedance control, for example, addresses the dynamic relation between a manipulator and its environment. Learning can be introduced at a higher supervisory level, where its output determines whether a completed action is accepted, repeated, or referred for inspection.

This division of responsibilities is relevant when training data are limited. A force and torque record may contain enough information to distinguish a normal execution from an abnormal event, yet provide no robot geometry, joint state, or recovery trajectory. Earlier work on flexible manufacturing-integrated monitoring, diagnosis, and learning within an execution supervisor provides a useful starting point for connecting recognition with task control. It also requires careful interpretation of the observation time. A record collected after an exception can support diagnosis of that exception; it does not establish advance warning before the physical event [3].

Anomaly monitoring illustrates why the meaning of a learned output matters. Detection, isolation, and identification are separate problems, with different sensing and modeling requirements. A binary abnormality score cannot specify a contact location or certify an acceptable contact force. Similar distinctions arise in collaboration. Assigning tasks among robots requires information about resource availability and task requirements. A classifier that recognizes an abnormal grasp does not solve the assignment problem. Its useful contribution is more specific: it can withhold permission for a dependent action until the workpiece state has been resolved [4].

The present study examines this supervisory interface. A compact signal representation supplies a classifier with information from a completed observation window and the current operation type. The classifier produces an abnormality score. A threshold converts that score into a release or hold decision, and a separate logical condition combines release with transfer-zone clearance. This separation follows the use of explicit states and enabling conditions in discrete-event representations such as Petri nets. Robot-to-robot coordination is considered through sequential use of a shared fixture; simultaneous cooperative carrying and human contact are outside the implemented model [5].

The empirical analysis uses the public Robot Execution Failures collection. A preliminary audit identifies repeated complete sensor trajectories across its five task files [6]. The evaluation therefore keeps identical trajectories together in both training and validation splits. Four classifiers or representations are compared under the same outer partitions. A nested procedure selects a decision threshold using training observations alone, and an additional evaluation excludes each operation type together with all matching sensor traces from training.

The study addresses three questions. First, how well can a small statistical representation support abnormal execution recognition under duplicate-aware evaluation? Second, how does a decision threshold change abnormal releases and unnecessary holds? Third, how far do the resulting scores transfer to an operation type absent from training? The contribution is an executable connection between public-data diagnosis and a clearly specified continuation rule, accompanied by an audit of repeated records [7]. The reported evidence concerns recognition and supervisory decisions. It provides a basis for subsequent integration with robot motion controllers, task planners, and independently validated protective functions.

2. Related Work

2.1. Learning and Motion Control

Robot learning from expert examples extracts task information from provided instances. The observed data in this study differs: each instance captures the sensor response linked to an execution outcome, without including corrective actions or expert-defined continuation strategies [7]. Consequently, the learned component estimates execution conditions, while the release policy is defined independently. This separation

ensures that diagnostic labels are not misinterpreted as comprehensive recovery instructions.

Classical robotic control methods provide complementary capabilities. Hybrid position–force control applies appropriate motion and interaction specifications within a task-oriented coordinate framework. Operational-space control defines motion and force behavior directly at the end-effector level. These approaches rely on detailed knowledge of robot dynamics and task geometry, which is unavailable in the publicly accessible sensor dataset. Within the proposed architecture, they serve as the foundational execution layer. Evaluation isolates this layer from the learning component and focuses exclusively on the supervisory decisions made following observation acquisition.

Control barrier functions provide a formal method for maintaining state constraints via constrained control optimization. Their theoretical guarantees require precise system dynamics, a well-defined safe region, and solvability of the associated control condition. A learned abnormality score does not supply these prerequisites. In environments shared with humans, speed and separation monitoring further depend on response latency, stopping characteristics, and sensor uncertainty [1]. These considerations necessitate a narrow interpretation of the continuation gate: it represents a process-level decision that must be embedded within a certified safety system prior to physical implementation.

2.2. Coordination and Execution Supervision

Multi-robot task allocation involves problems characterized by varying degrees of task interdependence and resource limitations. Earlier classification frameworks have been extended to incorporate interdependent performance outcomes and operational boundaries [1]. In a collaborative manufacturing process, an operation may require both the completion of its preceding step and the readiness of a supporting device. The present work adopts a fixed sequence for two robots. It focuses on evaluating whether the outcome of a prior operation is acceptable for continuation, rather than optimizing assignment across a larger group. This simplification ensures that diagnostic outputs correspond directly to a single, observable decision point.

Supervisory control of discrete-event systems provides a formal method for restricting event sequences to achieve desired behavioral properties [8]. A handoff procedure can be modeled using states such as observation, release, holding, placement, clearance, and retrieval. The controller must differentiate between an observation meeting acceptance criteria and a resource being operationally ready—either condition may inhibit progression to the next event. We employ an explicit logical conjunction of these conditions and verify its truth table independently of any classification scheme. This approach validates the implemented enabling rule within its specified Boolean framework, while deferring analysis of unmodeled sensor anomalies and physical dynamics.

Robot Operating System 2 (ROS 2) offers a modular software architecture supporting sensing, planning, and execution modules. Such middleware facilitates transmission of diagnostic metrics and task-state information; however, successful message delivery does not guarantee correct semantic interpretation [9]. Integration must still define which observation corresponds to which operation and precisely when an authorization is exercised. Our implementation constitutes an offline logical evaluation with a formal interface. It does not measure ROS 2 communication latency nor assert compliance with industrial real-time communication standards.

2.3. Signal Classification and Evaluation

Random forests integrate multiple randomized decision trees and effectively capture nonlinear relationships among sensor features. Randomized tree ensembles increase stochasticity in both feature selection and split point determination. Both approaches serve as practical baseline methods for short multichannel time-series records. Their output scores derive from the training data distribution and ensemble architecture; therefore, numerical outputs must undergo rigorous validation before being interpreted as calibrated event probabilities. A regularized logistic classifier provides a

complementary linear benchmark, where the decision boundary remains linear in the transformed feature space.

Time-series feature extraction reveals patterns in amplitude, variability, and temporal evolution using statistics that are invariant to class labels [10]. The Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh) framework offers a principled methodology for automated feature generation and selection. Alternative representation strategies include convolutional approaches. The Random Convolutional Kernel Transform (ROCKET) applies a large set of fixed random convolutional kernels for time-series classification, while MiniRocket introduces a computationally optimized variant. These developments underscore the importance of representation design alongside classifier selection. Our evaluation relies on raw signal samples and a fixed thirty-feature summary, without incorporating learned convolutional representations or asserting superiority over broader time-series modeling frameworks.

Probability calibration and decision performance are related yet distinct aspects of model evaluation. Comparative studies have examined the reliability of probability estimates produced by supervised learning algorithms. A model yielding well-calibrated ranking scores may still require threshold adjustment when operational requirements or cost asymmetries change. Threshold selection also constitutes an implicit model-tuning step. To avoid overoptimistic performance estimation, thresholds are determined independently within each outer training fold, with held-out outer test observations reserved exclusively for final evaluation [2].

Precision–recall analysis is particularly informative when class imbalance influences the interpretation of positive predictions, whereas receiver operating characteristic (ROC) analysis characterizes classification separation across varying decision thresholds [5]. Neither curve alone quantifies the number of abnormal tasks permitted under a given operational rule. Accordingly, we report ranking metrics jointly with class-specific release and hold rates. Research on concept drift highlights shifts in the statistical relationship between inputs and targets over time. Our excluded-task test assesses limited distributional variation across operation types, without modeling chronological drift in industrial settings.

Digital-twin research focuses on establishing robust alignment between computational models and their physical counterparts, emphasizing the fidelity and timeliness of information exchange. The present study constitutes a computational replay experiment conducted on archived observational data, with no real-time synchronization to a physical production cell [11]. Recent assessments of manufacturing collaboration highlight persistent gaps between theoretical capabilities and practical implementation. These insights reinforce the need for an evaluation framework that explicitly distinguishes classification accuracy, process permission logic, and physical execution—rather than conflating them into a single undifferentiated metric of intelligent collaboration.

3. Methodology

3.1. Scope and Supervisory Architecture

The architecture comprises a robot execution layer, an observation processor, a learned diagnosis module, and a collaboration supervisor. Figure 1 illustrates the information flow and sequential transfer arrangement. Robot A performs an operation linked to an observation checkpoint. A classifier assesses the available force and torque window. A normal release permits continuation of the process once the shared fixture is confirmed clear. A hold triggers redirection of the operation to inspection or recovery handling, whose physical implementation falls outside the scope of this experiment. Robot B retrieves the workpiece after Robot A has vacated the transfer zone [12].

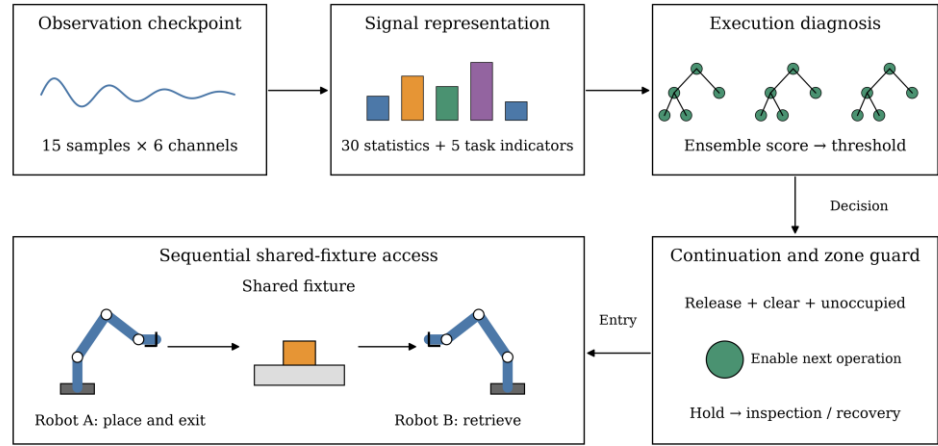


Figure 1. Supervisory Control Architecture and Sequential Robot Collaboration. Robot Arms, a Sensor Trace, and a Tree Ensemble Illustrate the Corresponding Modules. The Shared Fixture Is Used Sequentially.

The learned score does not directly regulate joint torque or end-effector velocity; instead, it determines a process-level outcome following the observation window. An operation placed on hold remains unresolved until an external inspection or recovery decision is provided. The study does not assign an automatic success rate or duration to such recovery actions [13]. This design ensures that every reported release or hold corresponds unambiguously to a measured record and a classifier output. No additional sensor measurements are generated by assuming successful completion of unobserved recovery actions.

The shared-zone rule relies on a centralized, atomic decision regarding occupancy. The clearance input indicates that the predecessor robot has fully exited the relevant zone [14]. The occupancy input signals that another operation currently occupies that zone. These are treated as distinct logical variables because real-world implementations may acquire readiness and resource status through separate interfaces. During offline testing, their values are explicitly supplied. The classifier contributes solely the process permission. This Boolean representation does not imply any conclusion about the reliability of physical clearance sensors.

3.2. Records and Feature Representation

Each episode i is represented by a matrix X_i containing $T = 15$ samples from $C = 6$ channels. The channel order is the three force components followed by the three torque components. An operation identifier specifies one of the five learning problems in the collection. The binary target y_i is assigned zero for the labels normal and ok, and one for all other labels. This mapping indicates whether the recorded execution state is acceptable for continuation [2]. It consolidates multiple failure categories and does not distinguish their individual recovery requirements.

For each channel c , the representation includes its mean, population standard deviation, minimum, maximum, and mean absolute first difference. Equations (1) and (2) define the average and change descriptors. The standard deviation uses denominator T , consistent with the implemented feature calculation. Minimum and maximum are computed over the full fifteen-sample window. Collectively, these five statistics yield thirty signal features. Five binary task indicators are concatenated, resulting in a thirty-five-dimensional input vector. In contrast, the raw comparison flattens all ninety sensor values and appends the same task indicators, producing a ninety-five-dimensional input [7].

$$\mu_{i,c} = \frac{1}{T} \sum_{t=1}^T x_{i,t,c}$$

$$\sigma_{i,c} = \sqrt{\frac{1}{T} \sum_{t=1}^T (x_{i,t,c} - \mu_{i,c})^2} \quad (1)$$

$$d_{i,c} = \frac{1}{T-1} \sum_{t=2}^T |x_{i,t,c} - x_{i,t-1,c}| \quad (2)$$

The entry $x_{i,t,c}$ denotes sample t from channel c of episode i . The symbols $\mu_{i,c}$, $\sigma_{i,c}$, and $d_{i,c}$ represent its channel-wise mean, standard deviation, and mean absolute first difference, respectively [3].

The means and extrema retain both direction and amplitude of each recorded channel. Standard deviation quantifies within-window variability, while the mean absolute first difference captures local temporal changes between adjacent samples. These descriptors are computed independently per episode and do not rely on class labels. Feature extraction can therefore be performed prior to data partitioning without information leakage across episodes. The logistic regression pipeline estimates centering and scaling parameters exclusively from the current training partition [15]. Tree-based ensembles operate directly on the extracted features and receive no global normalization derived from test observations.

Stored values retain their original numerical conventions. No force-to-torque norm or universal physical threshold is applied, as the collection does not define a shared scale for such aggregation in this analysis. The model learns channel-specific decision boundaries. Deployment requires calibrated sensors, explicitly declared units, and a consistent coordinate frame to align these feature definitions with live robotic systems. The operation identifier must be supplied by the execution state itself, rather than inferred from outcome labels.

3.3. Duplicate-Aware Model Development

A complete sensor matrix is converted to a deterministic byte representation and assigned a Secure Hash Algorithm 256 (SHA-256) digest. Records with identical matrices receive the same group identifier, including when they occur in different task files. Group identity excludes the outcome and operation labels. All records within a group remain on the same side of every training and evaluation split. The collection contains 251 distinct trajectory groups, and no group contains contradictory binary labels. File-level provenance records retain download addresses and file digests separately from these trajectory identities.

Five outer folds are generated for each of five random seeds, numbered zero through four. The split procedure balances the joint task and binary-label strata while preserving complete trajectory groups. Group preservation takes precedence when exact stratification is infeasible. Each record receives one out-of-fold score per repetition [7]. Metrics are computed over all 463 out-of-fold records in that repetition and then averaged across repetitions. Repeated partitions assess sensitivity to the finite size of this collection; they do not constitute five independent robot datasets.

Four fixed configurations are evaluated: random forest (RF) on raw inputs, logistic regression (LR) on statistical inputs, RF on statistical inputs, and randomized trees (RT) on statistical inputs. The tree ensembles contain 300 trees, require at least two training records per leaf, and consider the square root of the input dimension when selecting candidate features. Their remaining settings follow the recorded implementation. Logistic regression uses inverse regularization strength $C_{\text{reg}} = 1$ and a maximum of 2,000 optimization iterations [9]. These settings are fixed before outer scoring, with no hyperparameter search conducted.

For the statistical random forest, each tree supplies an abnormal-class proportion from the reached leaf. Equation (3) averages those values over $B = 300$ trees to produce score p_i . The score lies between zero and one. The primary diagnostic comparison uses threshold 0.5 for every model. This establishes a common operating convention, while area-based metrics evaluate score ordering. The forest configuration is selected as the supervisory reference before threshold comparisons, so the threshold experiment changes the decision rule while keeping the outer score generator fixed.

$$p_i = \frac{1}{B} \sum_{b=1}^B q_b(f_i), \quad 0 \leq p_i \leq 1 \quad (3)$$

Here f_i denotes the feature vector of episode i , and q_b denotes the abnormal-class proportion estimated by tree b at its reached leaf.

3.4. Decision Threshold and Collaboration Condition

The continuation decision involves two types of errors: a false release, which occurs when an abnormal episode is permitted to continue, and a false hold, which occurs when a normal episode is unnecessarily withheld. Relative penalties of five and one are assigned to these errors, respectively. These values reflect analytical design choices rather than empirically measured operational costs or safety impact assessments. Equation (4) selects an optimal decision threshold from a fixed grid spanning 0.1 to 0.9 in increments of 0.1. The indicator function evaluates to one if its condition holds, and zero otherwise.

$$L(\tau) = \frac{1}{N_v} \sum_{I \in V} [5 \cdot 1[y_I = 1 \wedge p_I < \tau] + 1[y_I = 0 \wedge p_I \geq \tau]]$$

$$\tau^* = \arg \min_{\tau \in G} L(\tau) \quad (4)$$

The set V comprises inner validation observations, N_v denotes its cardinality, and G represents the discrete threshold grid [16]. Each score contributing to the objective function is computed without training on the corresponding observation or any other member of its trajectory group.

For each outer training partition, three inner grouped folds yield validation scores for every training record. The threshold is determined by minimizing the mean weighted error across those inner scores. In cases of tie, the smaller threshold is selected, as the grid is traversed in ascending order. A statistical forest trained on the full outer training partition then produces scores for the outer test records. The selected threshold is applied unchanged to these test scores, without incorporating their labels or score distribution for further adjustment. Figure 2 illustrates the separation between threshold development and application.

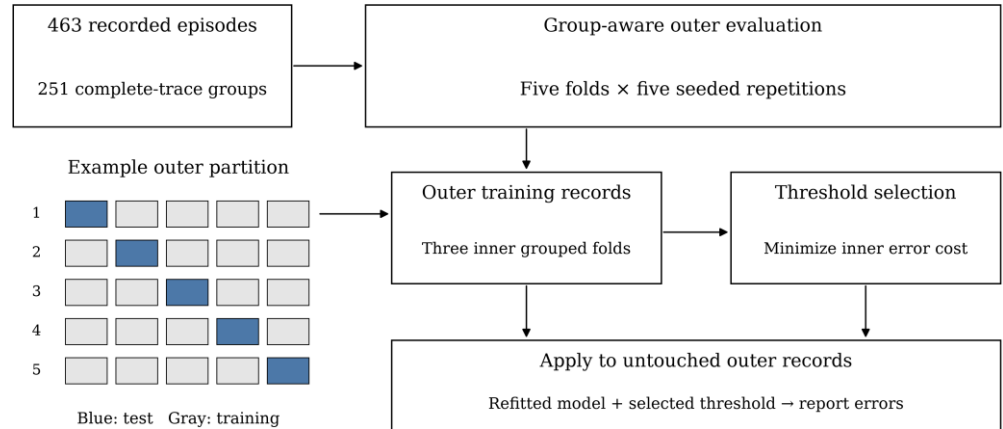


Figure 2. Grouped Development and Application. Identical Complete Traces Share One Group. Inner Validation Selects a Threshold; the Untouched Outer Fold Evaluates Both the Score and the Resulting Continuation Decision.

A score strictly below its assigned threshold grants process permission a_i ; equality results in a hold [17]. Equation (5) integrates this permission with three Boolean zone inputs: o indicates occupancy by another operation, g indicates predecessor clearance, and e indicates permission to enter. Logical negation and conjunction are evaluated prior to any modeled entry logic. Entry is permitted only when all three conditions are satisfied — process permission, an unoccupied zone, and predecessor clearance. The evaluation systematically covers all eight possible combinations of the three binary inputs.

$$a_i = 1[p_i < \tau^*], \quad e = a_i \wedge \neg o \wedge g \quad (5)$$

The truth-table verification confirms functional equivalence between the specified logical condition and its software implementation. Ensuring exclusive access during

physical motion would further require accurate state updates, atomic ownership transitions, and reliable clearance feedback. Additionally, any permission must be explicitly bound to a unique operation identifier before being acted upon. These integration constraints are defined as implementation boundaries. This experiment evaluates only the release decision and a stateless enabling condition, without asserting verification of a complete distributed cell control system.

3.5. Evaluation Measures and Transfer Test

The abnormal class is treated as the positive class. True positives correspond to abnormal episodes correctly retained by the supervisor; false negatives correspond to abnormal episodes incorrectly released. True negatives correspond to normal episodes correctly released, and false positives correspond to normal episodes incorrectly retained. Balanced accuracy is computed as the arithmetic mean of the recall values for both classes. Equation (6) defines this metric along with the weighted penalty used in decision evaluation. Macro F1 is the unweighted average of the F1 scores for the normal and abnormal classes, ensuring equal contribution from both classes regardless of their differing sample sizes.

$$BA = \frac{1}{2} \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right), \quad J = \frac{5FN + FP}{N} \quad (6)$$

The symbols TP, FN, TN, and FP denote the respective counts of true positives, false negatives, true negatives, and false positives; N represents their total sum. BA denotes balanced accuracy, and J denotes the mean weighted penalty [18].

The receiver operating characteristic area under the curve, average precision, and Brier score are computed directly from the unthresholded model outputs. The first two metrics assess the model's ability to distinguish between classes, while the Brier score quantifies the mean squared difference between predicted scores and binary ground-truth labels. No post-hoc calibration is applied. The abnormal-release rate is the ratio of false releases to all abnormal records. The normal-hold rate is the ratio of false holds to all normal records. The overall release fraction uses the total number of records as its denominator. These denominators remain consistent across all tables and figures.

A supplementary weighting scheme assigns equal aggregate weight to each distinct trajectory group: each record within a group of size m receives weight $1/m$, while training proceeds unchanged. This adjustment verifies whether repeated records unduly influence reported performance metrics. In the excluded-task test, one complete task type is fully withheld from training, and all training records whose sensor matrices match any held-out record are also removed. A single statistical forest is then trained using the fixed configuration and evaluated at a threshold of 0.5. This test evaluates generalization across operation types under strict trace-level exclusion—not transfer to novel robotic platforms or industrial environments.

4. Experiments and Results

4.1. Public Data and Integrity Checks

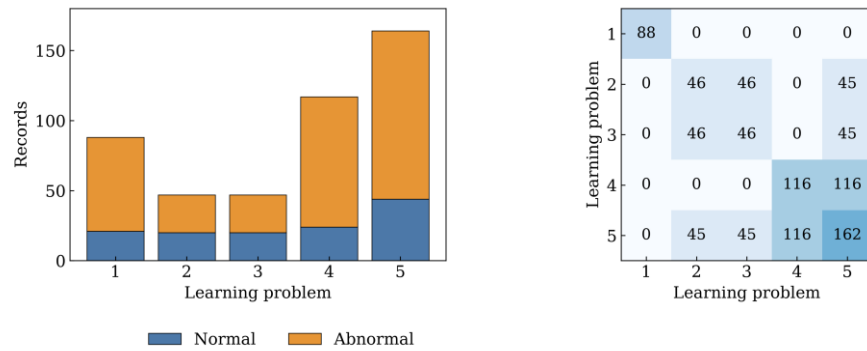
The experiment downloads the five plain-text files from the University of California, Irvine (UCI) Machine Learning Repository. The collection is associated with research on learning and supervision of automated systems. Its dataset identifier is 10.24432/C5M89N, and the public record specifies a Creative Commons Attribution 4.0 license. Each episode contains a class label followed by fifteen rows of six numerical measurements [13, 19]. The provider describes a 315 ms observation window collected after failure detection. The present classification therefore uses a completed diagnostic observation rather than a pre-failure forecasting interval.

Table 1 lists the task composition. The files contain 88, 47, 47, 117, and 164 episodes, respectively. Mapping normal and ok to the acceptable class gives 129 normal and 334 abnormal records. Every parsed matrix has the expected shape and finite entries, and no imputation is required. The original multiclass labels are retained in the downloaded files. Binary relabeling is performed only for the supervisory analysis, where the immediate decision is release or hold rather than selection of a particular recovery action.

Table 1. Public Episode Counts and Binary Targets. Unique Counts in Individual Files Overlap Across Files and Must Not Be Summed as Independent Trajectories.

Learning problem	Records	Normal	Abnormal	Unique traces
1	88	21	67	88
2	47	20	27	46
3	47	20	27	46
4	117	24	93	116
5	164	44	120	162
All files	463	129	334	251

The integrity audit finds 251 unique complete sensor trajectories across all files. Learning problems 2 and 3 share the same 46 unique trajectories. Forty-five of these also occur in problem 5. All 116 unique trajectories in problem 4 occur in problem 5. Problem 1 contributes 88 trajectories that do not occur in the other files. These overlaps make a naive record-level random split unsuitable for estimating recognition on unseen traces. Figure 3 shows the pairwise overlaps and record counts. Every reported main result uses the grouped partitioning described in Section 3.

**Figure 3.** Dataset Composition and Exact Trace Overlap. Left, Normal and Abnormal Record Counts. Right, the Number of Identical Complete Trajectories Shared by Each Pair of Task Files; Diagonal Values Are Within-File Unique Counts.

The retained record distribution determines the primary estimand: performance on records drawn with the frequencies present in the released collection [20]. It does not estimate a production failure rate. The abnormal class is common because this is an execution-failure collection. Task identifiers and duplication patterns also limit the number of independent operating conditions. Group-aware evaluation protects against exact trace leakage, while the supplementary group-weighted analysis changes the evaluation weighting. Neither procedure recovers unrecorded variation in payload, system hardware, sensor calibration, or operational timing.

4.2. Experimental Configuration

Table 2 records the fixed model and validation settings. Each outer repetition yields a complete set of 463 out-of-fold scores for all four configurations. Across five repetitions, this results in 2,315 scored record appearances per configuration, all drawn from the same underlying dataset. The twenty-five outer model fits per configuration employ identical group partitions. Threshold selection for the statistical forest incorporates three inner fits within each outer training partition. Random seeds and per-record fold assignments are preserved alongside the scores to ensure full reproducibility of every decision.

Table 2. Fixed Experimental Settings. Model Parameters Are Unchanged Across Outer Evaluations; Only the Supervisory Threshold Is Selected in Inner Validation.

Item	Setting
------	---------

Signal input	15 samples $\times$ 6 channels per record
Feature dimensions	Raw: 95; statistical: 35, including 5 task indicators
Tree ensembles	300 trees; minimum leaf size 2; square-root feature sampling
Logistic regression	Training-only standardization; inverse regularization 1
Outer evaluation	5 grouped folds $\times$ 5 repetitions; seeds 0–4
Inner selection	3 grouped folds; thresholds 0.1–0.9, step 0.1
Decision penalty	False release: 5; false hold: 1
Software	Python 3.12; NumPy 2.5.2; scikit-learn 1.9.0

The reported averages are derived from five repetition-level metrics. Where variability is shown, whiskers indicate the minimum and maximum values across repetitions; these represent descriptive ranges, not confidence intervals based on independent samples [7, 21]. The analysis explicitly treats each episode as a single predictive unit and does not disaggregate the fifteen time points within an episode into separate classification instances. Such disaggregation would introduce temporal leakage between training and testing sets and alter the fundamental prediction objective. All data preprocessing and model evaluations are performed using the retained scripts; numerical tables are generated directly from their saved outputs.

Observation duration and processing delay are distinct quantities. The archived window specifies the temporal span over which measurements were acquired relative to failure detection, but it does not quantify classifier inference time, message transmission latency, actuator response time, or supervisory transition duration on an industrial controller. Consequently, the experiment evaluates decisions only after the full observation window has been completed. A real-time implementation would require empirical measurement of each sequential delay on the target hardware and comparison of their cumulative duration against the system's task deadline. The current record-level results do not establish that deadline nor provide a worst-case response bound.

4.3. Recognition Performance

Table 3 compares the four configurations at threshold 0.5. The statistical random forest achieves balanced accuracy 0.9669 and macro F1 0.9687, compared with 0.9625 and 0.9555 for the raw-input forest [22, 23]. The statistical representation reduces the input dimension from ninety-five to thirty-five. This comparison supports the usefulness of the selected descriptors for these short traces. It does not isolate the contribution of each statistic, because all thirty descriptors are changed together when replacing the raw representation.

Table 3. Mean Diagnostic Performance Across Five Grouped Repetitions at Threshold 0.5. Higher Values Are Preferable Except for the Brier Score.

Configuration	Balanced accuracy	Macro F1	ROC area	Average precision	Brier score
Raw RF	0.9625	0.9555	0.9939	0.9977	0.0411
Stat LR	0.9558	0.9473	0.9881	0.9950	0.0450
Stat RF	0.9669	0.9687	0.9905	0.9960	0.0251
Stat ET	0.9675	0.9672	0.9940	0.9976	0.0227

Randomized trees have slightly higher balanced accuracy, 0.9675, and lower Brier score, 0.0227, than the statistical forest. The statistical forest has the higher macro F1, 0.9687 versus 0.9672. Logistic regression reaches balanced accuracy 0.9558 and macro F1 0.9473. The differences depend on the selected metric. The results support reporting a set of complementary measures rather than declaring one method dominant from a rounded accuracy value. Figure 4 presents performance curves from the first grouped repetition as an illustrative score comparison.

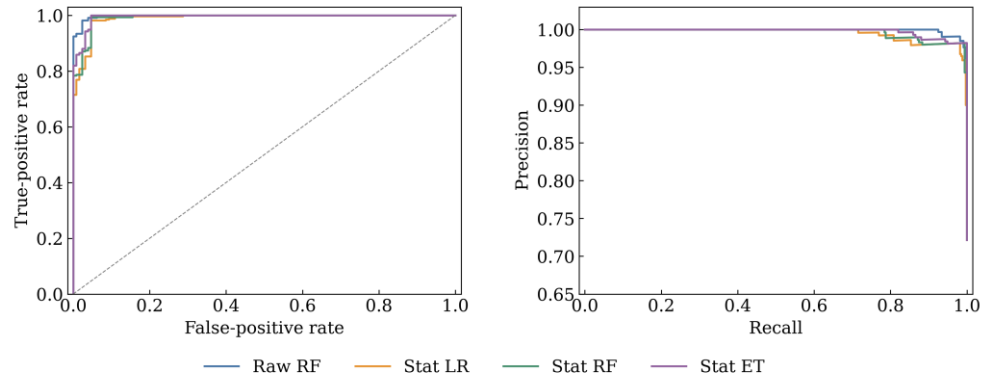


Figure 4. Receiver Operating Characteristic and Precision–recall Curves for the First Grouped Repetition. Aggregate Numerical Comparisons Use All Five Repetitions. Legends Are Outside Both Plotting Regions.

The raw forest has mean receiver operating characteristic area 0.9939, exceeding the statistical forest value of 0.9905, even though its thresholded macro F1 is lower. This difference shows that score ordering and a specific decision rule answer different questions. Average precision is high for all four configurations, with the abnormal class comprising 72.14% of the records. This prevalence should accompany interpretation of the precision–recall curves. The scores have not been calibrated for the frequency of abnormal events in a production cell [24].

At the fixed threshold, the statistical forest releases an average of five abnormal records per repetition. This count is calculated before averaging the metrics and represents a concrete residual diagnostic error, even when aggregate performance measures are close to one.

Giving every distinct trajectory group equal total evaluation weight changes statistical-forest balanced accuracy to 0.9777 and macro F1 to 0.9782. The corresponding raw-forest values are 0.9673 and 0.9555. The statistical representation remains competitive under this alternative weighting. Because the models are not retrained with group weights, these values are a sensitivity analysis of evaluation frequency rather than a new training method. They also remain estimates for the available trajectory groups, whose physical collection conditions are incompletely described in the released files.

4.4. Continuation Decisions and Cost Sensitivity

The decision experiment compares the statistical forest at threshold 0.5 with its training-selected thresholds. Table 4 gives the resulting error rates and weighted penalties. Fixed-threshold operation releases 1.50% of abnormal records and retains 5.12% of normal records. Cost-aware selection changes these rates to 1.14% and 9.61%. Thus, fewer abnormal episodes proceed, while more normal episodes trigger system response. The reduction in abnormal release is 0.36 percentage points; the increase in normal retention is 4.50 percentage points, calculated before rounding.

Table 4. Continuation Decisions under the Declared Five-To-One Error Penalty. Rates Use Their Class-Specific Denominators. Always Release and Always Hold Are Deterministic Reference Policies.

Policy	Abnormal release (%)	Normal hold (%)	Overall release (%)	Weighted penalty
Fixed RF	1.50	5.12	27.52	0.0683
Cost-aware RF	1.14	9.61	26.00	0.0678
Always release	100.00	0.00	100.00	3.6069
Always hold	0.00	100.00	0.00	0.2786

Mean weighted penalty changes from 0.0683 to 0.0678, a small difference. Its sign varies across repetitions: the selected threshold yields a lower penalty in two repetitions, a higher penalty in two, and an identical penalty in one. This pattern provides no evidence of consistent penalty improvement over the fixed threshold on this collection. The selected values range from 0.2 to 0.5. Selection variability reflects the small training partitions and the finite threshold grid [25]. The experiment reports that variability rather than selecting a favorable repetition for the main result.

Figure 5 separates abnormal-release and normal-retention rates. At the same time, the overall fraction of records released decreases from 27.52% to 26.00%. This fraction is not production throughput because the collection contains no recovery times, arrival process, or cycle durations. Always retaining every record eliminates abnormal release under the binary decision definition but retains every normal record. Its weighted penalty is 0.2786. Always releasing incurs penalty 3.6069. These policies provide useful checks on the direction and scale of the cost calculation [26, 27].

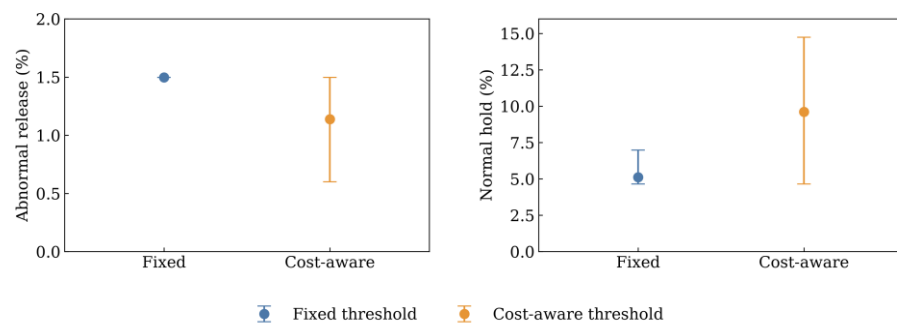


Figure 5. Class-specific Continuation Errors for Fixed and Cost-Aware Statistical Forests. Points Show Repetition Means; Whiskers Span the Five Repetition Values. Legends Are Outside the Axes.

The eight interlock input combinations all satisfy the specified entry condition. Only process permission combined with clearance and absence of occupancy enables entry. This result confirms the Boolean implementation, independent of whether a particular classifier decision is correct. It does not establish a physical proximity event rate, a stopping distance, or reliability under lost clearance messages. A real coordination controller would need stateful ownership and motion feedback around the tested condition, followed by workcell-level verification.

4.5. Excluded Tasks and Generalization

Figure 6 and Table 5 present the strict excluded-task test. Balanced accuracy ranges from 0.5250 for learning problems 2 and 3 to 0.9776 for problem 1. Results for problems 4 and 5 are 0.7917 and 0.7727, respectively. The test excludes all matching training traces, including identical instances appearing in other files. Consequently, excluding one task may remove a substantial portion of otherwise available training data [1]. This behavior is intentional: retaining such copies in training would violate the requirement for truly unseen traces.

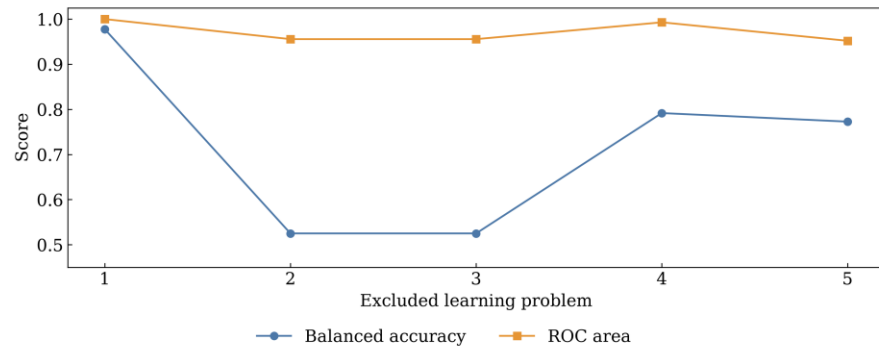


Figure 6. Performance on an Excluded Task. Balanced Accuracy at Threshold 0.5 and Receiver Operating Characteristic Area Describe Different Aspects of Transfer. Each Point Comes from One Strict Excluded-Task Fit.

Table 5. Statistical-forest Transfer to an Excluded Task at Threshold 0.5. Exact Held-Out Traces Are Removed from Every Training File.

Excluded task	Balanced accuracy	Macro F1	ROC area	Abnormal releases	Normal holds
1	0.9776	0.9552	1.0000	3	0
2	0.5250	0.4175	0.9556	0	19
3	0.5250	0.4175	0.9556	0	19
4	0.7917	0.8429	0.9928	0	10
5	0.7727	0.8145	0.9519	0	20

For problems 2 and 3, nineteen of twenty normal records are held out, while all twenty-seven abnormal records are held out. Their area under the receiver operating characteristic curve remains 0.9556, but balanced accuracy is only 0.5250. The scores preserve meaningful ordering, yet the default operating threshold yields suboptimal performance. Both problems yield identical results because their sensor trajectories and binary targets align under the current mapping. They therefore represent two labels for the same held-out binary recognition task, rather than independent validations of generalization [28].

These tests reveal a boundary not detectable through pooled grouped evaluation alone. Recognition performance benefits from knowledge of operational context and exposure to related conditions during training. Excluding an entire context alters score distributions and affects the number of normal tasks retained for evaluation. A deployment study should therefore collect task-specific validation records prior to selecting an operating threshold. The current results do not establish the minimum number of such records required, and no claims regarding model adaptation are supported by the excluded-task evaluation.

5. Discussion

The results demonstrate the functional role of learning in enabling the transition from observation to sustained process execution. Statistical descriptors support accurate discrimination between normal and abnormal operational traces, while a distinct authorization mechanism links this discrimination to sequential robot coordination [29]. Maintaining separation between these two functions ensures that errors in each remain identifiable. For instance, a classifier may erroneously authorize an abnormal operation even when the resource availability condition is correctly satisfied; conversely, a correct diagnostic assessment cannot override physical constraints such as occupancy of a transfer zone.

Data curation significantly influences interpretation. The publicly available dataset comprises 463 records but only 251 unique sensor trajectories, with substantial redundancy across task files. Grouped partitioning prevents exact trace duplication across

training and evaluation subsets. However, it does not ensure statistical separation among physically correlated trials when collection identifiers are absent. Consequently, the high aggregate performance metrics must be interpreted alongside the comparatively lower scores observed on excluded-task evaluations. These latter results indicate that a reliable ranking capability may persist even when the operational decision threshold proves inadequate—particularly when the definition of 'normal' varies across different operational contexts.

Threshold selection reflects an operational trade-off rather than a deterministic safety assurance. The specified five-to-one misclassification penalty prioritizes suppression of abnormal authorizations at the cost of increased delays for normal operations; however, its average improvement over a fixed-threshold baseline is modest and inconsistent across repeated evaluations. A manufacturing facility would need to determine intervention consequences, failure severity levels, and acceptable operational parameters based on its specific production environment. Such parameters cannot be inferred solely from class distribution statistics within a curated failure dataset.

The current experimental setup concludes at the level of supervisory decision-making and binary access control. It does not incorporate measurements of joint trajectories, contact forces from newly instrumented workcells, recovery success rates, or human-robot interaction scenarios. Subsequent validation should integrate the supervisor with a robot controller implementing operation-specific permissions and independently verified protective mechanisms. It should log timing of observation availability, effectiveness of hold actions in resolving underlying conditions, and impact of repeated operational pauses on throughput. Further testing should explicitly vary workpieces, task programs, and data collection sessions. These measures will determine whether diagnostic reliability persists under extended operation and whether the coordination interface adequately manages dynamic events not captured in static, offline recordings [26].

6. Conclusion

This paper investigated intelligent control and collaboration for industrial robots through an execution supervisor that integrates force and torque diagnosis with permission-based activation of a dependent task. The proposed architecture separates learned pattern recognition, threshold-driven continuation decisions, and a logical condition governing sequential access to a shared transfer zone. Experimental evaluation was conducted on post-observation process supervision using publicly available execution records; physical robot motion execution and recovery mechanisms are reserved for subsequent integration.

The analysis encompassed 463 execution records drawn from five learning problems. An integrity audit identified 251 distinct complete sensor trajectories and revealed substantial overlap across task files. Identical sensor traces were grouped consistently during both outer evaluation and inner threshold selection. Across five repeated grouped evaluations, a statistical random forest achieved a mean balanced accuracy of 0.9669 and a macro F1 score of 0.9687. A reduced input space of thirty-five dimensions retained diagnostic utility relative to the original ninety-five-dimensional representation. Alternative metrics favored the randomized ensemble, indicating that metric-specific interpretation is warranted. All evaluation scripts are retained to support independent numerical reproduction.

Thresholds selected during training reduced abnormal release rates from 1.50% to 1.14%, while increasing normal holding rates from 5.12% to 9.61%. The mean weighted penalty exhibited only marginal change, with directionality varying across repetitions. Evaluation on excluded tasks yielded balanced accuracy ranging from 0.5250 to 0.9776, highlighting limited generalizability of the default threshold. An independent interlock verification confirmed adherence to the specified Boolean entry rule across all eight input combinations.

These findings support compact, auditable diagnosis as one functional component within collaborative execution control. They further underscore the importance of identifying duplicate records and distinguishing pattern recognition from continuation decisions. Future work should employ session-separated measurements collected from a physical collaborative cell, validate operation-bound permissions, and quantify inspection and recovery outcomes. Linking these observations to cycle time, task completion rates, and protective-system behavior will provide essential evidence for assessing industrial utility beyond the current diagnostic benchmark and logical interface under real operating conditions.

References

1. A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, p. eaat8414, 2019.
2. A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human-robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.
3. V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.
4. J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
5. N. Hogan, "Impedance control: An approach to manipulation," in *1984 American Control Conference*, Jun. 1984, pp. 304–313.
6. L. M. Camarinha-Matos, L. S. Lopes, and J. Barata, "Integration and learning in supervision of flexible assembly systems," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 2, pp. 202–219, 1996.
7. S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.
8. B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.
9. T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
10. B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.
11. M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *IEEE Transactions on Robotics and Automation*, vol. 1, no. 2, pp. 126–133, 1981.
12. O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.
13. A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.
14. J. A. Marvel and R. Norcross, "Implementing speed and separation monitoring in collaborative robot workcells," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 144–155, 2017.
15. G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.
16. S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.
17. L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
18. P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.
19. A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020.
20. P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.
21. A. Dempster, D. F. Schmidt, and G. I. Webb, "Minirocket: A very fast (almost) deterministic transform for time series classification," in **Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining**, Aug. 2021, pp. 248–257.
22. M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.
23. A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd International Conference on Machine Learning*, Aug. 2005, pp. 625–632.
24. S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," *BMC Bioinformatics*, vol. 7, no. 1, p. 91, 2006.
25. T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, p. e0118432, 2015.

26. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014.
27. F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.
28. T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.
29. M. Dhanda, B. A. Rogers, S. Hall, E. Dekoninck, and V. Dhokia, "Reviewing human-robot collaboration in manufacturing: Opportunities and challenges in the context of industry 5.0," *Robotics and Computer-Integrated Manufacturing*, vol. 93, p. 102937, 2025.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of Publisher and/or the editor(s). Publisher and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.