AGENT FAILURE PREDICTION SURVIVES SAME-TASK TRACE SWAPS

Anonymous authors
Paper under double-blind review

ABSTRACT

A failure predictor can use an agent's initial request and partial execution record to predict its final outcome. However, a good overall score may not depend on observing the execution being scored. To examine this dependence, we applied task-matched process replacement to 982 held-out-task runs from the public τ^2-Bench archive. We kept the fitted predictor and initial request fixed and replaced the visible actions and feedback with those from another run of the same task and configuration. Donor selection did not use outcome labels. Before the third action, the default tree predictor achieved an AUROC of 0.694 with the original process, compared with 0.692 under same-task replacement and 0.652 under cross-task replacement. Across two tree settings and logistic regression at two observation points, same-task replacement retained similar or higher mean AUROC, while cross-task replacement lowered it in all six comparisons. Brier scores showed no clear advantage for the original process over same-task replacements. On 1,180 public SWE-Gym runs, adding process features improved tree AUROC, but an initial-only predictor achieved similar AUROC. These results showed that overall failure ranking can remain useful even when the process comes from another execution of the same task. Task-matched replacement tests this dependence without refitting the predictor and complements comparisons with initial-only baselines.

1 INTRODUCTION

Agent failure prediction estimates the final outcome of a run from its initial request and partial execution record. A useful score ranks failed runs above successful ones. The initial request may already explain part of this ranking. Actions and feedback may add information about the task without helping the predictor distinguish separate attempts at that task. When a predictor is used to monitor an ongoing execution, it is therefore important to test whether its performance depends on observing that execution.

Early outcome prediction has useful applications even when execution matching is not its objective. EarlyEval uses lightweight classifiers to reduce benchmark execution cost (Shi et al., 2026a). FailFast-RestartSmart uses prefix predictions to trigger a new rollout after an alarm (Wang et al., 2026). These applications motivate examining the evidence behind a score. Forecasting an outcome, identifying which repetition will fail, and improving success through an intervention are related but different evaluation problems.

An initial-only baseline tests whether adding process inputs improves prediction after fitting. It does not isolate whether a fitted joint predictor needs the process from the target execution. Removing a feature block and retraining changes the fitted model. Comparing different learners can also change the representation and regularization. To examine execution matching directly, we kept the predictor and initial request fixed and substituted the process from another run of the same task and released configuration. This comparison preserved the task and configuration while changing which execution supplied the process.

We applied this diagnostic to public repeated runs from τ^2-Bench. The default tree retained nearly the same overall ranking after same-task replacement, while replacement across tasks lowered mean AUROC. The same pattern appeared with stronger tree regularization, logistic regression, and a later observation point. We also analyzed SWE-Gym runs to compare process gains with a competitive initial-only predictor and examine which failures contributed additional alarms.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Our results showed that a predictor could retain favorable overall ranking after its process input was replaced by another execution's record. This finding separates outcome prediction from execution matching. The replacement procedure provides a direct test of that distinction on repeated agent runs, alongside the usual initial-only comparison.

2 Related work

Early outcome prediction and monitoring. EarlyEval trains success and failure classifiers and evaluates execution savings and evaluation fidelity. Its input-group ablations retrain predictors after removing task, action, or feedback features (Shi et al., 2026a). TRACER aggregates trajectory-risk signals for failure prediction in tool-using conversations (Tayebati et al., 2026). Last Step Matters examines uncertainty signals in long-horizon research agents and finds limited early predictive value (Li et al., 2026). These studies examine different signals and operating objectives. Our question concerns the source of the process input supplied to a fixed predictor.

FailFast-RestartSmart additionally evaluates a recovery policy that starts a fresh same-policy rollout and makes the interrupted repository diff available as an optional overlay (Wang et al., 2026). Our offline analysis does not evaluate restart or repair. The predictors tested here are archived adaptations trained for this study, not the published systems' fitted weights. Thus, our results concern the interpretation of these predictors' scores rather than a reproduction of those systems' reported outcomes.

Feature replacement and model reliance. Feature replacement is an established way to study predictive reliance. Fisher et al. (2019) distinguish reliance of a fixed model from measures involving refitting, and study loss estimates based on switching feature blocks across observations. Strobl et al. (2008) develop conditional permutation to account for correlated predictors. We use this general idea with repeated agent executions. Released task and configuration identifiers define the donor groups, and the target's initial request remains fixed. This is a task-matched diagnostic rather than fully conditional sampling of natural-language trajectories. Our contribution concerns its use to test execution matching in agent failure prediction.

3 Data and diagnostic design

3.1 Public archives and evaluation populations

We used Sierra-released runs from τ^2-Bench, a benchmark of conversational agents in a dual-control environment (Barres et al., 2025). The pinned release contains four trials for each of 278 tasks under three configurations, giving 3,336 records. We excluded 56 records with unknown official outcomes. The fixed common population with known outcomes and usable first-, third-, and fifth-action prefixes contains 3,210 runs. We assigned all configurations and repetitions of a task to the same split.

Tree predictors use 1,669 fitting runs, 556 calibration runs, and 985 validation runs. These splits contain 144, 49, and 85 tasks, respectively. The earlier block logistic regression (LR) fit uses the combined 2,225 fitting and calibration runs and the same validation split. These training regimes differ, so comparisons between model families are not a controlled architecture ranking. Our main comparisons hold each fitted model fixed.

Of the 985 validation runs, 982 from 84 tasks have another eligible execution of the same task and configuration. They include 205 failures and 777 successes and form the shared target population for both donor conditions. The three omitted singleton runs are successes, but donor availability, not outcome, determines inclusion. Because the population was fixed across observation points, results before the third action are conditional on reaching the fifth-action boundary.

The software-engineering archive is SWE-Gym's public OpenHands-Sampled-Trajectories release (Pan et al., 2025; SWE-Gym, 2024). It contains 6,055 records, 2,438 tasks, and 11 repositories. The fixed primary rule retains the first source-ordered record per task and configuration, then excludes flags for evaluation errors, test timeouts, and failed patch application. The resulting 5,814 records contain 3,486 fitting, 1,148 calibration, and 1,180 test runs. A fixed salted task hash within repository

determines the split, with all configurations of a task kept together. The test set contains 490 tasks, 1,094 failures, and 86 successes. Failure labels are inherited from the public evaluation outcomes, not a new adjudication of agent fault.

3.2 VISIBLE INPUTS AND PREDICTORS

For run i, let x_i denote the initial request, z_i the visible process, and $y_i \in \{0, 1\}$ the final failure label. A fitted model gives the score $p_i = f(x_i, z_i)$. In the conversational archive, we observe the prefix before the third or fifth substantive assistant action, excluding the fixed opening greeting. The process contains only actions and user or tool feedback visible at that point. It excludes future messages, terminal evaluation fields, hidden task specifications, and run identifiers.

The tree predictors use an EarlyEval-inspired representation with initial-request, accumulated-action, last-action, accumulated-feedback, and last-feedback text blocks, plus visible behavior counts (Shi et al., 2026b). The text blocks use term frequency and inverse document frequency (TF-IDF) features and singular-value reduction. We retain the fixed default and stronger-regularization settings. Replacement moves all process features together and leaves the initial block unchanged.

The block LR predictor uses separately normalized TF-IDF encodings for initial and process text. We did not normalize the concatenation again. For a target i and donor j, the replacement score is

$$p_{ij} = f(x_i, z_j) = \sigma\left(b + \beta_x^\top x_i + \beta_z^\top z_j\right). \tag{1}$$

Here x and z denote encoded vectors, b is the intercept, and β_x and β_z are the fitted coefficients for the two blocks. The logistic function σ maps the linear score to a failure probability. Replacement changes only the process term. We verified this calculation against direct sparse concatenation and archived own-process scores to an absolute tolerance of 10^{-12}. No predictor or encoder was refitted during replacement.

SWE-Gym uses the prefix before the third assistant message. The first user message is the initial input, while the preceding two assistant messages and visible tool feedback form the process. Its tree and LR models are separately fitted on that archive. They supply baseline and alarm analyses, not cross-domain transfer of a conversational predictor. Implementation details are in Appendix B.

3.3 TASK-MATCHED REPLACEMENT

For each target run i, the same-task donor set D_i contains all other eligible validation runs with the same task, domain, and released configuration. We evaluated the replacement score $p_{ij} = f(x_i, z_j)$ for every $j \in D_i$, using each tree or LR predictor in turn. We excluded the target itself and selected donors without using outcome labels or model scores. This gave 2,886 directed target-donor pairs at each observation point.

For the cross-task comparison, donors share the target's domain and configuration but have a different task. The same 982 targets yield 117,171 directed pairs per observation point. This condition changes task compatibility as well as execution identity. It is a stronger perturbation for comparison, not a causal decomposition of task and execution information.

Matching task identifiers does not make the initial natural-language requests identical. We therefore also examined donors with exactly matching initial text. This restriction leaves 11 targets from three tasks, all successful. It provides a coverage check but cannot support AUROC. Appendix A explains the corresponding loss comparison.

3.4 LOSS, RANKING, AND UNCERTAINTY

Our primary loss statistic averages the effect of one uniformly selected eligible donor for each target:

$$\Delta\,\text{Brier} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{|D_i|}\sum_{j\in D_i}(p_{ij} - y_i)^2 - (p_i - y_i)^2\right]. \tag{2}$$

Here N is the number of targets, p_i is the own-process score, and p_{ij} is the score after replacement. A positive value favors the original process. We enumerated every eligible donor, removing donor-sampling error for this statistic within the observed pool. Equal target weights prevent targets with

more donors from dominating the estimate. We averaged losses rather than probabilities, so the statistic describes a single donor's expected loss rather than an ensemble benefit.

We obtained 95% percentile intervals from 2,000 paired whole-task bootstrap draws stratified by domain. Each sampled task retained its configurations and repetitions. Fitted models and observed donor pools remained fixed. These intervals describe resampling of target tasks, not variability from retraining or collecting a new donor pool.

We also report area under the receiver operating characteristic curve (AUROC). For each of 512 draws, we independently sampled one donor uniformly for each target, allowing donor reuse. We calculated AUROC for each full assignment and averaged the 512 values. This differs from calculating AUROC from averaged prediction probabilities or from a one-to-one permutation. The plotted AUROC values are descriptive point estimates, not population confidence intervals.

The third-action default tree is the primary tree comparison. The linear extension fixed its third-action same-task comparison before scoring. Other boundaries and cross-task comparisons are exploratory complements. The replacement seed is 20260907. The validation data had been inspected in preceding research, so these analyses extend an exploratory study rather than constitute an untouched confirmatory test.

4 Prediction survives same-task replacement

4.1 A direct comparison of execution matching

The default tree retained nearly the same pooled ranking when its process input came from another run of the same task and configuration (Figure 1). Before the third action, own-process AUROC was 0.6937 and mean same-task AUROC was 0.6919. Cross-task replacement reduced mean AUROC to 0.6521. All three conditions used the same targets, fitted predictor, and target initial requests.

The Brier comparison showed no clear own-process advantage. The own Brier score was 0.194624, compared with an expected same-task donor Brier of 0.192728. The donor-minus-own difference was -0.001895, with interval $[-0.004879, 0.001121]$. The point estimate therefore slightly favored the donor process, but its interval also allowed a small own-process advantage.

4.2 The pattern extends across predictor settings

The stronger-regularization tree and block LR showed the same qualitative ranking pattern (Figure 1). At the third-action boundary, their own and same-task mean AUROCs were 0.6875 and 0.6912 for the tree, and 0.7261 and 0.7314 for LR. At the fifth-action boundary, same-task mean AUROC exceeded the own-process point estimate in all three settings. Cross-task replacement lowered mean AUROC in all six comparisons. Complete numerical values appear in Appendix A.

Brier differences varied in direction. Both trees had positive fifth-action point estimates, while block LR had a small negative estimate. All six same-task intervals included zero. These results do not establish statistical equivalence or imply that the predictors ignore every execution detail. Cross-task replacement also did not uniformly worsen Brier score. The primary finding concerns retained pooled ranking under same-task replacement, with loss results reported alongside it.

4.3 What the replacement test adds to initial-only baselines

The block LR initial-only comparator achieved AUROC 0.7122 on the same 982 runs. Its joint model achieved 0.7261, yet replacing the process with another execution's process retained a mean AUROC of 0.7314. A positive joint-versus-initial difference can therefore coexist with little observed advantage from matching the supplied process to the target execution.

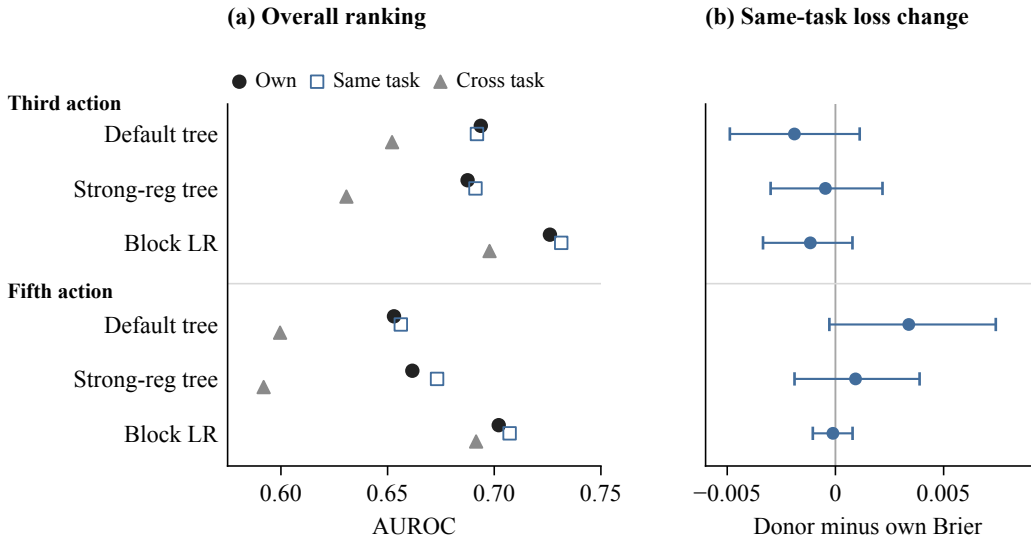

Figure 1: Execution matching and predictive performance on 982 conversational runs. Left: own-process AUROC and mean AUROC over 512 donor assignments. Symbols are descriptive point estimates without population uncertainty bars. Right: expected same-task donor minus own Brier score, with 95% whole-task bootstrap intervals. Positive loss differences favor the original process. Third and fifth refer to observation before the corresponding substantive action.

The earlier tree comparison reached a related conclusion through different metrics. On all 985 validation runs, adding process features increased pooled AUROC from 0.627 to 0.694, a paired increment of 0.0668 with interval $[0.0183, 0.1171]$. Within the same task and configuration, success-failure concordance instead changed by -0.0819 with interval $[-0.1661, 0.0000]$. This comparison uses 293 success-failure pairs from 365 runs in 92 mixed-outcome groups across 53 tasks. Concordance is the fraction of those pairs for which the failure receives the higher score, with half credit for a tie. It is not run-level classification accuracy.

The replacement analysis held the predictor fixed and tested its dependence on the supplied process directly. Its 982-run population excluded only the three runs without donors. Appendix A reports the earlier 985-run baseline analysis separately because it used a different population and, for LR, a different training regime.

5 Process Gains and Baseline Strength in Software Engineering

5.1 An Initial-Only Predictor Remains Competitive

On the 1,180-run primary SWE-Gym test set, adding process features increased tree AUROC by 0.0679, with interval $[0.0100, 0.1245]$. However, the joint tree's AUROC of 0.698 was close to the initial-only LR result of 0.693 (Table 1). Adding process features to LR reduced AUROC by 0.0277, with joint-minus-initial interval $[-0.1047, 0.0419]$. The positive tree increment and competitive initial-only LR support different statements about the value of observing the process.

The tree Brier change was -0.0006 with interval $[-0.0034, 0.0022]$, while LR's joint model increased Brier by 0.0125 with interval $[0.0058, 0.0205]$. The high failure prevalence also makes alarm precision an incomplete summary. We therefore examine counts at frozen thresholds in addition to ranking and probability loss.

Table 1: SWE-Gym results on 1,180 runs from 490 tasks, including 1,094 failures and 86 successes. Stratified AUROC compares outcomes within repository/configuration cells, not exact tasks. Higher AUROC and lower Brier are better. All eight fixed methods are shown.

Predictor	Pooled AUROC	Stratified AUROC	Brier
Constant	0.500	0.500	0.0678
Configuration prior	0.542	0.500	0.0670
Tree initial	0.631	0.568	0.0699
Tree process	0.617	0.536	0.0775
Tree joint	0.698	0.662	0.0694
LR initial	0.693	0.711	0.0635
LR process	0.665	0.615	0.0734
LR joint	0.665	0.655	0.0761

5.2 What the additional alarms detect

Thresholds were fixed using successful calibration runs, allowing at most $\lfloor 0.10n \rfloor$ strict exceedances among n calibration successes. The initial and joint trees flagged 300 and 375 test failures, respectively. Each flagged 11 of 86 successes. The joint tree's failure recall was 34.28% and its test false-positive rate was 12.79%. The calibration constraint did not guarantee a 10% false-positive rate on test tasks. Initial-only LR flagged 339 failures and 14 successes.

Table 2: Frozen-threshold SWE-Gym alarms. The net tree gain is 75 true alarms with no change in the false-alarm count. The failure groups describe archived output status, not independently determined failure causes.

Predictor	True alarms	False alarms
Tree initial	300	11
Tree joint	375	11
LR initial	339	14
Failure group		**Net tree gain**
With archived test output (640 failures)		6
Without archived test output (454 failures)		69

A post-result decomposition attributed 69 of the 75 net additional true alarms to failures without archived test output (Table 2). This group contained 429 empty patches, one null patch, and 24 nonempty patches. The 640 failures with documented output contributed six net additional alarms. These are differences in counts, not nested alarm sets. Missing output was not an exclusion criterion and does not identify known infrastructure failure.

The fixed population sensitivities retained a positive tree AUROC point increment both for all released test records, 0.0722 with interval $[0.0129, 0.1297]$, and for primary records with documented output, 0.0468 with interval $[-0.0157, 0.1076]$. Output status is observed after execution. This analysis describes the composition of the recorded gain, not the mechanism learned by the predictor or the value of an intervention.

5.3 Repeated-run coverage in SWE-Gym

The SWE-Gym primary population has one record per task/configuration by construction. Before that restriction, the 1,226-record test set had 37 task/configuration groups with two records each. Only four repeated groups contained both success and failure, and only 12 had different visible-feature hashes. We therefore used SWE-Gym for baseline and alarm analyses rather than as a second repeated-run replacement experiment.

6 DISCUSSION

Predictive value and execution matching answer different questions. A joint model may rank outcomes better than a matched initial-only learner while retaining its ranking with a different same-task process. Conversely, a model can be sensitive to replacement without adding much discrimination beyond initial input. The exploratory training comparisons in Appendix C illustrate the latter case. On held-out tasks, a matched-pointwise tree had an own-minus-swap within-task concordance difference of 0.0819, but its joint-minus-initial concordance difference was only 0.0068. Neither comparison can replace the other.

Using the diagnostic. To evaluate an outcome predictor, we suggest comparing joint and initial-only models on the same runs, including a competitive alternative initial-only learner. If repeated runs are available, task-matched replacement can test the joint predictor's reliance on its own execution record. Within-task success-failure comparisons then assess whether the score distinguishes alternative outcomes for the same task. For an alarm system, reporting counts at frozen thresholds identifies which successes and failures are flagged. Restart or repair benefits still require an intervention experiment.

Scope and alternative explanations. The retained ranking could arise because process features repeat task information, capture responses typical of a task, or contain execution information the fitted learner does not use effectively. Regularization and limited training data are also possible contributors. The present comparisons do not distinguish these mechanisms. Aggregate replacement loss can also conceal useful execution signals for particular subsets. The exploratory extensions contain such process sensitivity, so the conclusion is not that process information is universally unhelpful.

Our direct replacement evidence concerns two tree settings and a sparse linear predictor on one repeated-run archive. Same-task initial requests can differ, and combining a target request with a donor process can create incompatible inputs. Exact-text matching was too sparse to resolve this limitation. The common-prefix population omits short runs and unknown outcomes. The same validation set was also used in preceding analyses. Because the six comparisons share tasks and fitted components, they are not six independent replications.

7 CONCLUSION

Task-matched process replacement tests whether a fixed agent failure predictor benefits from the execution it is meant to monitor. Across the tested conversational predictors and early observation points, pooled ranking survived same-task replacement, while cross-task replacement reduced mean AUROC. The SWE-Gym analysis further showed that process gains depend on the baseline and on which failures contribute additional alarms. We recommend evaluating execution matching alongside initial-only baselines when using outcome scores to support claims about monitoring individual agent runs.

AI USE STATEMENT

AI tools were used for language polishing, grammar checking, and improving the clarity and organization of the manuscript.

REPRODUCIBILITY STATEMENT

Appendix B gives the pinned data revisions, text representations, model settings, and numerical checks. Section 3 defines target populations, donors, loss estimates, random assignments, and bootstrap units. Appendix A provides the complete main comparisons. The exploratory training results are separated in Appendix C. All reported analyses use public archived data. We do not redistribute raw trajectories or model weights in this manuscript.

ETHICS STATEMENT

This study analyzes public agent execution archives and does not collect new human-participant data. Released success and failure labels are used as evaluation outcomes rather than judgments about individual users or developers. Offline alarms can interrupt otherwise successful runs, and their false-positive rates can change across tasks. The reported scores should therefore not be treated as evidence that an untested intervention is safe or beneficial. Any reuse of the underlying archives remains subject to their source terms.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating conversational agents in a dual-control environment. arXiv:2506.07982v1, 2025. URL https://arxiv.org/abs/2506.07982v1.

Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81, 2019. URL https://jmlr.org/papers/v20/18-760.html.

Zongyue Li, Chengyue Yu, Lei Zang, Chenyi Zhuang, Linjian Mo, and Leilei Gan. Last step matters: Early uncertainty cannot predict failure in long-horizon agents. arXiv:2608.29685v1, 2026. URL https://arxiv.org/abs/2608.29685v1.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. arXiv:2412.21139v2, 2025. URL https://arxiv.org/abs/2412.21139v2.

Yuling Shi, Zhensu Sun, Junsen Dong, Chengcheng Wan, David Lo, and Xiaodong Gu. EarlyEval: Cheaper agent evaluation via early outcome prediction. arXiv:2609.02783v1, 2026a. URL https://arxiv.org/abs/2609.02783v1.

Yuling Shi, Zhensu Sun, Junsen Dong, Chengcheng Wan, David Lo, and Xiaodong Gu. EarlyEval: Cheaper agent evaluation via early outcome prediction. Computer software, 2026b. URL https://github.com/inphotoo/earlyeval. Revision 7fd1a8e5b755e1f7ab642bcae77473b53e2bf1d0.

Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC Bioinformatics*, 9:307, 2008. doi: 10.1186/1471-2105-9-307. URL https://doi.org/10.1186/1471-2105-9-307.

SWE-Gym. OpenHands-Sampled-Trajectories. Dataset, 2024. URL https://huggingface.co/datasets/SWE-Gym/OpenHands-Sampled-Trajectories. Revision baf3a4e4bff514d48ddc08a93a2ade5c126212c7.

Sina Tayebati, Divake Kumar, Nastaran Darabi, Davide Ettori, Ranganath Krishnan, and Amit Ranjan Trivedi. TRACER: Trajectory risk aggregation for critical episodes in agentic reasoning. arXiv:2602.11409v1, 2026. URL https://arxiv.org/abs/2602.11409v1.

Chenyu Wang, Yunbo Lyu, Junda He, Zhou Yang, Chenxing Zhong, Yaniv Harel, and David Lo. Fail-fast, restart-smart: Early failure prediction and restart for SWE agentic tasks. arXiv:2608.03222v1, 2026. URL https://arxiv.org/abs/2608.03222v1.

A COMPLETE REPLACEMENT AND BASELINE RESULTS

Tables 3 and 4 give the numerical values behind Figure 1. Each row compares inputs to one fixed predictor. All rows share 982 target runs, but the tree and block LR training populations differ as described in Section 3. No AUROC interval is inferred from variation across donor assignments. Brier intervals are unadjusted whole-task bootstrap intervals.

Table 3: Complete AUROC comparisons on 982 runs. Donor columns average AUROC over 512 complete single-donor assignments.

Boundary	Predictor	Own	Same task	Cross task
Third	Default tree	0.6937	0.6919	0.6521
Third	Strong-reg tree	0.6875	0.6912	0.6307
Third	Block LR	0.7261	0.7314	0.6978
Fifth	Default tree	0.6530	0.6562	0.5996
Fifth	Strong-reg tree	0.6616	0.6732	0.5919
Fifth	Block LR	0.7021	0.7072	0.6915

Table 4: Same-task Brier comparisons. The difference is expected donor loss minus own loss, calculated before rounding. Positive values favor the original process.

Boundary	Predictor	Own	Donor	Difference	95% interval
Third	Default tree	.194624	.192728	−.001895	[−.004879, .001121]
Third	Strong-reg tree	.164416	.163954	−.000463	[−.002995, .002175]
Third	Block LR	.157616	.156457	−.001159	[−.003346, .000787]
Fifth	Default tree	.189316	.192706	.003390	[−.000280, .007410]
Fifth	Strong-reg tree	.168882	.169811	.000929	[−.001890, .003889]
Fifth	Block LR	.159746	.159630	−.000116	[−.001045, .000792]

Table 5: Cross-task donor minus own Brier on the same 982 targets. Lower cross-task AUROC does not imply higher loss in every comparison.

Boundary	Predictor	Difference	95% interval
Third	Default tree	−.001002	[−.005143, .002815]
Third	Strong-reg tree	.004815	[.001291, .008400]
Third	Block LR	.002448	[−.000814, .005464]
Fifth	Default tree	.008403	[.002119, .015423]
Fifth	Strong-reg tree	.008111	[.002850, .013326]
Fifth	Block LR	.000441	[−.003282, .003663]

The earlier baseline comparison. Table 6 uses all 985 validation runs, including the three without donors. Its initial LR is from the smaller training regime used in the tree study, not the block LR extension trained on 2,225 runs. Macro AUROC equally weights domain/configuration cells. Exact-task concordance weights comparable success-failure pairs within task/configuration groups. The latter uses 293 pairs, rather than every possible failure-success comparison across tasks.

On the 982-run replacement population, the default and stronger-regularization initial trees have AUROCs of 0.6262 and 0.6445 and Brier scores of 0.192927 and 0.170179. Initial LR from the smaller regime has AUROC 0.6798 and Brier 0.162291. Initial LR from the larger block LR regime

Table 6: Earlier third-action comparisons on all 985 validation runs. Macro and pooled columns report AUROC. The last column reports exact-task concordance.

Predictor	Pooled	Macro	Exact task
Constant	.500	.500	.500
Configuration prior	.583	.500	.500
Initial LR	.680	.646	.459
Default tree initial	.627	.543	.539
Default tree process	.591	.484	.439
Default tree joint	.694	.614	.457
Strong-reg tree initial	.645	.580	.551
Strong-reg tree process	.688	.551	.486
Strong-reg tree joint	.688	.557	.485

has AUROC 0.7122 and Brier 0.157836. These are different fitted comparators, not inconsistent estimates of the same fit.

Exact-initial-text matching. The exact-text sensitivity analysis contained 11 successful targets from three tasks. When all targets in a donor group have identical initial inputs and labels, uniformly redistributing process scores within the group leaves its average loss unchanged. The near-zero loss difference in these groups follows from this property. AUROC is undefined because this population has no failures.

B Implementation details

B.1 Data versions and inputs

The conversational archive is from the public Sierra τ^2-Bench repository, revision `672227c6b6676edc20d57ea53b7000262aae77b9`. SWE-Gym trajectories use dataset revision `baf3a4e4bff514d48ddc08a93a2ade5c126212c7`. The reference feature implementation is EarlyEval revision `7fd1a8e5b755e1f7ab642bcae77473b53e2bf1d0`.

The conversational predictors use visible initial and process text and counts computed from that text. Task and configuration identifiers define splits and donor pools, not label-based donor selection. SWE-Gym predictors additionally share repository/configuration one-hot metadata across initial, process, and joint arms. Archived terminal test output is used for the post-result alarm decomposition, not as a prefix feature.

B.2 Conversational predictors

The tree text encoders use word unigrams and bigrams, minimum document frequency five, at most 30,000 features per text block, sublinear term frequencies, and L2 normalization. Each block is reduced to at most 64 singular-value dimensions. Encoders are fitted on fitting tasks only. Table 7 gives the complete tree presets. Both use the binary objective, 500 estimators, seed 42, deterministic single-thread CPU execution, and column-wise histogram construction. In the default setting, bagging frequency zero disables row subsampling despite the stored subsample parameter. Stronger regularization changes several parameters, not one isolated factor.

Block LR uses word unigrams and bigrams, minimum document frequency two, at most 20,000 features per block, sublinear term frequencies, and separate L2 normalization for initial and process blocks. Its classifier uses $C = 1$, the liblinear solver, a 2,000-iteration limit, and source seed 20260905. Both encoders remain fixed during replacement, including vocabularies and inverse-document-frequency weights. No cross-block normalization is applied after concatenation.

Table 7: Fixed conversational tree presets. Parameter names follow the implementation.

Parameter	Default	Strong regularization
Learning rate	.05	.03
Maximum leaves	63	31
Maximum depth	8	6
Minimum child samples	50	200
Subsample	.80	.75
Bagging frequency	0	1
Column sample fraction	.80	.70
L1 regularization	.10	.50
L2 regularization	1.0	10.0
Minimum split gain	.00	.01

B.3 SWE-Gym predictors and thresholds

The behavioral representation uses five text blocks truncated to 12,000 characters each and 27 counts from the visible prefix. Tree arms use 500 estimators, learning rate 0.05, 31 leaves, depth six, minimum child size 50, L1 regularization 0.1, and L2 regularization 1.0. Row and column fractions are one. LR uses standardized versions of these representations, $C = 1$, and a 2,000-iteration limit. It is not the sparse block LR used in the conversational replacement extension.

We selected alarm thresholds using only scores of successful calibration runs. An alarm requires a score strictly greater than the threshold. We allowed at most $\lfloor 0.10n \rfloor$ exceedances among the n calibration successes, retaining ties rather than forcing the target rate. Table 2 reports the observed test counts because the calibration constraint does not guarantee the same rate on new tasks.

B.4 Numerical checks

We checked own-process scores against the archived tree predictions and obtained exact agreement. Block LR's additive calculation agreed with direct sparse concatenation and archived scores within 10^{-12}. We also recomputed expected donor loss, bootstrap differences, and AUROC from stored scores and checked target and donor identities. These numerical checks did not change labels or assign causes of failure. Main replacement scores, data revisions, and code inputs were frozen before manuscript formatting.

C Exploratory training comparisons

C.1 Question and fixed comparison

To examine whether training on different outcomes for the same task improved execution matching, we compared a pairwise objective with a pointwise control trained on the same weighted run endpoints. We assessed own-versus-swapped sensitivity and joint-versus-initial predictive value separately. Unlike the main diagnostic, this analysis changed the training objective rather than only replacing inputs to an existing predictor.

The third-action fitting split contains 451 failure-success pairs from 142 mixed-outcome task/configuration groups. Their 562 distinct endpoints form the active training population. All runs remain in the source cohort, but pure-outcome groups receive zero endpoint training weight in both objectives. Each mixed group receives equal pair mass, distributed uniformly over its pairs. A run's matched-pointwise weight is half the summed mass of its incident pairs. Both objectives normalize total loss weight to 1,669, the size of the original fitting split.

The five text encoders and behavior counts follow Appendix B. The initial representation has 65 features, and the joint representation has 347. Both tree and linear models use these same features. LR additionally uses a scaler fitted on the fitting split. The representations are shared by eight models: initial and joint inputs under matched-pointwise and pairwise training, for each of LR and trees. Encoders and scalers are not fitted on calibration or validation data.

The pointwise objective is weighted logistic loss. For failure-success pairs (i, j), the pairwise objective is

$$L_{\text{pair}} = \sum_{(i,j)} w_{ij} \log(1 + \exp[-(s_i - s_j)]) \,, \tag{3}$$

where s_i and s_j are the failure and success run scores, and w_{ij} is the pair weight defined above. We fitted LR to score differences without an intercept, using $C = 1$, liblinear, a 2,000-iteration limit, and seed 20260908. Matched-pointwise LR included an intercept. Trees used the default preset in Table 7, with both custom objectives initialized at zero. We accumulated pairwise tree gradients at run endpoints and supplied the optimizer with diagonal Hessian terms, not cross-endpoint terms. These models produce ranking margins rather than calibrated probabilities.

C.2 Calibration and held-out-task scoring

The 556 calibration runs contain 140 comparable pairs from 44 groups. After examining this development split, all eight fitted models were frozen and scored on the original 985-run validation split. No vocabulary, scaler, model, or parameter was updated. The validation tasks are disjoint from fitting and calibration tasks, but this validation set had already been used in earlier project analyses.

This extension uses one fixed metadata-only circular donor assignment within task/configuration groups. It does not enumerate every donor as in the main experiment. All 556 calibration runs and 982 validation runs have donors. Validation's three singleton runs remain in own-score AUROC, and all 293 comparable validation pairs remain available for own-versus-swap concordance. Tables 8 and 9 retain all eight models.

Table 8: Training extension on calibration tasks. AUROC uses 556 runs. Concordance uses 140 failure-success pairs. These are exploratory development results.

Model and objective	Input	Own AUROC	Own concordance	Swap concordance
LR matched pointwise	Initial	.5585	.5643	.5643
LR matched pointwise	Joint	.5606	.5786	.4857
LR pairwise	Initial	.6154	.4857	.4857
LR pairwise	Joint	.5989	.5786	.4714
Tree matched pointwise	Initial	.6065	.5143	.5143
Tree matched pointwise	Joint	.6046	.5071	.5929
Tree pairwise	Initial	.6098	.5071	.5071
Tree pairwise	Joint	.5885	.5429	.5571

Table 9: The same eight frozen models on held-out tasks. Own AUROC uses 985 runs. Both concordance columns use the same 293 comparable pairs. This is not a ranking of independently tuned model families.

Model and objective	Input	Own AUROC	Own concordance	Swap concordance
LR matched pointwise	Initial	.5847	.4778	.4778
LR matched pointwise	Joint	.5458	.5017	.4881
LR pairwise	Initial	.6133	.4505	.4505
LR pairwise	Joint	.6450	.4812	.4778
Tree matched pointwise	Initial	.5560	.5495	.5495
Tree matched pointwise	Joint	.6072	.5563	.4744
Tree pairwise	Initial	.5337	.4983	.4983
Tree pairwise	Joint	.5364	.5392	.4812

C.3 What changed across task splits

The primary comparison was tree pairwise joint minus tree matched-pointwise joint within-task concordance. It was positive on calibration tasks but negative on validation tasks (Table 10). Pairwise

training did not outperform its matched-pointwise control in either model family on validation. We calculated each interval from 2,000 valid domain-stratified whole-task bootstrap draws, with shared task weights across methods. These exploratory intervals are unadjusted.

Table 10: All six fixed within-task concordance differences in the training extension. Each entry gives the estimate and its 95% interval. Pair and matched refer to training objectives. Unless stated otherwise, the input is joint.

Comparison	Calibration	Validation
Tree pair minus matched	$.0357\ [-.0461, .1309]$	$-.0171\ [-.0735, .0390]$
Tree pair joint minus initial	$.0357\ [-.0969, .1724]$	$.0410\ [-.0318, .1158]$
Tree pair own minus swap	$-.0143\ [-.1765, .1495]$	$.0580\ [-.0274, .1515]$
LR pair minus matched	$.0000\ [-.1024, .1040]$	$-.0205\ [-.1092, .0651]$
LR pair own minus swap	$.1071\ [-.0377, .2632]$	$.0034\ [-.0928, .0976]$
Tree pair minus LR pair	$-.0357\ [-.2222, .1395]$	$.0580\ [-.0512, .1656]$

LR pairwise own-minus-swap concordance fell from 0.1071 on calibration to 0.0034 on validation. The tree matched-pointwise model showed the opposite direction: its own-minus-swap difference was negative on calibration and positive by 0.0819 on validation. Yet the latter model's joint concordance of 0.5563 exceeded its initial-only concordance of 0.5495 by only 0.0068. Thus, sensitivity to the current process need not be accompanied by a comparable increment over the initial input.

Some fitted predictors were sensitive to their own process, but this sensitivity varied across task splits. Pairwise training did not improve validation concordance over matched-pointwise training in either model family. The tree result also showed why replacement sensitivity needs to be assessed alongside improvement over an initial-only predictor.