When Prediction Leaderboards Fail: Benchmarking the Prediction–Execution Gap

PEG-Bench Research Team
Public research release

Abstract

Sequential decision-making systems are often selected by offline predictive surrogates, even though deployment utility depends on costs, path dependence, risk controls, and distribution shift. Yet it remains unclear when an offline prediction leaderboard continues to support a deployment conclusion once realistic execution is introduced. To address this problem, we introduce the **Prediction–Execution Gap Benchmark (PEG-Bench)**, a benchmark for measuring when offline prediction rankings cease to remain valid under execution-aware evaluation. PEG-Bench fixes a 90-day multi-asset, multi-venue cryptocurrency limit-order-book evaluation contract with strict chronological splits, a five-level realism ladder from frictionless prediction evaluation ($R0$) to stress-tested execution evaluation ($R4$), and report cards for cost, risk, and stress. Because the underlying market feeds are licensed third-party data, the public artifact releases schemas, split manifests, evaluator code, execution contracts, normalized evidence, reproduction scripts, and sample or synthetic inspection shards, rather than raw feeds or reconstructable high-frequency data. We use cryptocurrency limit order books not to claim tradable alpha, but to provide a compact stress test for evaluation methodology, since they combine high-frequency decisions, explicit frictions, path dependence, rapid shift, and replayable execution in a single setting. In the completed 90-day artifact, the offline winner is stable but the deployment verdict is not: LightGBM is the strongest $R0$ predictor across all six populated time/volume panels, yet under the official benchmark protocol RR@1 is 1.0 in every panel and Kendall τ is negative throughout. At the 3 bps headline setting, the clean-active $R4$ winner changes in four of six panels, while the two unchanged panels remain negative after costs. Diagnostic wrapper and action-activity audits further show that execution outcomes are sensitive to the forecast-to-action interface, but these rows are reported separately from the official clean-active $R4$ leaderboard. Overall, PEG-Bench shows that prediction quality should be the starting point of deployment evaluation, not its endpoint.

1 Introduction

Sequential decision-making systems are widely studied in domains such as control, operations, recommendation, robotics, and finance, where model quality is ultimately judged by the decisions it induces under realistic constraints. However, system selection is still often driven by offline predictive surrogates such as accuracy, correlation, or ranking quality. This is justified only when offline predictive quality and realized deployment utility support the same conclusion. Once costs, path dependence, risk controls, and distribution shift are introduced, that alignment can fail. Despite its practical importance, this failure mode remains insufficiently studied as a benchmarked evaluation problem.

Existing work has mainly advanced individual parts of this pipeline rather than the mismatch between them. On the prediction side, prior work has developed strong models for limit order books (LOBs) and time series, including DeepLOB, TABL, LightGBM-style tabular predictors, and more recent

pretrained or foundation-style models such as iTransformer, TimesFM, Time-MoE, Moirai-MoE, WaveToken, TimeDART, and Kronos [23, 20, 8, 11, 2, 18, 10, 12, 21, 19]. On the benchmark side, recent efforts such as LOB-Bench have improved the standardized evaluation of financial sequence models and synthetic market data [13]. On the execution side, financial-RL environments, trading platforms, and replay tools have improved downstream policy evaluation and simulation realism [22, 9, 7, 6]. These directions are important, but they usually study either prediction quality, or model benchmarking, or execution realism in isolation. As a result, they provide limited evidence for a central benchmark question: when does an offline predictive advantage continue to support a deployment conclusion, and when does that conclusion change once the same method is evaluated under a shared execution-aware protocol?

To study this question, we introduce the **Prediction–Execution Gap Benchmark (PEG-Bench)**, a benchmark for measuring when offline prediction leaderboards cease to support deployment conclusions. PEG-Bench does not propose a new forecasting architecture or claim tradable alpha. Instead, it uses cryptocurrency LOBs as a compact stress test for evaluation methodology, because they combine high-frequency decisions, explicit frictions, path dependence, rapid distribution shift, and replayable execution under a common protocol. PEG-Bench fixes a shared data-and-evaluation contract: a 90-day multi-asset, multi-venue benchmark with strict chronological splits, replayable benchmark views, a common forecast/action interface, and a five-level realism ladder from frictionless prediction evaluation ($R0$) to stress-tested execution evaluation ($R4$). This design lets us evaluate the same method first as an offline predictor and then as an executable decision system, so that the scientific object of the paper is not model novelty, but whether the offline leaderboard still supports the deployment verdict once execution realism is added.

We instantiate this contract on BTCUSDT and ETHUSDT from Binance and OKX, and evaluate methods under shared cost, risk, and stress rules. In the completed 90-day artifact, the offline winner is stable but the deployment verdict is not: LightGBM is the strongest $R0$ predictor across all six populated time/volume panels, yet under the official benchmark protocol RR@1 is 1.0 in every panel and Kendall τ is negative throughout. At the 3 bps headline setting, the clean-active $R4$ winner changes in four of six panels, while the two unchanged panels remain negative after costs. These results motivate the benchmark's central claim that prediction quality should start, not end, deployment evaluation. Our contributions can be summarized as follows:

1. We introduce **Prediction–Execution Gap Benchmark (PEG-Bench)**, a benchmark for studying when offline prediction leaderboards fail to support deployment conclusions under a shared execution-aware protocol.
2. We define a fixed data-and-evaluation contract built around strict chronological splits, replayable benchmark views, a common forecast/action interface, and an $R0$–$R4$ realism ladder that jointly measures prediction quality, execution utility, ranking drift, and stress robustness.
3. We provide a completed 90-day artifact and empirical study showing systematic prediction–execution reversals, together with auditable evidence and explicit claim boundaries for completed, diagnostic, and target-only method rows.

2 Benchmark Contract: Data, Realism, and Claims

This section defines the PEG-Bench benchmark contract. PEG-Bench fixes the data window, the split protocol, the realism ladder, the report-card metrics, and the evidence boundary used for empirical claims. As in prior benchmark and evaluation frameworks, the protocol is defined separately from any single model family so that method comparison is performed under one shared contract [9, 22, 13].

2.1 A Fixed 90-Day Multi-Asset LOB Contract

PEG-Bench uses a fixed 90-day window from 2025-09-28 to 2025-12-26 UTC over BTCUSDT and ETHUSDT on Binance and OKX. Licensed incremental L2 updates and synchronized trades are canonicalized into a shared event schema and then converted into benchmark views, including reconstructed top-10 order books, 1-second bars, volume bars, and execution logs. These views follow standard limit-order-book representations used in market microstructure analysis and predictive modeling [1, 23, 20]. The split is strictly chronological: 60 training days, 15 validation days, and a

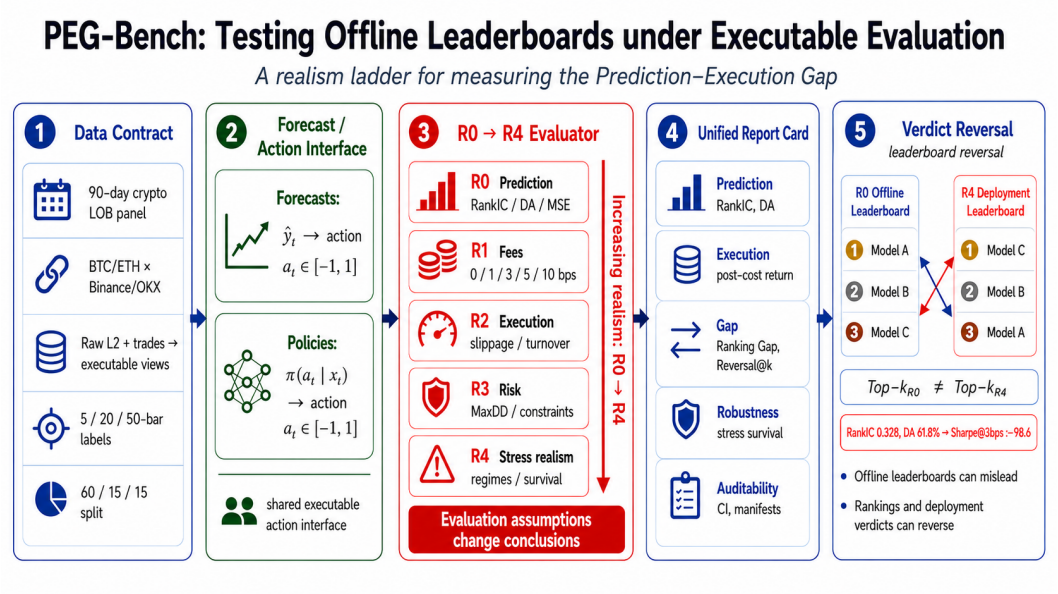

Figure 1: Overview of PEG-Bench. A fixed data contract feeds a shared forecast/action interface and an $R0$–$R4$ evaluator with increasing execution realism. The resulting report card measures prediction quality, execution utility, ranking drift, stress behavior, and auditability under one common protocol.

frozen 15-day test window. No normalization statistic, tokenizer, threshold, or cost calibration may use validation or test information during training.

The public artifact is designed for auditability rather than raw-data redistribution. The artifact releases schemas, split manifests, evaluator code, execution contracts, normalized evidence files, reproduction scripts, and synthetic or non-reconstructable inspection shards. The artifact does not release raw feeds, vendor API responses, tick-level trades, order-book updates, or fine-grained derived features that could substitute for the licensed data.

2.2 The $R0$–$R4$ Realism Ladder

PEG-Bench evaluates every method under a shared realism ladder. Level $R0$ measures frictionless prediction quality through offline metrics such as RankIC and directional accuracy. Level $R1$ adds exchange fees. Level $R2$ adds slippage. Level $R3$ adds risk constraints such as drawdown control and position accounting. Level $R4$ adds regime-aware and stress-aware evaluation. The ladder is designed to separate offline predictive quality from execution-aware deployment behavior under one protocol [22, 9, 6].

The main deployment verdict is anchored to $R4$, not to $R0$. The role of $R0$ is to measure predictive quality in isolation. The role of $R4$ is to test whether the same method remains credible once forecasts are converted into executable action paths under realistic frictions. PEG-Bench also fixes a cost ladder of 0/1/3/5/10 bps. The 90-day study uses 3 bps as the headline setting, while appendix tables report cost sensitivity for the available core methods across the full ladder.

2.3 Report Card and Prediction–Execution Gap Metrics

PEG-Bench reports prediction metrics and execution metrics under the same evaluation contract. RankIC and directional accuracy measure predictive signal. Post-cost Sharpe, MaxDD, turnover, and Calmar measure the quality of the induced action path. Worst-regime Sharpe and stress survival measure robustness under adverse slices.

PEG-Bench also reports four gap metrics that directly quantify the Prediction–Execution Gap. Metric Gap measures the cross-method association between the offline surrogate and $R4$ utility. Ranking Gap measures Kendall rank consistency between the $R0$ and $R4$ leaderboards. Reversal Rate@k measures how often offline top-k methods fall out of the execution-aware top-k. Fragility Gap

Component	Frozen PEG-Bench contract	90-day study evidence
Data and split	90D; BTC/ETH; Binance/OKX; 60/15/15 chronology	Time and volume views each pass 360 asset-days; h5/h20/h50 populated
Evaluation	$R0$–$R4$ ladder; 0/1/3/5/10 bps; stress slices	$R0$ and clean-active $R4$ report cards at 3 bps
Methods	Prediction, anchor, heuristic, wrapper, policy, and preference slots	LightGBM, Kairos, DeepLOB, Ridge-style controls, anchors, wrappers, and reported OG-PA diagnostics
Artifact	Schemas, split manifests, execution specifications, evaluator outputs, claim map, and sample/synthetic inspection shards	Normalized CSV/JSON evidence, generated tables and figures, result checklist, excluded-run notes, and SHA256 manifest

Table 1: Benchmark contract and 90-day study evidence. The *Frozen PEG-Bench contract* column summarizes the pre-specified benchmark protocol, and the *90-day study evidence* column summarizes the evidence populated under that protocol in the current study.

measures the shortfall from typical utility to worst-regime utility. A PEG-Bench result is interpretable only when prediction metrics, execution metrics, and gap metrics are tied to the same normalized evidence files.

2.4 Evidence Boundary for the 90-Day Study

PEG-Bench is organized around an evidence boundary rather than a model zoo. A benchmark row supports a main-text claim only when the row has normalized evidence under the frozen split and the shared evaluation interface. Diagnostic rows, lite rows, target rows, and audit-only rows may still be shown for transparency, but those rows do not count as ranked evidence.

The basic empirical unit is a panel, defined as one benchmark view paired with one return horizon. The completed 90-day study populates six panels: time and volume views at horizons 5, 20, and 50. Each populated panel induces an $R0$ leaderboard, an $R4$ leaderboard, and a report card over utility, risk, stress, and rank drift. This panel structure limits selective reporting because the same evaluation logic must be shown across all populated panels rather than only on a single favorable slice.

Table 1 summarizes the boundary between the fixed PEG-Bench contract and the evidence instantiated in the 90-day study. The *Frozen PEG-Bench contract* column states the pre-specified benchmark protocol, and the *90-day study evidence* column states the evidence populated under that protocol. The main empirical claims are supported by the evidence listed in Table 1.

2.5 Completed Evidence vs. Frozen Targets

The contract is designed around falsifiable evaluation questions rather than around a model zoo. A completed benchmark row must answer: what data view it consumes, what validation-only decisions it makes, what action path it emits, how it behaves under the shared cost and risk rules, and whether its $R0$ evidence still supports its $R4$ verdict. The claim rule is simple: a result supports a main-text claim only if it has normalized evidence under the frozen split and shared action interface. Diagnostic, lite, target, or audit-only rows are reported for transparency but excluded from official rankings. This is why Table 1 separates the frozen benchmark contract from the current 90D evidence.

The basic empirical unit is a panel: a benchmark view paired with a return horizon. The completed artifact populates six panels, namely time and volume views at horizons 5, 20, and 50. Each panel induces an $R0$ leaderboard, an $R4$ leaderboard, and a report card over utility, risk, stress, and rank drift. This panel structure prevents selective storytelling: the report cannot present only the single horizon where a claim looks strongest without also exposing the corresponding gap metrics and stress behavior for the other populated panels.

Table 1 distinguishes the benchmark specification from the subset instantiated in the 90-day study. Only the populated evidence is used for the main empirical claims. Target, diagnostic, and audit-only slots are reported separately for transparency and are not part of the main leaderboard.

3 A Shared Interface for Forecasts, Actions, and Audits

All ranked methods must produce either a forecast $\hat{y}_t$ that is mapped into an action a_t, or a direct action path $a_t \in [-1, 1]$ under the shared evaluator. Prediction-first models are first evaluated at $R0$ and then passed through the same action, cost, turnover, and stress accounting. Policy-learning methods are evaluated by $R4$ utility rather than by offline predictive metrics. Audit-only tools, such as HftBacktest, are excluded from the leaderboard.

The completed 90-day panel is intentionally compact. It includes non-ML anchors, a LOB heuristic, low-capacity predictors, LightGBM as the tabular baseline, DeepLOB and Kairos as LOB neural predictors, a cost-threshold/turnover wrapper, and OG-PA/DPO diagnostic rows where normalized outputs are available. These rows define the core benchmark panel used for the main empirical claims. Appendix E documents the broader full-release method panel and implementation provenance, but additional families are not part of the current ranking unless their normalized outputs are reported in the result tables.

The panel is designed to evaluate alignment between offline prediction quality and execution-aware deployment outcomes rather than model novelty alone. If a simple wrapper changes the $R4$ verdict, the result indicates sensitivity to the forecast-to-action interface. If a deployment-aware method remains negative under the frozen evaluator, the row is retained as a diagnostic result rather than presented as a positive benchmark finding.

The panel therefore serves four evaluation purposes. Anchors test whether active methods outperform no-trade and market beta after costs. Microstructure heuristics test whether simple LOB-native rules already account for the observed effects. Prediction-first baselines test whether an offline winner remains credible after forecast-to-action conversion. Wrapper and deployment-aware rows test whether execution-aware action design changes the deployment conclusion. Method families are included only when they address one of these evaluation roles.

The same protocol governs hyperparameters and action wrappers. Thresholds, smoothing parameters, and turnover penalties are selected on the validation split only, and test-time adjustment of the action mapping is not allowed. PEG-Bench therefore treats the forecast-to-action interface as part of the evaluated method definition rather than as a post hoc presentation choice. The appendix records both the broader intended method panel and the subset currently supported by normalized evidence.

4 Results: Stable Prediction, Unstable Deployment

Table 1 defines the empirical boundary. The completed artifact populates the core prediction-first report card on six panels: time and volume views at horizons 5, 20, and 50. It also reports anchors, heuristic rows, gap metrics, wrapper audits, stress diagnostics, and formal-method diagnostics. The result claims in this section refer only to rows with exported normalized evidence.

4.1 The Offline Winner Is Stable

The first result is deliberately simple: the offline comparison is not ambiguous. LightGBM is the strongest $R0$ predictor across the six completed panels, so the paper is not relying on a weak or unstable offline winner. This matters for the benchmark claim because the $R0$ result can be accepted on its own terms while the deployment question remains unresolved until forecasts are converted into cost-bearing actions.

4.2 The Deployment Verdict Changes

Figure 2 and Table 2 give the clean-active ranking result. LightGBM wins every $R0$ panel by RankIC, but the $R4$ top active method changes in four of six panels. In the two panels where LightGBM remains the $R4$ top active method, its post-cost Sharpe is still negative. Thus the execution-aware conclusion is not merely that another architecture sometimes wins; it is that the offline winner does not establish deployability.

This distinction matters for benchmark design. A conventional result table might stop after reporting RankIC or directional accuracy and conclude that the same method should be selected. PEG-Bench instead asks whether the selected method still clears a deployment threshold once action frequency,

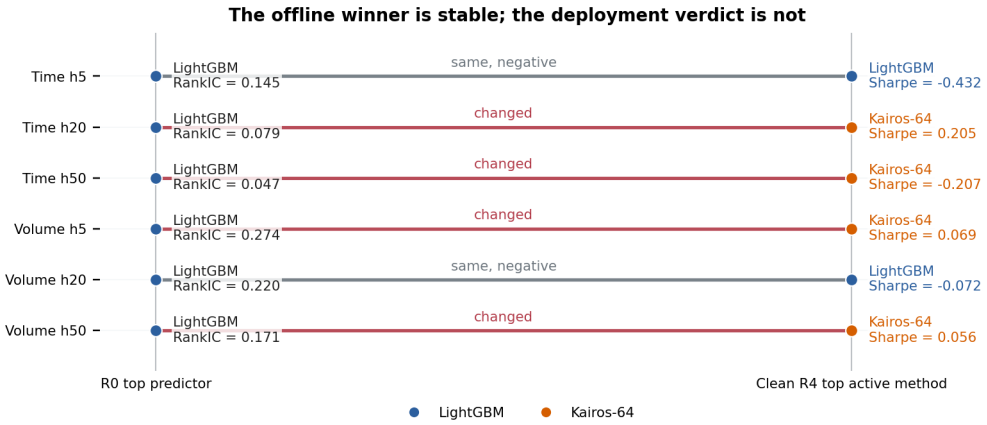

Figure 2: Clean-active result. The offline winner is stable, but the deployment verdict is not. The slopegraph connects the $R0$ prediction winner to the clean-active $R4$ winner at 3 bps for each completed panel; LightGBM wins every $R0$ panel, while four panels change winner and the two unchanged panels remain negative after costs. Positive $R4$ Sharpe values shown in the figure are point estimates.

Panel	h	$R0$ winner	RankIC	DA (%)	$R4$ winner	$R4$ Sharpe	Day-block 95% CI	Result
Time	5	LightGBM	0.145	54.7	LightGBM	-0.432	$[-0.630, -0.239]$	$R4$ winner unchanged; remains negative after costs
Time	20	LightGBM	0.079	54.0	Kairos-64	0.205	$[-0.137, 0.613]$	$R4$ winner changes; positive point estimate
Time	50	LightGBM	0.047	52.2	Kairos-64	-0.207	$[-0.529, 0.096]$	$R4$ winner changes; remains negative after costs
Volume	5	LightGBM	0.274	60.8	Kairos-64	0.069	$[-0.174, 0.325]$	$R4$ winner changes; positive point estimate
Volume	20	LightGBM	0.220	58.8	LightGBM	-0.072	$[-0.331, 0.012]$	$R4$ winner unchanged; remains negative after costs
Volume	50	LightGBM	0.171	57.2	Kairos-64	0.056	$[-0.256, 0.378]$	$R4$ winner changes; positive point estimate

Table 2: Clean-active $R0$–$R4$ comparison across the six populated panels. LightGBM is the $R0$ winner in all six panels, whereas the clean-active $R4$ winner at 3 bps changes in four panels. Positive $R4$ Sharpe values are reported as point estimates unless the day-block interval excludes zero.

friction, and adverse regimes are visible. In this evidence set, the answer is negative: even when the $R4$ identity does not change, the utility sign can still invalidate the deployment claim.

4.3 Gap Metrics Expose Leaderboard Drift

Table 3 reports the official Prediction–Execution Gap statistics for the completed prediction-first panels. Reversal Rate@1 equals 1.0 in every panel, and the ranking gaps are negative in all six panels.

Table 2 reports the clean-active $R4$ winner after excluding zero-action safety-floor anchors; it answers whether the active deployment winner changes. Table 3 reports official top-1 reversal under the deterministic prediction-first gap protocol, including the benchmark's tie-breaking and ranking rules; it answers whether the offline top method remains top under the official $R4$ ranking. The two tables therefore audit different parts of the same Prediction–Execution Gap.

Figure 3 explains why rank reversal is possible. A method can move rightward by improving RankIC while failing to move upward in post-cost Sharpe because turnover and execution frictions convert predictive signal into expensive action paths. The methods do not lie on a monotone frontier from better RankIC to better post-cost Sharpe; turnover and cost exposure create a second axis. This is why PEG-Bench reports both metric-level gaps and action-path diagnostics. The same view motivates the wrapper audit: modifying the forecast-to-action interface changes turnover exposure without

Panel	h	Metric Gap	Kendall $\tau(R0, R4)$	RR@1
Time	5	0.228	-0.333	1.000
Time	20	0.200	-0.111	1.000
Time	50	-0.044	-0.200	1.000
Volume	5	-0.978	-1.000	1.000
Volume	20	0.161	-0.600	1.000
Volume	50	-0.994	-1.000	1.000

Table 3: Official main result. Gap metrics show top-1 reversal in every completed panel. Metrics are computed under the deterministic prediction-first ranking protocol, including the benchmark's official ranking and tie-breaking rules; RR is reported at $k = 1$.

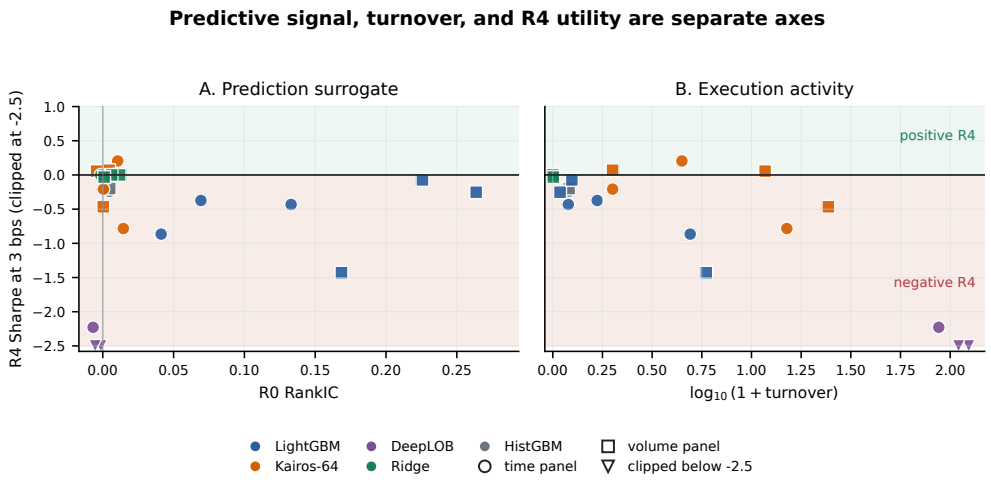

Figure 3: Diagnostic analysis. High RankIC does not imply positive post-cost Sharpe. This latest all-slice diagnostic view combines workbook RankIC rows with the 3 bps $R4$ execution rows; the left panel compares prediction signal with utility, the right panel compares turnover with utility, marker shape separates time and volume panels, and downward triangles mark values below the clipped display range at -2.5.

changing the underlying predictor. This is precisely why PEG-Bench treats the action bridge as part of the evaluated method rather than as an after-the-fact plotting choice.

4.4 Action Interfaces Can Change Verdicts

The negative Sharpe values should not be read as an argument that the domain is impossible or that every model family fails. They show that the benchmark is able to reject a deployment claim under a fixed evaluator. This is a useful property for a benchmark: if every offline winner also looked deployable after costs, the benchmark would not stress the evaluation protocol. The latest wrapper audit demonstrates the complementary behavior across 26 panel-method rows: changing the action interface can improve the same predictor without changing its underlying predictive model, but only the exported audit rows support this narrower diagnostic claim.

Table 4 is intentionally separated from the clean active leaderboard. Wrapper diagnostics test whether a cost-aware interface can change a deployment verdict: the LightGBM volume-h20 wrapper moves the Sharpe point estimate from negative to a positive point estimate. Formal-method diagnostics show the opposite boundary condition: OG-PA/DPO variants are negative or zero under the completed 90D evaluator and are therefore retained only as method-slot diagnostics. Stress diagnostics add the robustness view, because survival may be 1.0 while worst-regime Sharpe remains negative; PEG-Bench therefore reports utility and survival together.

Diagnostic	Scope	Key result	Interpretation
Wrapper	Volume h20, LightGBM	Sharpe p.e.: $-0.072 \rightarrow 0.629$	Cost-aware gating can flip the verdict point estimate.
Wrapper	Time h20, Kairos-64	Best wrapped Sharpe p.e.: 0.927	Interface tuning can also improve a strong predictor.
Wrapper matrix	26 panel-method rows	3 positive point estimates; 2 negative-to-positive flips	Wrapper evidence is diagnostic, not a clean leaderboard replacement.
Formal slot	OG-PA/DPO	Negative or zero-action variants	Diagnostic only; not a headline win.
Stress	Volume h20, LightGBM	Worst-regime Sharpe: -0.673; survival: 1.0	Survival is insufficient without utility.
Stress	Volume h20, Kairos-64	Worst-regime Sharpe: -4.804; survival: 1.0	R0 strength can remain stress-fragile.

Table 4: Diagnostic analysis. Action-interface rows separate execution-interface effects from architecture effects. The rows are reported separately from the clean ranked $R4$ leaderboard unless explicitly stated.

The wrapper result is not presented as a new algorithmic contribution. Its role is to show that even a minimal cost-aware bridge can alter the $R4$ conclusion, which is exactly the kind of sensitivity a benchmark should reveal. Likewise, the negative formal-method rows are not hidden. They are useful because they demonstrate that the artifact can host deployment-aware families while keeping the empirical claim boundary conservative.

Taken together, the completed results support three benchmark-level conclusions. First, $R0$ leaderboards are repeatable but not decisive. Second, action-interface choices can be as important as predictor architecture once costs are visible. Third, stress survival must be paired with utility because a strategy can survive by barely trading or by losing consistently in adverse regimes. These conclusions are about the evaluation object, not about a single model family. We therefore interpret the strongest completed evidence as identity reversal and ranking-gap instability; positive $R4$ Sharpe values are point estimates unless day-block intervals exclude zero.

5 Related Work

PEG-Bench relates to three lines of work. Prediction-first financial and time-series models, including LightGBM, DeepLOB, TABL, Kronos, iTransformer, and TimesFM [8, 23, 20, 19, 11, 2], provide strong forecasting and representation baselines, but they are typically compared by predictive metrics before deployment frictions are added. PEG-Bench contributes the missing comparison layer: it asks whether the same predictive ranking survives the conversion from forecasts to executable actions.

Trading and financial-RL benchmark ecosystems, including TradeMaster and FinRL-Meta [22, 9], provide useful environment and protocol infrastructure, but they do not make offline prediction-to-execution rank drift the central evaluation object. PEG-Bench differs by making the ladder itself scientific evidence: the same method can be inspected at $R0$, $R4$, and under stress, so the benchmark reports how the conclusion changes as realism is added.

Preference-learning and uncertainty-aware methods, including DPO and adaptive conformal inference [15, 3], motivate deployment-aware training ideas, but an internal alignment objective is not itself a deployment metric. In PEG-Bench these methods are useful only if they improve the frozen external report card; they are not allowed to replace the benchmark metric with an internal preference objective. This separation lets the appendix describe OG-PA/DPO interfaces while the main text remains a benchmark paper rather than a method paper.

More broadly, PEG-Bench follows the benchmark-paper principle that a useful evaluation artifact should change what evidence is considered sufficient. Existing model papers can still use their preferred training losses and architectures, but PEG-Bench asks them to report the same execution-aware evidence before making deployment claims. The contribution is the common yardstick, not a larger catalog of baselines.

6 Limitations and Responsible Release

PEG-Bench supports a narrow claim: offline predictive leaderboards are not sufficient deployment evidence under realistic sequential evaluation. In the completed 90-day artifact, rankings and deployment verdicts change between $R0$ and $R4$. For this reason, PEG-Bench reports post-cost utility, turnover, drawdown, and stress behavior together with offline metrics.

The current artifact does not populate the full intended method panel. Main claims are limited to rows with normalized evidence under the frozen split and shared action interface. Additional families in the appendix remain diagnostic, supplemental, or target rows unless explicitly populated. The replay evaluator also uses approximate frictions, so PEG-Bench should be read as an execution-aware benchmark rather than as a production simulator or evidence of tradable alpha.

The release is license-aware by construction. PEG-Bench uses third-party licensed market feeds internally, but the public artifact does not redistribute raw L2 updates, vendor responses, tick-level trades, order-book snapshots, or reconstructable fine-grained features. Instead, the release provides schemas, manifests, evaluator code, normalized evidence, reproduction scripts, metadata, and synthetic or non-reconstructable inspection shards. Full reruns require authorized access to the underlying market-data provider.

7 Conclusion

PEG-Bench reframes the Prediction–Execution Gap as a benchmark problem: whether evaluation realism changes rankings and deployment verdicts. In the completed 90D artifact, LightGBM is the strongest $R0$ predictor across all six panels, Reversal Rate@1 is 1.0 in every panel, and a diagnostic execution wrapper changes the volume-h20 LightGBM verdict from negative to a positive Sharpe point estimate outside the clean leaderboard. The contribution is a licensed-data evaluation contract, a realism ladder and report card for execution-aware comparison, and an executable artifact for reproducible benchmarking under shared assumptions.

The main message is that prediction quality should start, not end, a deployment evaluation. PEG-Bench supplies the missing intermediate object: a benchmark where the same model can be judged as a predictor, as an action generator, and as a stress-tested policy under one auditable protocol. PEG-Bench is released as a license-aware evaluation artifact: the protocol, evaluator, manifests, normalized evidence, and sample/synthetic shards are public, while full raw-data reconstruction is delegated to users with authorized access to the underlying market feeds.

References

[1] Emmanuel Bacry and Jean-François Muzy. *Financial Time Series: From Stochastic Processes to Order Book Dynamics*. Springer, 2020.

[2] Abhimanyu Das et al. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, 2024.

[3] Isaac Gibbs and Emmanuel Candès. Adaptive conformal inference under distribution shift. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[4] Google Research. Timesfm: Time series foundation model. `https://github.com/google-research/timesfm`, 2024.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[6] HftBacktest Developers. Hftbacktest: High-frequency trading backtesting tool. `https://hftbacktest.readthedocs.io/`, 2025.

[7] N. Kazu. Hftbacktest: High-frequency trading backtesting tool. `https://github.com/nkaz001/hftbacktest`, 2025.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[9] Xiao-Yang Liu, Hongyang Yang, Jincheng Gao, and Christina Dan Wang. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[10] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Junnan Li, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 38940–38962. PMLR, 2025. URL https://proceedings.mlr.press/v267/liu25an.html.

[11] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations*, 2024.

[12] Luca Masserano, Abdul Fatir Ansari, Boran Han, Xiyuan Zhang, Christos Faloutsos, Michael W. Mahoney, Andrew Gordon Wilson, Youngsuk Park, Syama Sundar Rangapuram, Danielle C. Maddix, and Yuyang Wang. Enhancing foundation models for time series forecasting via wavelet-based tokenization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 43248–43275. PMLR, 2025. URL https://proceedings.mlr.press/v267/masserano25a.html.

[13] Peter Nagy, Sebastian Frey, Kevin Li, Biplab Sarkar, Sergei Vyetrenko, Stefan Zohren, Andrei Calinescu, and Jakob Foerster. Lob-bench: Benchmarking generative ai for finance – an application to limit order book data. *arXiv preprint arXiv:2502.09172*, 2025.

[14] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[16] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[18] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations*, 2025.

[19] Yu Shi, Zongliang Fu, Shuo Chen, Bohan Zhao, Wei Xu, Changshui Zhang, and Jian Li. Kronos: A foundation model for the language of financial markets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(30):25366–25373, 2026. doi: 10.1609/aaai.v40i30.39730. URL https://ojs.aaai.org/index.php/AAAI/article/view/39730.

[20] Duy-Thanh Tran, Alexandros Iosifidis, Jussi Kanniainen, and Moncef Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 30(5):1407–1418, 2019.

[21] Daoyu Wang, Mingyue Cheng, Zhiding Liu, and Qi Liu. Timedart: A diffusion autoregressive transformer for self-supervised time series representation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 62627–62651. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25r.html.

[22] Zhengyao Wang et al. Trademaster: A holistic quantitative trading platform empowered by reinforcement learning. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[23] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019.

A Supplementary Evidence for the 90-Day Result

This appendix consolidates the evidence supporting the 90-day result: motivating-slice status tables, contract tables, and supplementary figures. Historical motivating-slice entries are retained solely for diagnostic context and are excluded from the 90-day benchmark evidence. The remaining material specifies the claim-to-evidence checklist, artifact and release boundaries, evaluator algorithms, baseline interfaces, and an OG-PA deployment-aware case-study slot. The appendix does not define an auxiliary leaderboard; instead, it records the diagnostics, artifact constraints, and implementation details that delimit the main empirical claims.

Evidence map. The headline reversal claim is supported by Tables 2, 3, 19, and 21, together with Figures 2, 3, 5, 12, and 13. Cost sensitivity is documented in Tables 11–12 and Figure 6; stress robustness is documented in Table 18 and Figure 11. Artifact and reproducibility boundaries are specified by the result-completion checklist, release tables, and SHA256 manifest. TABL-lite, iTransformer-lite, OG-PA diagnostics, HftBacktest fixed-action audits, and target-family recipes are excluded from headline leaderboard claims unless the corresponding normalized evidence is explicitly reported.

A.1 Motivating Slice and Contract Tables

This subsection preserves the detailed contract, motivating-slice, baseline, and audit tables that were removed from the compressed main text. The main paper now keeps only four core tables; the supporting tables below retain the same information for reproducibility and independent inspection.

Table 5 links the completed motivating slice to the frozen benchmark contract: the same schema and label logic are expanded from one BTCUSDT slice to the 90-day, two-asset, two-venue release.

The two blocks show why overlap control is essential. Overlap-allowing rows can inflate apparent prediction quality, while the de-overlapped rows motivate the execution-cost test under a defensible comparison.

Table 7 is the smallest concrete example of the Prediction–Execution Gap: predictive quality does not prevent severe post-cost collapse.

Table 8 defines the stable data object for comparable $R0$-to-$R4$ evaluation across model families.

The split is chronological to avoid look-ahead leakage: training fits parameters and preprocessing, validation selects thresholds and hyperparameters, and the final test window is reserved for report-card claims.

The ladder is the organizing principle of the paper: $R0$ records the selection proxy, while $R4$ records the cost-, risk-, and stress-aware deployment verdict.

Table 11 records the updated cost-sensitivity boundary. The 3 bps setting remains the headline deployment setting, but the latest result workbook also includes six-panel 0/1/3/5/10 bps leader and method-result rows for available core methods. Future target baselines and wrapper variants still need the same normalized cost-ladder treatment before they can be promoted to completed release rows.

Table 12 strengthens the main claim because the reversal is not a single 3 bps artifact. Costs are an evaluation axis: at 0 bps the top rows are all positive and winner changes are absent, while 1–10 bps introduce winner changes and eventually make every top row negative at 10 bps.

The report card is deliberately multi-axis so that prediction signal, deployment utility, conclusion drift, shift behavior, and reproducibility cannot be collapsed into a misleading single scalar.

Table 14 maps evaluation questions to method families: anchors set floors, heuristics test simple LOB signals, prediction-first rows test offline leaderboards, wrappers test interface alignment, policies target utility directly, and HftBacktest audits execution assumptions.

The audit rows show that representation choices and label overlap can materially change predictive metrics; only de-overlapped rows support clean protocol-controlled comparisons.

Table 16 places prediction and execution side by side for the motivating slice: nontrivial DA/RIC coexists with sharply negative 3 bps Sharpe.

Item	Motivating slice	Frozen contract
Assets / venues	BTCUSDT on Binance	BTCUSDT and ETHUSDT on Binance and OKX
Time window	2025-10-12, 2025-10-17, 2025-10-26, and 2025-10-27 UTC	2025-09-28 to 2025-12-26 UTC
Feeds	Licensed incremental L2 updates plus synchronized trades	Licensed incremental L2 updates and trades under the same canonicalized feed schema
Benchmark view	Top-10 reconstructed LOB, 1s bars, and volume bars	Internally reconstructed LOB, 1s bars, volume bars, and execution logs; public artifacts expose schemas and normalized evidence rather than market feeds
State / labels	OFI, toxicity, and forward log-return labels at 5/20/50 bars	Same state-construction logic and 5/20/50-bar predictive horizons
Paper role	Completed motivating evidence slice	Canonical 90-day benchmark release and report-card contract

Table 5: Diagnostic analysis. Completed motivating slice versus the frozen PEG-Bench contract. This historical slice motivates the benchmark but is not part of the six-panel official main result.

Model	DA	RIC	Notes
Overlap-allowing			
Ridge (tokens)	0.624	0.306	Hist. optimistic; seq 96, vocab 64
HistGBM (tokens)	0.691	0.586	Hist. optimistic; seq 96, vocab 64
LightGBM (LOB)	0.789	0.696	Stride 1; overlapping labels
Kairos (64-code)	0.594	0.146	Hist. optimistic; `return_5`
DeepLOB (time bars)	0.506	0.027	Hist. optimistic; 3 epochs, seq 50
Leakage-controlled			
Ridge (tokens)	0.479	0.048	De-overlapped; seq 64, vocab 64
HistGBM (tokens)	0.090	0.519	De-overlapped; seq 64, vocab 64
LightGBM (LOB)	0.648	0.262	Stride 50; no label overlap
Kairos (64-code)	**0.618**	**0.328**	**Stride 50; no label overlap**
DeepLOB (time bars)	0.506	0.027	Reused architecture on the de-overlapped slice

Table 6: Diagnostic analysis. Motivating-slice baselines under two audit settings. These rows are historical diagnostics, not completed 90D leaderboard evidence.

Table 19 is an aggregate robustness check over the six completed panels. The resampling unit is the panel, not the trading day or action path, so it should be interpreted as evidence that the six-panel aggregate is not dominated by a single panel.

Table 20 records the latest TABL-lite and iTransformer-lite supplemental rows without promoting them to official completed baselines. The useful signal is narrow: iTransformer-lite shows nontrivial $R0$ signal on volume panels, while TABL-lite is weaker and near-zero on time panels. The claim boundary is the important part of the table: these are benchmark-native lite rows for interface compatibility, not official external-repository reproductions and not clean-active leaderboard evidence.

Table 21 sharpens the uncertainty interpretation. The identity and ranking-gap evidence supports the Prediction–Execution Gap, but most positive $R4$ Sharpe point estimates should not be described as statistically stable positive utility because their day-block intervals cross zero.

Figure 4 complements Figure 1 by separating implementation detail from the main thesis diagram. It shows how prediction-first and policy-learning methods enter the same executable action interface,

Cost (bps)	Mean PnL (per bar)	Sharpe	Annual return
3	-5e-6	-98.6	-1.71
1	-1e-6	-70.0	-0.39

Table 7: Diagnostic analysis. Completed cost points on the motivating slice. These rows are historical diagnostics and should not be mixed with the official 90D cost-ladder summary.

Axis	Value	Benchmark role
Assets	BTCUSDT; ETHUSDT	Multi-asset coverage for ranking and robustness tests
Venues	Binance; OKX	Multi-venue comparison under shared evaluator rules
Licensed feeds	Incremental L2 + trades	Replayable microstructure and execution context used internally under authorized access
Window	2025-09-28 to 2025-12-26 UTC	90-day contract covering heterogeneous regimes
Units	Licensed-feed reconstruction; reconstructed LOB; 1s bars; volume bars; execution logs	Shared benchmark objects across model families; raw and reconstructable market feeds are not redistributed

Table 8: Audit-only specification. Frozen PEG-Bench data contract.

how the report card is decomposed into auditable layers, and why OG-PA is treated as a deployment-aware method slot without altering the completed leaderboard claim.

A.2 Additional 90-Day Result Figures

This subsection provides additional visual summaries generated directly from `peg_bench_90d_results_workbook.xlsx` and the derived audit tables in `audit_artifact/data/derived_tables/`. The winner-shift and RankIC-versus-Sharpe figures are promoted to the main text; the remaining figures below provide gap, heatmap, risk, wrapper, stress, and uncertainty views. These figures do not introduce additional experiments; they re-express the same R0, R4, gap, wrapper, stress, and uncertainty evidence used by the paper tables.

Figure 5 visualizes the same official gap metrics as Table 3. It should be read panel by panel: RR@1 answers whether the offline top method remains top under execution realism, and Ranking Gap answers whether the broader ordering is preserved. The consistent negative ranking behavior is the visual evidence that the gap is not confined to one horizon.

Figure 6 complements Tables 11–12. The left panel shows that costs quickly change the top selected method, while the right panel shows that mean top Sharpe moves from strongly positive without costs to negative under higher frictions. This supports the paper's benchmark framing: execution cost is not a nuisance parameter, but part of the evaluation object.

Figure 7 is the easiest way to audit active deployability. The exclusion of zero-action anchors keeps the figure focused on active methods, while hatched cells make evidence gaps visually distinct from zero-valued performance.

Figure 8 shows that rank drift is not only a top-1 event. A method can be consistently high under RankIC and still move down once the same forecasts are converted into cost-bearing actions. This view complements the official gap table by showing where individual methods move in the clean-active ranking.

Figure 9 should be read as a four-dimensional report card. A point with large predictive signal can still be unattractive if it sits in a high-turnover, high-drawdown, low-Sharpe region. It makes the paper's central design choice visible: prediction, cost exposure, risk, and realized utility are distinct axes that should not be collapsed into one offline score.

Split	UTC dates	Days	Use
Train	2025-09-28 to 2025-11-26	60	Fit standardization, tokenizers, codebooks, and supervised predictors
Val.	2025-11-27 to 2025-12-11	15	Hyperparameter selection, early stopping, and threshold tuning
Test	2025-12-12 to 2025-12-26	15	Frozen final report card, stress analysis, and leaderboard reporting

Table 9: Audit-only specification. Chronological split for PEG-Bench (test window frozen).

Lvl.	Stage	Definition
$R0$	Frictionless prediction	Offline predictive metrics only
$R1$	Fees only	Add exchange fees
$R2$	Fees + slippage	Add execution frictions
$R3$	Fees + slippage + risk	Add drawdown or position constraints
$R4$	Full realism	Add shift-aware stress evaluation

Table 10: Audit-only specification. PEG-Bench realism ladder (main-text rankings anchored to $R4$).

Cost (bps)	Current 90D artifact status	How to read it in this report
0	Populated in the 90D cost-ladder files for available core methods	Frictionless endpoint; usually favors LightGBM but is not the headline deployment setting.
1	Populated in the 90D cost-ladder files for available core methods	Low-friction sensitivity point; several panels already change winner relative to $R0$.
3	Populated headline setting for the completed 90D report card	Used by the clean active $R4$ leaderboards, wrapper audit, stress diagnostics, and generated appendix figures.
5	Populated in the 90D cost-ladder files for available core methods	Higher-friction sensitivity point; useful for testing whether a 3 bps reversal is a threshold artifact.
10	Populated in the 90D cost-ladder files for available core methods	High-friction stress point; all completed panels have negative top Sharpe at this cost.

Table 11: Supplemental analysis. Cost-ladder evidence status for the current 90D public artifact.

Cost (bps)	Winner changes	Positive top rows	Mean top Sharpe	Interpretation
0	0	6	1.702	Frictionless endpoint; the offline winner is not yet penalized by execution costs.
1	5	6	0.180	Low costs already change most top methods while preserving positive top Sharpe.
3	4	3	-0.063	Headline setting; winner identity often changes and average top utility turns slightly negative.
5	4	2	-0.129	Higher costs preserve rank instability and reduce positive utility cases.
10	4	0	-0.756	High-friction stress; all completed panels have negative top Sharpe.

Table 12: Six-panel cost-ladder summary for available core methods. The table aggregates the populated 0/1/3/5/10 bps rows from the latest result workbook and should be read as appendix sensitivity evidence, not as a replacement for the 3 bps headline table.

Layer	Main-text metrics	Appendix extensions	Benchmark role
Prediction	RankIC; directional accuracy	MSE/MAE; calibration when available	Prediction proxy
Execution	Post-cost Sharpe; MaxDD; turnover; Calmar	Net return; CVaR@95; average holding period	Deployment utility
Gap	Kendall τ; Reversal Rate@k	Spearman ρ; pairwise inversion rate	Conclusion drift
Robustness	Worst-regime Sharpe; stress-day survival	Regime variance; drawdown violation rate; recovery time	Shift fragility
Auditability	Mean $\pm$ 95% CI; shared split protocol	Seed sensitivity; runtime; split stability	Audit trail

Table 13: Audit-only specification. PEG-Bench report card.

Group	Methods	Benchmark role	Status in this paper
Anchors	No-trade; buy-and-hold	Risk floor and market beta check	Frozen in protocol; deterministic evaluator rows
LOB heuristic	OFI / queue-imbalance threshold	LOB-domain heuristic baseline	Frozen full-release target; validation-frozen rule
Low-capacity ML	Ridge; ElasticNet	Linear predictor sanity check	Ridge populated; ElasticNet reserved for robustness
Tabular ML	LightGBM	Main tabular predictor and leakage audit	Populated in both overlap and de-overlap audits
LOB neural models	DeepLOB; TABL	Canonical LOB neural and attention baselines	DeepLOB populated; TABL frozen for the full-release panel
Strong offline predictor	Kairos-64	Prediction-first stress case	Populated as the main completed collapse example
Financial/generic TS	Kronos; iTransformer	Financial pretraining and top-conference generic TS controls	Frozen full-release targets; TimesFM retained as an appendix alternative
Policy learning	PPO; SAC	Direct execution-aware policy baselines	Frozen full-release targets; scored on $R4$ utility, risk, and stress metrics
Execution-aware wrapper	Cost-threshold plus turnover penalty	Ablation for closing the prediction–execution gap	Frozen wrapper target for all prediction-first models
Audit-only	HftBacktest	External replay and latency/queue audit	Appendix audit tool; excluded from model ranking

Table 14: Audit-only specification. PEG-Bench method panel and current evidence status. Rows marked as frozen targets define the full-release benchmark panel and become completed empirical evidence only when explicitly populated in result tables.

Config	RIC	Notes
Tokenizer		
512 codes	0.012	Codebook collapse
64 codes	0.146	Balanced usage
LightGBM stride		
1	0.696	Label overlap
50	0.262	No overlap
Kairos		
stride=50	**0.328**	De-overlapped

Table 15: Diagnostic analysis. Audit checks for representation and overlap control. These historical diagnostic rows do not enter the completed 90D leaderboard.

Model	DA (%)	RIC	Sharpe @ 3 bps
Kairos	61.8	0.328	-98.6
LightGBM	64.8	0.262	-112.3

Table 16: Diagnostic analysis. Cross-metric evidence under 3 bps on the motivating slice. This table is not an official 90D main-result table.

Role	Method / scope	Raw Sharpe	Best Sharpe	Best turnover	Evaluation use
Wrapper audit	Time h20 Kairos-64	-	0.9270	122.002	Best exported smoothing wrapper; outside clean leaderboard.
Wrapper audit	Volume h20 LightGBM	-16.4700	0.6290	3.760	Negative-to-positive wrapper flip; outside clean leaderboard.
Wrapper audit	26 panel-method rows	-	-	-	3 positive point-estimate rows and 2 negative-to-positive flips.
Formal diagnostic	OG-PA no ACI	-7.1614	-7.1614	2.000	Diagnostic method slot; outside clean leaderboard.
Formal diagnostic	OG-PA ACI	-5.0909	-5.0909	1.997	Diagnostic method slot; outside clean leaderboard.
Formal diagnostic	OG-PA low-turnover	0.0000	0.0000	0.000	Zero-action diagnostic; not a headline win.
Formal diagnostic	OG-PA conservative	-5.0908	-5.0908	1.999	Diagnostic method slot; outside clean leaderboard.

Table 17: Execution-aware wrapper and formal-method rows separated from the clean ranked R4 leaderboard. Wrapper rows summarize the latest raw-vs-best audit and should not be read as clean active leaderboard replacements.

View	h	Method	Worst-regime Sharpe	Stress survival
Time	20	DeepLOB	-8.5369	1.0000
Time	20	Kairos-64	-3.6498	1.0000
Time	20	LightGBM	-0.8719	1.0000
Volume	20	Kairos-64	-4.8038	1.0000
Volume	20	LightGBM	-0.6734	1.0000

Table 18: Stress diagnostics for representative horizon-20 methods in the completed 90D release.

Metric	Estimate	95% CI	Unit
Winner-change rate	0.667	[0.333, 1.000]	fraction
Mean top R4 Sharpe	-0.063	[-0.237, 0.090]	Sharpe
Mean Metric Gap	-0.238	[-0.667, 0.162]	correlation
Mean Ranking Gap	-0.541	[-0.852, -0.259]	Kendall tau
Mean RR@1	1.000	[1.000, 1.000]	fraction
Negative Ranking Gap fraction	1.000	[1.000, 1.000]	fraction

Table 19: Panel-bootstrap uncertainty over the six completed 90D panels. This aggregate check resamples panels, not days; it supports robustness auditing but does not replace day-block confidence intervals from raw action paths.

Panel	Supplemental row	n	DA	RankIC	Claim boundary
Time h5	TABL-lite	11	0.436 ± 0.080	0.021 ± 0.006	Benchmark-native lite row; not an official TABL reproduction.
Time h5	iTransformer-lite	11	0.533 ± 0.043	0.035 ± 0.012	Benchmark-native lite row; not an official iTransformer reproduction.
Time h20	TABL-lite	17	0.526 ± 0.035	0.009 ± 0.005	Benchmark-native lite row; not an official TABL reproduction.
Time h20	iTransformer-lite	17	0.519 ± 0.027	0.016 ± 0.006	Benchmark-native lite row; not an official iTransformer reproduction.
Time h50	TABL-lite	14	0.504 ± 0.008	0.002 ± 0.003	Benchmark-native lite row; not an official TABL reproduction.
Time h50	iTransformer-lite	14	0.506 ± 0.006	0.004 ± 0.005	Benchmark-native lite row; not an official iTransformer reproduction.
Volume h5	TABL-lite	11	0.528 ± 0.012	0.067 ± 0.002	Benchmark-native lite row; not an official TABL reproduction.
Volume h5	iTransformer-lite	11	0.596 ± 0.007	0.239 ± 0.002	Benchmark-native lite row; not an official iTransformer reproduction.
Volume h20	TABL-lite	20	0.527 ± 0.011	0.083 ± 0.007	Benchmark-native lite row; not an official TABL reproduction.
Volume h20	iTransformer-lite	20	0.559 ± 0.031	0.138 ± 0.082	Benchmark-native lite row; not an official iTransformer reproduction.
Volume h50	TABL-lite	11	0.535 ± 0.002	0.084 ± 0.002	Benchmark-native lite row; not an official TABL reproduction.
Volume h50	iTransformer-lite	11	0.563 ± 0.002	0.155 ± 0.002	Benchmark-native lite row; not an official iTransformer reproduction.

Table 20: Supplemental benchmark-native lite target rows from the latest workbook. Values are mean rows over the exported lite runs. These rows test interface compatibility for LOB-attention and generic time-series families, but they are not official external-repository reproductions and are excluded from the completed clean-active leaderboard.

Panel	$R4$ winner	Table 2 Sharpe p.e.	Day-block 95% CI	Conservative interpretation
Time h5	LightGBM	-0.432	$[-0.630, -0.239]$	Negative utility is stable under day-block resampling.
Time h20	Kairos-64	0.205	$[-0.137, 0.613]$	Positive point estimate is not sign-stable because the interval crosses zero.
Time h50	Kairos-64	-0.207	$[-0.529, 0.096]$	Utility remains near or below zero under day-block resampling.
Volume h5	Kairos-64	0.069	$[-0.174, 0.325]$	Positive point estimate is not sign-stable because the interval crosses zero.
Volume h20	LightGBM	-0.072	$[-0.331, 0.012]$	Utility remains near or below zero under day-block resampling.
Volume h50	Kairos-64	0.056	$[-0.256, 0.378]$	Positive point estimate is not sign-stable because the interval crosses zero.

Table 21: Day-block confidence intervals for clean active $R4$ winners. The point-estimate column matches Table 2; intervals use 15 test-day blocks and 1000 bootstrap draws from the latest result workbook.

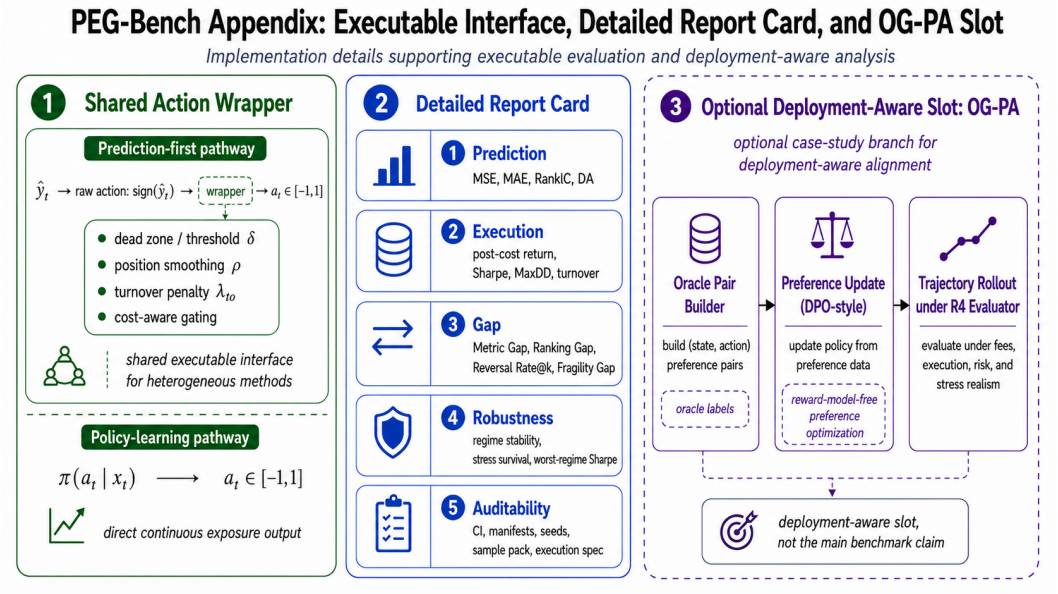

Figure 4: Appendix overview of executable interfaces and diagnostic slots. The left block specifies the shared forecast-to-action and policy-learning pathways used by the benchmark evaluator, the middle block decomposes the report-card layers used throughout the appendix, and the right block identifies OG-PA as a deployment-aware case-study branch rather than a main benchmark claim.

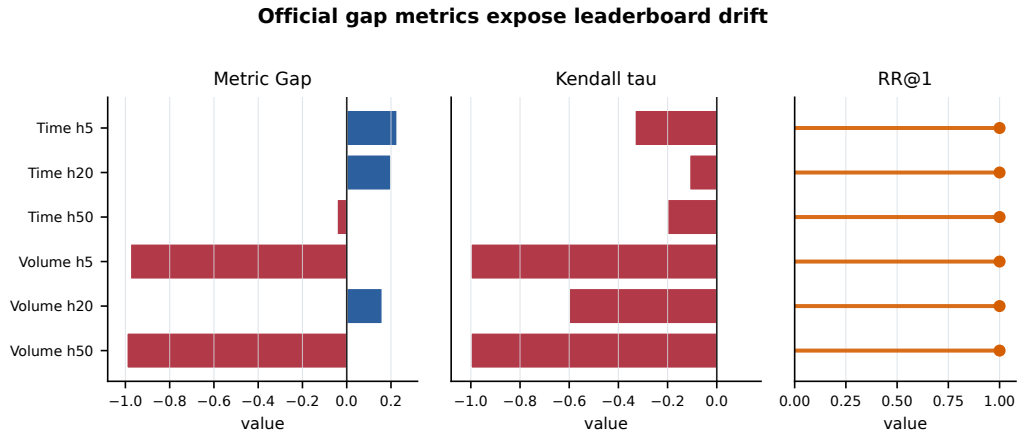

Figure 5: Official main result. Prediction–Execution Gap metrics by panel. Reversal Rate@1 is one in all six panels, while Ranking Gap is negative across the completed panels, indicating that predictive ranking and execution ranking disagree under the shared R4 evaluator.

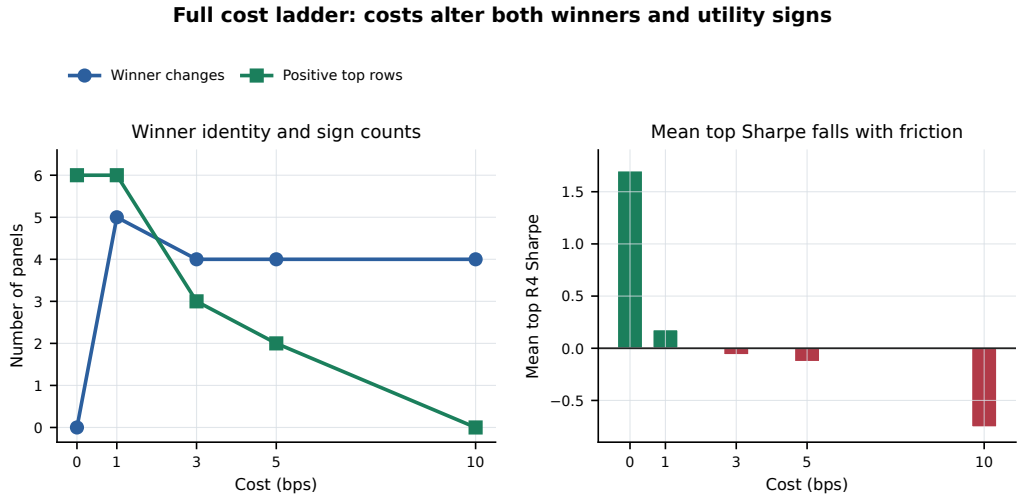

Figure 6: Supplemental analysis. Cost-ladder summary for available core methods. Winner changes, positive top-row counts, and mean top Sharpe are aggregated over the six completed panels at 0/1/3/5/10 bps, showing that the 3 bps headline setting is one point on a broader friction axis rather than an isolated operating point.

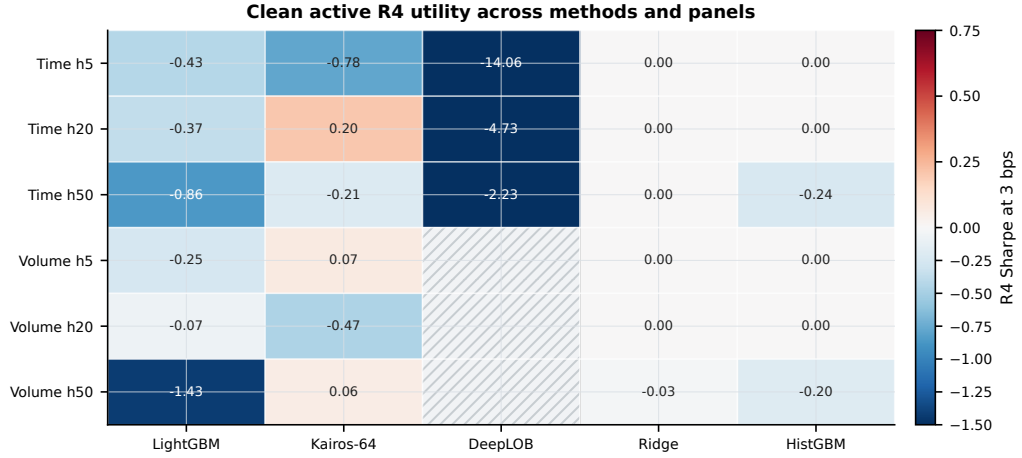

Figure 7: Clean-active result. R4 Sharpe heatmap. The colormap is centered at zero, warm colors indicate positive point estimates, cool colors indicate negative point estimates, and hatched gray cells mark method-panel pairs not populated in the current 90-day artifact.

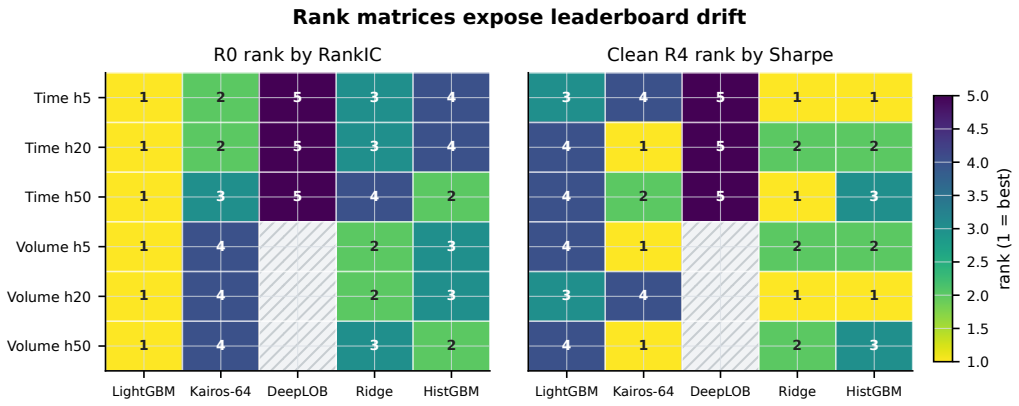

Figure 8: Diagnostic analysis. Clean-active rank-drift matrices. The left matrix ranks methods by R0 RankIC; the right matrix ranks the same populated rows by R4 post-cost Sharpe after removing zero-action anchors. This diagnostic view complements the official Ranking Gap in Table 3, which is computed under the benchmark's deterministic gap protocol.

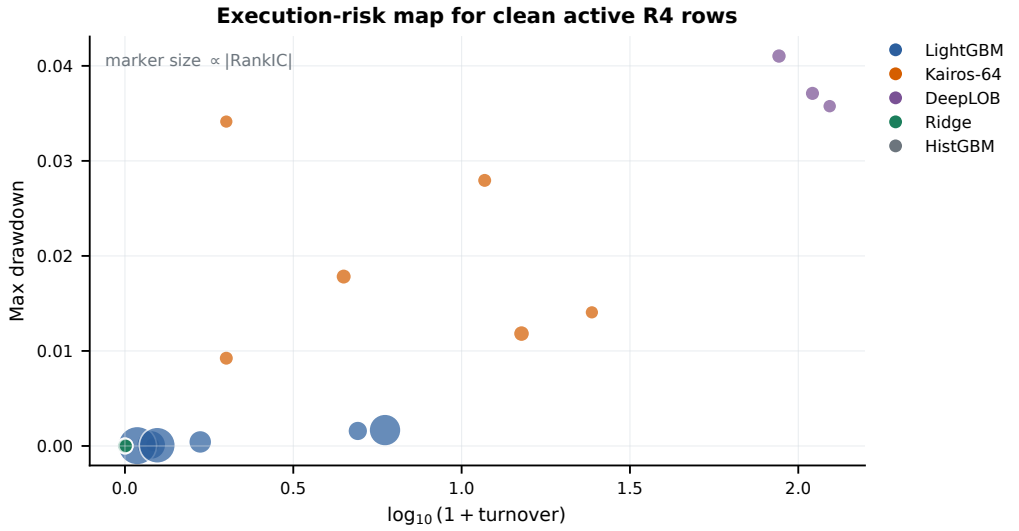

Figure 9: Diagnostic analysis. Execution-risk map for clean-active R4 rows. Turnover is shown on a log scale, drawdown is on the vertical axis, color encodes post-cost Sharpe, and marker size tracks absolute RankIC. The map separates predictive strength from execution risk.

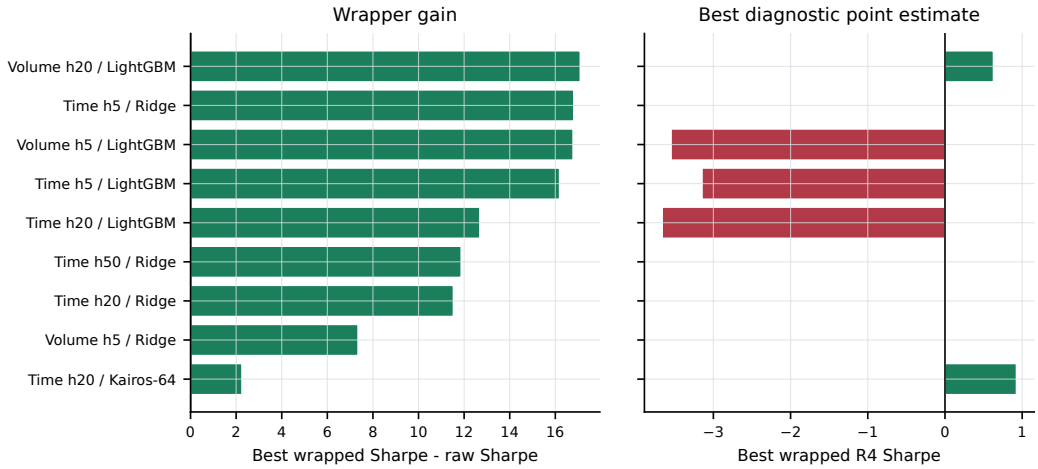

Figure 10: Diagnostic analysis. Wrapper audit across the largest action-interface changes. The left panel shows the best wrapped-minus-raw Sharpe gain, and the right panel shows the best wrapped Sharpe point estimate. These rows audit action-interface sensitivity and are reported separately from the clean-active leaderboard.

Figure 10 summarizes the largest action-interface changes in the latest 26-row wrapper audit. Large gains can come from suppressing costly actions, but only a small number of rows become positive point estimates. This is why the paper interprets wrapper results as interface sensitivity rather than as a new official leaderboard.

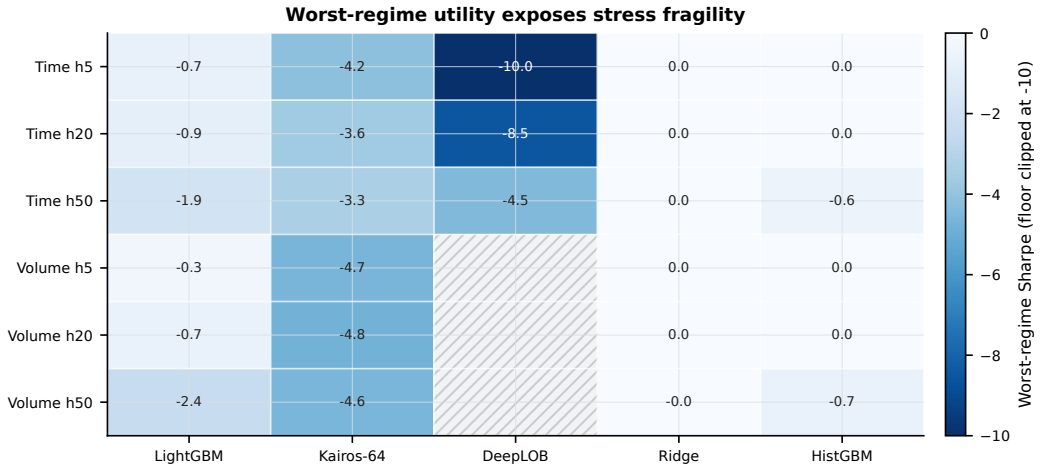

Figure 11: Diagnostic analysis. Worst-regime Sharpe heatmap. The colormap is centered at zero and clipped at -10 for readability, annotations report unclipped values, and hatched gray cells mark method-panel pairs not populated in the current artifact.

Figure 11 reports adverse-regime utility rather than average utility. Negative cells identify methods whose action paths become especially poor under the selected stress slices, even when their average or predictive metrics look acceptable. This supports the benchmark requirement that stress robustness be reported alongside standard post-cost utility.

Figure 12 is the main visual guardrail against overclaiming positive deployment utility. The rank-reversal evidence is stable at the panel level, but most positive winner-Sharpe point estimates cross zero under day-block resampling. The benchmark claim should therefore be read as a claim about evaluation-induced rank and verdict drift, not as a claim that all positive R4 estimates are sign-stable.

Figure 13 shows how the headline conclusions vary when completed panels are resampled. The intervals are useful for auditing whether the six-panel aggregate is dominated by a single panel; the latest workbook also includes day-block/action-path bootstrap files, which support the conservative reading that ranking drift is more stable than the sign of small positive post-cost Sharpe estimates.

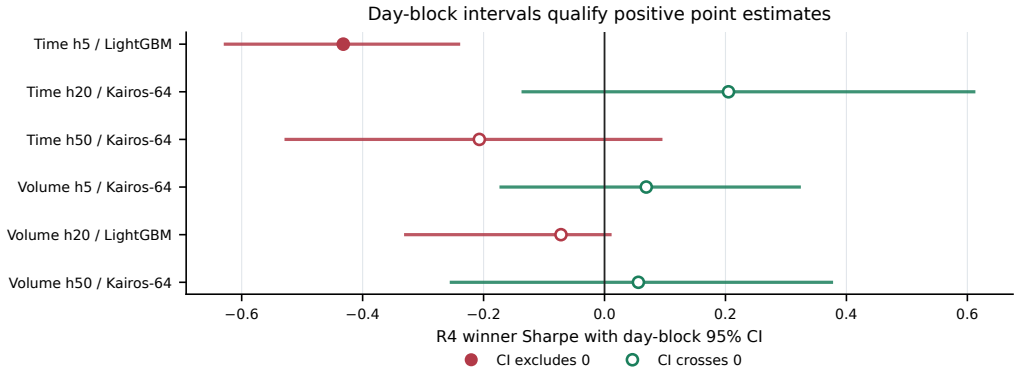

Figure 12: Supplemental analysis. Day-block confidence intervals for the clean-active R4 winners. Filled markers indicate intervals that exclude zero, while open markers indicate intervals that cross zero; the figure supports the conservative interpretation that positive winner Sharpe values are point estimates unless the interval excludes zero.

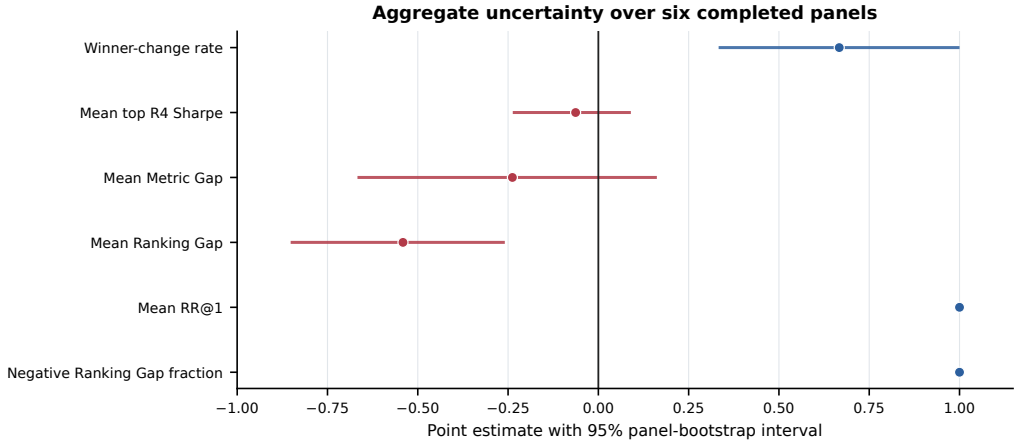

Figure 13: Supplemental analysis. Panel-bootstrap uncertainty over the six completed panels. Intervals are computed by resampling completed panels, not individual days or action paths, so this is an aggregate robustness check.

B Claim-to-Evidence Completion Checklist

This appendix records the complete result checklist for the current 90-day public artifact. The goal is to make the empirical boundary auditable: a row marked *Done* has normalized evidence in `peg_bench_90d_results_workbook.xlsx` or `audit_artifact/data/derived_tables/` and appears in a report table or figure; a row marked *Partial* has usable evidence but does not yet cover the full-release benchmark scope; a row marked *Target* is part of the frozen benchmark contract but is not used as completed evidence in this report.

Table 22 is the artifact-level gate for data claims. A row marked Done means the corresponding data or protocol object is present in the normalized release and is used by the manuscript. It also clarifies that raw-feed reconstruction and broader execution logs require authorized market-data access, while the public artifact supports inspection of the reported 90D evidence without redistributing licensed or reconstructable market data.

Table 23 links each headline claim to a concrete paper table or figure. Its purpose is to make the evidence chain auditable: if a claim is made in the main text, the reader can locate the normalized evidence family and the displayed artifact that supports it. The uncertainty rows are deliberately conservative: panel bootstrap supports the aggregate reversal claim, while day-block intervals prevent positive Sharpe point estimates from being overread as stable positive utility.

The next two tables are a full-release roadmap, not current evidence for the headline leaderboard. They are included so that the public benchmark specification is explicit, while the report claims remain tied to the completed rows in Tables 22 and 23.

Table 24 should be read as a roadmap table. It preserves the intended public-release method scope without implying that every listed family is already part of the completed leaderboard. The current empirical claim remains the completed core result: stable $R0$ selection can fail to support the $R4$ ranking and deployment verdict under the populated evaluator.

Table 25 is the display and audit roadmap for the long-lived public benchmark. It documents what would make the public release broader, but it does not expand the claim boundary of the current report. Supplemental and audit rows remain transparency material unless their source coverage, protocol-clean status, and reproduction commands match the clean active leaderboard standard.

The claim boundary is therefore simple. The main paper may claim only rows marked *Done*, with the explicit caveats attached to *Partial* rows. Rows marked *Target* are frozen benchmark obligations for the full release and should not be described as completed results unless their normalized evidence, table, figure, and reproduction command are added to the artifact.

Required item	Current evidence	Paper use	Status
90-day data scope	Status file reports BTCUSDT and ETHUSDT on Binance and OKX, with 360 time-bar asset-days and 360 volume-bar asset-days passing the guard.	Supports Table 1, Table 26, and the data-contract claim.	Done
Chronological split	Frozen 60/15/15 train/validation/test split over 2025-09-28 to 2025-12-26 UTC.	Supports Table 9 and all completed leaderboard claims.	Done
Benchmark views and horizons	Time and volume views are populated at horizons 5, 20, and 50 from internally reconstructed licensed feeds; the public artifact releases normalized evidence and non-reconstructable inspection samples rather than raw L2 or execution-log data.	Supports the six completed panels used in Tables 2–3.	Done
Normalized evidence bundle	Release directory contains report cards, R0/R4 leaderboards, gap metrics, stress files, cost-ladder rows, wrapper/formal diagnostics, panel bootstrap, day-block/action-path bootstrap files, and completion matrix.	Supports all generated paper tables and Figures 2–13.	Done
Reproduction scripts	Table and figure generation scripts regenerate the displayed LaTeX snippets and all generated 90D figures from normalized CSV/JSON evidence.	Supports artifact inspection without requiring raw-feed reconstruction.	Done

Table 22: Audit-only specification. Data and protocol checklist for the current 90D artifact.

Result family	Evidence in release	Displayed as	Claim supported	Status
R0 leaderboard	R0 leaderboard and report-card rows over the six completed time/volume by horizon panels.	Table 2; Figure 3.	LightGBM is the strongest $R0$ predictor in all six completed panels.	Done
Clean active R4 leaderboard	Clean R4 leaderboard after separating zero-action safety-floor rows.	Table 2; Figures 2, 7, and 8.	Clean active R4 winner changes in four of six panels; unchanged panels remain negative.	Done
Official gap metrics	Metric Gap, Ranking Gap, and Reversal Rate@1 for the six completed prediction-first panels.	Table 3; Figure 5.	Official RR@1 is 1.0 in every completed panel and Ranking Gap is negative in every panel.	Done
Base R4 and anchors	Unwrapped R4 rows plus no-trade, buy-and-hold, and OFI anchor/heuristic evidence.	Table 14; anchor rows separated from clean active rankings.	Safety-floor and market-beta controls are present but not confused with active model wins.	Done
Execution diagnostics	Wrapper/formal diagnostics and selected execution-risk rows.	Table 4; Table 17; Figure 10.	Wrapper can change a deployment verdict on selected audit rows; formal-method rows are diagnostic only.	Done
Stress robustness	Worst-regime Sharpe and stress survival for completed rows.	Table 18; Figure 11.	Stress survival alone is insufficient; worst-regime utility must also be reported.	Done
Risk map	Turnover, drawdown, post-cost Sharpe, and RankIC shown jointly for clean active R4 rows.	Figure 9.	Predictive strength and execution risk occupy different axes.	Done
Cost-ladder sensitivity	Six-panel 0/1/3/5/10 bps leader and method-result rows for available core methods.	Tables 11–12.	Costs are an evaluation axis: 0 bps is positive and stable, while higher costs induce winner changes and 10 bps makes every top row negative.	Done for available core methods
Aggregate uncertainty	Panel-bootstrap intervals over six completed panels, plus day-block/action-path bootstrap files for appendix audit.	Tables 19 and 21; Figure 13.	Ranking drift is stable at the aggregate level; positive utility signs should be read conservatively when day-block intervals cross zero.	Partial

Table 23: Audit-only specification. Completed evidence and displayed table/figure checklist. This is an evidence-boundary table, not an additional leaderboard.

Method / roadmap status	Public artifact evidence	Public-release completion condition
Ridge, HistGBM, LightGBM, Kairos-64 (Partial)	Populated on all six time/volume horizon panels under the shared evaluator; current compare roots are sufficient for the paper's prediction-first claims and available-core cost ladder.	Full-release status requires a protocol-clean refresh with the final context length and validation-tuned wrappers.
DeepLOB (Partial)	Populated on time-view panels; current implementation is time-bar oriented.	Full-release status requires either volume-view DeepLOB rows or a documented not-applicable boundary.
No-trade, buy-and-hold, OFI (Partial)	Deterministic anchor/heuristic rows are present across the six completed panels and excluded from clean active model wins where appropriate.	Full-release status requires the deterministic anchor and heuristic set declared by the frozen protocol.
Wrapper audit (Partial)	Full wrapper activity audit exports 26 panel-method rows, including Time h20 Kairos-64 best wrapped Sharpe 0.927 and Volume h20 LightGBM best wrapped Sharpe 0.629.	Full-release status requires the same wrapper protocol for each final prediction-first family; wrapper rows remain outside the clean active leaderboard unless explicitly promoted.
OG-PA / ACI case study (Partial)	Volume h20 deployment-aware diagnostic rows are populated; ACI improves over the no-ACI diagnostic row but does not produce a headline win.	General method claims require normalized cross-panel diagnostic evidence under the same evaluator.
ElasticNet and TABL (Partial / Target)	ElasticNet has supplemental rows; TABL-lite is reported only as benchmark-native lite evidence in Table 20, not an official external-repository reproduction.	Official TABL completion requires official implementation provenance, validation protocol, R0/R4 report-card rows, and clean active ranking entries.
Kronos, iTransformer, TimesFM (Target / Supplemental)	iTransformer-lite has supplemental benchmark-native rows in Table 20; Kronos and TimesFM remain target-only without verified normalized R0/R4 artifacts.	Official completion requires bar-compatible inputs, validation-only tuning, and exported R0/R4/gap/stress rows.
PPO and SAC (Target)	Specified as policy-learning targets; no normalized 90D policy outputs in this artifact.	Official completion requires frozen-validation policy outputs scored only by R4 utility, risk, and stress metrics.
HftBacktest audit (Partial)	Fixed-action-path sensitivity rows are present for Time h20 and Volume h20 over LightGBM and Kairos-64; this is not a full raw-L2 queue/latency replay.	Full audit status requires documented latency and queue-position grids with audit-only discrepancy reporting.

Table 24: Audit-only specification. Full-release method roadmap, separate from current evidence. The table records how the public method panel would be completed beyond the current 90D artifact; only normalized rows explicitly linked to Tables 22–23 support current manuscript claims.

Display / roadmap status	Public artifact evidence	Public-release completion condition
Cost-ladder sensitivity for target families (Partial)	Six-panel 0/1/3/5/10 bps rows exist for available core methods.	Full-release status requires the same ladder for target baselines and wrapper variants once normalized action paths exist.
Day-block confidence intervals (Partial)	Panel-bootstrap aggregate intervals plus test-day/action-path bootstrap files over 15 test-day blocks.	Full-release uncertainty status requires the declared authorized order-level and seed-sweep interval package.
Per-asset and per-venue breakdown (Partial)	Per-asset, per-venue, and per-asset-venue report-card sheets exist as appendix robustness evidence.	Full-release display status requires documented source coverage and fast-replay caveats.
Full wrapper matrix (Partial)	Wrapper raw-vs-best table covers 26 panel-method rows.	Full-release status requires wrapper-before/after evidence for the final prediction-first method set.
Full method-family leaderboard (Target)	Completed paper focuses on prediction-first, anchor, heuristic, wrapper, and OG-PA diagnostic evidence.	Full-release status requires normalized R0 and R4 rows for the declared target families.
External execution audit (Partial)	Fixed-action-path HftBacktest sensitivity rows are available, but not a full raw-L2 queue/latency replay.	Full audit status requires fee, slippage, latency, and queue-position sensitivity under a documented external replay configuration.
Split and seed stability (Target)	Current split is frozen and reproducible; seed sensitivity is not fully tabulated.	Full-release status requires the declared seed-sweep and alternate-window evidence while preserving the frozen test window.
Licensed-feed reconstruction audit (Target)	Public artifact contains normalized evidence and non-reconstructable inspection samples, not raw or reconstructable market-data redistribution.	Archival-release status requires authorized raw-feed acquisition metadata, checksums, schema logs, and shard-level reconstruction statistics.

Table 25: Audit-only specification. Full-release display roadmap, separate from current evidence. The rows list additional public-release display and audit obligations; they support headline claims only after their normalized evidence is explicitly populated and linked by the completed-evidence checklist.

Layer	Benchmark contract	Evidence reported in this paper
Data scope	90 days; BTCUSDT and ETHUSDT; Binance and OKX; licensed feeds used internally to construct replayable benchmark views	90D guard passes 360 time-bar and 360 volume-bar asset-days
Split and realism ladder	Strict 60/15/15 chronology; $R0$–$R4$ realism ladder; fixed 0/1/3/5/10 bps ladder	Core report card and cost-ladder results are reported for the six study panels for which source files exist
Prediction-first panel	Shared feature contract, leakage controls, and evaluator	LightGBM, Kairos, DeepLOB, and Ridge/HistGBM-style controls are reported when implemented; ElasticNet and lite target baselines are supplementary if normalized rows are available
Anchors and heuristics	No-trade, buy-and-hold, and deterministic microstructure rules	Reported separately from active ranked methods to avoid safety-floor leaderboard pollution
Gap and stress metrics	Metric Gap, Ranking Gap, Reversal Rate@k, Fragility Gap, stress survival	Gap metrics, stress diagnostics, and bootstrap uncertainty are reported for the 90D result set used in this paper
Deployment-aware slot	Preference-based family slot under the shared evaluator	OG-PA/DPO is documented as a diagnostic formal-method slot rather than benchmark-winning evidence
Artifacts	Evaluation protocol, schemas, split manifests, execution specifications, normalized evidence, metadata, and sample/synthetic inspection subset	The public artifact includes CSV/JSON evidence, paper tables, generated figures, cost/bootstrap/wrapper outputs, claim maps, metadata, SHA256 manifest, and non-reconstructable inspection samples; no raw or reconstructable market-data redistribution

Table 26: Benchmark contract and evidence reported in this manuscript.

C Artifact and Release Specification

This section complements Section 2 by specifying which components are inspectable in the public research artifact and which components are deferred to the archival release. The canonical benchmark scope, chronological split, and evaluator contract are fixed in the main text; the material below defines the artifact boundary, release packaging, and implementation anchors.

C.1 Research Artifact vs. Archival Release

PEG-Bench is broader than the completed model panel reported in the main manuscript. Table 26 makes this boundary explicit by separating the fixed benchmark contract from the evidence normalized and reported here.

Table 26 states the evidence boundary used throughout the appendix. The left column records the PEG-Bench protocol, whereas the right column records the evidence normalized and reported in this manuscript. Future extensions, lite baselines, supplemental fast replays, fixed-action audits, and sidecar diagnostics do not enter the main leaderboard unless they appear explicitly in the result tables.

C.2 Release Layout and Reproduction Commands

The public research artifact contains the normalized result snapshot used in this manuscript. In particular, `artifact_snapshot/release/` contains CSV/JSON evidence; `paper_tables/` contains generated LaTeX tables; `figures/generated_90d/` contains regenerated result figures; `artifact_snapshot/scripts/reproduce_tables.py` regenerates the tables from normalized release files; and `scripts/generate_result_figures.py` regenerates Figures 2–13. Cost-ladder, bootstrap, wrapper-matrix, per-asset/per-venue supplemental, and fixed-action execution-audit files are included when their source CSV/JSON artifacts are present. The public artifact does not re-

Release component	Location	Contents
Root metadata	Dataverse root	`README.md`, `LICENSE.txt`, `CITATION.cff`, `CHANGELOG.md`, `VERSION`
Machine-readable metadata	Dataverse `metadata/`	`croissant.json`, `schema.json`, `dataset_card.md`, `rai_metadata.md`, `provenance.md`, `checksums.sha256`
Split manifests	Dataverse `manifests/`; HF `splits/`	`file_manifest.csv`, `split_manifest.csv`, and train/val/test/stress day lists
Inspection shards / synthetic samples	Dataverse `samples/`; HF `sample/`	Synthetic or non-reconstructable sample shards for schema inspection, loader tests, and smoke evaluation; not raw licensed market data or high-frequency derivatives
Authorized reconstruction outputs	Local authorized workspace only	Processed benchmark views generated by users with authorized market-data access; not included in the public artifact mirror
Execution contract	Dataverse and HF `execution/`	`fee_schedule.yaml`, `slippage_model.yaml`, `risk_limits.yaml`, `simulator_contract.md`
Reports and quickstart	Dataverse `reports/`; HF `reports/`	Report-card template, descriptive statistics, and quickstart notebook
Loaders and verification	Dataverse code/	Release loaders, checksum verification, and evaluation examples

Table 27: Canonical archival release layout.

distribute raw L2 archives, vendor API responses, tick-level trades, order-book updates, or reconstructable high-frequency features. Its purpose is to audit the reported 90-day evidence and reproduce the manuscript tables and figures.

The benchmark is released under the name PEG-BENCH. The code and evaluator-facing artifact are hosted at `https://github.com/computational-decision-lab/peg-bench`. Public access is documented at `https://github.com/computational-decision-lab/peg-bench`. The GitHub repository contains the benchmark tooling, scripts, metadata, tests, and normalized evidence needed to inspect and reproduce the reported tables and figures; the Hugging Face repository serves as a lightweight gated dataset mirror for public inspection. Because the underlying market feeds are licensed third-party data, the public release excludes raw feeds and reconstructable high-frequency market paths. Users with authorized market-data access can regenerate the full benchmark views locally under the same manifests and execution contract.

Table 27 specifies the release layout for long-term benchmark reuse rather than one-time manuscript reproduction. Metadata, manifests, execution specifications, normalized evidence, inspection samples, reports, and verification code are separated so that a result can be audited without ambiguity about the split or evaluator definition. The same layout also clarifies why the public artifact contains normalized evidence and non-reconstructable inspection samples, while full market-path reconstruction remains local to users with authorized raw-feed access.

Consequently, artifact inspection does not require reconstructing the licensed market archive. The public artifact exposes the normalized evidence used in this manuscript, the generated tables and figures, and the commands required for reproduction. From the report-package root, the principal commands are:

```
python3 scripts/generate_latest_result_tables.py

python3 scripts/generate_latest_result_figures.py

latexmk -pdf -interaction=nonstopmode -halt-on-error main.tex
```

The audit snapshot in `audit_artifact/` stores standardized result workbooks, derived CSV tables, generated table snippets, generated figures, generation scripts, source maps, and a SHA256 manifest. The archival release will additionally provide execution specifications, machine-readable metadata,

Artifact component	License / terms status	Disclosure status	Release interpretation
Evaluator code, schemas, manifests	Author-owned / open-source license	Released	Supports independent audit and full reruns by users with authorized market-data access.
Normalized CSV/JSON result evidence	Author-owned derived evidence	Released	Supports paper table and figure reproduction; not a replacement for market data.
Synthetic/sample shards	Author-owned synthetic or non-reconstructable data	Released	Supports format inspection, loader checks, and smoke tests only.
Raw Tardis / venue feeds	Vendor and venue terms	Not released	Users must reacquire the underlying market feeds under applicable terms.
Fine-grained Tardis-derived features	Restricted / potentially reconstructable	Not released	Excluded to avoid substituting for licensed market data.
Third-party baseline repositories	Repository-specific licenses	Linked, not bundled unless compatible	Used via citations, source links, and installation instructions.

Table 28: License and asset-disclosure status of the public artifact.

Experiment group	Observed environment	Runtime / memory	Reproducibility effect
90D prediction-first panels	Remote Ubuntu container; CPU SKU not recovered	Partial	Point estimates are complete; per-run memory and wall-clock logs are missing.
Anchors and heuristics	Same remote container; no GPU observed in read-only audit	Partial	Deterministic rows are low-compute and reproduced from normalized outputs.
Wrapper audit	Same remote container; no GPU observed in read-only audit	Partial	Wrapper results are available; runtime logs are not included.
OG-PA diagnostics	Same remote container; memory blocker noted in audit	Partial with caveats	Diagnostic-only rows; not used as positive headline evidence, and numeric rows should be treated as source-required unless their source artifact is present.
Local table/figure reproduction	Local workstation; normalized CSV/JSON files	Complete	Tables and figures regenerate from the public snapshot where source files are present; unsupported historical motivating-slice numbers are marked as missing-source or historical diagnostics.

Table 29: Compute-resource disclosure status for reproducing and auditing the reported evidence.

provenance records, and synthetic or non-reconstructable inspection samples. Users with appropriate market-data access can generate processed benchmark views locally.

Table 28 is a transparency statement rather than a performance result. It identifies which assets are released, which third-party feeds must be reacquired under external terms, and which repository dependencies are linked rather than bundled. These disclosure boundaries are part of the reproducibility claim: the public artifact supports audit of the reported evidence without redistributing licensed or reconstructable market data.

Table 29 separates result reproduction from full experiment reruns. Local table and figure regeneration is complete from normalized evidence, whereas some remote-run hardware and wall-clock metadata are only partially recovered. The table therefore serves as a compute-provenance disclosure boundary rather than a claim of complete per-run resource accounting.

Repository path	Protocol role	Inspectable component
`src/data/impact_cost.py`	Cost model	Fee plus square-root impact implementation and batch cost computation for $R1$/$R2$
`src/data/04_select_stress_periods.py`	Stress slicing	Percentile-based stress extraction from volatility, return, volume, and regime-shift signals
`src/evaluation/10_evaluate_aci.py`	Auditability	Confidence-interval and uncertainty evaluation under temporal dependence
`src/evaluation/12_plot_pareto_frontier.py`	Audit plots	Risk-return and return-drawdown appendix visualizations
`PEG_Bench_Prediction_Execution_Gap/`	Public artifact	Main report, appendix, figures, and report-card definitions aligned with the frozen contract

Table 30: Implementation anchors for the PEG-Bench protocol.

C.3 Implementation Anchors

Table 30 links the abstract benchmark contract to concrete repository anchors. The listed paths are not an exhaustive code inventory; rather, they show that each major benchmark axis maps to inspectable code or public artifact assets.

The paths map conceptual benchmark components to implementation anchors. Although not exhaustive, the table establishes concrete locations for costs, stress slicing, uncertainty auditing, plotting, and manuscript artifacts, thereby supporting independent inspection of the evaluator contract.

D Evaluator Algorithms and Metric Definitions

D.1 Algorithm A1: Standardized state construction

Input. Licensed L2 updates, trade feed, sampling parameters.

Output. Benchmark state tensor set $\mathcal{S}$.

1. Replay order-book updates to reconstruct the top-N book.
2. Align the trade stream with each reconstructed book state.
3. Sample synchronized states into time bars and volume-clock bars.
4. Compute standardized benchmark features.
5. Form context windows of fixed length.
6. Attach forward labels and audit metadata.
7. Return the state collection $\mathcal{S}$ together with split manifests.

D.2 Algorithm A2: Realism-ladder evaluation

Input. Method m, benchmark states $\mathcal{S}$, cost model C, risk rules $\mathcal{R}$.

Output. Ladder results $\{U_{R0}, \ldots, U_{R4}\}$.

1. Score m offline on predictive metrics to obtain U_{R0}.
2. Apply fees to obtain U_{R1}.
3. Apply fees plus slippage to obtain U_{R2}.
4. Apply fees, slippage, and risk rules to obtain U_{R3}.
5. Evaluate U_{R3} across regime and stress slices to obtain U_{R4}.
6. Export per-level metrics and confidence intervals.

D.3 Algorithm A3: Gap metric computation

Input. Offline scores $S_{\text{off}}(m)$ and full-realism utilities $U_{R4}(m)$ over a method set.

Output. Metric Gap, Ranking Gap, Reversal Rate@k, and Fragility Gap.

1. Compute cross-method correlation between $S_{\text{off}}(m)$ and $U_{R4}(m)$.
2. Rank methods by S_{off} and by U_{R4}.
3. Compute Kendall τ or Spearman ρ between the two rankings.
4. For the chosen k, measure how many methods in the offline top-k drop out of the $R4$ top-k.
5. For each method, compute worst-regime shortfall relative to typical performance.
6. Report gap metrics with clear indication of whether the panel is fully populated.

D.4 Algorithm A4: Stress-slice construction

Input. Session statistics, return paths, spread and liquidity summaries.

Output. Stress subset $\mathcal{E}$.

1. Compute daily or session-level volatility, spread, and liquidity summaries.
2. Flag windows exceeding threshold conditions such as extreme return, spread spike, or liquidity collapse.
3. Merge adjacent flagged windows into stress intervals.
4. Exclude intervals that violate benchmark split rules.
5. Freeze the resulting set $\mathcal{E}$ before model comparison.

D.5 Algorithm A4b: Regime tagging and stress-day selection

Input. Daily asset–venue summaries containing returns, realized volatility, spread statistics, depth measures, and jump statistics.

Output. Regime labels and the benchmark stress-day subset.

1. Compute day-level summary statistics for every asset–venue pair in the frozen benchmark window.
2. Assign regime tags such as `trend_up`, `trend_down`, `high_vol`, and `liquidity_stress` from the frozen rule set.
3. Add a day to the stress candidate pool if it exceeds the benchmark percentile threshold on any frozen stress criterion.
4. Deduplicate candidate days and retain the fixed stress subset used for the benchmark report card.
5. Export regime manifests and `stress_days.txt` together with the summary statistics used to derive them.

D.6 Algorithm A5: Block-bootstrap confidence intervals

Input. Metric samples or per-day evaluation slices $\{x_t\}_{t=1}^{T}$, block length b, number of bootstrap draws B.

Output. Point estimate and 95% confidence interval for a benchmark metric.

1. Partition the evaluation series into contiguous blocks of length b.
2. Draw B bootstrap resamples by sampling blocks with replacement until the original horizon length is recovered.
3. Recompute the target metric on each bootstrap resample.
4. Form the empirical 2.5th and 97.5th percentiles of the bootstrap metric distribution.
5. Return the original point estimate together with the percentile interval and the bootstrap configuration.

D.7 Algorithm A6: Report-card assembly and claim tagging

Input. Metric tables over methods, realism levels, regimes, and costs.

Output. A benchmark report card with explicit claim status per cell or metric family.

1. Populate prediction, execution, gap, robustness, and auditability layers from the evaluated outputs.
2. Mark a metric family as *reported* only if all inputs required by its definition are present under the frozen benchmark contract.
3. Mark partially reported layers when the evidence is sufficient for a bounded claim but not for a benchmark-wide conclusion.
4. Mark unavailable layers as protocol extensions and exclude them from the main-text verdict.
5. Export a main-text table containing verdict-changing metrics and an appendix table containing explanatory and audit metrics.

D.8 Closed-form report-card metric definitions

This subsection states the report-card metrics from Table 13 in explicit mathematical form. Algorithms A6b–A6g provide the corresponding computation procedures. Let T denote the number of evaluated bars; let $\hat{y}_{1:T}$ and $y_{1:T}$ denote predicted and realized forward returns; let a_t denote benchmark actions or exposures; let r_t denote realized returns; let c_t denote per-step execution costs; and let $p_t = a_t r_t - c_t$ denote net step return.

Definition A6.1 (Prediction-layer metrics). Under the frozen evaluation index, the prediction-layer metrics are:

(a) *RankIC*:

$$\mathrm{RankIC} = \frac{\sum_{t=1}^{T} \tilde{r}_t^{\hat{y}} \tilde{r}_t^{y}}{\sqrt{\sum_{t=1}^{T} (\tilde{r}_t^{\hat{y}})^2} \sqrt{\sum_{t=1}^{T} (\tilde{r}_t^{y})^2}},$$

where $\tilde{r}_t^{\hat{y}}$ and $\tilde{r}_t^{y}$ are centered mid-rank transforms of $\hat{y}_t$ and y_t.

(b) *Directional accuracy*:

$$\mathrm{DA} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}[\mathrm{sgn}_0(\hat{y}_t) = \mathrm{sgn}_0(y_t)].$$

where $\mathrm{sgn}_0(x) = 1$ if $x > 0$, $\mathrm{sgn}_0(x) = -1$ if $x < 0$, and $\mathrm{sgn}_0(x) = 0$ if $x = 0$.

(c) *MSE* and *MAE*:

$$\mathrm{MSE} = \frac{1}{T} \sum_{t=1}^{T} (\hat{y}_t - y_t)^2, \qquad \mathrm{MAE} = \frac{1}{T} \sum_{t=1}^{T} |\hat{y}_t - y_t|.$$

(d) *Calibration error* when calibration is reported:

$$\mathrm{CalErr} = \sum_{b \in \mathcal{B}} \frac{n_b}{T} \, |\bar{\hat{y}}_b - \bar{y}_b|,$$

where $\mathcal{B}$ is the frozen bin set, n_b is the bin count, $\bar{\hat{y}}_b$ is the mean predicted value, and $\bar{y}_b$ is the mean realized value in bin b.

Definition A6.2 (Execution-layer metrics). Under the frozen execution contract, the execution-layer metrics are:

(a) *Net return path and equity*:

$$E_0 = 0, \qquad E_t = \sum_{i=1}^{t} p_i, \qquad M_t = \max_{0 \le i \le t} E_i.$$

(b) *Post-cost Sharpe*:

$$\bar{p} = \frac{1}{T} \sum_{t=1}^{T} p_t, \qquad s_p^2 = \frac{1}{T-1} \sum_{t=1}^{T} (p_t - \bar{p})^2, \qquad \mathrm{Sharpe}_{\mathrm{post}} = \frac{\sqrt{A}\,\bar{p}}{s_p + \varepsilon}.$$

(c) *Maximum drawdown*:

$$\mathrm{MaxDD} = \max_{1 \le t \le T} (M_t - E_t).$$

(d) *Turnover*:

$$\mathrm{Turnover} = \frac{1}{T-1} \sum_{t=2}^{T} |a_t - a_{t-1}|.$$

(e) *Net return* and *Calmar*:

$$\mathrm{NetRet} = \sum_{t=1}^{T} p_t, \qquad \mathrm{Calmar} = \frac{A\bar{p}}{\max(\mathrm{MaxDD}, \varepsilon)}.$$

(f) *CVaR@95*:

$$\mathrm{CVaR}_{0.95} = -\frac{1}{|\mathcal{I}_{0.05}|} \sum_{t \in \mathcal{I}_{0.05}} p_t, \qquad \mathcal{I}_{0.05} = \{t : p_t \le q_{0.05}\}.$$

where $q_{0.05}$ is the empirical 5th percentile of $\{p_t\}_{t=1}^{T}$ and $\mathcal{I}_{0.05}$ includes all ties at that threshold.

(g) *Average holding period*:

$$\mathrm{AHP} = \begin{cases} 0, & J = 0, \\ \frac{1}{J} \sum_{j=1}^{J} (e_j - s_j + 1), & J > 0, \end{cases}$$

where $\{[s_j, e_j]\}_{j=1}^{J}$ are maximal contiguous nonzero same-sign holding segments.

Definition A6.3 (Gap-layer metrics). Under the frozen tie policy, let π_{off} and π_{R4} be strict total orders over the M-method panel obtained by deterministic tie-breaking from the benchmark's primary and secondary keys. Then the gap-layer metrics are:

(a) *Kendall τ*:

$$\tau_K = \frac{C - D}{\binom{M}{2}},$$

where C and D are concordant and discordant pair counts.

(b) *Spearman ρ*:

$$\rho_S = \mathrm{corr}_{m \in \mathcal{M}}(\mathrm{pos}_{\mathrm{off}}(m), \mathrm{pos}_{R4}(m)).$$

(c) *Reversal Rate@k*:

$$\mathrm{RR@}k = \frac{1}{k} \left| \mathrm{Top}_k(\pi_{\mathrm{off}}) \setminus \mathrm{Top}_k(\pi_{R4}) \right|.$$

(d) *Pairwise inversion rate*:

$$\mathrm{PairInv} = \frac{1}{\binom{M}{2}} \sum_{1 \le i < j \le M} \mathbf{1}[(\pi_{\mathrm{off}}(i) - \pi_{\mathrm{off}}(j))(\pi_{R4}(i) - \pi_{R4}(j)) < 0].$$

Definition A6.4 (Robustness-layer metrics). Let $\mathcal{R}$ denote the frozen regime set and let $\mathcal{S}$ denote the frozen stress-slice set. Then the robustness-layer metrics are:

(a) *Worst-regime Sharpe*:

$$\mathrm{WorstSharpe} = \min_{r \in \mathcal{R}} \mathrm{Sharpe}_r.$$

(b) *Stress-day survival*:

$$\mathrm{Survival}_{\mathrm{stress}} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathbf{1}[F_s = 0].$$

(c) *Regime variance*:

$$\mathrm{RegVar}(U) = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} (U_r - \bar{U})^2, \qquad \bar{U} = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} U_r.$$

(d) *Drawdown violation rate*:

$$\mathrm{DDViolRate} = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \mathbf{1}[\mathrm{MaxDD}_s > d_{\max}].$$

(e) *Recovery time*:

$$\mathrm{RecTime} = \begin{cases} 0, & |\mathcal{D}| = 0, \\ \frac{1}{|\mathcal{D}|}\sum_{j\in\mathcal{D}} \left(\tau_j^{\mathrm{rec}} - \tau_j^{\mathrm{dd}}\right), & |\mathcal{D}| > 0. \end{cases}$$

where $\mathcal{D}$ is the set of drawdown events, τ_j^{dd} is the trough time of event j, and τ_j^{rec} is its first recovery time to the previous peak under the benchmark censoring rule.

Definition A6.5 (Auditability-layer metrics). Under the frozen evaluation and resampling protocol, the auditability-layer metrics are:

(a) *Mean and 95% confidence interval*:

$$\bar{q} = \frac{1}{B}\sum_{b=1}^{B} q^{(b)}, \qquad \widehat{\mathrm{CI}}_{95}(q) = \left[Q_{0.025}(\{q^{(b)}\}_{b=1}^{B}),\, Q_{0.975}(\{q^{(b)}\}_{b=1}^{B})\right].$$

(b) *Shared split protocol*:

$$\mathcal{D} = \mathcal{D}_{\mathrm{train}} \sqcup \mathcal{D}_{\mathrm{val}} \sqcup \mathcal{D}_{\mathrm{test}}, \qquad (|\mathcal{D}_{\mathrm{train}}|, |\mathcal{D}_{\mathrm{val}}|, |\mathcal{D}_{\mathrm{test}}|) = (60, 15, 15) \text{ days}.$$

(c) *Seed sensitivity*:

$$\bar{q}_{\mathrm{seed}} = \frac{1}{S}\sum_{s=1}^{S} q_s, \qquad \mathrm{SeedSD}(q) = \sqrt{\frac{1}{S-1}\sum_{s=1}^{S}(q_s - \bar{q}_{\mathrm{seed}})^2}.$$

(d) *Runtime audit*:

$$\mathrm{GPUHours} = \sum_{j=1}^{J} n_j h_j.$$

(e) *Split stability*:

$$\Delta_{\mathrm{split}}(q) = |q^{\mathrm{nbr}} - q^{\mathrm{base}}|, \qquad \Delta_{\mathrm{rank}}(m) = |\,\mathrm{pos}^{\mathrm{nbr}}(m) - \mathrm{pos}^{\mathrm{base}}(m)|.$$

D.9 Algorithm A6b: Predictive metric computation

Input. Prediction vector $\hat{y}_{1:T}$, realized target vector $y_{1:T}$, and optional calibration bins.

Output. RankIC, directional accuracy, MSE, MAE, and calibration diagnostics.

1. Align predictions and realized forward-return targets on the frozen evaluation index and discard any missing pairs. Let T denote the number of retained pairs, let $\mathrm{rk}(\cdot)$ denote the benchmark's mid-rank transform, and define centered ranks $\tilde{r}_t^{\hat{y}} = \mathrm{rk}(\hat{y}_t) - \overline{\mathrm{rk}(\hat{y})}$ and $\tilde{r}_t^{y} = \mathrm{rk}(y_t) - \overline{\mathrm{rk}(y)}$.
2. Compute *RankIC* as

$$\mathrm{RankIC} = \frac{\sum_{t=1}^{T} \tilde{r}_t^{\hat{y}} \tilde{r}_t^{y}}{\sqrt{\sum_{t=1}^{T}(\tilde{r}_t^{\hat{y}})^2}\sqrt{\sum_{t=1}^{T}(\tilde{r}_t^{y})^2}}. \tag{1}$$

3. Define $\mathrm{sgn}_0(x) = 1$ if $x > 0$, $\mathrm{sgn}_0(x) = -1$ if $x < 0$, and $\mathrm{sgn}_0(x) = 0$ if $x = 0$. Compute *directional accuracy* as

$$\mathrm{DA} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}[\mathrm{sgn}_0(\hat{y}_t) = \mathrm{sgn}_0(y_t)]. \tag{2}$$

4. Compute *MSE* and *MAE* as

$$\mathrm{MSE} = \frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2, \tag{3}$$

$$\mathrm{MAE} = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t|. \tag{4}$$

5. If calibration is reported, partition predictions into frozen bins $b \in \mathcal{B}$ with counts n_b, bin means $\bar{\hat{y}}_b$, and realized means $\bar{y}_b$, then summarize binned calibration error as

$$\mathrm{CalErr} = \sum_{b \in \mathcal{B}} \frac{n_b}{T} |\bar{\hat{y}}_b - \bar{y}_b|. \tag{5}$$

6. Export predictive metrics together with the target horizon, bar type, and split manifest used to compute them.

D.10 Algorithm A6c: Execution metric computation

Input. Action or exposure path $a_{1:T}$, realized returns $r_{1:T}$, fee/slippage costs $c_{1:T}$, and risk-rule events.

Output. Post-cost return path, Sharpe, MaxDD, turnover, Calmar, net return, CVaR@95, and average holding period.

1. Convert model scores into frozen benchmark actions or exposures a_t using the evaluator's shared sizing rule.
2. Compute gross and net step returns as

$$g_t = a_t r_t, \tag{6}$$
$$p_t = g_t - c_t. \tag{7}$$

3. Form cumulative equity and its running peak as

$$E_0 = 0, \qquad E_t = \sum_{i=1}^{t} p_i, \qquad M_t = \max_{0 \le i \le t} E_i. \tag{8}$$

4. Compute the post-cost return mean, volatility, and Sharpe ratio as

$$\bar{p} = \frac{1}{T} \sum_{t=1}^{T} p_t, \qquad s_p^2 = \frac{1}{T-1} \sum_{t=1}^{T} (p_t - \bar{p})^2, \tag{9}$$

$$\mathrm{Sharpe}_{\mathrm{post}} = \frac{\sqrt{A}\,\bar{p}}{s_p + \varepsilon}, \tag{10}$$

where A is the annualization constant implied by the bar clock.

5. Compute *MaxDD* as

$$\mathrm{MaxDD} = \max_{1 \le t \le T} (M_t - E_t). \tag{11}$$

6. Compute *turnover* under the benchmark's normalization as

$$\mathrm{Turnover} = \frac{1}{T-1} \sum_{t=2}^{T} |a_t - a_{t-1}|. \tag{12}$$

7. Compute *net return* and *Calmar* as

$$\mathrm{NetRet} = \sum_{t=1}^{T} p_t, \tag{13}$$

$$\mathrm{Calmar} = \frac{A\bar{p}}{\max(\mathrm{MaxDD}, \varepsilon)}. \tag{14}$$

8. Let $q_{0.05}$ denote the empirical 5th percentile of $\{p_t\}_{t=1}^{T}$ and let $\mathcal{I}_{0.05} = \{t : p_t \le q_{0.05}\}$, including all observations tied at the threshold. Compute *CVaR@95* as

$$\mathrm{CVaR}_{0.95} = -\frac{1}{|\mathcal{I}_{0.05}|} \sum_{t \in \mathcal{I}_{0.05}} p_t. \tag{15}$$

9. Let $\mathcal{H} = \{[s_j, e_j]\}_{j=1}^{J}$ denote the maximal contiguous nonzero same-sign holding segments. If $J = 0$, set *average holding period* to 0; otherwise compute it as

$$\mathrm{AHP} = \begin{cases} 0, & J = 0, \\ \frac{1}{J}\sum_{j=1}^{J}(e_j - s_j + 1), & J > 0. \end{cases} \tag{16}$$

10. Export the full execution metric tuple together with the cost level, sizing rule, and risk-rule manifest.

D.11 Algorithm A6d: Robustness and survival metric computation

Input. Regime-labeled or stress-labeled execution results $\{p_t, E_t\}$, regime assignment map, and failure criteria.

Output. Worst-regime Sharpe, stress-day survival, regime variance, drawdown violation rate, and recovery time.

1. Partition the execution path by the frozen regime labels and stress subset.
2. For each regime or stress slice, recompute post-cost Sharpe, cumulative return, volatility, MaxDD, and the chosen utility metric on that slice only.
3. Compute *worst-regime Sharpe* as

$$\mathrm{WorstSharpe} = \min_{r \in \mathcal{R}} \mathrm{Sharpe}_r. \tag{17}$$

4. Let $F_s \in \{0, 1\}$ indicate whether stress slice $s \in \mathcal{S}$ violates the benchmark failure rule. Compute *stress-day survival* as

$$\mathrm{Survival}_{\mathrm{stress}} = \frac{1}{|\mathcal{S}|}\sum_{s \in \mathcal{S}} \mathbf{1}[F_s = 0]. \tag{18}$$

5. Let U_r denote the regime-level utility and $\bar{U} = |\mathcal{R}|^{-1}\sum_{r \in \mathcal{R}} U_r$. Compute *regime variance* as

$$\mathrm{RegVar}(U) = \frac{1}{|\mathcal{R}|}\sum_{r \in \mathcal{R}}(U_r - \bar{U})^2. \tag{19}$$

6. Given the benchmark drawdown threshold $d_{\max}$, compute *drawdown violation rate* as

$$\mathrm{DDViolRate} = \frac{1}{|\mathcal{S}|}\sum_{s \in \mathcal{S}} \mathbf{1}[\mathrm{MaxDD}_s > d_{\max}]. \tag{20}$$

7. Let τ_j^{dd} denote the trough time of drawdown event $j \in \mathcal{D}$ and let τ_j^{rec} denote its first recovery time to the previous peak, censoring unrecovered events according to the benchmark manifest. If $|\mathcal{D}| = 0$, set *recovery time* to 0; otherwise compute it as

$$\mathrm{RecTime} = \begin{cases} 0, & |\mathcal{D}| = 0, \\ \frac{1}{|\mathcal{D}|}\sum_{j \in \mathcal{D}}\left(\tau_j^{\mathrm{rec}} - \tau_j^{\mathrm{dd}}\right), & |\mathcal{D}| > 0. \end{cases} \tag{21}$$

8. Export regime-conditioned and stress-conditioned metrics with the regime manifest and failure-rule version.

D.12 Algorithm A6e: Ranking and inversion metric computation

Input. Offline ranking π_{off}, execution-aware ranking π_{R4}, and a chosen top-k list.

Output. Kendall τ, Spearman ρ, Reversal Rate@k, and pairwise inversion rate.

1. Sort methods by the offline metric to obtain π_{off} and by the execution-aware metric to obtain π_{R4} under the frozen benchmark tie policy. When two methods have identical scores, break ties deterministically using the benchmark's secondary key so that both rankings are strict total orders and the top-k set is unique.

2. Let C and D denote the concordant and discordant pair counts over the M-method panel. Compute *Kendall* τ as
$$\tau_K = \frac{C - D}{\binom{M}{2}}. \tag{22}$$

3. Let $\mathrm{pos}_{\mathrm{off}}(m)$ and $\mathrm{pos}_{R4}(m)$ denote the ordinal positions of method m. Compute *Spearman* ρ as
$$\rho_S = \mathrm{corr}_{m \in \mathcal{M}}(\mathrm{pos}_{\mathrm{off}}(m), \mathrm{pos}_{R4}(m)) \,. \tag{23}$$

4. Compute *Reversal Rate@k* as
$$\mathrm{RR@}k = \frac{1}{k} \left| \mathrm{Top}_k(\pi_{\mathrm{off}}) \setminus \mathrm{Top}_k(\pi_{R4}) \right| . \tag{24}$$

5. Compute *pairwise inversion rate* as
$$\mathrm{PairInv} = \frac{1}{\binom{M}{2}} \sum_{1 \le i < j \le M} \mathbf{1}[(\pi_{\mathrm{off}}(i) - \pi_{\mathrm{off}}(j))(\pi_{R4}(i) - \pi_{R4}(j)) < 0] \,. \tag{25}$$

6. Export ranking metrics together with the exact offline score definition, execution score definition, and candidate method panel used to form the rankings.

D.13 Algorithm A6f: Fragility and gap summary computation

Input. Offline scores $S_{\mathrm{off}}(m)$, full-realism utilities $U_{R4}(m)$, and regime-conditioned utilities $\{U_r(m) : r \in \mathcal{R}\}$.

Output. Metric Gap, Fragility Gap, and per-method fragility ranking.

1. For the populated method panel $\mathcal{M}$, compute *Metric Gap* as
$$\mathrm{MetricGap} = \mathrm{corr}_{m \in \mathcal{M}}(S_{\mathrm{off}}(m), U_{R4}(m)) \,. \tag{26}$$

2. For each method m, compute its typical regime utility as
$$U_{\mathrm{typ}}(m) = \mathrm{median}_{r \in \mathcal{R}} \, U_r(m). \tag{27}$$

3. Compute *Fragility Gap* as
$$\mathrm{FragilityGap}(m) = U_{\mathrm{typ}}(m) - \min_{r \in \mathcal{R}} U_r(m). \tag{28}$$

4. Rank methods by Fragility Gap, with larger values indicating greater regime sensitivity and weaker deployment robustness.
5. Export both the cross-method Metric Gap and the per-method Fragility Gap table together with confidence intervals if the panel size permits them.

D.14 Algorithm A6g: Audit metric computation

Input. Metric values over seeds, splits, regimes, and timing logs.

Output. Confidence intervals, seed sensitivity, split stability, and runtime audit statistics.

1. For each headline metric q, apply the frozen day-level or block-bootstrap procedure with B resamples to obtain the point estimate and 95% confidence interval,
$$\widehat{\mathrm{CI}}_{95}(q) = \left[Q_{0.025}(\{q^{(b)}\}_{b=1}^{B}), \, Q_{0.975}(\{q^{(b)}\}_{b=1}^{B}) \right] . \tag{29}$$

2. Over repeated random seeds $s = 1, \ldots, S$, compute *seed sensitivity* through the seed mean and standard deviation,
$$\bar{q}_{\mathrm{seed}} = \frac{1}{S} \sum_{s=1}^{S} q_s, \tag{30}$$
$$\mathrm{SeedSD}(q) = \sqrt{\frac{1}{S-1} \sum_{s=1}^{S} (q_s - \bar{q}_{\mathrm{seed}})^2}. \tag{31}$$

3. Compute *split stability* on a neighboring split through metric and ranking shifts,

$$\Delta_{\text{split}}(q) = \left|q^{\text{nbr}} - q^{\text{base}}\right|, \tag{32}$$

$$\Delta_{\text{rank}}(m) = \left|\text{pos}^{\text{nbr}}(m) - \text{pos}^{\text{base}}(m)\right|. \tag{33}$$

4. Compute *runtime audit* from accelerator usage as

$$\text{GPUHours} = \sum_{j=1}^{J} n_j h_j, \tag{34}$$

where n_j is the accelerator count and h_j is the wall-clock hours for run segment j; report training time, inference time, and accelerator type alongside this aggregate.

5. Export audit metrics in appendix-only tables so they explain confidence and reproducibility without replacing the main-text deployment verdict.

D.15 Algorithm A7: Oracle-guided preference-pair construction

Input. Candidate trajectories $\{\tau_i\}$, ex-post evaluation tuple $g(\tau_i)$, risk preference condition z, and preference margin δ.

Output. Winner/loser preference pairs $\mathcal{P}$ for OG-PA-style training.

1. Roll each candidate trajectory through the shared evaluator under the chosen condition z.
2. Compute an ex-post deployment tuple $g(\tau_i)$ containing quantities such as post-cost return, drawdown, turnover, and survival.
3. Rank trajectories by the benchmark's deployment preference rule under z.
4. For each candidate pair, retain (τ^w, τ^l) only if the utility gap exceeds margin δ.
5. Filter duplicate or contradictory pairs and export the resulting pair set $\mathcal{P}$ together with the evaluator manifest used to construct it.

D.16 Algorithm A8: Continuous-action preference optimization

Input. Preference pairs $\mathcal{P}$, trainable policy π_θ, frozen reference policy π_{ref}, temperature β, and optimizer configuration.

Output. Aligned checkpoint $\pi_\theta^\star$ and training audit logs.

1. Freeze the reference policy parameters and initialize the trainable policy from the chosen backbone checkpoint.
2. For each minibatch of winner/loser pairs, compute trajectory log-likelihoods under both π_θ and π_{ref}.
3. Form the DPO-style preference loss from the winner/loser log-likelihood ratio difference.
4. Add entropy regularization and any covariance-floor parameterization required for stable continuous-action training.
5. Update θ with the configured optimizer, clipping gradients and logging preference accuracy, logit margin, and action statistics.
6. Save checkpoints together with the evaluator manifest so each aligned model can be reinserted into the shared report card.

D.17 Algorithm A9: Uncertainty-aware pair filtering and deployment gating

Input. Candidate preference pairs, estimated trajectory-outcome intervals, overlap threshold η, and optional action-gating rule.

Output. Filtered pair set $\mathcal{P}_{\text{safe}}$ and an uncertainty-aware deployment policy.

1. Estimate uncertainty intervals for the winner and loser trajectory outcomes with the benchmark's chosen CI or conformal procedure.

2. Reject any pair whose intervals overlap by more than threshold η, or whose estimated margin is below the benchmark's reliability floor.
3. Record the fraction of rejected pairs and the uncertainty configuration in the run manifest.
4. At deployment time, use the same uncertainty signal to downscale or clip action magnitude in high-uncertainty regions.
5. Re-evaluate the resulting policy under the same $R0$–$R4$ ladder and report any change in execution or robustness columns separately from the base aligned model.

D.18 Algorithm A10: License-aware release-bundle assembly

Input. Normalized evidence files, schemas, manifests, execution specifications, metadata files, evaluator code, and synthetic or non-reconstructable sample subset.

Output. A canonical Dataverse release and a lighter Hugging Face mirror.

1. Assemble the canonical release with schemas, split manifests, execution specifications, normalized evidence, metadata, reports, synthetic or non-reconstructable samples, and helper code.
2. Validate checksums, metadata completeness, and path naming conventions.
3. Assemble the Hugging Face mirror using schemas, split manifests, normalized evidence, execution contracts, and inspection samples.
4. Attach the execution specification and sample subset to both hosts.
5. Document that full benchmark-view reconstruction is performed locally by users with authorized market-data access and is not part of the public artifact mirror.
6. Publish version identifiers and release notes describing any protocol-breaking change.

E Baseline Interfaces and Execution Recipes

This section defines the method-family interfaces used to evaluate PEG-Bench data. Table 31 is the central object: each row states the method family, the data view it consumes, and the action or audit output passed to the shared evaluator. The remaining tables provide guardrails for provenance, scope control, and reproducible implementation anchors. A row is treated as benchmark evidence only when it is explicitly reported in the result tables with source artifacts.

E.1 Method Families Under the Shared Interface

Table 31 defines the operative full-release panel. It is organized around the evaluation object rather than model novelty. Each row records the method family, implementation source, interface to the frozen PEG-Bench contract, and role in interpreting the shift from $R0$ prediction metrics to $R4$ execution realism. The main text reports only the explicitly included subset; lite, smoke, supplemental fast-replay, and audit-only rows are labeled separately.

Table 31: Method families under the shared PEG-Bench interface. Each row reports the baseline source or venue, method type, execution recipe under the shared benchmark contract, and role in the Prediction–Execution Gap analysis. A row counts as benchmark evidence only when it is explicitly reported in the result tables.

Role	Model or method	Source or venue	Method class	PEG-Bench interface	Rationale / status
Risk floor	No-trade	Self-implemented control	Deterministic zero-exposure policy with no learned signal.	Skip training; set $a_t = 0$ for every test bar and pass the path through the shared cost, drawdown, regime, and stress evaluator.	Deployment floor: models that cannot beat doing nothing after costs have no credible $R4$ value. Deterministic anchor.
Market beta	Buy-and-hold	Self-implemented control	Fixed long-exposure control for market drift.	Set $a_t = +1$ on test and replay returns through the common evaluator; always-short is appendix sanity only.	Separates model value from directional market beta. Deterministic anchor.

Continued on next page.

Table 31: Method families under the shared PEG-Bench interface (continued).

Role	Model or method	Source or venue	Method class	PEG-Bench interface	Rationale / status
LOB heuristic	OFI / QI threshold	Microstructure heuristic	Non-ML rule using order-flow imbalance, queue imbalance, spread, and microprice tilt.	Standardize features, select threshold θ on validation under post-cost utility and turnover constraints, freeze θ, and deploy long/short/flat actions.	Tests whether simple LOB-native signals already explain the effect. Included as a benchmark target with a validation-frozen rule.
Low-capacity ML	Ridge / ElasticNet [14]	scikit-learn / JMLR 2011	Linear return predictors with explicit regularization.	Fit flattened top-10 LOB states plus engineered features; select regularization on validation; map $\hat{y}_t$ to raw and wrapped actions.	Distinguishes feature signal from model capacity. Ridge is reported when available; ElasticNet is included as a supplementary target baseline where normalized rows are exported.
Tabular ML	LightGBM [8]	2017	Gradient-boosted decision-tree baseline for tabular features.	Train on tabular LOB features; use de-overlapped stride-50 for leaderboard; keep stride-1 for leakage/overlap audit.	Strong reproducible tabular baseline that supports the claim that protocol choices change offline conclusions. Used in the reported core comparison.
LOB neural	DeepLOB [23]	IEEE TSP 2019	Canonical LOB neural model with convolutional, inception-style, and sequence modules.	Feed reconstructed top-10 LOB windows; train chronologically; map return or direction forecasts to the shared action interface.	Required LOB-domain neural control, not only tabular comparison. Reported when available.
LOB attention	TABL [20]	IEEE TNNLS 2019	Lightweight temporal-attention and bilinear LOB predictor.	Intended official interface: use the DeepLOB-compatible window contract; tune only on validation; evaluate raw and wrapped forecast-to-action variants on test.	Official TABL remains a broader release target in the public artifact. Benchmark-native TABL-compatible or TABL-lite rows are supplementary evidence only and are not described as official reported baseline results.
Stress predictor	Kairos-64	PEG-Bench artifact	Strong prediction-first stress case using volume-clock LOB tokenization, 64-code quantization, and an autoregressive transformer.	Generate volume-clock tokens, predict forward returns, and run raw plus wrapped actions across the 0/1/3/5/10 bps cost ladder and stress slices.	Motivating evidence: high $R0$ RankIC/DA can collapse under realistic execution. Used in the reported stress-case comparison.
Financial FM	Kronos [19]	arXiv 2025	Financial sequence foundation-model baseline for bar-compatible market views.	Intended full-release interface: convert data to volume-bar or OHLCV-like views; choose zero-shot, supervised, or light fine-tuning on validation; avoid licensed L2 tensors unless natively supported.	Broader release target only in the public artifact. The extension pass found a verified-checkpoint/adapter blocker, so Kronos does not contribute a reported leaderboard row in this paper.

Continued on next page.

Table 31: Method families under the shared PEG-Bench interface (continued).

Role	Model or method	Source or venue	Method class	PEG-Bench interface	Rationale / status
Generic TS	iTransformer [11] / TimesFM [2, 4] / Time-MoE [18] / WaveToken [12] / TimeDART [21]	ICLR / ICML time-series baselines	Generic and foundation-style multivariate time-series comparators, not financial LOB-native models.	Intended interface: build multivariate 1s-bar or volume-bar views, train or adapt chronologically, and map forecasts through the common action wrapper.	Benchmark-native iTransformer-lite rows are supplementary only and not official reported rows. TimesFM remains target-only because no verified checkpoint and adapter were available in the extension pass. Other TSFM-style alternatives remain broader release targets unless normalized PEG-Bench $R0/R4$ artifacts are produced.
Policy learning	PPO / SAC via SB3 [17, 5, 16]	PPO arXiv 2017; SAC ICML 2018; SB3 JMLR 2021	Direct policy-learning baselines that emit continuous exposure rather than forecasts.	Intended interface: construct a Gym-style environment with action $a_t \in [-1, 1]$ and reward equal to net return minus cost, turnover, and drawdown penalties; tune on validation and freeze before test.	Broader release target only in the public artifact. The extension pass found environment/checkpoint blockers and no completed > 8 hour frozen-test training run, so PPO and SAC are not part of the reported leaderboard evidence.
Wrapper	Cost-threshold / turnover wrapper	PEG-Bench artifact	Uniform execution-aware post-processor for prediction-first models.	Apply to Ridge, LightGBM, DeepLOB, TABL, Kairos, Kronos, and iTransformer-style models; select threshold, turnover penalty, and smoothing on validation; report raw and wrapped variants.	Tests whether a simple execution interface changes the $R4$ ranking. Claims in this paper are limited to exported wrapper-matrix rows.
Audit only	HftBacktest [7, 6]	GitHub audit tool	External replay engine for fee, latency, slippage, and queue-position sensitivity.	Translate fixed PEG-Bench action paths into documented order rules and replay selected windows under latency and queue-position grids where the audit artifact supports it.	Audits execution realism but is not a model and never enters the leaderboard. Fixed-action latency/queue sensitivity rows are audit-only and are not described as full raw-L2 leaderboard evidence.

Table 31 is an interface map for the report card; it does not imply that every listed family is reported in the public artifact. The non-ML anchors and OFI/QI heuristic produce actions directly and calibrate deployment floors. Reported prediction-first rows, including LightGBM, Kairos, DeepLOB, and implemented linear controls, produce forecasts $\hat{y}_t$ and enter the common forecast-to-action bridge.

E.2 Panel Guardrails

The panel is deliberately compact: a family is retained only when it tests a distinct evaluation question under the shared $R0$–$R4$ interface. All preprocessing, thresholds, wrappers, and hyperparameters are selected on validation only, and every ranked method must emit either a forecast mapped to a_t or a direct action $a_t \in [-1, 1]$. The panel spans multiple method families and publication venues: LightGBM is the tabular baseline, iTransformer is the generic multivariate time-series comparator, TimesFM is a time-series foundation-model alternative, Time-MoE is a recent TSFM reference, WaveToken and TimeDART are time-series representation references, and Kronos is a financial foundation-model target [11, 2, 18, 12, 21, 19]. Recent LOB benchmark work such as LOB-Bench addresses complementary generative order-book evaluation rather than the prediction-to-execution report-card setting studied here [13]. In the public evidence bundle, official target-family rows are reported only if normalized $R0$/$R4$ artifacts exist. Lite, smoke, dependency-resolution, supplemental fast-replay, and audit-only rows are labeled separately. TradeMaster/FinRL-Meta, DPO/ACI, and HftBacktest are cited as related systems, method background, or audit tools rather than ranked LOB baselines.

E.3 Optional and Excluded Candidates

Table 32 lists candidates intentionally kept outside the ranked panel. Exclusion indicates that a candidate is not required for the core PEG-Bench question; it is not a statement of irrelevance.

Candidate	Disposition	Reason
TransLOB / Axial-LOB	Appendix optional	Useful LOB Transformer checks, but overlap with the Kairos/TABL attention role.
TimesFM / Time-MoE / WaveToken / TimeDART [2, 18, 12, 21]	Appendix alternatives	Strong recent TSFM or self-supervised time-series references; kept out of the compact main panel unless normalized PEG-Bench rows are produced.
PatchTST / TimesNet / Autoformer	Not main panel	Adding several generic TS models would shift PEG-Bench toward a forecasting benchmark.
TradeMaster / FinRL-Meta	Related work only	Financial-RL benchmark ecosystems, not single LOB predictors under the PEG-Bench action interface.
sd-PNHP / market-making DRQN	Related or future task	Event-stream generation and market making solve different tasks from exposure ranking.
LOBCAST / LOBFrame / LOB-Bench [13]	Implementation or related-evaluation note	Useful LOB experimentation and generative-evaluation ecosystems, not individual prediction-to-execution leaderboard methods in the public panel.

Table 32: Disposition of optional or excluded baseline candidates.

Table 32 formalizes scope control. Excluded candidates are omitted because they would duplicate an existing role or shift the manuscript from evaluation science toward a broad model zoo. The disposition column preserves the main benchmark focus while leaving explicit paths for appendix or future-release extensions.

E.4 Implementation Anchors

Table 33 centralizes repository links and artifact anchors; formal publications remain in the bibliography.

Component	Implementation anchor	Repository or artifact
Ridge / ElasticNet	scikit-learn linear models	`sklearn linear models`
LightGBM	Official package	`lightgbm-org/LightGBM`
DeepLOB	Public reference implementation	`zcakhaa/DeepLOB`
TABL	Public reference implementation	`viebboy/TABL`
Kronos	Financial sequence FM	`shiyu-coder/Kronos`
iTransformer	Generic multivariate TS comparator	`thuml/iTransformer`
TimesFM alternative	General TS foundation transfer	`google-research/timesfm`
Time-MoE alternative	Sparse-MoE TSFM reference	`Time-MoE/Time-MoE`
PPO / SAC	Stable-Baselines3 policies	`DLR-RM/stable-baselines3`
HftBacktest	Audit-only replay engine	`nkaz001/hftbacktest`
Anchors, OFI, wrapper, Kairos	PEG-Bench artifact code	Released with evaluator, manifests, and execution specifications

Table 33: Implementation anchors for reproducible baseline runs.

Table 33 provides concrete implementation anchors for each retained or target method family. Bibliographic citations establish scientific provenance, whereas repository links establish practical reproducibility paths. Components listed as PEG-Bench artifact code are released with the evaluator and manifests rather than being imported from an external method repository.

E.5 Common Evaluation Loop

Input. Baseline family m, frozen train/validation/test manifests, cost ladder $\mathcal{C} = \{0, 1, 3, 5, 10\}$ bps, realism levels $R0$–$R4$.

Output. A report-card row for m.

1. Build only the views allowed for the family: locally reconstructed LOB views generated under authorized market-data access for LOB-native models, bar views for foundation models, and benchmark features for tabular models.
2. Train or fit m on the training split. Deterministic anchors skip this step.
3. Select hyperparameters on validation using the family-appropriate validation objective stated in the run manifest.
4. Freeze all preprocessing, weights, thresholds, wrappers, and risk controls before the test split.
5. Compute $R0$ prediction metrics when the method emits forecasts.
6. Convert forecasts or policies into test-time actions a_t through the frozen action interface.
7. Run the shared evaluator over all cost levels, regimes, and stress slices to produce $R1$–$R4$ metrics.
8. Export prediction, execution, gap, robustness, and audit metrics with the exact split, seed, cost, and implementation manifest.

E.6 Deterministic Anchors and Microstructure Rules

Input. Reconstructed benchmark states, validation split, test split, candidate threshold grid Θ.

Output. Frozen anchor and heuristic policies.

1. For no-trade, set $a_t = 0$ for every bar and evaluate the path without training.
2. For buy-and-hold, set $a_t = +1$ for every bar; optionally evaluate $a_t = -1$ only as an appendix sanity check.
3. For the microstructure heuristic, compute a normalized signal

$$s_t = w_1 \mathrm{OFI}_t + w_2 \mathrm{QI}_t + w_3 \mathrm{MicropriceTilt}_t - w_4 \mathrm{Spread}_t, \tag{35}$$

   with weights fixed by the benchmark implementation or selected from a small validation grid.
4. Choose $\theta \in \Theta$ on validation using post-cost Sharpe subject to a turnover cap.
5. Freeze θ and deploy

$$a_t = \begin{cases} +1, & s_t > \theta, \\ -1, & s_t < -\theta, \\ 0, & \text{otherwise.} \end{cases} \tag{36}$$

6. Evaluate the frozen policy through the same cost ladder, drawdown rules, and stress slices as learned models.

E.7 Prediction-First Models and Execution-Aware Wrappers

Input. Forecasts $\hat{y}_t$ from Ridge, ElasticNet, LightGBM, DeepLOB, TABL, Kairos, Kronos, iTransformer, or the TimesFM appendix alternative; validation grid for threshold δ, turnover weight λ_{to}, and smoothing ρ.

Output. Raw and wrapped action paths for each prediction-first model.

1. Evaluate the raw predictor at $R0$ with MSE, MAE, RankIC, and directional accuracy.
2. Define the unwrapped action as $a_t^{\mathrm{raw}} = \mathrm{sgn}(\hat{y}_t)$, with optional flat action when $|\hat{y}_t|$ is below a validation-selected dead zone.
3. For the cost-threshold wrapper, convert the forecast to

$$\tilde{a}_t = \mathrm{sgn}(\hat{y}_t)\, \mathbf{1}\left[|\hat{y}_t| > \frac{\mathrm{cost\_bps}}{10^4} + \delta \right]. \tag{37}$$

4. Smooth positions by
$$a_t = \rho a_{t-1} + (1-\rho)\tilde{a}_t. \tag{38}$$
5. Select δ, ρ, and λ_{to} on validation by maximizing
$$J_{\text{val}} = \text{Sharpe}_{\text{post}} - \lambda_{\text{to}} \frac{1}{T} \sum_{t=1}^{T} |a_t - a_{t-1}|. \tag{39}$$
6. Report both the raw model and the wrapped version so the benchmark can measure whether a simple execution-aware bridge changes the $R4$ ranking.

E.8 Foundation and Generic Time-Series Protocol

Input. Frozen bar views, Kronos, iTransformer, TimesFM, Time-MoE, WaveToken, TimeDART, or compatible time-series foundation-model interface, validation budget.

Output. Foundation-model forecasts and benchmark actions.

1. Convert the benchmark data to the model-compatible view: volume bars or OHLCV-like bars for Kronos, and 1s or volume-bar multivariate views for iTransformer, TimesFM, Time-MoE, WaveToken, TimeDART, or compatible TSFM-style baselines [19, 11, 2, 18, 12, 21].
2. Do not expose licensed L2 tensors to a foundation or generic TS model unless the interface natively supports that representation.
3. Run either zero-shot inference, supervised training, or a light fine-tuning protocol selected on validation and recorded in the manifest.
4. Produce forward-return forecasts at the benchmark horizons.
5. Use the wrapper protocol in Section E.7 to map forecasts into actions and to evaluate raw versus wrapped variants.
6. Report financial foundation, generic TS, and LOB-native models as separate families so representation mismatch is visible rather than hidden. Smoke, lite, or dependency-resolution rows are not promoted to reported benchmark rows unless normalized $R0$/$R4$ artifacts are exported.

E.9 Policy-Learning Baselines

Input. PEG-Bench Gym-style environment, policy family in {PPO, SAC}, training split, validation split.

Output. Frozen policy checkpoint and $R0$–$R4$ report-card row.

1. Define state x_t from the same benchmark features available to prediction-first models.
2. Define continuous exposure action $a_t \in [-1, 1]$, where -1 is full short, 0 is flat, and $+1$ is full long.
3. Use the step reward
$$r_t^{\text{env}} = a_t r_t - C(a_t, a_{t-1}) - \lambda_{\text{to}} |a_t - a_{t-1}| - \lambda_{\text{dd}} \max(0, \text{DD}_t - \text{DD}_{\max}). \tag{40}$$
4. Train PPO and SAC only on the training split, selecting reward penalties and model hyperparameters on validation.
5. Freeze the selected checkpoint and evaluate on the test split with the same evaluator used by all other methods.
6. Report policy learning as a separate family because its optimization target is already execution-aware and need not improve $R0$ prediction metrics.

E.10 External Execution Audit Protocol

Input. A fixed strategy or action path, PEG-Bench execution log or fixed-action replay artifact, HftBacktest configuration when available, latency and queue-position grids.

Output. Audit-only sensitivity table.

1. Select fixed policies from the frozen panel, such as no-trade, OFI threshold, Kairos raw, and Kairos wrapped.
2. Translate benchmark actions into orders using a documented order-sizing and limit/market-order rule.
3. Replay the same windows under fee, latency, and queue-position configurations that match or bracket the PEG-Bench evaluator.
4. Compare fill rate, realized cost, slippage, turnover, Sharpe, and MaxDD between PEG-Bench and the external or fixed-action audit replay.
5. Report discrepancies as appendix audit statistics only; do not include HftBacktest as a ranked model.
6. Treat large discrepancies as evaluator-risk findings that require a versioned execution-spec update before leaderboard claims are made.

Accordingly, HftBacktest-related rows in the public artifact are fixed-action execution-sensitivity audits. They are neither ranked model results nor a replacement for the PEG-Bench evaluator.

F OG-PA as a Deployment-Aware Case Study

This section documents OG-PA as a benchmark-compatible case-study method slot. The emphasis is on its interface, supervision signal, optimization recipe, and audit trail under the shared PEG-Bench evaluator. Repository-backed defaults are stated as implementation facts; broader method advantages remain hypotheses unless supported by populated report-card results elsewhere in the manuscript. The section specifies the interface required to host OG-PA in PEG-Bench and is not used to support a positive method claim in the main text.

OG-PA is an optional method-family slot that reuses the PEG-Bench state construction, split builder, shared evaluator, and report card. Its only distinction is the training interface: trajectories are replayed through the evaluator, converted into risk-conditioned preference pairs, and used for a DPO-style update before the resulting policy is reinserted into the same $R0$–$R4$ scoring flow.

F.1 Case-Study Design Rationale

OG-PA targets settings in which the final deployment outcome is observable ex post but is difficult to optimize through a differentiable one-step loss. In PEG-Bench, a reference predictor may estimate direction or ranking quality, but the deployment decision remains path-dependent once costs, drawdown, and turnover are included. OG-PA therefore replaces scalar reward engineering with preference learning over complete trajectories.

The case study is defined by four interface choices that distinguish it from generic preference learning:

1. **Oracle supervision from replayable execution outcomes.** Preference labels are not human annotations and are not inferred by a reward model. The labels are constructed from replayed trajectory outcomes under the same evaluator used by the benchmark.
2. **Risk-conditioned preference construction.** The conditioning variable z exposes a family of business priorities. In the prototype, z is instantiated as a risk-aversion scalar λ; more generally, z can index lexicographic ranking templates.
3. **Trajectory-level, not step-level, comparison.** Winner/loser labels are attached to complete paths so that path-dependent effects such as cumulative cost, drawdown, and survival can shape the supervision signal.
4. **Reference-anchored continuous-action alignment.** The aligned policy is trained relative to a frozen baseline policy, which stabilizes learning and keeps the search close to auditable behavior.

F.2 From Kairos to OG-PA

Table 34 summarizes the interface changes between the Kairos baseline and the deployment-aware case-study method.

Table 34 clarifies why OG-PA is discussed without turning the manuscript into a method paper. Kairos represents a strong prediction-first family, whereas OG-PA shifts the training supervision and

Dimension	Kairos	OG-PA / KAIROS-X	Expected benchmark effect
Input	LOB tensor or token sequence	Same benchmark state plus risk condition z	Preserves comparability under the shared contract
Output	Return score or ranking signal	Continuous-action policy or conditioned trajectory distribution	More direct control over execution and turnover
Training supervision	Pointwise surrogate losses such as MSE and RankIC	Winner/loser preferences from ex-post trajectories	Moves supervision closer to execution-aware metrics
Training unit	Single sample or single bar	Complete trajectory	Lets path dependence shape the learning signal
Where cost enters	Mostly only during backtest	Already enters preference labeling	More likely to affect post-cost Sharpe and turnover
Where risk enters	Mainly in posterior evaluation	Drawdown, survival, and low-turnover logic enter comparator	More likely to affect MaxDD and stress survival
Multi-objective control	No explicit risk knob	z or λ spans aggressive-to-conservative frontier	Better suited to frontier analysis
Stability source	Supervised regularization and model bias	Reference anchor, entropy regularization, optional uncertainty filter	More auditable behavior under alignment updates
Likely advantage zone	May still look better at $R0$	Need not improve $R0$, but targets $R4$	Should be judged by ranking shifts under realism

Table 34: Kairos versus OG-PA as method families.

action interface toward trajectory-level preferences. Any expected benchmark effect is treated as a hypothesis about $R4$ behavior and must be verified by reported report-card rows before becoming an empirical claim.

F.3 Problem Setup and Notation

The core OG-PA objects are introduced in the order in which they are used during rollout and evaluation:

1. **Benchmark state.** Let x_t denote the benchmark state at bar t, consisting of the replay-derived order-book tensor, derived microstructure features, asset and venue identifiers, and any policy-side context.
2. **Risk condition.** Let z denote the risk preference condition that determines which return–risk–cost trade-off the policy should follow.
3. **Policy action.** The policy outputs a continuous trading action
$$a_t \sim \pi_\theta(\cdot \mid x_t, z), \tag{41}$$
where $a_t \in [-1, 1]$ represents normalized short-to-long exposure.
4. **Trajectory.** A trajectory over a replay window of length T is
$$\tau = \{(x_1, a_1), \ldots, (x_T, a_T)\}. \tag{42}$$
5. **Evaluator tuple.** The evaluator maps τ to an ex-post outcome tuple
$$g(\tau) = \big(R(\tau), S(\tau), \mathrm{MDD}(\tau), \mathrm{TO}(\tau), \mathrm{Surv}(\tau)\big), \tag{43}$$
where R is post-cost return, S is a Sharpe-style risk-adjusted score, MDD is maximum drawdown, TO is turnover, and Surv indicates whether the trajectory remains within benchmark risk constraints. The exact tuple can be extended, but the key point is that the tuple is computed by the same shared evaluator used elsewhere in PEG-Bench.

F.4 Continuous-Action Parameterization

The OG-PA prototype uses a predictive backbone as the reference policy source. The continuous-action parameterization consists of two steps:

1. **Gaussian action head.** The code instantiates a Gaussian policy on top of the base model:

$$\mu_t, \sigma_t = f_\theta(x_t, z), \qquad u_t \sim \mathcal{N}(\mu_t, \sigma_t^2), \qquad a_t = \tanh(u_t). \tag{44}$$

   This parameterization is implemented by the `GaussianPolicy` wrapper in `src/training/09_train_dpo.py`. The Gaussian head produces an action mean and a learned log-standard-deviation parameter, making the case study a continuous-action rather than discrete-token preference-learning setting.

2. **Position smoothing.** The trajectory generator smooths actions into positions using

$$p_t = \begin{cases} a_1, & t = 1, \\ \rho p_{t-1} + (1-\rho) a_t, & t > 1, \end{cases} \qquad \rho = 0.9, \tag{45}$$

   so that rapid action oscillations are damped before costs are applied. This smoothing is an implementation-level realism choice that approximates the constraint that deployable trading systems rarely switch instantaneously between full short and full long exposure on every bar.

F.5 Oracle Rollout and Execution-Aware Supervision

The execution-aware supervision signal is constructed in the following sequence:

1. **Risk-conditioned action scaling.** For each risk condition z, the generator samples multiple replay windows and rolls the policy forward over the same historical path. In the prototype, z is implemented as a risk-aversion parameter λ, and the raw action proposal is scaled as

$$a_t = \tanh\left(\frac{\hat{y}_t}{\lambda}\right), \tag{46}$$

   where $\hat{y}_t$ is the base model output. Smaller λ values induce more aggressive action magnitudes; larger λ values produce more conservative trajectories.

2. **Transaction-cost-adjusted return.** Given positions $\{p_t\}_{t=1}^T$ and benchmark returns $\{r_t\}_{t=1}^T$, the implementation computes transaction-cost-adjusted returns as

$$c_t = |p_t - p_{t-1}| \cdot \frac{\text{fee\_bps}}{10^4}, \qquad \tilde{r}_t = p_t r_t - c_t, \tag{47}$$

   with $p_0 = 0$.

3. **Trajectory summary.** The trajectory-level summary statistics are then computed from $\{\tilde{r}_t\}_{t=1}^T$, including cumulative return, volatility, a Sharpe-style score, and maximum drawdown. This step makes the supervision signal execution-aware rather than purely predictive.

F.6 Preference Construction

Preference construction is the central OG-PA interface. The public repository instantiates this step through the `OraclePreferenceGenerator` class in `src/training/08_generate_oracle_preferences.py`. The procedure is:

1. Choose a set of risk conditions. The default set is $\{0.1, 0.5, 1.0, 2.0, 5.0\}$.
2. For each risk condition, sample multiple trajectories on replay windows of fixed length. The default is 1,000 trajectories per condition with $T = 100$.
3. Evaluate every trajectory under the shared execution contract and compute the ex-post tuple.
4. Compare trajectories pairwise across different risk conditions to avoid trivial within-condition comparisons.
5. Keep only pairs whose preference margin exceeds a user-defined threshold. The default minimum margin is 0.01.
6. Sort retained pairs by margin and keep the strongest subset. The default target is 50,000 pairs, although the script is parameterized for larger sets.

The prototype uses a simplified lexicographic comparator:

1. **Primary key: Sharpe-style score.** If $|S(\tau_i) - S(\tau_j)| > 0.1$, prefer the higher Sharpe-style trajectory.

Condition	Name	Lexicographic priority
z_1	Aggressive	Maximize cumulative return; liquidated trajectories rank last
z_2	Balanced	Maximize Sharpe-style score; tie-break by lower MaxDD
z_3	Conservative	Maximize Calmar-style return-to-drawdown ratio; liquidated trajectories rank last
z_4	Low-turnover	Maximize post-cost net return subject to low turnover

Table 35: Canonical OG-PA risk conditions.

2. **Secondary key: total return.** If the Sharpe-style scores are too close, compare total return and require a gap larger than 0.001.
3. **Tertiary key: maximum drawdown.** If return is also too close, prefer the lower-drawdown trajectory, again with threshold 0.001.

This simplified rule corresponds to a balanced risk preference. The broader OG-PA interface is more general: the comparator can be conditioned on z, so that aggressive, balanced, conservative, and low-turnover preferences correspond to different lexicographic orderings over the ex-post tuple. That conditioning logic is part of the case-study design, even though the prototype instantiates only the balanced variant.

F.7 Preference Rule and Optimization Rationale

OG-PA avoids collapsing deployment goals into a single weighted-sum reward. In execution settings, the relative importance of return, drawdown, turnover, and survival can be discontinuous: a liquidated trajectory is unacceptable regardless of raw return, and two paths with similar cumulative return may differ substantially in turnover or drawdown. Lexicographic preference construction preserves these priorities directly and makes the oracle rule auditable.

This interface also distinguishes OG-PA from standard offline RL. The supervision signal is neither a learned reward model nor an estimated value function; it labels which complete trajectory is preferable under an explicit business rule computed by the same evaluator used elsewhere in PEG-Bench. This evaluator-defined labeling is the sense in which the method is *oracle-guided*.

F.8 Risk Conditions

The broader OG-PA interface allows the comparator to depend on a risk condition z. Table 35 records the canonical semantics of that conditioning variable. The current prototype instantiates a simpler λ grid, but the benchmark interface accommodates the more interpretable preference modes shown below.

Risk conditions define how the preference oracle ranks trajectories under different business priorities. They are included to make future OG-PA runs auditable: for example, a conservative run should not be interpreted as optimizing raw return alone. In this manuscript, the conditions define the interface; empirical claims remain limited to reported diagnostic rows.

F.9 Training Protocol and Uncertainty Extension

Once preference pairs (τ^w, τ^l) are constructed, the aligned policy is trained relative to a frozen reference checkpoint using the DPO objective

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\mathbb{E}\left[\log \sigma\left(\beta\left(\log \frac{\pi_\theta(\tau^w \mid s, z)}{\pi_{\mathrm{ref}}(\tau^w \mid s, z)} - \log \frac{\pi_\theta(\tau^l \mid s, z)}{\pi_{\mathrm{ref}}(\tau^l \mid s, z)}\right)\right)\right]. \tag{48}$$

In the current implementation, winner and loser actions are scored under a Gaussian policy, the DPO temperature is $\beta = 0.1$, entropy regularization is set to 0.01, AdamW uses learning rate 10^{-5}, gradients are clipped at norm 1.0, and the preference dataset is split 90/10 into train and validation partitions. Training logs include loss, preference accuracy, and logit statistics. Benchmark-relevant evaluation still happens only after reinserting the checkpoint into the shared $R0$–$R4$ evaluator.

The repository also contains adaptive confidence-interval utilities in `src/evaluation/10_evaluate_aci.py`. These utilities can reject ambiguous preference pairs during data construc-

Component	Prototype default	Role in OG-PA
Risk set $\mathcal{Z}$	5 values from 0.1 to 5.0	Spans aggressive-to-conservative action scaling
Trajectory length T	100 bars	Defines the horizon over which path-dependent outcomes are measured
Trajectories per condition	1,000	Controls coverage of each risk slice
Preference target size	50,000 pairs	Keeps the pair dataset large enough for stable minibatch training
Minimum margin	0.01	Rejects nearly tied winner/loser examples
Transaction cost in pair generation	3 bps	Makes the oracle execution-aware rather than frictionless
DPO temperature β	0.1	Controls winner/loser separation in the objective
Learning rate	10^{-5}	AdamW step size for preference optimization
Batch size / epochs	32 / 10	Current training budget for the case-study prototype
Entropy coefficient	0.01	Reduces collapse to overly deterministic actions
Warmup / grad clip	100 steps / 1.0	Stabilizes early optimization and limits exploding gradients
Train / val split	90% / 10% on pair set	Supports pair-level validation before benchmark evaluation

Table 36: OG-PA prototype implementation defaults.

Record	Logged fields	Audit rationale
Reference checkpoint	Training split, seed, and frozen evaluator version	Prevent silent drift between reference and preference phases
Trajectory pool	Rollout horizon, number of candidates, and action constraints	Makes preference coverage and compute budget auditable
Preference rule	Utility tuple, tie-breaking rule, and minimum margin δ	Defines which deployment behavior is being favored
Optimization settings	β, update schedule, stopping rule, and rejection filters	Explains stability or collapse during alignment
Evaluation manifest	Cost ladder, risk rules, regimes, and stress slices used for scoring	Ensures case-study claims remain on the shared benchmark contract

Table 37: Recommended OG-PA run sheet.

tion or cap action magnitude in high-uncertainty regions. In this manuscript, the uncertainty-aware path remains a supported extension rather than a reported empirical result.

F.10 Implementation Defaults from the Current Repository

Table 36 records the prototype settings required to reproduce or audit the diagnostic OG-PA rows. It also identifies implementation choices that can affect outcomes: trajectory length, risk grid, preference margin, cost level, DPO temperature, and entropy regularization all shape the learned action path. The table is therefore a run-manifest summary, not evidence of method superiority.

F.11 Case-Study Run Sheet

The run sheet specifies the metadata required before an OG-PA result can be treated as benchmark evidence. It links the reference checkpoint, trajectory pool, preference rule, optimization settings, and final evaluator manifest, allowing a future deployment-aware claim to be traced from training

data to the $R4$ report card. This is the evidence standard required to move OG-PA from a diagnostic slot to a reported leaderboard row.