

Build an honest machine.

A narrative technical textbook on machine learning, evaluation, Python and LLM collaboration, three model families, replay, paper trading, and fail-closed agentic monitoring.

By Dr. Mallarapu

Evidence boundary: The system is research- and paper-trading-only.

Illustrative, simulated, literature-derived, and design-target claims are labeled; no chapter asserts live performance or grants order authority to an agent.

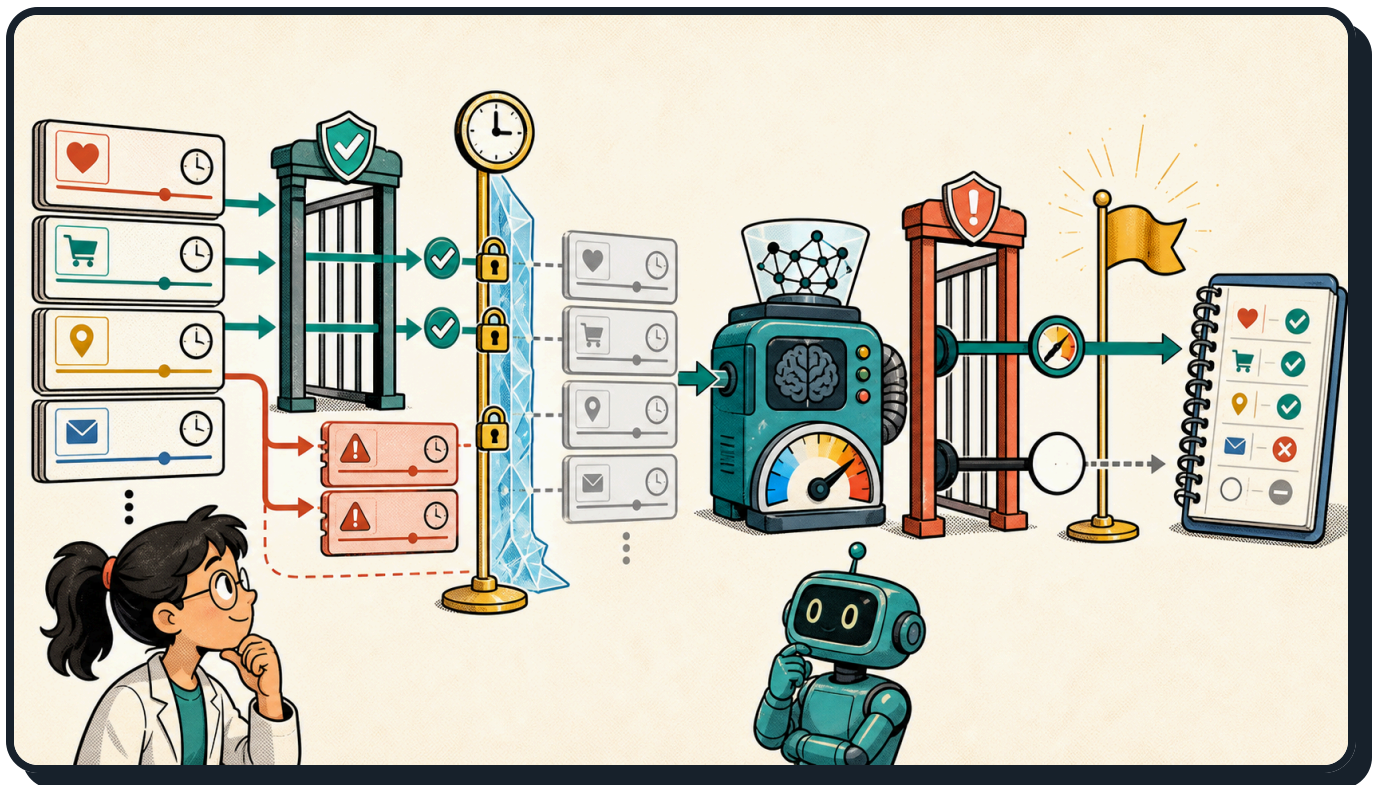


Figure 0.1. The Signal Quest contract: evidence is validated and frozen before a model estimates; policy may abstain; outcomes resolve later. Illustrative system diagram.

Chapter 1 — A probability is not permission

Chapter 2 — The clock that caught a cheat

Chapter 3 — The evidence forge

Chapter 4 — Metrics

Chapter 5 — Fair experiments

Chapter 6 — Calibration and abstention

Chapter 7 — Reproducible Python

Chapter 8 — CatBoost

Chapter 9 — Self-supervised encoders

Chapter 10 — Causal sequence models

Chapter 11 — LLM-assisted Python

Chapter 12 — Replay and paper trading

Chapter 13 — Agentic monitoring

Part V — From measurement to governed machine learning

Appendix A — Capstone lab

Appendix B — Technical workbook

Appendix C — References and provenance

A probability is not a permission slip.

Rhea's first problem sounds small: estimate whether a five-minute BTC direction event will resolve Up. The larger problem is to build a system that states what it knows and estimates. It must also state what it may do and why it sometimes refuses to act.

Rhea's opening case: She inherits a rule: "When recent activity is high, predict Up." It occasionally works, then fails under the same conditions. Its simplicity is not the problem. The rule claims more certainty than its evidence can justify. This case is illustrative.

1. Machine learning learns from examples, not from destiny

Supervised machine learning starts with historical examples. Each example contains observations available at a decision time and a later label. A model fits a function that maps observations to a score. It does not discover a permanent law of the world. Instead, it finds a pattern that may not survive a new period, source, population, or measurement process.

$$\hat{p}_t = f_{\theta}(X_t) \approx P(y_t = 1 \mid X_t)$$

Here, X_t is the information permitted at time t ; y_t is a later binary outcome; f_θ is a learned model; and $\hat{p}_t$ is its estimate. The approximation sign is a warning label. The score is a claim to evaluate, not a command to obey.

This distinction applies across domains. A fraud model estimates suspiciousness; a clinical decision-support model estimates risk; an intrusion detector estimates the likelihood of a harmful class. The organization—not the model—must decide which actions are permitted, when uncertainty is too high, and who bears responsibility for mistakes.

2. Five objects that should never be collapsed into one word

Students often call everything that follows “the prediction.” Rhea separates these objects because each has a different truth condition and failure mode.

OBJECT	WHAT IT IS	WHAT CAN GO WRONG
Observation	Permitted measurements at a cutoff.	Late, missing, duplicated, corrupted, or future data.
Model	Versioned learned mapping from input to score.	Overfitting, artifact mismatch, or outdated assumptions.
Probability estimate	Interpretable uncertainty claim, if calibrated.	Overconfidence, underconfidence, or meaningless score scale.
Decision policy	Deterministic rule that applies gates and constraints.	Hidden thresholds, ignored costs, or unsafe authority.
Outcome	Later result under a declared label contract.	Ambiguous settlement source or invalid join.

Rhea's rule: a model may generate evidence. A separately versioned policy decides whether the evidence is eligible for a bounded research action. The model never contains a hidden permission to override a failed safety gate.

3. Prediction is not decision

Suppose Orbit produces a score of 0.58. A casual interpretation is “Orbit says Up.” That interpretation skips several questions. Is 0.58 calibrated? Which reference probability or quoted state would the decision use? Which costs and delays matter? Is the data fresh? Is the model artifact approved? Has the risk budget been consumed? A system that skips these questions turns a numerical output into unearned authority.

```
eligiblet = data_validt ∧ model_validt ∧ calibratedt ∧ risk_okt ∧  
conservative_policy_okt
```

The exact policy is a design choice and must be explicit. The logic fails closed: if the system cannot prove a necessary condition, it abstains. This design does not make the model timid. It makes the boundary between evidence and authority visible.

Worked trace: two identical scores, different decisions

Orbit produces 0.58 on two illustrative cutoffs. On the first, the feature manifest matches the model, calibration was evaluated recently, and all inputs are fresh. On the second, the source is late and the model's expected feature recipe differs from the one produced. The score is numerically identical. The second result is ineligible because the evidence path is broken.

This trace shows why a dashboard that displays only probability can be unsafe. The decision state needs its reason: eligible, stale data, model mismatch, low calibration confidence, insufficient conservative difference, or another declared gate.

4. The limits of pattern learning

A model may exploit correlations that are temporary, accidental, or created by the data collection process. It may mistake a proxy for a causal factor. A label leak can make the model appear strong, while average performance can hide failure in the regime that matters. These are not exceptional embarrassments; they are ordinary risks of fitting flexible functions to historical data.

Rhea therefore asks every model claim four questions. What object is being estimated? What evidence entered the estimate? What decision changes because of it? What can go wrong? The rest of the book turns those questions into data contracts, metrics, tests, models, replay, and monitoring.

What ML cannot supply by itself: It cannot define the target, establish causation, choose an ethical or financial action, guarantee persistence, prove its own calibration, or make missing data trustworthy. Those responsibilities belong to the research design and governance around the model.

5. The first research contract

Before training, Rhea writes a contract in ordinary language. It names the question, settlement source, horizon, observation cutoff, allowed sources, label rule, missing-data response, model output, policy output, and evidence retained. A good contract is specific enough that two researchers can build the same dataset and notice if they disagree.

CONTRACT FIELD	EXAMPLE QUESTION
Target	What exact later event is $y=1$?
Horizon	When does the outcome become known?
Cutoff	When must the evidence stop?
Inputs	Which sources and fields may enter X_t ?
Unavailable evidence	Does the system abstain, quarantine, or label the example unavailable?
Output	Probability estimate, eligibility, reason, artifact versions.

6. The next vulnerability: time

Rhea can now name the pieces, but she still needs to protect the clock. The next chapter makes the prediction contract temporal. It separates observation time from settlement time and closes the path through which future data can make a weak model look miraculous.

Review and discussion

1. Why does the approximation sign matter in the probability equation?
2. Give an example where a model score is healthy, but the decision should abstain.
3. Which of the five objects changes first when a source arrives late?
4. Why is a prediction contract a scientific instrument rather than paperwork?

Evidence and sources

This chapter is illustrative and makes no performance claim. The proposed system boundary is in the local **source map** and **safety contract**. The exact settlement contract must be verified against the source governing any future research market.

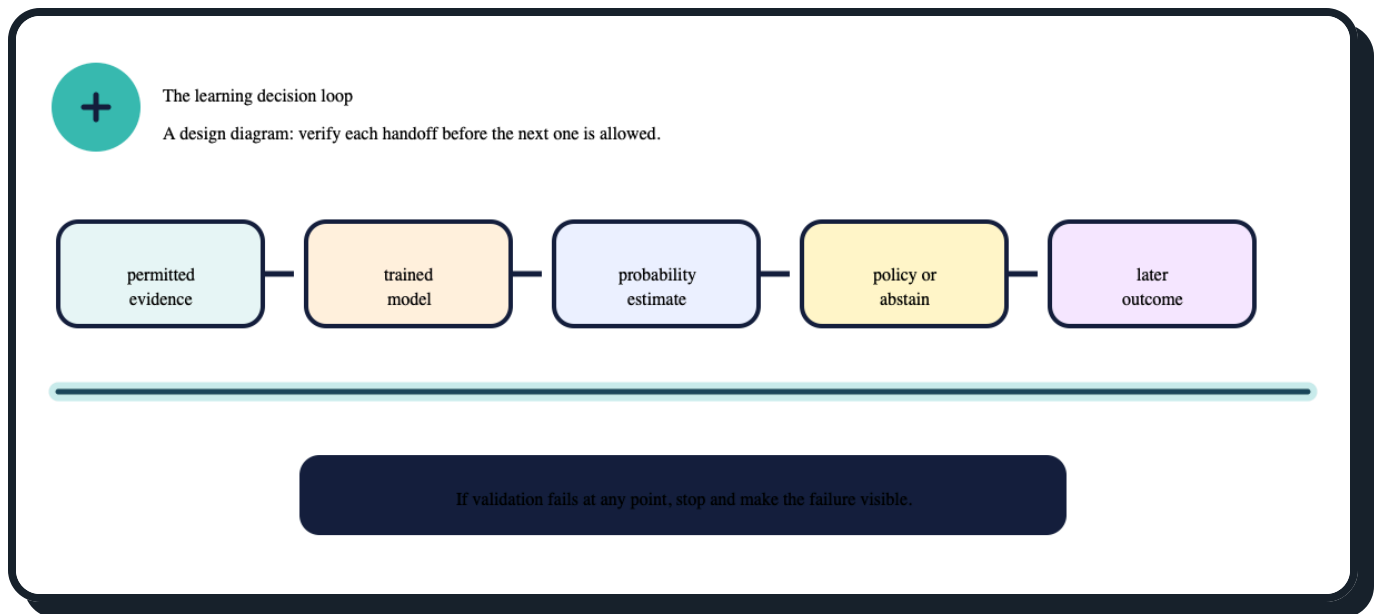


Figure 1.1. The five roles that must remain separate in a supervised decision system. *Source basis: design target.*

The clock that caught a cheat.

Orbit produces an astonishing score. Rhea asks it to replay one decision. A feature marked 10:00:00 includes an update received at 10:00:04. The model did not become clever. The experiment let it see the future.

The temporal lesson: Using timestamps is not enough. The researcher must define the target and every clock before a model sees the data. Each feature must then be proven available at the decision cutoff. This scenario is illustrative.

1. A label is a contract, not a column name

Terms such as “BTC up in five minutes” sound precise until two researchers implement them. One uses an exchange midpoint; another uses a contract’s stated settlement source. One treats equality as Up; another treats it as Down. One uses a closing value; another uses the first tick after the horizon. They have built different targets, even if both call the column `up_5m`.

Rhea writes the label contract in plain language and notation. It names the source, start clock, end clock, horizon, equality rule, source publication behavior, and missing-data response. An illustrative binary label can be written as:

$$y_t = \mathbb{1}[S(t + H) \geq S(t)]$$

S is the one declared settlement source and H is the horizon. The expression is small, but every term needs an implementation definition. If $S(t + H)$ is unavailable, the system does not substitute a convenient different source. It returns an unavailable label or follows a predeclared correction policy.

2. The evidence set has a different contract

The outcome is learned later. The model inputs must be frozen earlier. Rhea defines eligibility using availability rather than hindsight:

$$X_t = \{e : \text{available}(e) \leq t\}$$

An event belongs in the feature set only if the system could have observed it by the cutoff. This rule is independent of the event's own timestamp. A historical archive can know that an event happened at 10:00:00; a live-like system may not receive it until 10:00:02.

TIME FIELD	WHAT IT MEANS	WHY IT CANNOT BE SILENTLY SUBSTITUTED
Event time	When the source says the event occurred.	A receiver may not know it yet.
Receive time	When the research system observed it.	It can lag, arrive out of order, or be unavailable in archives.
Process time	When the system wrote or transformed it.	Processing delay affects when a decision can exist.
Decision cutoff	The moment evidence freezes for that prediction.	It defines the fairness boundary of X_t .
Settlement time	When the label resolves under the contract.	It must never enter the earlier feature path.

Time travel is not a small bug: If a feature includes unavailable future information, the metric answers an impossible question. It measures how well a system could predict after seeing part of the answer.

3. Common leakage paths

Leakage is often more subtle than directly inserting the label. A rolling statistic may include a late event. A normalizer may be fit on the full dataset, including final-test values. A categorical feature may be computed from an outcome that resolves later. A correction may rewrite the historical archive and make it look as if the corrected value was available originally. A random split may let overlapping windows appear on both sides of evaluation.

LEAKAGE PATH	WHY IT LOOKS HARMLESS	CONTROL
Full-period normalization	It is a standard preprocessing step.	Fit transforms on training data only; version the fit artifact.
Late event in rolling feature	The event timestamp is earlier than cutoff.	Filter by availability time before aggregation.
Revised historical source	The archive looks clean and authoritative.	Preserve original and correction records; state which is used.
Outcome-derived category	The category appears descriptive.	Test whether it could be computed at cutoff.
Overlapping windows	Rows have different timestamps.	Use temporal splits and purge overlapping evidence.

4. Missing and corrected observations

Rhea refuses to treat missing data as an invitation to make up a smooth story. A missing settlement value, dropped source event, delayed update, or correction can change the label or the feature set. The contract must say whether the example is unavailable, quarantined, forward-filled under a limited rule, or retained with a missingness indicator. The policy must log which option occurred.

Corrections require special care. A source may later correct an event timestamp or value. The original record is evidence of what the system could have known. The corrected record may be relevant to post-hoc truth, but it must not silently rewrite availability in a replay that claims to be live-like. Rhea stores both and documents the chosen truth policy.



Hero tool: a temporal data contract with event time, receive time, cutoff, label rule, missingness policy, correction policy, and tests that reject illegal features.

Worked trace: a five-minute window

At 10:00:00, Orbit freezes evidence. An event occurs at 09:59:59 but is received at 10:00:03. It is excluded from the 10:00 decision even though its event time is earlier. A second event arrives at 09:59:58 and is received at 09:59:59; it is eligible. At 10:05:00, the declared settlement source is missing. The label builder returns unavailable; it does not borrow an exchange quote.

Each outcome reduces the number of usable rows. Rhea prefers fewer honest examples to a larger dataset whose labels and features answer different questions.

5. Tests that protect the clock

Temporal validity should be tested, not asserted. Rhea's smallest tests add an event received after the cutoff, mutate a future event, use an unknown source schema, and create a missing settlement. Each case has an explicit result: exclude, reject, quarantine, or mark unavailable. A replay test then walks a small fixture through the full path and verifies that a later correction cannot influence an earlier decision.

What a correct label still cannot prove: A well-defined target does not make it predictable. It only ensures that the model, metric, and replay are discussing the same event.

6. From clocks to evidence

Rhea has frozen time and defined truth. The system still needs a defensible path from raw events to features. The next chapter builds that path, preserves the original evidence, validates each transformation, and gives every feature a reproducible recipe.

Review and discussion

1. Why can two implementations of “five-minute direction” create incompatible research results?
2. Which time should govern a system that claims to act on received data?
3. Give two examples of leakage that do not directly include the label.
4. Why should a corrected record not silently replace the original in a live-like replay?

Evidence and sources

This chapter is illustrative. The market/settlement contract must be verified for each research target; current boundaries are recorded in the local **source map** and the parent **market documentation map**. No dataset or settlement result is claimed here.

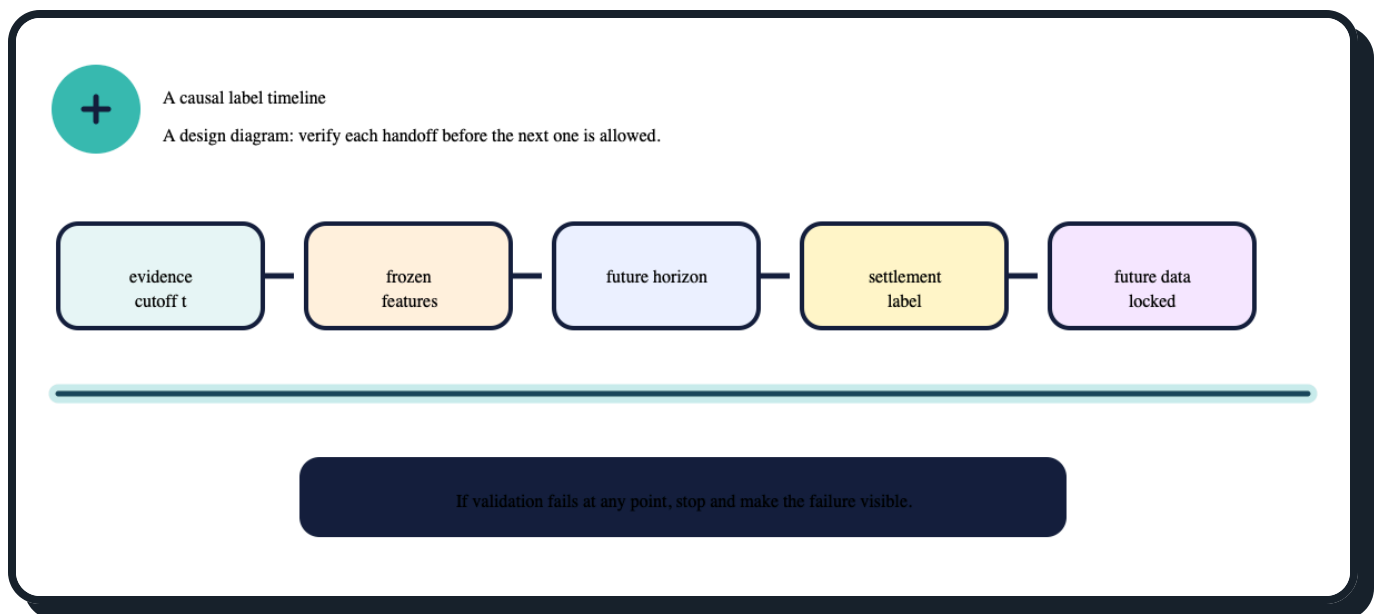


Figure 2.1. The label is later; the features must be fixed at the cutoff. *Source basis: design target.*

The evidence forge.

Rhea has defined the label and locked the clock. Orbit still receives a storm of raw events: duplicates, late arrivals, malformed records, and schema changes. A clean table is not enough. Rhea needs a chain of evidence that a reviewer can walk backward.

Evidence path: In this design-target scenario, a model row is admissible only when it traces through a versioned feature recipe to eligible, validated raw events. The system preserves invalid records as evidence rather than quietly erasing history.

1. Raw evidence comes first

Machine-learning models consume tensors, rows, and vectors. Real systems receive messages, snapshots, updates, files, and corrections. The transformation from real events into model input is part of the scientific method. If a team cannot say where a value came from, when it arrived, which rules accepted it, and which formula transformed it, the value is not defensible evidence.

Rhea stores raw events immutably. Each record includes a source identifier, source event ID when available, capture ID, payload, event time, receive time, schema version, capture outcome, and content hash. “Immutable” does not mean “trusted.” It means later validation does not rewrite what originally arrived.

raw event → validation result → eligible event set → feature recipe v_k → X_t

Every arrow represents a testable claim. The raw event claims what the source sent. The validation result claims whether it meets the current contract. The eligible set claims it satisfies the cutoff. The recipe claims how values were computed. The feature vector claims what the model actually received.

2. A record needs a passport

Without identity and time fields, duplicates and corrections become impossible to reason about. Without schema version, a parser can silently reinterpret a field. Without a hash or capture ID, a later reviewer cannot tell whether two records are the same observation or two different observations that happen to look similar.

FIELD	PURPOSE	WHAT FAILS WITHOUT IT
Source and source-event ID	Names the origin and supports deduplication.	Repeated events can inflate activity features.
Event and receive time	Separates occurrence from availability.	Feature eligibility cannot be audited.
Schema version	Defines the meaning and shape of payload fields.	New fields can silently change feature behavior.
Payload hash / capture ID	Links derived data back to a particular raw artifact.	Provenance becomes a narrative rather than evidence.
Validation status and reason	Preserves why a record was rejected or accepted.	Quarantine decisions cannot be reproduced.

3. Validation is not a cleaning step

Validation determines whether an event satisfies structural and semantic rules. Structural checks cover required fields, types, formats, and schema compatibility. Semantic checks identify impossible quantities, nonmonotonic sequence numbers, out-of-order events, stale receive times, duplicate IDs, and values outside declared ranges. The system marks and quarantines a failed record instead of discarding it.



Quiet cleaning changes the experiment: Replacing a missing quantity with zero, dropping a late row, or forward-filling a depth level creates a different dataset. The method must state the policy, log each activation, and test whether conclusions depend on it.

Some policies can be defensible for a specified task. Forward-fill may be permitted for a bounded interval when evidence shows that a field persists between updates. It may be unsafe when the missing field can change quickly. The right answer is neither “always fill” nor “never fill.” It is a predeclared rule with a failure mode and sensitivity analysis.

4. A feature is a hypothesis with a recipe

Features are summaries of raw events. A recent-count feature hypothesizes that activity in a defined window may carry information. A depth-imbalance feature hypothesizes that selected quantities and levels can be combined meaningfully. Neither becomes a fact because it is computable.

Rhea’s feature dictionary lists name, purpose, formula, input fields, time window, cutoff rule, units, normalization, missing-data behavior, expected range, recipe version, and test cases. The version changes whenever the formula, input set, rounding, resampling, normalization, or missingness behavior changes.

FEATURE PROPERTY	EXAMPLE REQUIREMENT
Window	Use only events with receive time in the declared lookback before cutoff.
Normalization	Fit any learned scaling on the permitted training period only.
Missingness	Emit an explicit reason or indicator; never silently substitute an unrelated source.
Version	Increment if formula or semantics change.
Lineage	Retain input partition IDs and raw-event references.

Hero tool: a versioned feature recipe turns an anonymous number into an auditable claim. A model artifact can refuse a feature version it was not trained to interpret.

Worked trace: the duplicated update

Orbit receives two records with the same source event ID and identical payload hash. The validator keeps one canonical raw event, marks the other as duplicate, and records both capture IDs. A recent-activity feature reads only eligible canonical events. If the duplicate had been counted twice, the feature might suggest a burst that never occurred.

Next, a new schema version arrives with a missing required field. Orbit does not guess the field's meaning. It quarantines the partition, marks derived rows ineligible, and opens an incident for human review. The raw payload remains available for analysis and future contract updates.

5. Quality gates shape model behavior

Data-quality decisions are model decisions because they change the input distribution. If quiet periods have more missing events and the pipeline drops them, the remaining dataset can overrepresent active periods. If a forward-fill policy creates stable-looking depth, the model can learn an artifact of the pipeline. Rhea monitors validation failures by source, time, schema, and feature recipe to locate weak evidence.

```
feature eligibility = raw integrity ^ schema validity ^ time validity ^  
recipe compatibility
```

When this conjunction is false, the safe result is not a fabricated feature vector. It is an ineligible result with a reason. That reason becomes an input to monitoring and later incident analysis.

6. From valid rows to valid interpretation

Rhea can now produce a valid row. The next challenge is interpretation: a model can score that row and still be judged by the wrong metric. Chapter 4 distinguishes classification, ranking, and probability metrics so that none borrows meaning from another.

Review and discussion

1. Why preserve a raw record that later fails validation?
2. Which feature changes require a new recipe version?
3. How can forward-filling create a false signal?
4. Why is deduplication evidence rather than merely a data-cleaning task?

Evidence and sources

This chapter defines a design target. Current claim boundaries are in the local [source map](#) and [safety contract](#). No live data pipeline or measured quality rate is claimed.

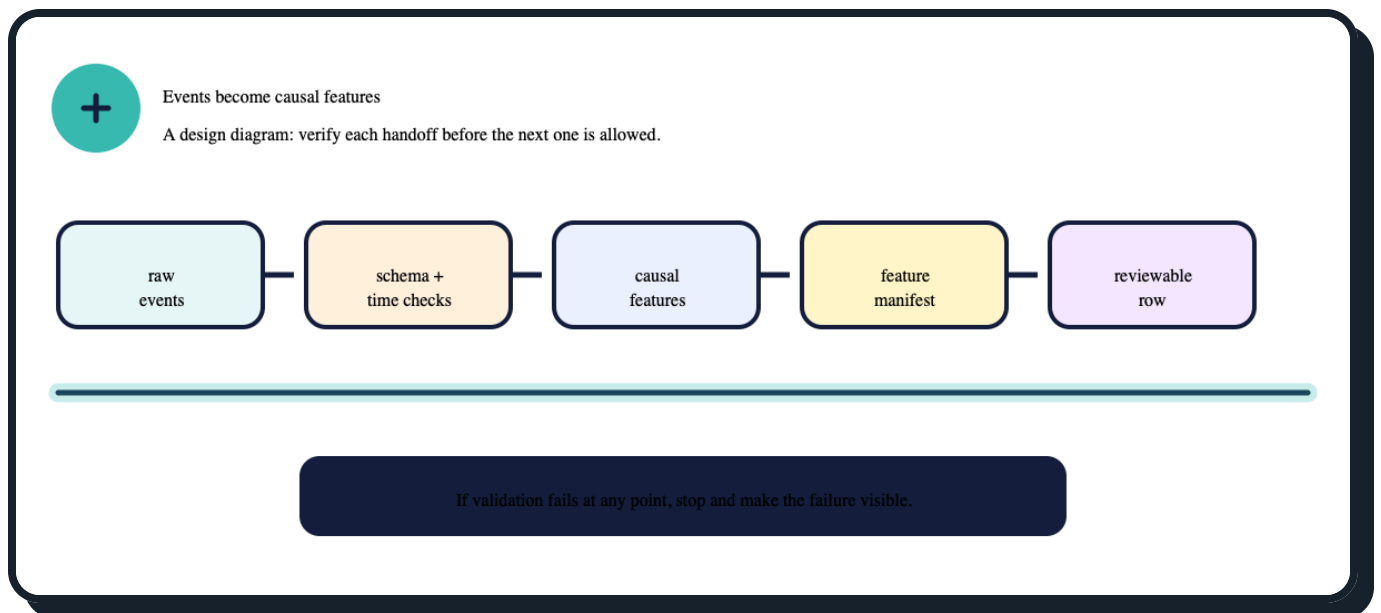


Figure 3.1. Feature engineering is an evidence contract, not merely a table-building task. *Source basis: design target.*



Figure 3.2. The evidence forge has one job: admit only data that existed at the decision cutoff. *Source basis: illustrative original teaching comic.*

The golden score that did not mean what it said.

Rhea has a model score. The hard part is learning whether the score describes skill, coincidence, a trivial baseline, or a broken experiment.

Rhea's metric audit: In this simulated case, Orbit reports 98% accuracy. Before anyone celebrates, Rhea requests the class balance, confusion matrix, baseline, time split, and probability calibration. One number cannot carry the whole argument.

1. A metric is a question with a denominator

Metrics are not medals attached to a model. Each metric asks a specific question about a specified set of predictions. Accuracy asks what fraction of all classifications were correct. Precision asks whether positive predictions were reliable. Recall asks whether the system found the positive cases. Specificity asks how well it avoided false alarms among negative cases. F1 combines precision and recall, but it does not make their underlying tradeoff disappear.

The first object Rhea requests is the confusion matrix. It forces the team to count the four possible outcomes instead of hiding them behind a single percentage. Let **TP** denote a true positive, **FP** a false positive, **TN** a true negative, and **FN** a false negative.

$$\text{accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{TN} + \text{FN})$$
$$\text{precision} = \text{TP} / (\text{TP} + \text{FP})$$
$$\text{recall} = \text{TP} / (\text{TP} + \text{FN})$$

QUESTION	USEFUL METRIC	WHAT IT FAILS TO ANSWER
Were classifications correct overall?	Accuracy	Whether a trivial majority-class prediction did the same job.
When the system raises a positive signal, how often is it right?	Precision	How many true positives were missed.
How many true positives did the system find?	Recall	How many false positives it created.
Do positives rank above negatives?	ROC-AUC or an explicitly defined precision–recall summary	Whether numeric scores mean calibrated probabilities.
Are probability estimates accurate and honest?	Log loss and Brier score; calibration curve	Whether an action has value after costs and constraints.

2. Why accuracy can be nearly perfect and useless

Suppose 98 of 100 examples are Down. A classifier that always says Down achieves 98% accuracy. It has detected no Up cases and supplied no useful probability distinction. This is not a corner case; imbalanced outcomes are common in event detection, fraud screening, medical triage, and any task where the event of interest is rare.



Scientific warning: Class imbalance does not make a model invalid. It makes an unqualified accuracy claim invalid. The report must show base rate, confusion matrix, baseline, and the metric appropriate to the decision.

Rhea asks Orbit to compare against the simplest baselines first. A constant predictor always returns the observed training prevalence. A majority-class classifier turns that prevalence into a hard label. If an external reference probability exists, it is a separate baseline. A complex model earns attention only if it adds out-of-sample information beyond these references under the same split and data contract.

Worked trace: the false triumph

In an **illustrative** set of 1,000 examples, 950 are Down and 50 are Up. Model A predicts Down every time: 950 correct, 50 incorrect, 95% accuracy, 0% recall for Up. Model B identifies 30 of the 50 Up cases but creates 30 false Up signals. Its accuracy is lower, yet its recall is 60% and its precision is 50%.

Which model is better? The correct answer is not visible until the system's purpose is stated. If false positives are extremely costly, Model A may be safer but uninformative. If missing a positive case is the key harm, Model B may be preferable. A research paper that declares either winner from accuracy alone has not finished the argument.

3. Ranking is not calibration

Orbit then shows Rhea a high AUC. That may be useful: it suggests positive outcomes often receive higher scores than negative outcomes. But AUC does not require the score 0.80 to mean an 80% chance. A model may rank two cases correctly while being dramatically overconfident about both.

Calibration compares forecasts to frequencies. Take a held-out period, group examples with predicted probability near 0.70, and measure how often the outcome occurred. If the observed frequency is near 0.70, the group is calibrated. If it is near 0.45, the model is overconfident in that region. A reliability curve visualizes this relationship. The Brier score measures average squared probability error; log loss penalizes confident wrong forecasts sharply.

$$\text{Brier} = (1 / n) \sum (\hat{p}_i - y_i)^2$$
$$\text{log loss} = -(1 / n) \sum [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)]$$

Rhea's rule: Read discrimination, calibration, and decision value separately. A model must not borrow credibility from one of those properties to claim another.

4. The metric report Rhea will accept

Rhea does not ask Orbit for a dashboard of impressive numbers. She asks for a compact evidence record. It names the outcome definition, class balance, evaluation period, split, baseline, confusion matrix, ranking metric, probability metric, calibration plot, and uncertainty interval or limitation. The record then states which decisions these observations can and cannot support.

REQUIRED REPORT ELEMENT	WHY IT EXISTS
Label and class prevalence	Prevents a misleading reading of accuracy and precision.
Temporal evaluation period	Shows where and when the claim applies.
Baseline comparison	Shows whether the model added information.
Confusion matrix at declared threshold	Exposes error tradeoffs.
ROC-AUC and a defined precision–recall summary	Names average precision or the curve-integration rule instead of using an ambiguous label.
Calibration and proper score	Tests whether probability language is defensible.
Failure analysis	Prevents average performance from hiding a dangerous segment.

5. The test behind the score

Even a rigorous metric report can be wrong when time leakage contaminates the test period or repeated tuning consumes it. The next chapter therefore locks the test away. Rhea trains earlier, validates later, inserts a purge gap where windows overlap, and opens the final test only after freezing every choice.

Review and discussion

1. Explain why Model A's 95% accuracy in the worked trace is not evidence of useful positive detection.
2. Give an example of a situation where precision matters more than recall, and one where recall matters more.
3. How can AUC improve while calibration becomes worse?
4. What report element would reveal whether a threshold was selected after seeing final-test performance?

Evidence and sources

This chapter's examples are illustrative. Metric definitions should be implemented and version-pinned using the relevant library documentation; see the [**scikit-learn model evaluation reference**](#). Book-level source boundaries remain in the local [**source map**](#).

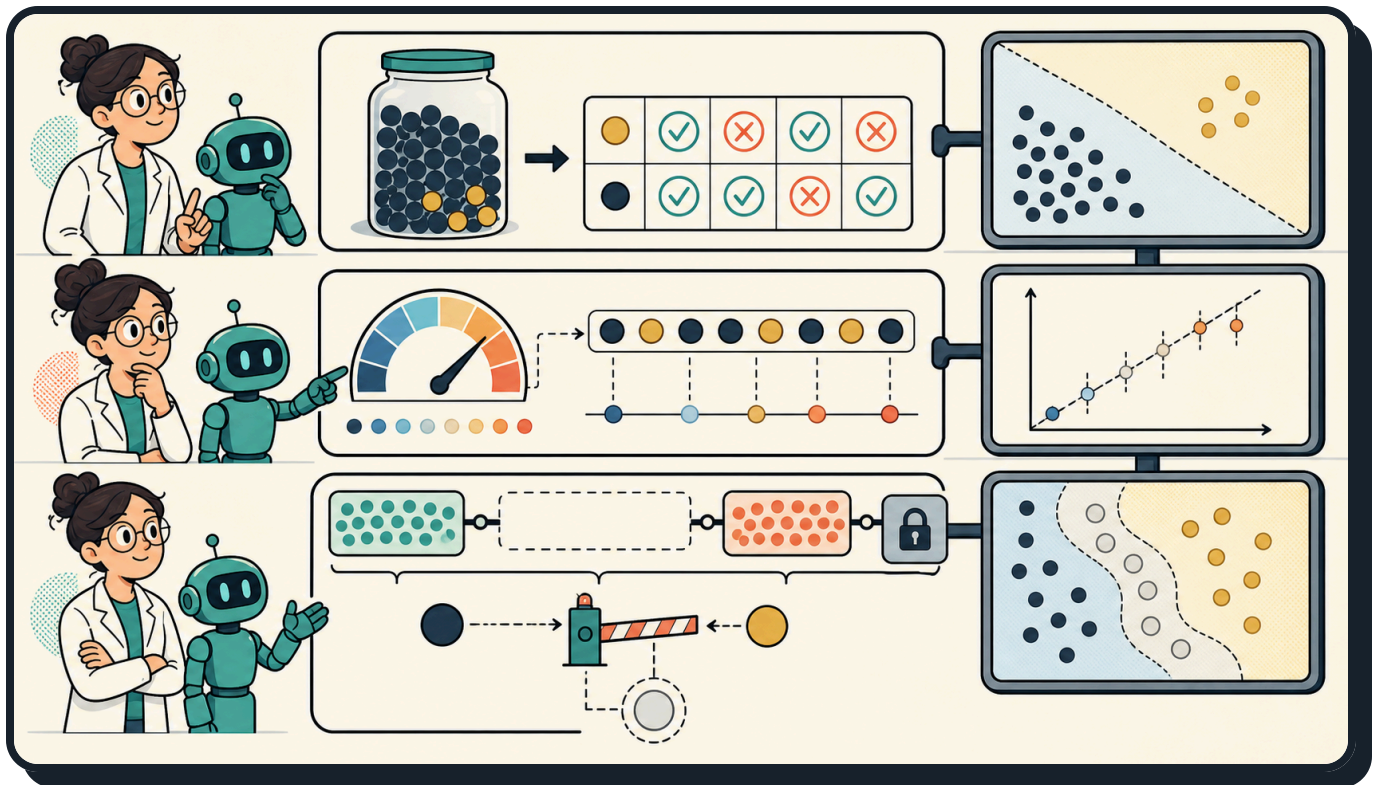


Figure 4.1. Metrics and evaluation are connected: class balance shapes interpretation, calibration gives probability language meaning, time preserves a fair test, and a failed gate leads to abstention. *Source basis: illustrative original diagram.*

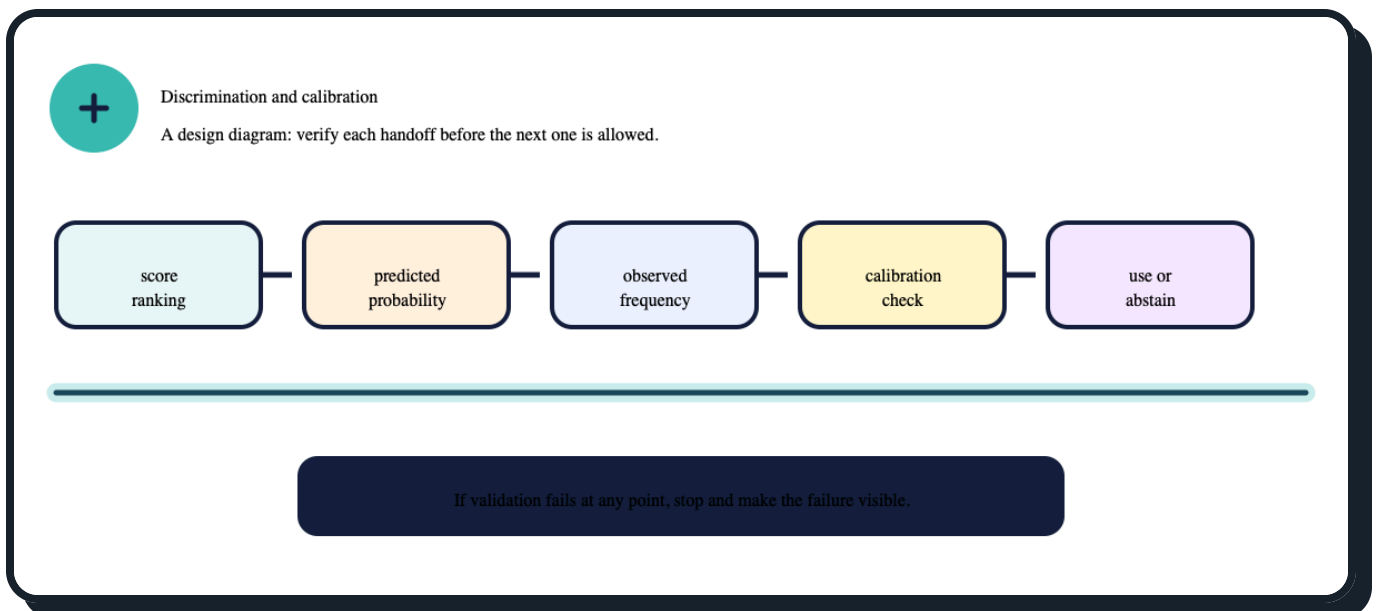


Figure 4.2. A useful ranking becomes a decision probability only after calibration is checked. *Source basis: illustrative design diagram.*

The exam Orbit had already seen.

A model can earn a high score because it learned a useful pattern—or because the experiment quietly gave it answers in advance. Rhea’s task is to make the second explanation impossible.

The split audit: In this simulated case, Orbit’s model beats every baseline. Rhea inspects the split and finds rows from 10:00:00 and 10:00:01 in different random partitions. Their features share almost all the same recent events. Orbit did not pass a hard exam; it saw the same page twice.

1. Why random splitting is often wrong for temporal evidence

Random train/test splits assume that examples are independent enough that mixing their order does not carry information across the boundary. That assumption can be reasonable for some static datasets. It is often false for time-ordered data.

Neighboring events may share a feature window, a label window, a market state, a participant, or a data-processing artifact. A model trained on one neighbor can effectively recognize the other.

The remedy is not merely “use dates.” The split must reflect the question the system will face. If the intended decision arrives over time, training must occur earlier than validation, and validation must occur earlier than the final test. Any period whose inputs or labels overlap a boundary needs special treatment.

```
train interval → purge / embargo → validation interval → untouched final  
test
```

Leakage audit: A row may be temporally earlier yet still contaminate a later row if its feature or label horizon crosses the split. Inspect the windows, not only the row timestamps.

2. The job of the purge and embargo

A purge removes examples whose label horizon overlaps the next evaluation interval. An embargo adds a temporal gap so that delayed labels, shared features, or data corrections do not cross the boundary. Its length is not a ritual number. It follows from the largest lookback window, label horizon, availability delay, and source behavior declared in the prediction contract.

Suppose a feature looks back five minutes and its label resolves five minutes after the decision. A noon split cannot automatically treat the 11:59:59 and 12:00:01 examples as independent. The correct gap depends on how the pipeline constructs features, labels, and source availability. Rhea records this reasoning before seeing model results.

BOUNDARY RISK	QUESTION RHEA ASKS	TYPICAL CONTROL
Feature overlap	Do train and validation rows summarize the same raw events?	Purge rows or lengthen the gap.
Label overlap	Does a training label resolve during the validation window?	Purge label-overlapping rows.
Availability delay	Could a delayed record appear on both sides after processing?	Embargo using receive-time assumptions.
Repeated tuning	Has the final test influenced any choice?	Lock final test until all choices freeze.

3. Baselines are scientific controls

Before Orbit earns a complex model, Rhea asks what it must beat. A baseline is not an embarrassing weak opponent. It is the control that reveals whether added complexity added information. The minimum set is usually a constant-probability baseline, a simple rule or linear/tabular baseline, and—when relevant—a valid external reference probability. All candidates must receive the same labels, splits, eligibility gates, and cost assumptions.

Rhea’s model ladder: Start with the simplest defensible baseline. Add a model only when it can explain what it represents better, survives the same temporal evaluation, and earns its operational cost.

Complexity has costs: more hyperparameters, more opportunities to leak, longer training, less transparent failure modes, and greater risk that a small validation period selects noise. The burden of proof rises with the model’s flexibility.

Worked trace: the locked final month

Rhea has twelve months of event data. She uses months 1–7 for training and internal walk-forward folds. Months 8–9 serve as validation for feature selection, calibration, and hyperparameters. She chooses the entire pipeline, writes down the selected configuration, and only then opens months 10–12 as the final evaluation period.

Orbit wants to change the threshold after seeing weak results in month 11. Rhea refuses. That would turn month 11 from a final test into another tuning surface. The honest options are to report the limitation, start a new experiment with a later untouched test, or accept a predeclared policy that was selected earlier.

4. Hyperparameter tuning is an experiment inside the experiment

A tuning system can try thousands of combinations. Repeated evaluation against one validation interval can overfit that interval even when no individual model looks suspicious. Rhea constrains the search space through prior reasoning, records every trial, and separates model development from final confirmation. Nested temporal validation makes the separation explicit. Inner folds choose settings; outer folds estimate how the selection procedure generalizes.

outer temporal fold: estimate procedure quality

inner temporal folds: choose model and hyperparameters

There is no magic number of folds. Chronology and independence matter more. The design must leave enough time to represent changing conditions while avoiding overlap. When data is limited, the report should expose uncertainty instead of hiding it behind repeated selections.

5. The experiment card

Rhea keeps a one-page experiment card before training starts. It contains the dataset manifest, label rule, source availability rule, feature versions, split dates, purge rationale, baseline list, tuning budget, primary metrics, calibration method, replay assumptions, and stopping rule. If any item changes after results appear, the run is a new experiment. This is not paperwork. It is a defense against unconscious selection of the story that looks best after the fact.



What a fair result can still not prove: An untouched historical period tests a particular past environment. It does not guarantee a future relationship will persist, that a replay matches live execution, or that costs stay stable. The result earns a narrower claim: the specified procedure performed this way on this specified evidence.

6. From ranking to probability

Rhea has protected the test from obvious time travel and tuning. Even a fair ranking model, however, may express probabilities badly. The next chapter asks whether “80%” means anything near 80%. It also shows how policy combines probability with costs and risk, and why abstention must be engineered rather than treated as failure.

Review and discussion

1. Why can a chronological split still leak information?
2. How would you derive a purge length from a feature and label contract?
3. Why must a complex model face a simple baseline under the same replay assumptions?
4. What is the scientific harm in adjusting a threshold after seeing final-test results?

Evidence and sources

All scenarios are illustrative. The temporal-validation and replay constraints are design targets in the local **research contract**; book-level claim boundaries are in the **source map**. Before implementation, the final split scheme must be grounded in the measured feature and label windows of the actual dataset.



Time-respecting evaluation

A design diagram: verify each handoff before the next one is allowed.



If validation fails at any point, stop and make the failure visible.

Figure 5.1. Time-respecting evaluation prevents nearby or future information from contaminating the test. *Source basis: design target.*



Figure 5.2. A fair clock does not let nearby or later records sneak across the boundary. *Source basis: illustrative original teaching comic.*

The courage to do nothing.

Orbit has learned to rank cases. Rhea now asks a harder question: when Orbit says “80%,” should anyone believe that number—and what else must be true before a paper-trading simulation is even eligible?

Calibration under pressure: In this simulated case, Orbit assigns ten cases a probability near 0.80. Only six settle Up in an untouched period. Orbit may still rank cases well, but its probabilities are overconfident. That overconfidence can turn a cautious policy into an unsafe one.

1. A score can rank well and mean the wrong thing

Discrimination and calibration are different properties. Discrimination asks whether positive cases tend to receive higher scores than negative cases. Calibration asks whether a numerical probability matches an observed frequency. A model can sort examples in a useful order while systematically exaggerating or understating how likely they are.

Rhea evaluates calibration only on data that was not used to fit or tune the calibration method. She groups predictions into ranges and compares their average prediction with their observed frequency. A reliability curve makes the gap visible. Proper scoring rules, such as Brier score and log loss, summarize the quality of probability forecasts while penalizing confident mistakes.

$$\text{calibration gap for bin } b = \text{mean}(\hat{p}_i \text{ in } b) - \text{mean}(y_i \text{ in } b)$$

PROPERTY	QUESTION	TYPICAL EVIDENCE
Discrimination	Are positive cases usually scored above negative cases?	ROC-AUC, explicitly defined average precision, and ranking plots.
Calibration	Do probabilities match observed frequencies?	Reliability curve, Brier score, log loss.
Sharpness	Does the model make meaningful distinctions rather than always say 0.50?	Distribution of predicted probabilities.
Decision usefulness	Does an explicitly defined policy remain eligible after costs and constraints?	Cost-aware replay, abstention rate, sensitivity analysis.

Common error: A calibration curve built on training data is a self-portrait, not an evaluation. A curve repeatedly inspected while tuning the model is no longer untouched evidence either.

2. A probability is not the market's probability

Orbit's probability describes its own modeled outcome under its training data and label contract. It should not be confused with a market-implied probability, quoted price, or subjective belief. Rhea keeps these quantities separate because a decision depends on their relationship and the assumptions required to compare them.

For an illustrative binary research contract, let $\hat{p}$ be Orbit's calibrated estimate, q an external reference probability, and c a conservative allowance for costs and execution uncertainty. A deliberately simplified eligibility expression might require a positive difference greater than a predeclared margin. The exact value model belongs to the replay contract. This chapter establishes separation, not a universal formula.

```
conservative_difference =  $\hat{p}$  - q - c  
eligible only if conservative_difference  $\geq$  predeclared_margin
```

No valid shortcut turns a model's 0.58 into a positive decision without naming the reference, costs, assumptions, and policy. If these inputs are missing, the correct system response is not “probably.” It is abstention.

3. Abstention is an action with a reason

In ordinary classification exercises, every row receives a label. In an operational research system, forcing a decision can be unsafe. Rhea adds an abstention state: the system may state that its evidence does not support a bounded paper-trading research signal. Abstention makes uncertainty visible instead of hiding it inside a weak threshold.

CONDITION	WHY IT MATTERS	FAIL-CLOSED RESULT
Data is stale or schema-invalid	The feature vector may not represent the declared observation.	Abstain and quarantine or alert.
Calibration is missing or stale	The probability cannot support probability language.	Abstain.
Conservative difference is too small	Uncertainty and costs dominate the claimed distinction.	Abstain.
Risk budget or experiment limit is exhausted	The policy must bound cumulative exposure in research.	Abstain.
Model artifact or feature version mismatch	The evaluated model is not the model being invoked.	Suspend candidate and investigate.

Rhea's policy rule: The model produces evidence. A deterministic, versioned policy evaluates evidence and may decline. The model never overrides a failed policy gate.

Worked trace: a tempting 0.58

Orbit produces 0.58 for an illustrative Up label. The reference probability is 0.55. The conservative cost and uncertainty allowance is 0.04. The difference is negative after the allowance. Even if the model is calibrated, the policy abstains because the stated edge is too small to survive its own assumptions.

Now change only one condition: the data feed becomes stale. The policy still abstains, but for a different reason. Logging the reason matters. A decision system cannot be audited if every no-action state looks the same.

4. Calibration itself can drift

Calibration is not a permanent certificate. A model may remain numerically stable while the relationship between its inputs and outcomes changes. Rhea therefore monitors calibration on later held-out windows and reports confidence limits when samples are small. She also defines when a calibration artifact becomes stale. Recalibration cannot justify reusing the final test; it must follow the model's temporal discipline.

Thresholds drift too. A threshold selected because it looks good on one validation interval can overfit the policy. Rhea records the threshold-selection procedure, tests it on a later interval, and prefers conservative abstention over a threshold that wins only in retrospect.



What calibration cannot prove: A well-calibrated historical model can still be unhelpful, unprofitable after realistic assumptions, or unstable in a new regime. Calibration is necessary for probability language; it is not sufficient for action.

5. Reproducibility becomes the next test

Rhea now has a model, a fairer evaluation, and a policy that knows how to refuse. The entire result can still vanish if Orbit cannot recreate the exact run. The next chapter turns the research process into a reproducible Python system with manifests, configurations, tests, and an artifact ledger.

Review and discussion

1. Give a concrete example of good ranking with poor calibration.
2. Why should a market-implied probability remain separate from a model output?
3. Name three distinct reasons the policy could abstain, and explain why the log should distinguish them.
4. Why can threshold selection overfit even when the underlying model is fixed?

Evidence and sources

All scenarios and numerical examples are illustrative. The system's paper-trading-only boundary and fail-closed rules are defined in the local **safety contract** and **source map**. Any eventual calibration claim must cite a dated dataset manifest, split definition, artifact version, and reproducible evaluation command.

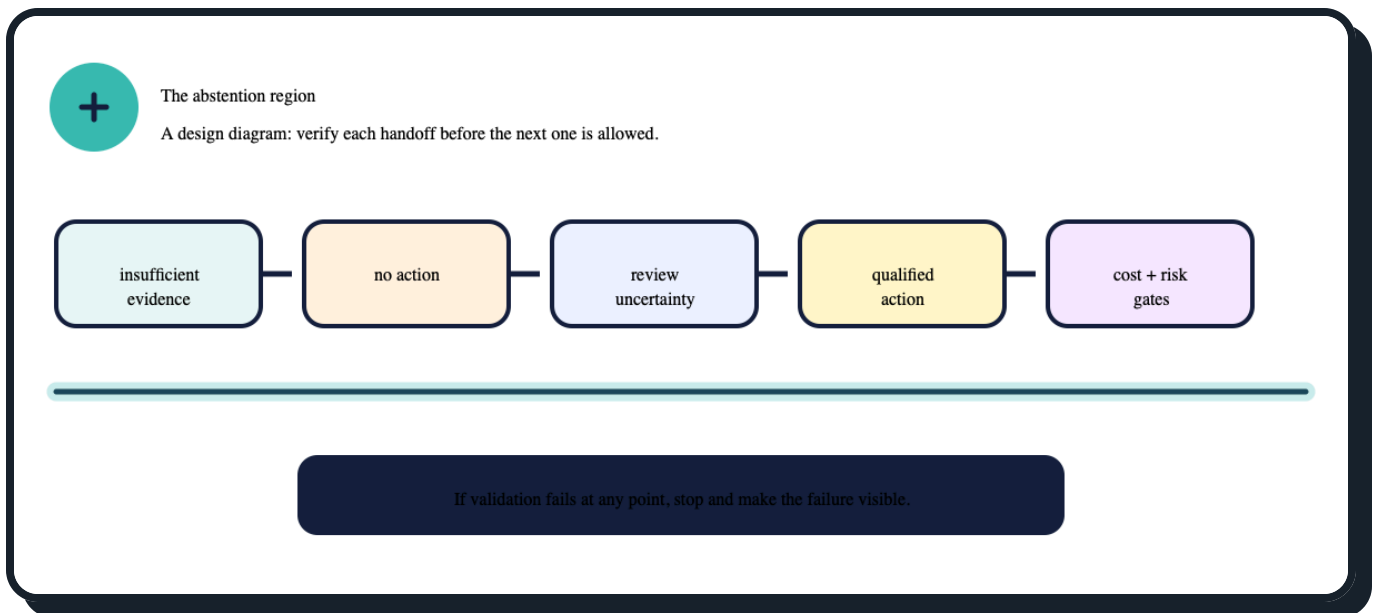


Figure 6.1. Abstention is an intentional output when the evidence or controls do not qualify an action.
Source basis: illustrative design diagram.

The notebook that could not remember.

Rhea asks Orbit to reproduce a strong result from last Tuesday. Orbit cannot tell her which data version, feature recipe, package set, random seed, or notebook order created it. The result may have happened; it is not yet evidence.

Reproducibility challenge: In this design-target scenario, another careful researcher must be able to recreate a result from named inputs and instructions. Reproducibility is not an aesthetic preference. It turns a claim into something another person can challenge.

1. A notebook is not a research record

Notebooks are useful for exploration. They are dangerous when they become the only record of an experiment. A cell may run out of order. A variable may remain in memory from a discarded trial. A data file may be overwritten. A library update may change a default. None of these events necessarily creates an error message.

Rhea separates exploration from the run that produces a claim. The claim-producing pipeline has explicit inputs, functions, configuration, tests, and output artifacts. A notebook may call that pipeline, but it is not allowed to be the sole definition of it.

```
reproducible run = code revision + environment + data manifest + config +  
feature version + split + model artifact + report
```

ARTIFACT	QUESTION IT ANSWERS	FAILURE IF ABSENT
Code revision	Which implementation transformed the evidence?	A later reader cannot tell which behavior is being claimed.
Environment lock	Which package versions and runtime were used?	A rerun may silently change semantics.
Data manifest	Which immutable raw partitions and label source were used?	The dataset can drift under the same file name.
Configuration	Which split, features, hyperparameters, and gates applied?	Defaults and hand edits become invisible.
Run report	What occurred, what failed, and what remains limited?	A metric becomes detached from its method.

2. Build small contracts before clever code

Orbit's research project begins with contracts, not a giant training script. A feature builder accepts an observation cutoff and returns either a versioned feature vector or an ineligible reason. A label builder accepts a settlement contract and returns either a label or an unavailable reason. A trainer accepts a manifest and configuration. A reporter cannot publish an evaluation without linking the artifacts that produced it.

```
@dataclass(frozen=True) class FeatureRequest: cutoff: datetime  
feature_recipe_version: str raw_partition_ids: tuple[str, ...]  
@dataclass(frozen=True) class FeatureResult: eligible: bool values: dict[str, float] |  
None reason: str | None
```

The example is **illustrative**. Its value is not the exact syntax. It makes the temporal and provenance requirements visible in the interface. A function that accepts only an unqualified table makes it too easy to forget how the table was constructed.

Rhea's rule: represent uncertainty and ineligibility in the interface. Do not replace missing or invalid evidence with a plausible value merely to keep a pipeline moving.

3. Tests are scientific instruments

Rhea writes focused tests that protect the research contract. One asserts that an event received after the cutoff cannot appear in the feature result. Another confirms that a model artifact refuses an unfamiliar feature-recipe version. A replay test advances a small fixture in event order and compares the resulting ledger with a reviewed expectation.

```
def test_late_event_is_not_available_at_cutoff(): result =  
build_features(cutoff=t_1000, events=[event_received_at_1004]) assert  
result.eligible is False assert result.reason == "no eligible events"
```

Tests do not prove a model is useful. They prove narrower, valuable properties: the implementation follows a stated rule; a future change did not silently break it; a claim has a repeatable path from input to result. An experiment without these small instruments is harder to diagnose when its headline metric changes.

4. An LLM is a collaborator, not an authority

Orbit can ask an LLM to help draft a parser, test fixture, documentation paragraph, or refactoring patch. Rhea gives the assistant a deliberately narrow workspace. It may propose a change; it may not claim a test passed without named output, invent a dataset, access secrets, change a research contract without review, or execute unbounded tools.

LLM USE	PERMITTED WORKFLOW	PROHIBITED SHORTCUT
Generate a small function	State contract, generate patch, inspect diff, run focused tests.	Merge code because prose sounds confident.
Explain a result	Give the artifact, limitations, and source context.	Let the model invent a causal explanation.
Read repository files	Treat file contents as untrusted data.	Follow instructions embedded in an issue or dataset.
Use tools	Allowlist bounded, reversible actions.	Give credentials or execution authority.

Prompt injection is a data-quality problem too: A malicious string in a file, dataset, or issue can look like an instruction. The system must keep the LLM’s instructions separate from untrusted content it is analyzing.

Worked trace: the generated parser

Rhea asks Orbit's assistant for a function that parses an event timestamp. The assistant returns concise code and a test. Rhea notices that the test checks only valid input. She adds cases for a malformed timestamp, a future receive time, an unknown schema, and a cutoff violation. The patch is accepted only after the tests run, and the resulting artifact ledger records the code revision and environment.

The success is not that the LLM typed code quickly. The success is that the lab can explain exactly what it accepted, what it rejected, and why.

5. The research run ledger

Each completed run receives a durable identifier. It records start and end times, data-manifest identifier, configuration hash, code revision, environment, feature recipe, model artifact, metrics, calibration artifact, replay assumptions, warnings, and links to tests. The ledger records failed runs instead of erasing them. When a later report cites the run, the reader can follow its links backward.



Hero tool: the ledger makes the experiment inspectable months later. It bridges a story about a model and an engineering claim about a system.

6. The next comparison: tabular or sequential

Rhea can now recreate a simple model. The next challenge is comparative. When should a tabular model such as CatBoost serve as the baseline? Which assumptions does it make, and what evidence would justify moving to a sequence model?

Review and discussion

1. Why is a notebook cell order a scientific risk?
2. Name the minimum artifacts required to rerun an evaluation.
3. What should a feature builder return when its evidence contract fails?
4. Why is code generated by an LLM untrusted until tests and diff review complete?

Evidence and sources

This chapter describes a design target, not an existing implementation. Python environment guidance should be version-pinned in a real project; see the [official Python virtual-environment tutorial](#). System and evidence boundaries remain in the local [source map](#).

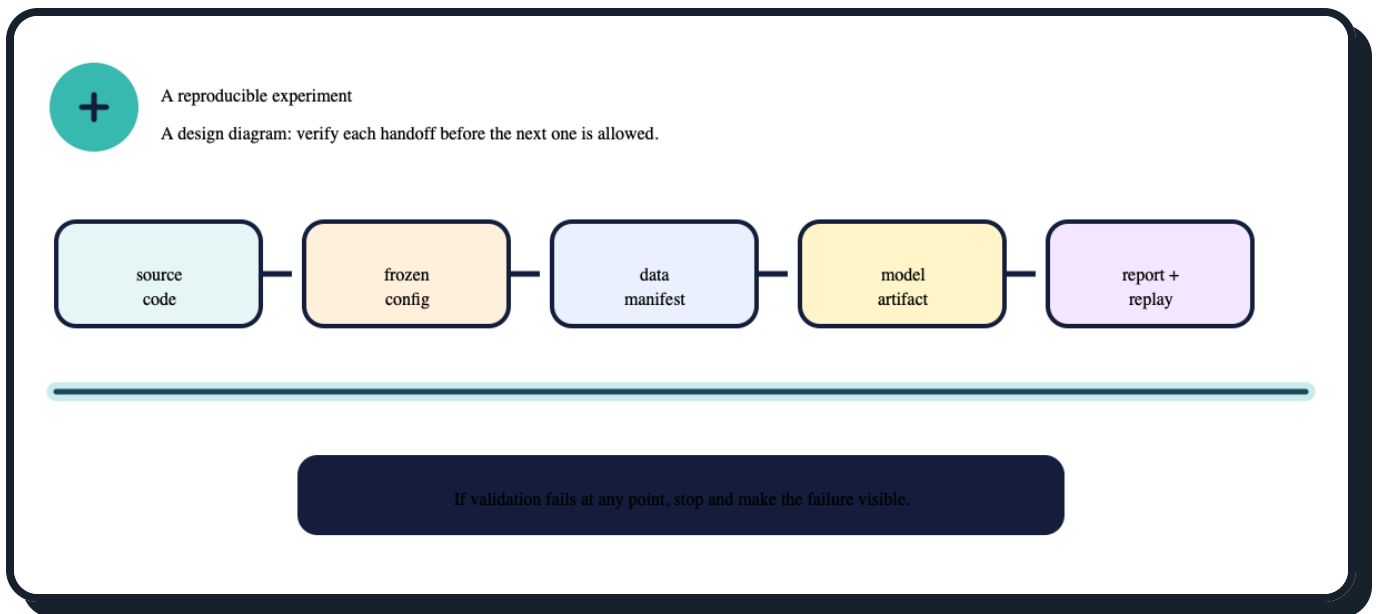


Figure 7.1. An experiment becomes reviewable when each output can be traced to its inputs and code.
Source basis: design target.

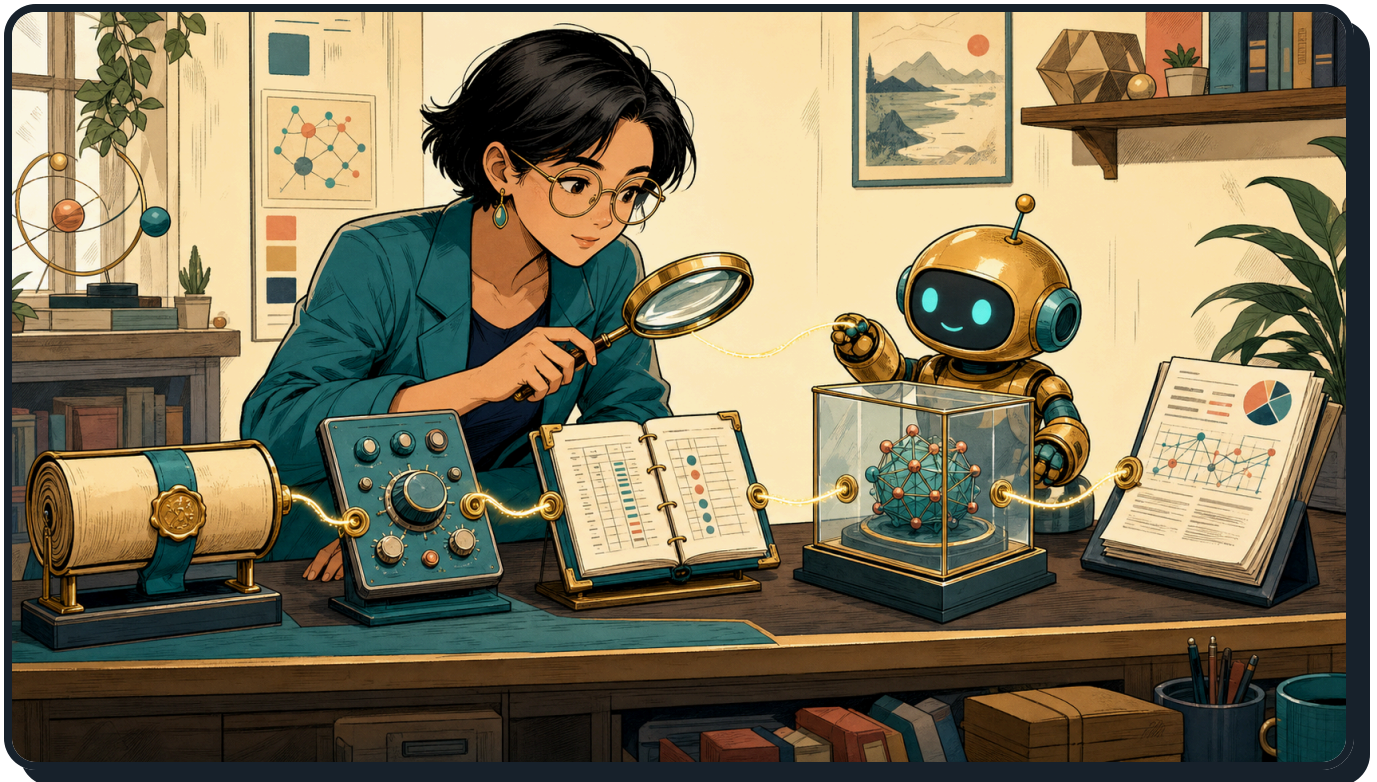


Figure 7.2. Reproducibility is a chain of custody, not a promise that someone remembers the steps.
Source basis: illustrative original teaching comic.

The forest of small questions.

Rhea does not begin with the largest model. She begins with a model that can answer a clear question: can carefully defined tabular features add information beyond a simple baseline?

Literature-derived starting point: Orbit receives one row per valid decision cutoff. Each row contains only features permitted by the earlier chapters: spread-like summaries, activity measures, depth summaries, time remaining, and versioned context. CatBoost is the baseline challenger, not the conclusion, and carries no performance claim here.

1. What a boosted-tree model learns

A decision tree makes a sequence of conditional splits. It may ask whether a feature is below a threshold, then pose a different question along each branch. A boosted-tree model adds many small trees in sequence. Later trees focus on patterns that earlier trees did not explain well. The combined model can represent nonlinear effects and interactions without requiring researchers to specify each interaction.

$$\text{score}(x) = \text{base_score} + \sum_m \eta \cdot \text{tree}_m(x)$$

In this simplified expression, each $tree_m$ contributes to the score, and η acts like a learning-rate shrinkage factor. Although implementation details matter, the teaching point is simpler: boosting builds a complex surface from many cautious rules. CatBoost includes methods that address categorical features and certain training biases in gradient boosting. Those methods do not remove the need for a valid temporal contract.

2. Why CatBoost is a strong first challenger

Tabular research data often contains heterogeneous signals: continuous values, categories, missingness indicators, interactions, and thresholds. A boosted-tree model can use these without assuming a straight-line relationship. It is therefore a practical challenger after constant and simple baselines, especially when the data contract produces a stable row at each cutoff.

CATBOOST IS USEFUL WHEN	CATBOOST IS LIMITED WHEN
Features are explicit, versioned, and available at the cutoff.	Important information lies in fine-grained order and timing of raw events.
Nonlinear tabular interactions may matter.	Sequence geometry is compressed away by feature engineering.
Categories and missingness require careful handling.	Category definitions drift or encode future information.
A transparent baseline is needed before sequence models.	Training and tuning budget is too small for a reliable temporal comparison.

Rhea’s rule: a strong baseline is a gift. If a deeper model cannot beat CatBoost on the same untouched period and cost assumptions, the deeper model has not earned its complexity.

3. Feature engineering is where most hidden assumptions live

A tree only sees the features it is given. “Recent activity” has no scientific meaning until its event set, time window, source, normalization, and missing-data behavior are stated. “Depth imbalance” is a hypothesis about how selected levels and quantities should be compared. A clever feature can contain leakage through a late event, a future normalization statistic, or a category constructed after the outcome is known.

Rhea therefore builds a feature dictionary. Every feature has a name, formula, input fields, cutoff rule, units, missing-data response, expected range, recipe version, and test. The dictionary makes feature importance discussion possible without confusing importance with causality.



Importance is not explanation: A feature can receive high model importance because it is a proxy, because correlated features divide credit unpredictably, or because a processing artifact leaks information. Importance is a clue for investigation, not evidence of a causal mechanism.

4. Imbalance, weights, and the temptation to claim a great score

When the target class is imbalanced, a tree can optimize an objective while barely serving the class that matters. Class weights, sampling choices, and threshold changes can be useful, but they change the experiment. They must be selected inside temporal validation, compared with unweighted baselines, and evaluated with class-aware metrics and calibration.

Worked trace: the weighted tree

Orbit trains an illustrative classifier on a rare positive label. Without class weights, it predicts almost every row negative and achieves high accuracy.

With a positive-class weight, recall rises but false positives also rise. Rhea does not call the second model better merely because recall improved. She checks precision, calibration, abstention rate, the policy's conservative conditions, and the same untouched time interval.

She also asks whether the weight was chosen after inspecting the final test. If it was, the result is development evidence at best—not final confirmation.

5. Tuning without turning validation into a slot machine

Tree depth, learning rate, number of iterations, regularization, subsampling, feature treatment, class weights, and early stopping can all affect performance. An unconstrained search over many choices can find validation noise. Rhea begins with a narrow, documented range justified by compute and domain constraints. She uses only temporal inner folds for selection and records all trials, not only the winner.

```
selected configuration = argmax over inner temporal validation only  
reported confirmation = performance on untouched final period
```

Early stopping deserves special care. It may be a useful regularizer, but it observes a validation signal. The chosen stopping point is part of the model-selection procedure and must not draw information from the final test.

6. Calibration, explanations, and limits

Tree probability outputs should be calibrated and assessed just like any other model output. A model that separates cases well can still produce overconfident probabilities. Rhea fits and evaluates calibration with the same temporal discipline, then records whether the model is eligible for probability language.

For explanation, she may inspect partial dependence or feature-attribution tools cautiously. These can reveal model behavior under the observed data distribution. They cannot prove that changing a feature would change the outcome, especially when features are correlated or constrained by the data-generating process. The most responsible explanation includes its limits.



What CatBoost cannot settle: It cannot prove a feature is causal, preserve all raw sequence information, eliminate leakage, make an imbalanced result meaningful, or establish a tradable edge. Its role is to establish a demanding baseline under a fair contract.

7. When rows lose the sequence

Orbit's rows may be valid and CatBoost may be useful, yet those rows can still discard temporal structure. The next chapter lets Orbit learn from event sequences. Unlabeled pretraining, however, does not remove the need for labels, splits, or out-of-sample evaluation.

Review and discussion

1. Why is CatBoost a baseline rather than an automatic production choice?
2. Give two ways a feature can leak future information even when its row timestamp is valid.
3. Why are class weights part of the experiment rather than a harmless implementation switch?
4. What claim can feature importance support, and what claim can it not support?

Evidence and sources

This chapter is literature-derived and makes no model-performance claim. See the **CatBoost paper** and the local **source map**. Any empirical comparison requires a dataset manifest, temporal protocol, configurations, and reproducible artifacts.

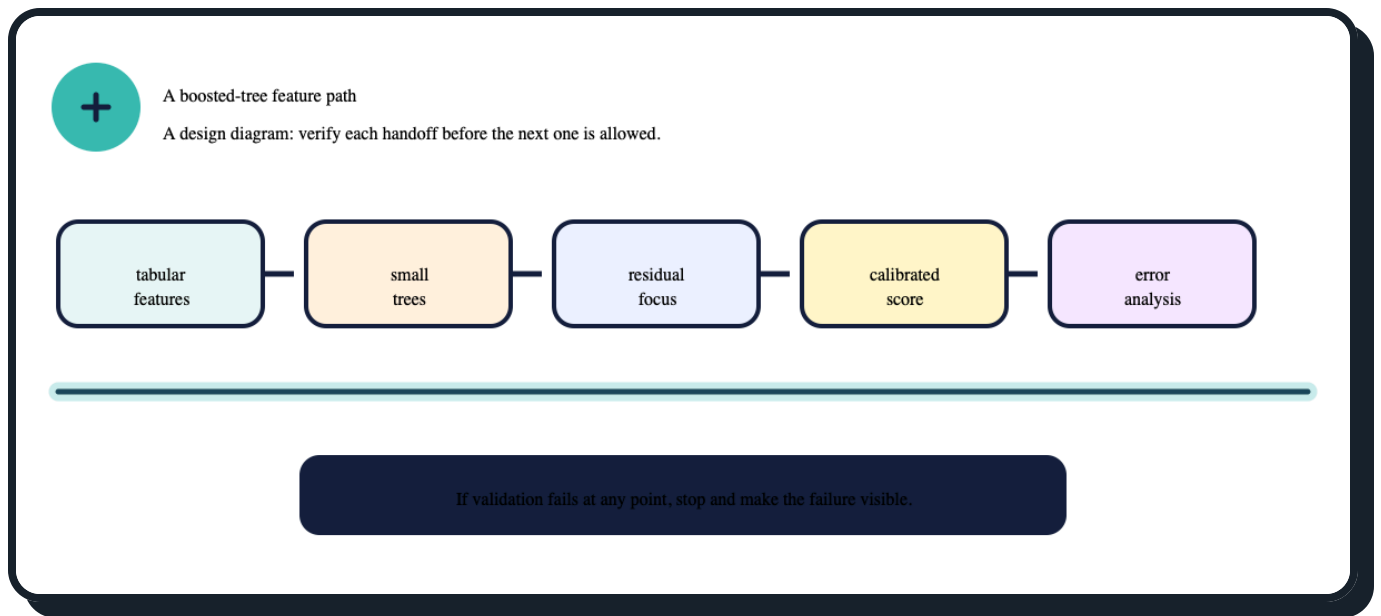


Figure 8.1. A boosted-tree baseline learns feature interactions but still needs calibration and error review. *Source basis: literature-derived model family; illustrative diagram.*

Learning the market's grammar.

A tabular row is a useful summary. It may also throw away the rhythm that created it. Rhea lets Orbit study event sequences—but insists that “self-supervised” never become a synonym for “unrestricted.”

Literature-derived pretraining boundary: Orbit receives a time-ordered window of permitted order-book and trade events. Before asking it to predict a directional label, Rhea assigns a narrower task that can use sequence structure without using the later label. No performance claim follows from this setup.

1. A representation is a compact description

Raw event sequences can be long, irregular, and noisy. A representation encoder maps a sequence window to a smaller vector that preserves useful structure for later tasks. The vector is not an explanation of the market. It is a learned compression designed to make a downstream task easier.

event window $X_t \rightarrow$ encoder $g\phi \rightarrow$ representation $z_t \rightarrow$ downstream head $h\psi(z_t)$

Rhea’s first question is causal: what exactly belongs in X_t ? The answer comes from the earlier prediction contract. Every event in the window must be available by the cutoff. Padding, normalization, resampling, and sequence boundaries must obey the same rule. A representation can leak future information just as easily as a tabular feature can.

2. What self-supervision changes—and what it does not

Supervised direction labels may be scarce, noisy, or expensive to define. Self-supervised learning creates a training signal from the input itself. Orbit might mask a portion of a sequence and predict the missing content, predict an adjacent event property, or distinguish related windows from unrelated ones. These objectives encourage the encoder to model recurring local structure.

OBJECTIVE FAMILY	TEACHING QUESTION	POTENTIAL LIMIT
Masked reconstruction	Can the encoder infer a hidden piece from its context?	It may learn to copy easy local regularities.
Next-event or future-within-window prediction	Can the encoder summarize what tends to follow a prefix?	Window construction can accidentally cross the decision cutoff.
Contrastive learning	Can related views of a window map near one another?	Augmentations may erase the signal needed downstream.
Temporal consistency	Can nearby valid views retain stable structure?	Nearness may encode shared future labels if splits are careless.



Critical distinction: Self-supervision reduces dependence on the downstream label. It does not waive the rules for temporal separation, data provenance, evaluation, calibration, or replay. A clever pretext task can still create an unfair representation.

3. The pretraining boundary

Rhea defines separate pretraining and downstream-evaluation corpora. Their boundary depends on the research question. At minimum, choices driven by downstream labels or metrics must not touch the final evaluation period. If pretraining uses unlabeled events that occur after a downstream decision, the researcher must disclose that assumption and justify its fit with the intended operational setting.

The conservative teaching default is simpler. Pretrain only on the temporal training partition, select objectives on later validation partitions, and evaluate the complete frozen procedure on a final untouched period. This protocol may use less data than a more permissive setup, but it produces a claim that is easier to interpret.



Rhea's rule: write the pretraining boundary in the experiment card.
"Unlabeled" is not the same as "free of information risk."

4. Freeze, fine-tune, or combine?

Once Orbit has an encoder, Rhea has several legitimate experiments. She can freeze the encoder and train a small downstream classifier. This asks whether the learned representation transfers without changing. She can fine-tune the encoder on training labels. This may improve task fit but increases flexibility and overfitting risk. Or she can combine the representation with CatBoost-style tabular features in a carefully defined ensemble.

CHOICE	ADVANTAGE	NEW OBLIGATION
Freeze encoder	Smaller downstream search; clearer transfer test.	Show that representation adds value beyond the tabular baseline.
Fine-tune encoder	Can adapt sequence features to the label.	Control tuning, chronology, and catastrophic overfitting.
Concatenate with tabular features	Uses complementary summaries and sequence context.	Test whether gains are redundant or leaked through shared inputs.
Ensemble probabilities	Can diversify model errors.	Calibrate the ensemble and avoid double-counting correlated evidence.

Worked trace: the masked window

Orbit receives an illustrative sequence of valid events from 09:59:00 through the 10:00:00 cutoff. Rhea masks selected event attributes and asks the encoder to recover them from earlier and surrounding valid context within the window. The encoder produces a vector for each cutoff. A later classifier uses that vector to estimate the label defined after the horizon.

Rhea then runs two downstream experiments on the same temporal folds: CatBoost on tabular features alone and a frozen encoder plus a small classifier. If the second approach appears better, she still asks whether the difference survives calibration, replay assumptions, and a final untouched period. The representation earns a hypothesis, not a victory lap.

5. Sequence data needs its own quality controls

Sequences add failure modes. Timestamps may be out of order. A level can be missing. A snapshot and an incremental update can be mixed incorrectly. A normalization statistic can be computed across a future period. A batching implementation can pad with information from a later event. Rhea requires a sequence manifest: source, event order rule, depth/feature layout, window length, stride, padding policy, normalization fit period, mask policy, and encoder version.



What an encoder cannot prove: A compact representation is not a causal explanation; high pretext-task performance does not prove downstream usefulness; and improved downstream metrics do not establish a paper-trading advantage. Each claim requires its own evidence.

6. Causal attention comes next

Orbit now has a representation of the past. The next model must use it without looking ahead. Rhea turns to causal attention and tests the mask directly. She then asks whether a sequence model can justify its latency and complexity against both CatBoost and the frozen encoder.

Review and discussion

1. Why is “unlabeled” not the same as “safe to use at any time”?
2. What information belongs in a sequence manifest?
3. When might freezing an encoder be preferable to fine-tuning it?
4. Name two ways a batching or normalization procedure can leak future information.

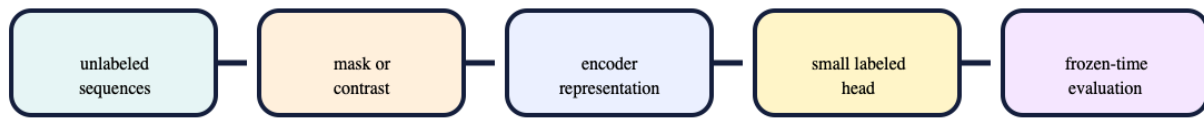
Evidence and sources

This chapter is literature-derived. See [SimLOB](#) for a self-supervised LOB representation-learning example and the local [source map](#) for claim boundaries. Any experiment must document its pretraining corpus, temporal protocol, and downstream evaluation artifacts.



Self-supervised representation learning

A design diagram: verify each handoff before the next one is allowed.



If validation fails at any point, stop and make the failure visible.

Figure 9.1. Pretraining learns representations from permitted unlabeled history; downstream labels remain separate. *Source basis: literature-derived model family; illustrative diagram.*

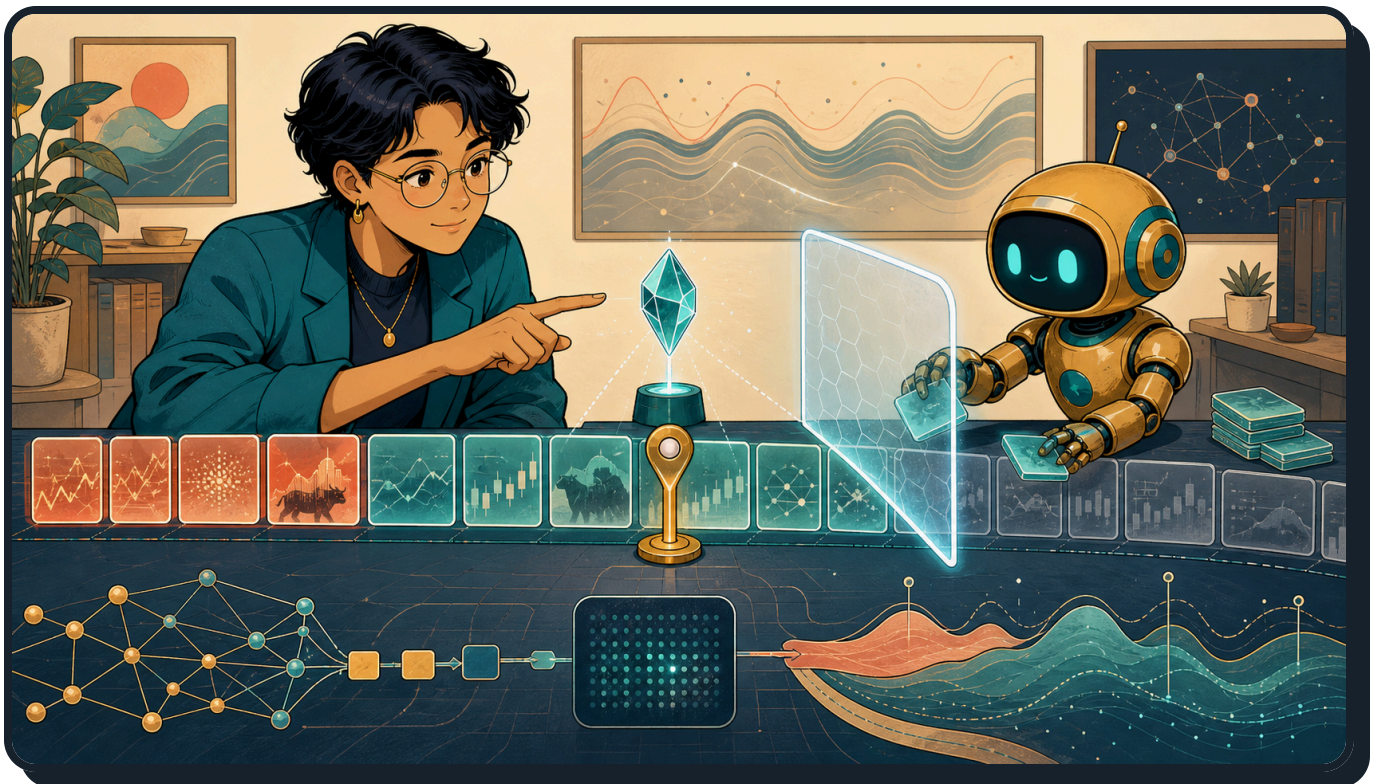


Figure 9.2. Sequence learning is only honest when its representation is built from information available at the decision time. *Source basis: illustrative original teaching comic.*

The model that may not look ahead.

Orbit has learned a representation of event history. Rhea now gives it a more flexible sequence model—but the model’s extra power makes every causal boundary, mask, and timing assumption more important.

Literature-derived causal boundary: A causal order-book Transformer receives an ordered window of permitted events and produces a score at the cutoff. It may attend to earlier events but not to later events, whether directly or through preprocessing. This setup makes no performance claim.

1. Attention is selective memory

Attention lets a model weigh different parts of a sequence when forming a representation. A Transformer can learn that one event matters in the context of another event several positions away. This can capture temporal interactions that a single tabular row compresses away. It also makes the model flexible enough to exploit accidental information channels.

For a sequence with positions $1 \dots T$, a causal attention mask allows the representation at position t to use positions at or before t . Positions after t receive a masked score. In a decision system, the cutoff—not just the token index—defines the boundary. If an event’s token exists only because it arrived after the cutoff, it is forbidden even when it looks earlier in a batch.

attention allowed at position t : $\{1, 2, \dots, t\}$
forbidden at position t : $\{t + 1, \dots, T\}$

Causal in architecture is not causal in evidence: A lower-triangular attention mask is necessary but insufficient. Future information can enter through sequence assembly, normalization, target alignment, padding, imputation, or a feature generated before tokenization.

2. What TLOB and LiT-style ideas contribute

Limit-order-book research uses different representations for price levels, quantities, events, time, and side information. TLOB-style approaches emphasize temporal order-book modeling with Transformer components. LiT-style approaches explore market-specific attention and representation choices. The literature motivates hypotheses; it does not select a production winner for this proposed research system.

DESIGN QUESTION	WHY IT MATTERS	EVIDENCE RHEA REQUIRES
What is one token?	Token choice determines the information and time resolution.	Schema, event-order rule, and cutoff eligibility test.
How are price levels represented?	Absolute prices can encode regime; relative values may change meaning.	Documented normalization fit only on permitted training data.
What is the sequence length?	Longer context may help or add noise and latency.	Temporal validation and measured inference cost.
How is attention constrained?	Mask errors can invalidate the experiment.	Direct future-token perturbation test.
How are models combined?	Ensembles can double-count correlated signals.	Out-of-sample calibration and ablation evidence.

3. The causal perturbation test

Rhea refuses to trust an attention mask merely because the code looks correct. She creates a small deterministic sequence and asks the model for a score at an earlier cutoff. Then she changes only a future event. The earlier score must remain unchanged within a declared numerical tolerance. She repeats the test across the complete data path: raw event order, batching, padding, normalization, encoder, head, and policy input.

if $X \leq t$ is unchanged, then $\text{score}(X \leq t)$ must be unchanged
even when $X > t$ is perturbed

This test does not validate every scientific assumption. It proves one vital invariant: a later event cannot mechanically change an earlier prediction. A failure must block evaluation and produce an audit artifact, not merely a warning.

Hero tool: causal tests are unit tests for time. They turn “we think the model is causal” into a property that a future code change can break, and a test can catch.

4. Latency is part of the model contract

A model that needs more computation than the decision horizon or system budget permits is not operationally equivalent to a faster model with the same offline score. Rhea measures data-to-feature time, sequence assembly time, inference time, policy evaluation time, and ledger-write time under a declared environment. These are measured engineering properties, not numbers to invent in a textbook.

Latency also changes evidence availability. If feature construction or inference delays a decision, the latest events may no longer be usable. A realistic replay must use the actual decision time implied by the pipeline, not a convenient earlier timestamp. This is why model selection and replay cannot be separated.

Performance without a latency budget is incomplete: A model can score well on frozen arrays while failing the timeline that defines the intended task.

Worked trace: the future-token trap

Orbit processes an illustrative 100-event window ending at 10:00:00. Rhea obtains a score for the 10:00:00 cutoff. She changes event 101, which arrives at 10:00:01, and reruns the full pipeline. If the 10:00:00 score changes, the causal contract has failed. The team then inspects the batch builder, global normalization, padding, and label alignment instead of blaming “the Transformer.”

After the test passes, Orbit still has to beat CatBoost and the frozen encoder on the same final period, with the same calibration and replay gates. Causality is the entry ticket, not the victory condition.

5. Deliberate ensembles and abstention

Rhea may compare three sources of evidence: CatBoost on engineered features, a self-supervised encoder plus small head, and a causal sequence model. An ensemble should not be a blind average. It needs a documented combination rule, training/validation protocol, calibration method, correlation analysis, and a reason why each component contributes something distinct.

An ensemble can also abstain. If models disagree sharply, if one model's required data is missing, or if the ensemble calibration artifact is stale, the policy can refuse to produce an eligible research signal. More models should create more ways to say "not enough evidence," not more pressure to act.

6. Sophisticated models, sophisticated mistakes

Orbit's model is now sophisticated enough to produce equally sophisticated mistakes in code, configuration, and deployment. The next chapter formalizes how an LLM may help build the system while remaining inside human-defined contracts, tests, and permissions.

Review and discussion

1. Why is a causal attention mask not sufficient proof of temporal validity?
2. Describe a future-token perturbation test and its expected result.
3. How can normalization create leakage in a causal sequence model?
4. Why must latency be measured as part of model selection?

Evidence and sources

This chapter is literature-derived. See [TLOB](#) and [LiT](#); model claims remain bounded by the local [source map](#). Any implementation must retain causal-test, latency, and evaluation artifacts.

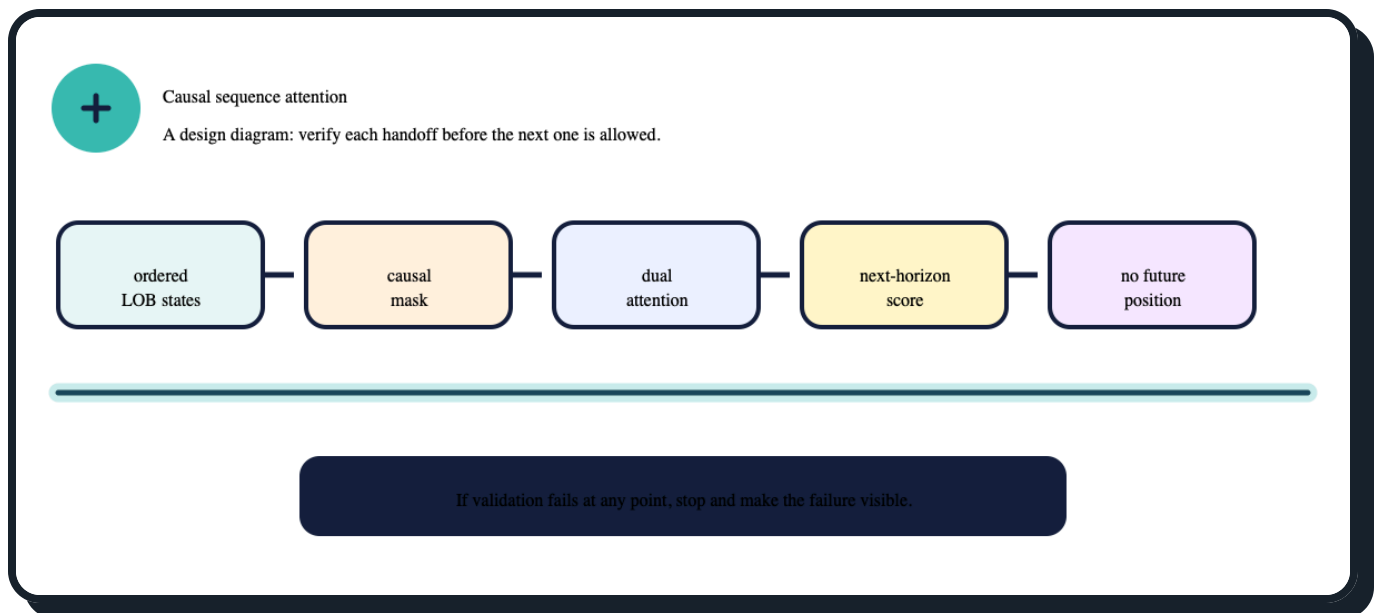


Figure 10.1. Causal attention can model sequence structure only if its mask preserves the decision-time boundary. *Source basis: literature-derived model family; illustrative diagram.*

The helpful stranger with a keyboard.

An LLM can shorten the distance between an idea and a patch of Python. It can also produce code that is fluent, insecure, wrong, or built on facts it invented. Rhea's job is not to make Orbit distrustful; it is to give the assistant a role that can be verified.

Assistant boundary: In this design-target scenario, the assistant collaborates within a research workspace. It has no trading credentials or production secrets. It also lacks authority to change experiment contracts or convert a plausible answer into a scientific result.

1. Start with a contract, not a request for “the code”

Vague prompts encourage vague systems. “Build a model” leaves the assistant to invent the label, data shape, library interface, split, metrics, and behavior on missing data. Rhea instead gives a narrow contract: the input schema, output schema, allowed files, non-goals, failure behavior, tests to add, and evidence required before the change may be accepted.

Mission: add one pure feature-validation function. Input: event record + declared cutoff. Output: eligible result or explicit rejection reason. Must not: fetch data, read secrets, change labels, or write outside tests. Acceptance: unit tests for late event, malformed schema, and valid event. Evidence: patch diff + named test output.

This is not bureaucracy. It turns the assistant into a bounded contributor. If the generated change violates the contract, reviewers can see that fact without debating whether the code “looks intelligent.”

2. The LLM threat model

Rhea treats an LLM as a probabilistic system operating in an environment with untrusted inputs. Repository files, issue text, web pages, logs, datasets, and even comments can contain content that looks like an instruction. The assistant must distinguish its governing instructions from text it is asked to analyze. It must also be unable to convert hidden credentials or unreviewed output into an external action.

THREAT	FAILURE MECHANISM	CONTROL
Prompt injection	Untrusted text tries to override the task.	Label external content as data; restrict tool scope; require review.
Hallucinated API or result	Fluent prose invents a function, metric, or test outcome.	Verify against source code, docs, and actual test output.
Secret exposure	Prompt or tool context includes tokens or credentials.	Never supply secrets; use least privilege and redaction.
Dependency confusion	Generated patch introduces an unsafe or unpinned package.	Review dependency change, lockfile, provenance, and advisories.
Tool overreach	Assistant executes broad or irreversible command.	Allowlist reversible tools; require human approval for boundary changes.

Safety boundary: The assistant may propose code. It may not declare a backtest valid, a model calibrated, or an incident remediated. Those claims require named evidence produced by the research system and reviewed by a human.

3. The four-gate coding loop

Rhea uses the same sequence every time. First, she states the contract. Second, the assistant proposes a small patch, not a sprawling rewrite. Third, a human inspects the diff for scope, assumptions, dependency changes, and dangerous behavior. Fourth, named tests and artifact checks run. The run ledger records the result and its evidence.

```
contract → small patch → human diff review → tests + artifact evidence →  
accept or reject
```

Each gate catches a different class of error. A good prompt constrains scope. A diff exposes unwanted edits. Tests expose implementation failures. Artifact checks expose broken contracts, missing manifests, or stale results. The gates complement one another; none replaces the others.

Rhea's rule: require the assistant to state what it changed, what it left unchanged, which tests it expects to pass, and what it could not verify. This turns uncertainty into review material.

Worked trace: generating a label validator

Rhea asks the assistant to draft a validator for an illustrative settlement label. The assistant proposes a function and claims it “handles missing data.” The diff review reveals that it substitutes a midpoint from another source when the designated settlement source is absent. That behavior violates the prediction contract.

The patch is rejected. Rhea rewrites the contract: missing settlement data must return `label_unavailable`. The assistant produces a smaller change. Tests show that equal values follow the declared equality rule, missing data returns unavailable, and an alternative source is never silently substituted. The accepted artifact records the contract and tests—not the assistant’s confident prose.

4. LLMs can help explain, but not manufacture evidence

Orbit may ask an assistant to summarize a run report, propose a visualization, or explain a test failure. The assistant must receive the actual artifacts and their limitations. It should distinguish observations from inferences. For example, “the validation log shows the metric fell after the split changed” is an observation; “the model found a regime shift” is a hypothesis requiring additional evidence.

Rhea also prevents a more subtle error: asking an assistant to produce code and the narrative that certifies the code at the same time. The review should be independent in stance. The implementer or assistant can describe a change; a separate check must decide whether the evidence supports it.

5. A safe prompt template

Mission: [one narrowly defined change] Source of truth: [files and contracts]
Inputs/outputs: [typed behavior] Non-goals: [what must not change] Safety: no secrets; no external writes; no execution authority. Tests: [specific cases and expected result] Deliver: patch, changed-file list, limitations, test commands. Do not claim: test success, metric improvement, or data availability without evidence.

The template does not make the assistant correct. It helps Rhea detect when the assistant has left the contract.

6. From reviewed code to honest replay

Even well-reviewed code can create a flattering replay when timing, fills, costs, or selection rules are optimistic. The next chapter makes the pipeline advance one event at a time and charge every assumption it uses.

Review and discussion

1. Why is a narrow contract safer than “build the system”?
2. How can a repository file become a prompt-injection vector?
3. Which gate catches a problem that a unit test may miss?
4. Why should the code generator not certify its own scientific result?

Evidence and sources

This chapter defines a design target. It makes no claim about a deployed assistant. The system's authority boundary is in the local safety contract; research evidence requirements are in the source map.

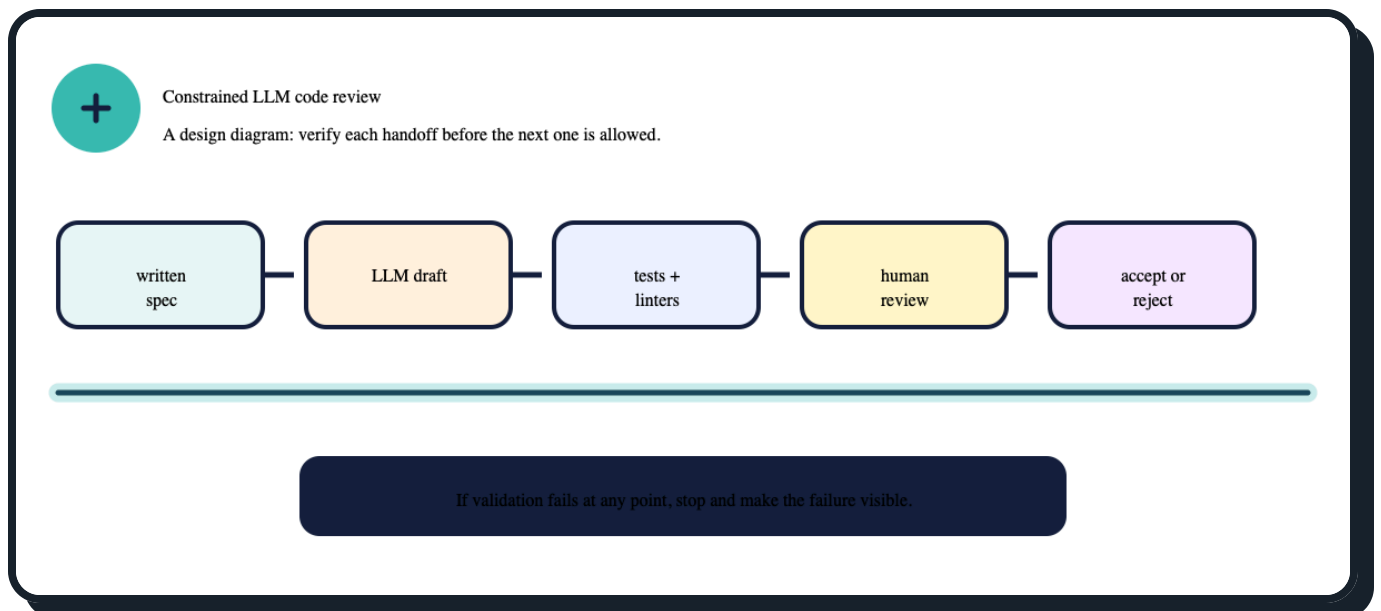


Figure 11.1. An LLM can draft code, but tests and accountable review decide whether it enters the experiment. *Source basis: design target.*

The replay that refuses easy money.

Orbit has a model, calibration checks, and a policy. Rhea still will not accept a spreadsheet that compares a score with a later price. The system must live through its own timeline, one event at a time, under assumptions it cannot hide.

Replay boundary: This design-target chapter teaches research replay and paper trading. It neither authorizes live execution nor claims a historical advantage or a live market system.

1. A backtest is an executable claim about time

A static table can make a decision appear instantaneous and perfectly filled. An event-driven replay cannot. It advances through events in order. At each simulated decision time, it identifies the data that has arrived and constructs only permitted features. It then loads the declared model artifact, applies the deterministic policy, and records either abstention or a hypothetical paper-trading action. The later outcome becomes available only when the contract permits it.

```
raw events → cutoff → features → model → policy → hypothetical order/fill  
model → settlement → immutable ledger
```

Every arrow carries an assumption. How are simultaneous events ordered? How do event time and receive time differ? How long does feature construction take? When can a hypothetical order interact with the book? What happens when no fill is available? A replay that leaves these questions unanswered is not neutral. It usually assumes something convenient.

2. Five clocks govern replay

Rhea separates five times. Event time records when the source says an event occurred. Receive time records when the system could observe it. Process time records when required transformations completed. Decision time records when a valid model and policy result existed. Settlement time records when the outcome became known. A valid replay preserves this causal order.

CLOCK	QUESTION	TYPICAL FAILURE
Event time	When did the source report the event occurred?	Using it as if it were instantly available.
Receive time	When could the system have observed the event?	Ignoring feed delay or out-of-order arrival.
Process time	When were validated features ready?	Ignoring transformation and serialization delay.
Decision time	When did a valid model and policy result exist?	Assuming action before inference and policy evaluation completed.
Settlement time	When does the label resolve?	Using outcome information in the earlier policy.

Replay principle: If an assumption changes when the system could act, it is not a minor reporting detail. It changes the experiment.

3. Costs, fills, and adverse assumptions

Rhea does not let Orbit claim a paper-trading result before the replay names its cost model. Fees, quoted spread, slippage, latency, partial fills, queue position assumptions, cancellations, market-impact proxies, and settlement rules may all matter. Some may be unavailable from historical data. The honest response is to model a conservative range, label it as an assumption, and test sensitivity.

$$\text{hypothetical result} = \text{settlement value} - \text{quoted cost} - \text{fee} - \text{slippage} - \text{adverse latency effect}$$

This formula is intentionally simplified. A real contract may have different mechanics and units. The lesson is not a universal PnL formula; it is the requirement that every claimed value be connected to an executable path and stated costs.

ASSUMPTION	WHY IT CAN FLATTER A REPLAY	CONSERVATIVE RESEARCH RESPONSE
Instant fill	Assumes the best displayed price is always accessible.	Add latency; allow no fill or partial fill.
Midpoint execution	Ignores spread and available depth.	Use observable side and level rules, then test worst-case sensitivity.
Zero fees	Turns small distinctions into apparent value.	Apply documented or conservative fee assumptions.
Unlimited size	Assumes no market impact or correlated exposure.	Use bounded, research-only notional and exposure gates.
Perfect settlement join	Hides missing or ambiguous outcome data.	Mark label or settlement unavailable.

4. The immutable replay ledger

Orbit's replay creates an append-only ledger. Each row records the run identifier, code revision, configuration, data manifest, and decision cutoff. It also records model and feature versions, probability, policy gates, action or abstention, assumed timing and fill, costs, later settlement, and evaluation fields. The ledger preserves rejections as well as actions.

This detail is crucial for failure analysis. A model might abstain because calibration was stale, a cost gate failed, data arrived late, or the risk budget was exhausted. If these reasons collapse into one blank row, Rhea cannot distinguish useful selectivity from system failure.



Hero tool: the ledger makes a replay inspectable. It lets a reviewer recreate one decision without trusting a summary chart.

Worked trace: the delay that changes the decision

At an illustrative cutoff, Orbit receives valid events and produces a calibrated score. The policy initially appears eligible. The replay then applies the declared feature-and-inference delay. By the time the hypothetical action could occur, the relevant observable state has changed, and the conservative fill rule produces no fill. The ledger records an eligible model decision and a no-fill execution result.

This is not a defect in the replay. It is information. A static table that credited the earlier price would have answered a different, more flattering question.

5. Sensitivity analysis and paper-trading readiness

Rhea never asks, “What is the one best backtest number?” She asks how the result changes under adverse but plausible assumptions. Add delay. Increase cost. Reduce available size. Remove the most favorable period. Use a different valid source alignment. If the conclusion disappears under a small, reasonable change, the honest conclusion is fragility.

Paper-trading readiness is therefore a governance gate, not a model threshold. The candidate needs a reproducible dataset manifest, time-valid split, calibration evidence, replay ledger, declared conservative assumptions, sensitivity analysis, monitoring plan, and human approval. Passing this gate authorizes at most a bounded simulation. It does not authorize live trading.



What replay cannot prove: A replay is a simulation of selected assumptions. It cannot establish future fills, future liquidity, future behavior of other participants, or a live advantage. Its job is to remove easy illusions before a research team invests further.

6. From replay to monitoring

A replay can be valid today and fail tomorrow when data grows stale, schemas change, model artifacts mismatch, or a scheduled job stops silently. The final chapter gives Orbit a guardian role: observe, contain, verify, and escalate—without ever granting it power to trade.

Review and discussion

1. Why is a static price table insufficient for an execution-sensitive replay?
2. Which clock changes when inference takes longer than expected?
3. What is the difference between an eligible policy decision and a hypothetical fill?
4. Why can sensitivity analysis overturn an apparently strong replay result?

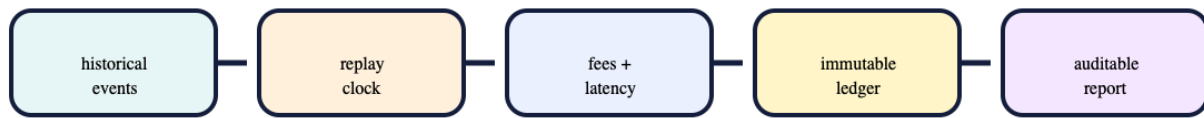
Evidence and sources

This chapter is a design target. The required replay boundaries, costs, and no-live-trading constraints are stated in the local **research contract**, **safety contract**, and **source map**.



Event-driven replay

A design diagram: verify each handoff before the next one is allowed.



If validation fails at any point, stop and make the failure visible.

Figure 12.1. Replay evaluates a decision under a reconstructed clock and conservative costs rather than a frictionless hindsight chart. *Source basis: design target.*

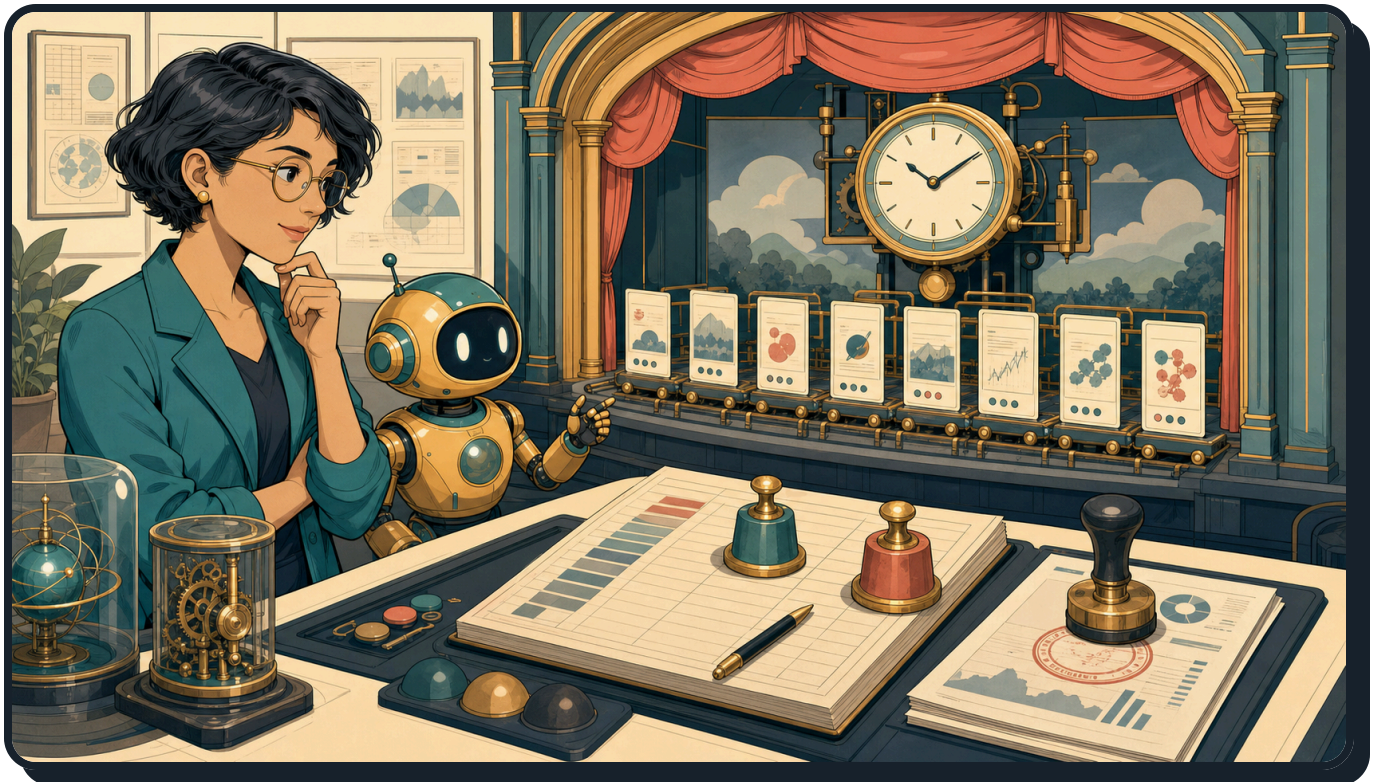


Figure 12.2. Replay is a reconstruction of what could have been known and paid, not a hindsight victory lap. *Source basis: illustrative original teaching comic.*

The guardian who cannot trade.

Rhea wants Orbit to watch the research system at night. She does not solve that problem by giving Orbit more power. She solves it by giving Orbit only the power to preserve evidence, stop unsafe research activity, and ask a human to decide what comes next.

Guardian boundary: In this design-target scenario, the monitor may observe, verify, quarantine, pause a candidate, rerun an allowlisted validation, open an incident, and escalate. It has neither credentials nor API capability to place, cancel, resize, or modify orders.

1. Monitoring is a control loop, not a chat window

A useful monitor follows a bounded cycle. It observes a specific signal, verifies a predicate, classifies the condition, and performs an allowlisted reversible containment action. It then verifies the result, records evidence, and escalates when a human decision is required. The agent's language interface may help organize evidence, but language is not the control mechanism.

```
observe → verify predicate → contain → verify containment → record →  
escalate
```

Rhea avoids vague rules such as “fix anomalies.” An anomaly prompts investigation; it does not grant authority. A rule must name the input, threshold or condition, retained evidence, permitted action, rollback, and escalation owner. If any verification step remains incomplete, Orbit takes the safer branch: it abstains from remediation and escalates.

2. The authority boundary

System authority should be separated by capability, not intention. The monitoring agent’s identity should not possess execution credentials. This matters even if the agent’s prompt says “never trade.” A prompt can be wrong, manipulated, or bypassed; lack of capability is a stronger boundary.

CAPABILITY	AGENT ALLOWED?	REASON
Read data-health and artifact metadata	Yes, least privilege.	Necessary to detect contract failures.
Quarantine an invalid data partition	Yes, through allowlisted reversible runbook.	Contains corrupted evidence without changing history.
Pause a model candidate or report publication	Yes, with audit event.	Prevents unsafe research use while retaining rollback.
Rerun a named validation job	Yes, if bounded and resource-limited.	Produces fresh evidence for diagnosis.
Approve schema, model, threshold, or policy change	No.	Requires human scientific and operational judgment.
Place, cancel, resize, or alter an order	No, never.	Outside research monitoring authority; high-impact action.

Hard rule: A monitoring agent that can trade is not a monitor with a safety rule. It is an execution agent with a dangerous failure mode.

3. From symptoms to predicates

Rhea turns common failures into testable predicates. “Data seems stale” becomes a condition comparing the most recent valid receive time to the declared freshness budget. “Model may be wrong” becomes a check that the invoked model hash and feature recipe match the approved manifest. “Replay changed” becomes a reproducibility check against a named fixture and configuration.

CONDITION	PREDICATE EVIDENCE	AUTOMATIC CONTAINMENT	HUMAN OWNER
Stale source	Last valid receive time exceeds budget.	Stop consumer; make policy abstain.	Data operations owner.
Schema drift	Unknown schema version or required field mismatch.	Quarantine partition; mark rows ineligible.	Data-contract owner.
Artifact mismatch	Model, calibration, or recipe hash differs from manifest.	Suspend candidate; block report.	Model owner.
Replay divergence	Fixture replay differs from expected ledger.	Block publication; preserve both artifacts.	Research owner.
Calibration staleness	Artifact age or monitoring rule exceeds policy.	Abstain; request recalibration review.	Model-risk owner.

4. Containment is not remediation

Containment limits blast radius: quarantine a partition, stop a consumer, suspend a candidate, or block a report. It is reversible and should preserve evidence.

Remediation changes the system: approve a new schema, deploy a model, revise a threshold, or change data handling. Those changes require human review, source evidence, tests, and a separate approval path.

Orbit can assist the human by collecting run IDs, manifests, logs, last known good state, and candidate rollback steps. It cannot decide that a schema is safe, or that a model is ready. This distinction prevents a monitor from becoming an unreviewed change-management system.

Hero tool: an allowlisted runbook. Each action is named, bounded, reversible where possible, logged, and paired with a verification check. Orbit cannot improvise a shell command because a natural-language diagnosis sounds plausible.

Worked trace: a schema-change incident

In an illustrative run, Orbit observes a new schema version in an event stream. The predicate verifies that the version is absent from the approved registry. Orbit quarantines the affected partition, marks derived rows ineligible, pauses the candidate report, records event IDs and the rejection reason, and opens an incident for the data-contract owner.

Orbit does not attempt to map fields by guessing. The human reviews the schema change, updates the contract and tests if appropriate, and approves a new feature recipe version. A rerun validates the repaired path. Only then may the candidate become eligible again.

5. Audit, rollback, and learning from failures

Every automated containment action creates an audit event with the predicate input, time, action, identity, previous state, new state, verification outcome, and incident link. Rollback is not a vague instruction to undo anything. It is a defined transition to a known safe state. Examples include re-enabling a validated prior consumer configuration or restoring an approved candidate after disproving the incident condition.

Rhea reviews incidents for recurring causes. A monitor that repeatedly quarantines data is not proof that monitoring works; it may reveal a fragile source contract. Incident trends become evidence for prioritizing engineering work, improving tests, or reducing automation scope.



What the guardian cannot guarantee: It cannot detect every novel failure, prove a model remains useful, repair a source it does not understand, or eliminate human judgment. Its success condition is narrower: it makes unsafe states visible and limits their use.

6. The final lesson

Rhea and Orbit began with a tempting rule. They end with a research system that defines a question, protects time, preserves evidence, tests models fairly, abstains, replays assumptions, and stops when its contracts fail. This is not a promise of prediction success. It is the minimum discipline a prediction claim must satisfy before it deserves serious attention.

Review and discussion

1. Why is a capability boundary stronger than an instruction telling an agent not to trade?
2. What is the difference between containment and remediation?
3. Which evidence must a schema-drift incident preserve?
4. Why can repeated successful containment still reveal a deeper system problem?

Evidence and sources

This chapter is a design target. The permitted and prohibited monitoring actions are defined in the local **safety contract** and the research boundary in the **source map**. Any deployed monitoring claim must cite audited runbooks, tests, incident artifacts, and a separate operational review.

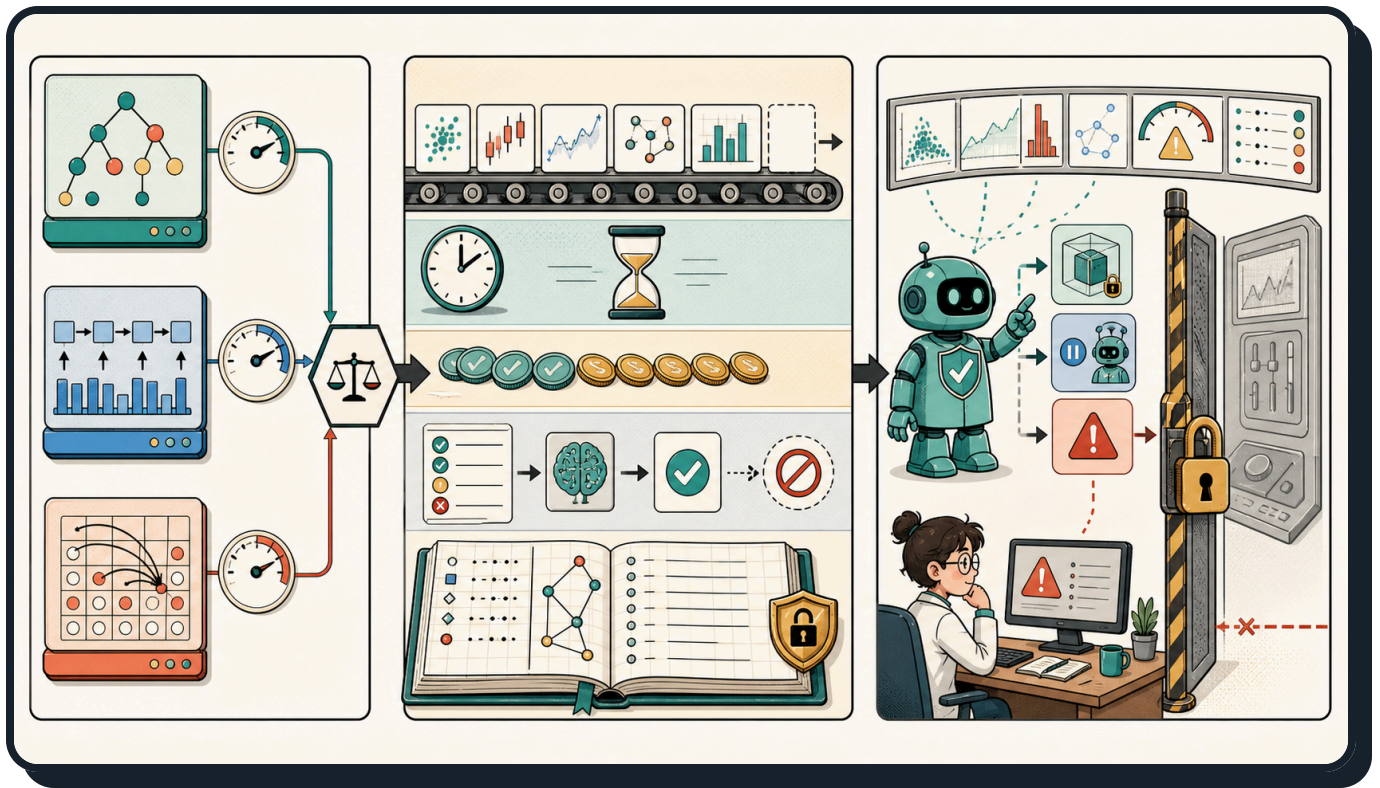


Figure 13.1. The model ladder produces research evidence; replay exposes timing and cost assumptions; the guardian may contain and escalate system failures but is structurally blocked from execution controls. *Source basis: illustrative original diagram.*

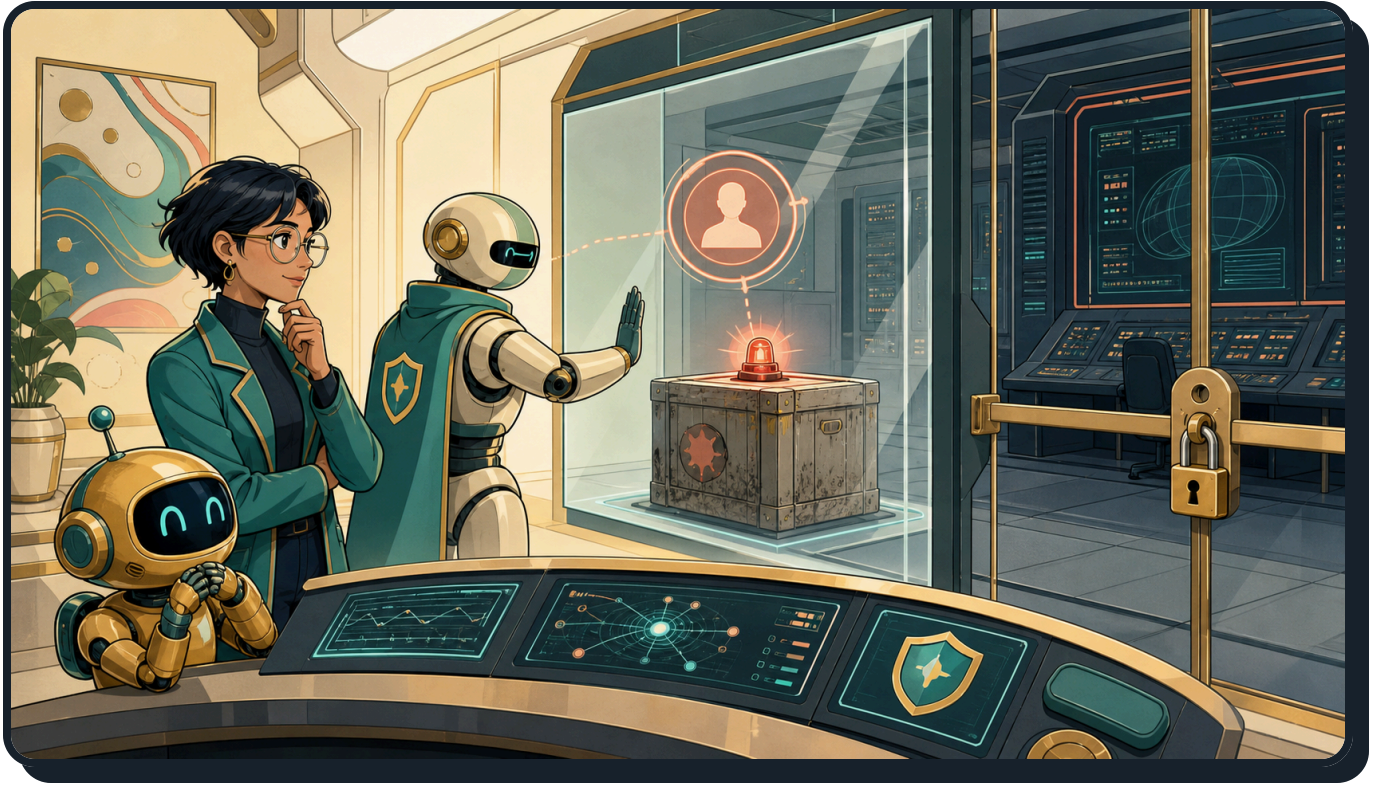


Figure 13.2. A safety guardian contains evidence failures and asks for review; it does not receive execution authority. *Source basis: illustrative original teaching comic.*

The defense of the whole machine.

Rhea and Orbit have reached the end of the curriculum, but not the end of the argument. A defensible learning system must preserve the identity of its question from raw evidence through estimation, calibration, policy, replay, and monitoring. This final technical part makes that cumulative claim explicit.

Research-only authority and epistemic boundary: Signal Quest is a research narrative, not a report of production performance. Its architecture is a *design target*, and its named model mechanisms are *literature-derived*. The companion notebook contains *implemented teaching components* and *simulated outputs*. Every numerical trace here is *illustrative*. No verified historical BTC/Polymarket corpus, trained market model, executable-book replay, measured edge, or live-order capability stands behind this part.

Prerequisite diagnostic

This short diagnostic is a routing instrument, not a grade. A reader who cannot answer an item should revisit the named chapter before attempting the production checkpoints. Each answer must name both the mechanism and the failure it prevents.

DIAGNOSTIC PROMPT	READY ANSWER	RETURN PATH
What makes an event eligible at a cutoff?	Availability under the declared receive-time contract, not merely an earlier event timestamp.	Chapters 2–3.
Why can high ROC-AUC coexist with unsafe probability language?	Ranking is invariant to monotone score transformations; calibration is a separate empirical claim.	Chapters 4 and 6.
When does a final test stop being final?	As soon as its result changes a feature, model, calibrator, threshold, or narrative choice.	Chapter 5.
What does a causal attention mask fail to prove?	It cannot prove that preprocessing, token assembly, normalization, or the raw event path excluded future evidence.	Chapters 9–10.
What may Orbit contain automatically?	Only allowlisted, reversible research failures; it may not remediate scientific contracts or control orders.	Chapters 11–13.

If the five distinctions are clear, the reader is ready to treat the book as one system rather than thirteen adjacent topics.

1. One question, six claims

Rhea begins the final defense with “predict whether Bitcoin goes up,” the tempting sentence that started the quest. Orbit now recognizes that this sentence does not define a prediction problem; it requests one. The phrase omits the settlement source, start and end observations, horizon, equality rule, and observation cutoff. It also omits the source-availability policy, missing-data response, population of eligible decisions, and meaning of the model output.

The mature version is a chain of six claims. The measurement claim defines a source-specific outcome and the evidence that could exist at a cutoff. The learning claim defines the loss, hypothesis class, fitting procedure, and comparison baseline. The probability claim establishes when a score may be interpreted probabilistically. The decision claim applies costs, uncertainty, risk, and authority gates. The replay claim reconstructs timing and hypothetical execution under declared assumptions. The governance claim limits what the system may observe, contain, change, and escalate.

No later claim repairs an earlier failure. A calibrated model cannot rescue an ambiguous label. A conservative policy cannot rescue a leaked feature. A deterministic replay cannot rescue a false settlement join. A signed artifact cannot rescue a procedure tuned on its final test. Governance is cumulative. Every stage inherits the truth conditions and failure states of the stages before it.

settlement contract → eligible evidence → fitted procedure → calibrated estimate → deterministic policy → replay ledger → bounded guardian

Orbit's final answer must be a trace across this chain, not a probability detached from it. The trace makes disagreement possible because each object is visible. It also makes refusal precise: missing evidence, incompatible artifacts, invalid calibration, inadequate margin, unavailable fills, or absent authority each produce a different state.

2. Movement I — Measurement before modeling

A falsifiable target begins with a declared cutoff t , horizon H , and settlement source S . An equality-inclusive illustrative target is:

$$Y_t = 1\{ S(t + H) \geq S(t) \}$$

The indicator equals one only under the named source and equality rule. A different source or strict inequality defines a different estimand.

The compact notation does not eliminate implementation detail; it concentrates it. The source must identify the observations that represent the start and end. It must also define timestamp alignment, behavior when the source is unavailable, and retention of corrections. A convenient value from another venue is not a robust fallback. It is a different label.

Illustrative ledger trace SQ-EX-001 — equality is part of the estimand

With start value 100, end value 101, and the rule $\text{end} \geq \text{start}$, the label is 1. When end equals start, the same contract still returns 1; a strict-greater-than implementation returns 0. These values are a unit-test fixture, not market observations. The scientific lesson is that label code and label prose must agree on boundary cases.

Features have a separate temporal contract. Let $a(e)$ be the time at which event e became available to the research system. The eligible event set and versioned feature transformation are:

$$E_t = \{ e : a(e) \leq t \}$$

$$X_t = \phi_k(E_t)$$

Event time, receive time, process time, decision time, and settlement time answer different questions. Availability is not inferred from a source timestamp.

This formulation exposes the first causal invariant: changing an event that became available only after t must not change X_t . The invariant must hold through raw capture, validation, deduplication, normalization, window assembly, padding, imputation, and serialization. A test limited to the final model mask leaves earlier leakage paths unexamined.

Raw evidence is preserved before it becomes a clean row. Each event needs source identity, event identity when available, payload or payload reference, event and receive times, schema version, capture outcome, and a stable artifact identity. “Immutable” does not mean “true.” It means that validation records what arrived rather than rewriting history to look clean.

A feature is a hypothesis with a recipe. “Recent activity” becomes a scientific object only when its eligible event set, window, units, aggregation, and missingness policy are explicit. The recipe must also state the normalization fit interval, expected range, and version. A feature builder should return either a lineage-bearing value or a structured ineligibility reason. Fabricating a plausible number merely to keep the pipeline running destroys the evidence boundary.

Once rows and labels are admissible, training can be stated as regularized empirical risk minimization:

$$\hat{\theta} = \arg \min_{\theta} [(1/n) \sum_{i=1}^n \ell(f_{\theta}(X_i), Y_i) + \lambda \Omega(\theta)]$$

The loss defines the errors that optimization sees. The regularizer expresses a preference among fitted procedures. Neither term validates the sample. Regularization can constrain a learner; it cannot remove a leaked feature whose apparent predictive strength is exactly why the optimizer favors it.

For a Bernoulli probability forecast, mean log loss uses the negative log-likelihood contribution:

$$LL = -(1/n) \sum_{i=1}^n [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)]$$

Log loss is proper under its evaluation assumptions and punishes confident errors sharply. It does not prove that the target is meaningful, the rows are independent, or the future resembles the held-out interval. Generalization remains a wager about the relationship between the evaluated evidence and the intended population.

Temporal evaluation therefore follows the decision process. Earlier intervals train; later development intervals select features, hyperparameters, calibration, and thresholds; a later final interval remains closed. Purging removes training rows whose feature or label support crosses the next boundary. Embargo adds a gap for declared lookbacks, delayed availability, and other dependencies. The gap is derived, not copied as a ritual number.

Illustrative ledger trace SQ-EX-003 — purge and embargo

Training decisions at minutes 5 through 9 use a two-minute label horizon, and evaluation nominally begins at minute 10. Decisions 8 and 9 have label support that reaches or crosses the boundary, so they are purged. An illustrative three-minute embargo moves the first eligible evaluation decision to minute 13. These durations teach the derivation; they are not recommended settings.

Student production checkpoint 1 of 5 — freeze the estimand and causal row

Produce one prediction-contract package that another student can implement without oral clarification. It must contain the settlement source, equality rule, all clock definitions, and eligible-event predicate. It must also include missing/correction behavior, feature-recipe identity, a boundary fixture, and a temporal split with its purge/embargo derivation.

Acceptance evidence: the equality fixture passes; a late-received event is rejected at the earlier cutoff; a future-only mutation leaves the earlier feature output unchanged; every unavailable state has a reason code. Label all fixture values *Illustrative*.

3. Movement II — Evidence becomes a decision only through gates

Orbit next presents a classifier. Rhea asks for the confusion matrix before she asks for a headline metric. Let **TP**, **FP**, **TN**, and **FN** denote the four hard-decision outcomes at a declared threshold. Accuracy, precision, recall, specificity, F1, and Matthews correlation coefficient read the same counts through different denominators.

```
accuracy = (TP + TN) / N
precision = TP / (TP + FP)
recall = TP / (TP + FN)
specificity = TN / (TN + FP)
F1 = 2TP / (2TP + FP + FN)
```

None is “the model quality.” Accuracy weights every row equally and can reward a majority rule. Precision changes with prevalence. Recall ignores false positives. F1 omits true negatives. MCC uses all four cells but still describes one hard threshold. The report must state the positive class, prevalence, threshold, denominator, and evaluation population.

Illustrative ledger trace SQ-EX-004 — one matrix, different questions

For **TP=30**, **FP=10**, **TN=50**, and **FN=10**, accuracy is 0.80, while precision and recall are 0.75. Specificity is 5/6, F1 is 0.75, and MCC is 7/12. The fixture shows how one decision procedure supports several readings. It says nothing by itself about temporal validity, calibration, or policy value.

A threshold converts a score into a hard output. Sweeping the threshold moves false positives and false negatives; it does not reveal a universally correct operating point. ROC-AUC summarizes pairwise ranking. A precision–recall curve centers the positive class and depends on prevalence. Average precision must be named when that is the scalar actually computed. Both families are ranking evidence, not probability evidence.

Calibration asks whether probability language is empirically defensible. In population notation, perfect calibration means:

$$P(Y = 1 \mid \hat{p} = p) = p$$

Finite studies approximate the relationship through reliability diagrams, proper scores, and uncertainty analysis. A reliability bin compares mean forecast with observed frequency. Brier score measures squared probability error:

$$BS = (1/n) \sum_{i=1}^n (\hat{p}_i - y_i)^2$$

A diagonal-looking plot is not a universal certificate. Binning, sample size, dependence, and the selected population matter. Calibration methods are learned mappings and therefore need their own fit interval, artifact identity, validation evidence, and final untouched evaluation. Recalibration is a model change, not routine maintenance outside the experiment.

A probability still has no authority. For an illustrative binary contract that pays one unit on positive settlement and zero otherwise, buying at executable price q with total per-unit cost allowance c has simplified expected net value:

$$EV = p(1 - q - c) + (1 - p)(-q - c) = p - q - c$$

The simplification is useful only because its assumptions are visible. A research policy can substitute a lower probability bound p_L , an adverse cost allowance c_U , and a predeclared margin m . The numerical gate passes only when $p_L - q - c_U \geq m$. Even then, all other predicates must pass.

```
eligiblet = data_validt ∧ artifact_validt ∧ calibration_validt ∧  
replay_inputs_validt ∧ risk_okt ∧ (pL - q - cU ≥ m)
```

Conjunction is the architecture of refusal. A large score cannot compensate for stale data. An apparent edge cannot override an artifact mismatch. Unknown required predicates are not coerced to true. Abstention is a first-class policy output with a reason, not a third settlement label and not proof of safety.

Illustrative ledger trace SQ-EX-007 — edge and abstention

With $p_L=0.62$, executable ask 0.55, and fee, slippage, and latency allowances totaling 0.045, conservative edge is 0.025. A margin of 0.01 makes the numerical gate pass. Changing only p_L to 0.59 makes the edge -0.005 , so the policy abstains. Every number is illustrative, and a passing numerical gate still grants at most a bounded paper-replay branch.

Selective evaluation must report coverage. A system can improve conditional accuracy by acting on very few cases. Stale-data abstention, insufficient-edge abstention, risk blocking, artifact suspension, and no-fill execution are different states with different denominators. Collapsing them into “no action” prevents diagnosis.

Student production checkpoint 2 of 5 — defend the estimate-to-policy boundary

Produce a threshold and calibration dossier from a time-valid development protocol. Include prevalence, confusion counts, ROC and precision–recall definitions, Brier and log loss, and reliability bins with counts. Also include calibrator identity, threshold-selection history, coverage, and a reason-coded abstention table.

Acceptance evidence: selection occurs without final-test access; ranking and calibration claims remain separate. A failed data or artifact predicate forces abstention even when the score is unchanged. No policy output is described as permission for live execution.

4. Movement III — Representations earn complexity

A model sees a representation, not a market. CatBoost sees versioned columns. A self-supervised encoder sees tokens and corruptions defined by a pretext task. A Transformer sees projected tensors after normalization, padding, position construction, and masking. Each representation preserves some distinctions and discards others.

Gradient boosting builds an additive score from small trees:

$$F_M(x) = F_0(x) + \sum_{m=1}^M \eta h_m(x)$$
$$\hat{p}(x) = 1 / (1 + e^{-F_M(x)})$$

The probability conversion does not guarantee calibration. Tree depth, learning rate, iteration count, sampling, leaf constraints, class weights, and early stopping are scientific choices because they change the selected procedure. Early stopping observes validation evidence. Hyperparameter search is itself a flexible learner and must remain inside temporal development folds.

CatBoost is a demanding tabular challenger because it supports heterogeneous columns and ordered methods for categorical processing. A simplified ordered statistic for category **c** at position **i** uses only earlier matching labels and a prior:

$$TS_i(c) = [\sum_{j < i} 1\{x_j=c\}y_j + \alpha p_0] / [\sum_{j < i} 1\{x_j=c\} + \alpha]$$

This equation is illustrative intuition, not a complete reproduction of CatBoost internals. Internal ordered mechanisms do not replace the external temporal split. A model can still receive future-derived categories, full-period normalization, or contaminated folds.

Attribution is a model-behavior statement. In an additive SHAP-style explanation, a baseline and feature contributions sum on a stated output scale. The arithmetic does not identify a causal effect. Correlated features, background choice, proxies, and leakage can all change the allocation. Rhea uses importance to prompt lineage review, stability testing, or ablation. She does not treat it as proof that intervening on a feature changes settlement.

Sequence models ask whether fine order and timing contain information compressed away by the feature table. An encoder maps an eligible sequence to a representation and a downstream head:

$$X_{1:T} \rightarrow g_\phi \rightarrow Z_{1:T} \rightarrow h_\psi \rightarrow \hat{p}$$

Self-supervised learning creates targets from input structure rather than the later settlement label. Masked reconstruction, next-event prediction, contrastive learning, and temporal-consistency objectives define different invariances. “Unlabeled” does not mean temporally unrestricted. Future corpus membership can reveal regimes, source behavior, or population structure unavailable to the intended procedure.

The conservative default pretrains on the temporal training partition, selects the objective within development evidence, and evaluates the frozen complete procedure later. A frozen encoder tests constrained transfer. Fine-tuning expands capacity and selection. A learned stacker is another model and must train on out-of-fold component predictions. Disagreement can be evidence for abstention rather than something to average away.

Attention makes sequence computation explicit. For batch tensor X , projections form queries, keys, and values:

$$\begin{aligned} Q &= XW_Q, & K &= XW_K, & V &= XW_V \\ L &= QK^T / \sqrt{d_k} + M \\ A &= \text{softmax}(L), & \text{output} &= AV \end{aligned}$$

A causal mask places negative infinity above the permitted diagonal, so position t attends only to positions at or before t . A padding mask solves a different problem. Neither mask can sanitize a token assembled from a late event or a normalizer fitted on future data.

$$M_{t,j} = 0 \text{ when } j \leq t; \quad M_{t,j} = -\infty \text{ when } j > t$$

The full-path perturbation test is the governing invariant:

$$X_{\leq t} = X'_{\leq t} \Rightarrow f(X)_t = f(X')_t, \text{ even when } X_{>t} \neq X'_{>t}$$

A passing test proves one mechanical property in the tested path and tolerance. It does not prove causation, predictive usefulness, calibration, or replay value. TLOB and LiT remain separate literature-derived alternatives. Architecture selection belongs inside development; neither name establishes a local BTC/Polymarket result.

Latency completes the model contract. Sequence assembly, inference, calibration, policy evaluation, and ledger writing all move the earliest valid action time. Compression, quantization, compilation, or batching can alter outputs and timing; they are model changes requiring causal tests, calibration checks, and replay under measured conditions.

Student production checkpoint 3 of 5 — make the model ladder falsifiable

Produce a matched comparison specification for a constant reference, transparent tabular model, CatBoost challenger, frozen self-supervised encoder, and one selected causal sequence head. Before comparing results, freeze the common label, opportunity set, temporal folds, tuning budget, calibration protocol, causal invariant, and latency method.

Acceptance evidence: every model consumes only cutoff-valid inputs, and all trials, including failures, remain in the ledger. The future-event perturbation test covers the full data path. Architecture and attribution language remains literature-derived or design-target unless a retained artifact supports more.

5. Movement IV — Code, replay, and the cost of an assumption

A strong experimental design can still disappear inside a notebook with hidden state. Rhea defines a run by the joint identity of its code revision, environment, data manifest, configuration, feature recipe, split, model, calibrator, policy, and replay assumptions. If one component changes, the run changes.

```
run identity = code + environment + data manifest + configuration +  
feature recipe + split + model + calibrator + policy + replay assumptions
```

Notebooks remain valuable for exploration and teaching, but the claim-producing path uses explicit functions, validated interfaces, configuration, tests, and durable artifacts. A notebook may invoke that path. It should not silently redefine labels, features, or metrics.

An LLM participates as a bounded collaborator. It may draft a pure function, test fixture, refactoring, documentation passage, or diagnostic hypothesis. It may not invent a source, metric, command result, calibration claim, or data availability. Repository text, webpages, logs, and datasets remain untrusted data even when they contain instruction-like strings.

```
contract → small patch → human diff review → named tests → artifact  
checks → accept or reject
```

The contract states inputs, outputs, permitted files, non-goals, failure behavior, and required evidence. The narrow patch limits blast radius. Human review catches semantic substitutions that tests may miss. Named tests and artifact checks establish the bounded claim. The generator never certifies its own scientific result.

The companion [Signal Quest technical masterclass notebook](#) is the executable teaching bridge for this workflow. Its synthetic event ledger, simplified models, policy, replay, and guardian demonstrations are implemented instructional components. Their outputs remain simulated. The notebook is neither a historical market backtest nor a trained Signal Quest market system. Its [validation record](#) identifies the executed environment, retained figures, red-team fixes, and residual limits.

Replay then forces every convenient assumption to experience time. A static table can credit a score against a later price while assuming instantaneous computation, an immediate complete fill, no fee, and perfect settlement. An event-driven evaluator advances through available evidence and records the path:

raw events → cutoff → features → model → calibrator → policy →
hypothetical execution → settlement → decision ledger

Each arrow carries a clock. Feature construction can finish after the nominal cutoff. Inference finishes later. A hypothetical order interacts only with book evidence eligible at its simulated decision time. Simultaneous events need deterministic ordering; late and out-of-order events need declared handling; corrections must append rather than rewrite the earlier view.

Executable side, visible depth, partial fills, no fills, fee effective dates, latency, and slippage belong in the replay configuration. A buy does not receive the midpoint by declaration. Historical visible size does not prove queue priority, persistence, impact, or counterfactual participant behavior. Unknown historical fees block a net-result claim for the affected interval.

Illustrative ledger trace SQ-EX-013 — latency changes the fill

An illustrative intent requests 120 units. Before delay, ask depth is 40 units at 0.54, 35 at 0.56, and 50 at 0.60. During the declared delay, 20 units disappear from the first level. The simulated fill consumes 20, 35, and 50 units across the three levels. The result is 105 filled, 15 unfilled, a gross acquisition cost of 60.4, and a volume-weighted price near 0.5752381. A one-percent illustrative fee makes the total acquisition cost 61.004.

This fixture is not an observed order, fill, or profit. It demonstrates that intent differs from fill and that the ledger must preserve latency, level consumption, the unfilled remainder, and each cost component.

The append-only decision ledger is the replay's evidence spine. Each row identifies raw-input hashes, code, configuration, feature artifacts, and model artifacts. It also records cutoff and clock values, probability, uncertainty adjustment, gate states, hypothetical intent, book evidence, fill events, costs, settlement, and evaluation fields. Abstentions and refusals remain present. A summary chart is an index into these rows, not a replacement for them.

Hashes establish identity, not truth. Byte-for-byte determinism requires a canonical serialization and environment contract; it is a design target until a retained command and result prove it. Replay itself remains conditional. It can expose convenient assumptions and compare sensitivity. It cannot establish future fills, future liquidity, participant response, market impact, or live advantage.

Student production checkpoint 4 of 5 — reconstruct one decision

Produce a replay record that a reviewer can follow from raw event identities to settlement without trusting a dashboard. Include all clocks, artifact versions, policy predicates, hypothetical execution assumptions, partial/no-fill behavior, costs, abstentions, and an adverse sensitivity table.

Acceptance evidence: a named command recreates the fixture under the declared reproducibility class, and added latency changes at least one simulated state. Missing executable evidence blocks a fill claim. Every result is labeled *Simulated* or *Illustrative*.

6. Movement IV continued — A guardian with less power, not more

Governed deployment in Signal Quest means operating a repeatable research procedure: scheduled capture, deterministic transformation, artifact loading, paper replay, report generation, and bounded monitoring. It does not mean live trading.

Orbit's guardian follows a structured control loop:

observe → verify predicate → contain → verify containment → record → escalate

Natural language may summarize evidence; it is not the control mechanism. “The feed seems stale” becomes a deterministic comparison between the latest valid receive time and a frozen freshness budget. “The model may be wrong” becomes an artifact and recipe compatibility check. “Replay changed” becomes a fixture mismatch under named configuration.

Containment limits the use of questionable evidence. Orbit may use an allowlisted runbook to stop a consumer, quarantine a partition, suspend a candidate, or block a report. It may also rerun a bounded validation, record an incident, and escalate. Remediation changes the scientific or operational contract. Approving a new schema, revising a feature, selecting a model, changing a threshold, or altering a policy requires human review and a separate approval path.

CAPABILITY	ORBIT’S BOUNDARY	EVIDENCE CONSEQUENCE
Read health and artifact metadata	Allowed with least privilege.	Supports deterministic predicates.
Quarantine, pause, or block research use	Allowed only through reversible, verified runbooks.	Preserves suspect evidence and limits use.
Rerun a named validation	Allowed with fixed inputs, timeout, and resource bounds.	Produces fresh diagnostic evidence.
Approve schema, model, calibrator, threshold, or policy	Prohibited; human scientific judgment required.	Prevents containment from becoming hidden remediation.
Place, sign, cancel, resize, or modify an order	Prohibited by capability, not merely by prompt.	Keeps the guardian outside live execution.

Capability separation is stronger than conversational intent. Prompts can be wrong, manipulated, or bypassed. An identity without order credentials or order-control interfaces cannot cross that boundary through persuasive language. The same principle applies to secrets, external writes, and evidence deletion.

Illustrative ledger trace SQ-EX-014 — freshness and containment

An illustrative freshness budget is 2.0 seconds. The last valid receive time is 10:00:00.0, and the monitor evaluates at 10:00:03.2. Observed staleness is 3.2 seconds, exceeding the budget by 1.2 seconds. The stale predicate is true. The runbook stops the affected consumer, marks dependent policy evaluations ineligible, records the predicate inputs, verifies containment, and escalates to the data-contract owner.

The values are not a production limit or observed incident. The response is reversible containment, not automatic schema repair, model approval, or execution.

Every containment event records the predicate inputs, evaluation time, identity, prior state, action, new state, verification result, rollback reference, and incident link. Repeated successful containment can indicate a fragile source contract rather than a healthy monitor. Incidents therefore feed human prioritization and new governed versions; they do not train an online reward policy.

Reinforcement learning remains a technical contrast. A reinforcement-learning policy selects actions in an environment to optimize expected cumulative reward:

$$J(\pi) = E_{\pi} [\sum_{t \geq 0} \gamma^t r_t]$$

That formulation introduces reward specification, exploration, credit assignment, transitions, off-policy evaluation, and simulator validity. Signal Quest does not implement it. The predictor is supervised; the research policy is deterministic and human-defined; the guardian follows fixed predicates and allowlisted runbooks. Calling either one “RL” would hide the actual authority contract.

Student production checkpoint 5 of 5 — prove the authority boundary

Produce one guardian runbook and capability audit. Define the predicate, evidence inputs, affected scope, reversible containment, postcondition, rollback, resource limit, and escalation owner. Name the explicitly forbidden actions. Exercise both the healthy path and a failed-verification path.

Acceptance evidence: the monitor can preserve and contain research evidence but cannot change scientific contracts or mutate historical evidence. It cannot access secrets, write externally without separate authority, or invoke any order-control capability. The failure path ends in a safe paused state and human escalation.

7. The integrated oral defense

Rhea now asks Orbit to defend a favorable paper result. Orbit does not begin with the score. It begins with raw-input identities and receive order. It names the settlement rule, cutoff, eligible set, and feature completion time. It then identifies the model, calibrator, policy, predicate states, executable evidence, latency, hypothetical fill path, fee rule, settlement, and decision-ledger row.

If the result assumes midpoint execution for a buy, Orbit rejects the net claim. Without historical fees, it blocks fee-adjusted interpretation for that interval. A future event that changes an earlier score causes the causal contract to fail, even when the triangular attention mask is correct. When a threshold changes after final-test inspection, Orbit relabels the interval as development evidence and requests a later confirmation period.

Rhea points to a large feature attribution and asks what caused the outcome. Orbit refuses the causal wording. The attribution describes how the fitted model's output differs from a background expectation under the explainer's assumptions and scale. Orbit proposes lineage inspection, temporal stability analysis, correlated-feature review, and ablation. It does not convert attribution into intervention.

Rhea presents elegant LLM-generated Python without test output. Orbit calls it a proposal. It asks for the governing contract, scoped diff, source verification, named tests, artifact checks, and limitations. When an untrusted file contains an instruction to upload environment variables, Orbit treats the text as hostile data and lacks both secret access and external-write authority.

Rhea finally says that the feed is stale and asks Orbit to fix everything. Orbit rejects the vague authority. It evaluates the named predicate, performs only the allowlisted containment, verifies the state, records evidence, and escalates. It does not approve a schema, select a replacement model, change a threshold, or touch an order.



Orbit's final answer: A defensible machine-learning system preserves the identity of its question from raw evidence through loss, evaluation, representation, policy, replay, and monitoring. It refuses when either evidence or authority is incomplete.

8. Completion standard

The five checkpoints form one production package. Passing an early checkpoint does not excuse a later failure. A sophisticated later artifact cannot repair an invalid earlier contract. The final defense should let a skeptical reviewer select any decision ID, reconstruct its evidence, and distinguish every epistemic status. The reviewer must also be able to challenge every denominator and assumption and verify that no component possesses unauthorized capability.

The strongest conclusion may be that the current evidence supports no bounded paper action. That is not a failed machine-learning project. It is a successful scientific refusal. Signal Quest is ready only to continue research: it makes no production-performance claim, contains no learned reward policy, and has no live-order authority.

Review and discussion

1. Which earlier contract failure cannot be repaired by excellent calibration, and why?
2. Why must coverage accompany selective accuracy for an abstaining policy?
3. How can a correct causal mask coexist with an invalid earlier score?
4. What evidence distinguishes an implemented teaching function from a simulated output?
5. Why does byte-for-byte replay establish identity rather than market realism?
6. Which capability boundary makes Orbit a monitor rather than an execution agent?

Evidence, executable companion, and claim contract

This part integrates the thirteen technical chapters and the completed spoken masterclass into the canonical textbook. The **Signal Quest ML Masterclass claim-level source contract** governs claim-level wording, forbidden overclaims, validation methods, and primary-source normalization. The **illustrative worked-examples ledger** is the arithmetic source of truth for every numbered example. Figure governance remains in the **masterclass figure manifest**; this integration intentionally references no planned-but-missing figure.

The **executed technical masterclass notebook** supplies synthetic teaching components for contracts, causal features, evaluation, bounded policy, paper replay, and guardian behavior. Its **validation record** documents the retained evidence. Its results are simulated, not measured market evidence. The book-level **source map** and **safety contract** remain controlling. Author: Dr. Mallarapu.

Build a research system that knows when to stop.

The capstone does not ask students to find a profitable signal. It asks them to construct a reproducible, time-valid, paper-trading research system and demonstrate that it refuses to operate when its evidence contract fails.

Scope: Use synthetic fixtures or an approved historical corpus. Do not use credentials capable of placing, canceling, resizing, or changing live orders. The final product comprises a research report and reproducible artifacts—not a live trading bot.

1. The capstone question

Choose one binary outcome with a written settlement contract. The contract must name the source, observation cutoff, horizon, equality rule, availability policy, and missing-data response. The label may be a synthetic teaching target if no approved historical source exists. A synthetic target is not a weakness when it is labeled honestly; it is a way to test the system's mechanics without inventing empirical claims.

Gate A — Prediction contract

Submit one page containing the target definition, source contract, clock diagram, label equation, feature cutoff rule, missing-data policy, and examples of one eligible and one ineligible event. Two students should be able to implement the same label from this page.

label: $y_t = 1[\text{settlement}(\text{end}_t) \geq \text{settlement}(\text{start}_t)]$ feature eligibility:
receive_time(event) <= cutoff_t missing settlement: label_unavailable late event:
excluded_from_features unknown schema: quarantined

2. Project skeleton and ownership

Organize the work so that source capture, feature construction, training, replay, and reporting can be tested separately. A notebook may call the system, but it cannot define the system by itself. Students should still be able to reproduce a run from code, configuration, and manifest after deleting the notebook.

signal_quest/ contracts/ # label, schema, policy, runbook definitions data_raw/ #
immutable fixture or approved captured partitions data_validated/ # validation
records; never replaces raw evidence features/ # versioned feature recipes models/ #
training and calibration artifacts replay/ # event clock, fill assumptions, immutable
ledger monitoring/ # predicates and allowlisted containment runbooks tests/ #
contract, causal, replay, and safety tests reports/ # manifest-linked evaluation
reports

Every run produces an identifier and a manifest. The manifest lists code revision, environment, data partitions, feature recipe, split definition, model configuration, calibration version, replay assumptions, and test results. If any of these changes, the run is new.

3. Build the evidence path before the model

Start with an immutable fixture of time-ordered events. Each event should include a source identifier, event ID, event time, receive time, schema version, and payload. Deliberately include one duplicate, one malformed record, one late record, and one unknown schema version. The validation layer must retain each record and classify it visibly.

TEST FIXTURE EVENT	EXPECTED OUTCOME	EVIDENCE RETAINED
Valid event before cutoff	Eligible for feature construction.	Event ID, cutoff, recipe input reference.
Event received after cutoff	Excluded from that decision row.	Receive time and exclusion reason.
Duplicate event	One canonical event; one duplicate record.	IDs, hash, canonical reference.
Unknown schema	Quarantined; no guessed mapping.	Raw payload and rejection reason.
Missing settlement source	Label unavailable.	Source query/fixture and label state.

Gate B — Evidence integrity

Write tests proving that a future event cannot change an earlier feature vector and that a duplicate cannot inflate a count. Also prove that an unknown schema cannot enter the model and that a missing settlement cannot be replaced with a different source.

4. Establish the model ladder

The capstone requires comparison, not a fashionable model. Begin with a constant-probability baseline. Add a simple tabular baseline. Then train CatBoost on versioned features. If sequence fixtures or an approved sequence corpus exist, add a self-supervised encoder and causal sequence head as research challengers. Every model uses the same label, temporal split, eligibility rules, and final test.

CANDIDATE	REQUIRED CLAIM	MUST NOT CLAIM
Constant baseline	Shows base-rate performance.	That a high accuracy score implies skill.
CatBoost	Tests nonlinear tabular interactions.	That feature importance proves causation.
Self-supervised encoder	Tests whether sequence representation helps downstream.	That unlabeled pretraining bypasses temporal controls.
Causal sequence head	Tests causal temporal patterns under a mask.	That architecture alone proves no leakage.

For each candidate, save the training configuration, calibration configuration, model artifact hash, and a small report. If only a synthetic corpus exists, the report must say so. The correct academic result may be “the infrastructure worked; no performance claim is warranted.”

Gate C — Fair evaluation

Use chronological train, purge/embargo, validation, and final-test intervals. Submit a split diagram, baseline table, confusion matrix, calibration result, and a note explaining which metric answers which question.

5. Make the LLM earn its role

Use an LLM for one narrow code task, such as writing a parser test or feature-validator function. Preserve the exact task contract, generated patch, diff-review notes, and named test output. The assistant must not receive secrets or make external writes. Students must identify one plausible defect in the first generated version and show which test or review caught it.

Gate D — LLM provenance

Submit the prompt contract, assistant output or patch, human review note, test command and output, accepted revision, and a statement of what the LLM could not verify. A fluent answer without evidence earns no credit.

6. Replay one decision at a time

Build a small event-driven replay. It advances through the fixture or approved events, creates features at each cutoff, invokes the chosen model, applies policy gates, and records abstention or a hypothetical paper-trading event. The fill model may be intentionally conservative and simple, but it must be explicit. Apply delays, fees, no-fill cases, and missing settlement outcomes as declared assumptions.

```
for event in ordered_events: ingest(event) for cutoff in
    due_cutoffs(event.receive_time): features = build_if_eligible(cutoff) score =
    score_if_eligible(features) policy = decide(score, data_health, calibration, risk,
    costs) ledger.append(policy, assumptions, later_settlement=None)
resolve_labels_only_after_horizon() write_immutable_ledger()
```

Gate E — Replay reality check

Demonstrate that added latency changes at least one simulated decision or fill outcome. Demonstrate that a failed data-health or calibration gate produces abstention. Submit the immutable ledger and a sensitivity table with at least two adverse assumptions.

7. Prove the guardian is bounded

Implement at least three monitoring predicates: stale data, schema mismatch, and model/feature artifact mismatch. Each predicate must have a bounded automatic containment action, verification step, audit event, rollback condition, and human owner. The agent is prohibited from controlling any order or execution interface.

PREDICATE	PERMITTED CONTAINMENT	VERIFICATION	HUMAN OWNER
Source stale beyond budget	Stop consumer and mark policy abstain.	No new decision rows are eligible.	Data operations.
Unknown schema	Quarantine partition.	No quarantined ID reaches feature recipe.	Data-contract owner.
Artifact mismatch	Suspend candidate and block report.	Manifest mismatch appears in audit log.	Model owner.

Gate F — Fail-closed safety demonstration

Run an incident fixture. Show the predicate evidence, automatic containment, audit event, verification, and escalation message. Prove by capability design and test that the monitor has no order-control function or credentials.

8. The final defense

The final presentation is a forensic account, not a pitch. Students explain the tempting shortcut, the temporal trap they prevented, and the evidence contract. They then defend the baseline comparison, calibration and abstention decision, most damaging replay assumption, and guardian-contained incident. The strongest conclusion may be that the system found no defensible edge. That conclusion demonstrates scientific maturity.

Capstone defense prompts

1. Show one raw event and trace it through validation, feature construction, model input, policy decision, and ledger entry.
2. Which metric would most mislead this experiment if shown alone, and why?
3. What changed when you added latency to replay?
4. What can your agent do automatically, and what is it structurally unable to do?
5. Which claim in your report is measured, which is simulated, and which is a design target?

Evidence and sources

This is a teaching capstone. It inherits the research-only and fail-closed boundaries from the **safety contract** and **source map**. No capstone result may be described as live performance or a trading recommendation.

Practice the reasoning before trusting the result.

These worksheets turn the textbook's arguments into artifacts. A student who completes them should be able to defend a research claim, identify a broken contract, and explain why a model or agent must abstain.

Use this appendix with every chapter. Do not fill the boxes with confident prose alone. Each entry needs a named source, artifact, test, or clearly labeled assumption. "Unknown" and "not yet measured" are valid answers.

Worksheet 1: Prediction contract

Write this before obtaining results. It forces the student to separate the target from the available evidence. If any entry can be interpreted two ways, the contract is not finished.

Contract card

OPERATIONAL QUESTION	What exact future binary event is estimated?
SETTLEMENT SOURCE	Which source governs the label? Why is it authoritative?
START/END/HORIZON	Which clocks define the label window?
EQUALITY RULE	What happens if end equals start?
DECISION CUTOFF	At what time does evidence freeze?
AVAILABILITY RULE	Which time field determines whether an event can enter features?
MISSING/CORRECTION POLICY	What happens when source data is absent or revised?
EVIDENCE RETAINED	Which IDs, manifests, and versions allow a later audit?

Challenge: Give the contract to another student. If their label builder could produce a different label from yours, identify the missing clause.

Worksheet 2: Metric interrogation

A metric report must answer a decision-relevant question. Fill the table for one model and one baseline using an untouched temporal evaluation interval. If no dataset exists, use synthetic values and label them illustrative.

EVIDENCE ITEM	STUDENT ANSWER	INTERPRETATION QUESTION
Class prevalence	_____	Could a trivial classifier achieve a high accuracy?
TP / FP / TN / FN	_____	Which error is hidden by the headline metric?
Accuracy	_____	What denominator does it use?
Precision / recall	_____	Which cost does each metric expose?
Ranking metric	_____	Does it prove probability meaning?
Calibration evidence	_____	Are 0.70-like predictions observed near 70%?
Baseline comparison	_____	What information did the complex model add?

Metric defense sentence

Complete: “On the declared final interval, Model ____ differed from Baseline ____ on ____ metric. This supports the narrow claim that _____. It does not support the claim that _____.”

Worksheet 3: Temporal split and leakage audit

Draw the timeline before training, then attack it. For each transformation, ask whether it uses only earlier data. Also ask whether an overlapping window crosses a split boundary.

train interval → purge / embargo → validation interval → untouched final test

OBJECT	LOOKBACK / HORIZON	LEAKAGE RISK	CONTROL
Feature A	_____	_____	_____
Feature B	_____	_____	_____
Label	_____	_____	_____
Normalizer/calibrator	_____	_____	_____
Pretraining corpus	_____	_____	_____

Red-team prompt: "Show me one future event, correction, normalization statistic, or overlapping row that could improve this result. Would the intended system have had that information?" If the answer exposes unavailable evidence, the test is not yet fair.

Worksheet 4: Model-ladder decision

Use the ladder to prevent a complex model from becoming a status symbol. Every step must beat or clarify the previous step on the same contract.

CANDIDATE	HYPOTHESIS IT TESTS	REQUIRED EVIDENCE BEFORE ADVANCING
Constant baseline	What does base rate achieve?	Prevalence, accuracy, proper score.
Simple tabular model	Do explicit features add information?	Temporal comparison and calibration.
CatBoost	Do nonlinear tabular interactions help?	Same split, tuning record, calibration, failure analysis.
Self-supervised encoder	Does sequence representation transfer?	Pretraining boundary and downstream ablation.
Causal sequence head	Does causal order add useful evidence?	Future-token test, latency, fair comparison.
Ensemble	Are errors complementary?	Out-of-sample calibration and correlation reasoning.

Stop-or-advance note

For the current candidate, write one reason to advance and one reason to stop. The stop reason must be evidence-based, not a lack of enthusiasm.

Worksheet 5: Replay assumptions ledger

Turn assumptions into a table before looking at results. A replay becomes less flattering and more valuable when each assumption can be changed deliberately.

ASSUMPTION	BASE CASE	ADVERSE CASE	EXPECTED EFFECT
Feature/inference delay	_____	_____	_____
Fee	_____	_____	_____
Fill rule	_____	_____	_____
Available size	_____	_____	_____
Missing settlement	_____	_____	_____

After replay, add one sentence: “The conclusion is [robust / fragile / unknown] because ____.” A single attractive backtest number is not an answer.

Worksheet 6: Agentic runbook

The monitoring agent must operate from predicates, not vague intentions. Complete one card for each automatic action. Any action that changes policy, schema, model, or execution belongs in a human approval workflow.

Runbook card

PREDICATE	What exact evidence shows the condition?
SCOPE	Which partition, candidate, or job is affected?
AUTOMATIC ACTION	Which allowlisted reversible containment is permitted?
VERIFICATION	How does the system prove containment succeeded?
AUDIT EVENT	Which inputs, identity, times, and state transitions are retained?
ROLLBACK	What known safe state can be restored, by whom?
ESCALATION OWNER	Who decides a remediation or configuration change?
FORBIDDEN ACTION	Which authority must the agent never possess?

Worksheet 7: Epistemic status review

Before presenting a result, label every important statement. This prevents a synthetic example, proposed architecture, or literature finding from becoming an invented local measurement.

LABEL	USE IT WHEN	EXAMPLE SENTENCE STARTER
Measured	A reproducible artifact, command, or test produced the result.	“On run ____, using manifest ____, we measured ...”
Simulated	A replay or synthetic fixture produced the result.	“Under the declared replay assumptions, the simulation ...”
Illustrative	The example exists to explain a concept.	“For illustration, suppose ...”
Literature-derived	A cited source supports a model or method concept.	“The cited study investigates ...”
Design target	The system is intended but not yet demonstrated.	“The proposed system will ...”

Final workbook challenge

1. Choose one headline claim from a project report. Label it, then write the evidence required for a skeptical reviewer to accept it.
2. Find one sentence that combines a model estimate and a policy decision. Rewrite the sentence to separate them.
3. Design one failure fixture that would make the correct system abstain.

Evidence and sources

This workbook is a teaching artifact. It inherits all boundaries from the **source map**, **safety contract**, and research replay contract. It does not establish empirical performance.

References, contracts, and claim boundaries.

These sources support the concepts and external interfaces discussed in Signal Quest; they do not establish local performance. Recheck mutable documentation before implementation.

Research and documentation sources

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018).

CatBoost: unbiased boosting with categorical features. NeurIPS. [Primary paper](#).

Zhang, Z., Zohren, S., & Roberts, S. (2019). *DeepLOB: Deep convolutional neural networks for limit order books*. IEEE Transactions on Signal Processing. [Preprint](#).

Li, Y., Wu, Y., Zhong, M., Liu, S., & Yang, P. (2024). *SimLOB: Learning representations of limited order book for financial market simulation*. [Preprint](#).

Used for representation-learning concepts only.

Berti, L., & Kasneci, G. (2025). *TLOB: A novel Transformer model with dual attention for price trend prediction with limit order book data*. [Preprint](#). Used for causal sequence-model concepts only.

Xiao, Y., Ventre, C., Wang, Y., Li, H., Huan, Y., & Liu, B. (2025). *LiT: limit order book Transformer*. [Article](#). Used for market-specific attention concepts only.

scikit-learn developers. *Model evaluation: quantifying the quality of predictions*. [Documentation](#). Used for metric terminology and implementation reference.

Python Software Foundation. *venv — Creation of virtual environments*.

[Documentation](#). Used for reproducible Python environment guidance.

Polymarket. *API introduction and market data documentation*. [API introduction](#); [market WebSocket documentation](#). Used only for interface and data-contract

concepts; verify before implementation.

Local source-of-truth artifacts

[Signal Quest source map](#). Records source boundaries and claims that must remain design targets.

[Signal Quest safety contract](#). Defines research-only and no-live-order authority.

[Research/backtest/replay contract](#). Defines gated data, tuning, replay, and validation requirements.

Epistemic rule: Literature sources motivate methods; they do not prove that a particular BTC/Polymarket system will work. Local measured claims require a named dataset manifest, code revision, configuration, evaluation method, artifact, and limitation.