The Four Leakage Gates

Walk-forward backtests can fail in subtle ways. The most common failure mode is **temporal leakage** — when the model accidentally sees data it shouldn't have seen at the point it was making a prediction.

The Signal — Live runs four hard-assert checks before every model fit. If any check fails, the run aborts immediately. These are not warnings, they are blocking gates.

You see the result of all four in every Monday email's governance block.

Gate 1: No Future in Features

The check: Verify the latest timestamp present in the training feature matrix is strictly BEFORE the prediction date.

The leak it catches: A common bug pattern where, due to off-by-one indexing or a forgotten `.shift(1)` lag, a feature for date D accidentally includes information available only after D. Even a single day of look-ahead inflates backtest Sharpe by 0.3-0.5 because the model gets to "see" tomorrow's returns when scoring today's picks.

Plain-language version: The model isn't allowed to use tomorrow's data to make today's prediction.

Gate 2: Feature-Target Correlation Below 0.5

The check: Compute the absolute correlation between every individual feature column and the binary forward-return target. Fail the gate if any single feature has $|r| \geq 0.5$.

The leak it catches: A feature accidentally containing target information directly — for instance, a "future_return_5w" column that someone left in the dataset by mistake. A feature legitimately predictive of returns might hit $|r| = 0.10$. A feature that IS the target (or trivially derives from it) will hit $|r| = 0.99$. The 0.5 threshold catches the obvious leak cases without flagging the actually-informative features.

Plain-language version: No single feature is allowed to be a giveaway answer.

Gate 3: No Target-Like Column Names

The check: Scan all feature column names for banned tokens: `forward`, `future`, `target`, `label`, `_fwd_`, `_y_`.

The leak it catches: The naming convention slippage where a feature engineering script generates `momentum_forward_5w` (meant to be informational) but accidentally includes it in the model's input frame. The model trains on what is, in effect, the answer, and produces fantastic backtest results that completely collapse in production.

Plain-language version: Even the NAMES of feature columns can't suggest future-looking data.

Gate 4: Time-Split Integrity with Embargo

The check: Verify that the prediction date is strictly AFTER (train end + 1 week embargo). The 1-week embargo prevents subtle contamination at the train/test boundary where a forward-shifted label from the final training week could overlap with the prediction window.

The leak it catches: Boundary leakage. In a walk-forward setup where train ends at week 100 and predict happens at week 101, the forward 1-week label for the LAST training observation (week 100) corresponds to the return from week 100 → week 101 — which IS the prediction window. Without an embargo, the model's training "knew" about the prediction-window return. Embargoing one week (dropping the final training week) eliminates this.

Plain-language version: There's a 1-week buffer between what the model trained on and what it's predicting, to prevent any sneaky overlap.

What you see each Monday

Every Premium picks email and every Free Letter includes a governance block with all four gates marked ✓ or ✗:

```
LEAKAGE GATES
✓ no_future_in_features
✓ feature_target_correlation
✓ no_target_columns_in_features
✓ time_split_integrity
```

If any of these show ✗ (which would be a critical bug), the email won't go out. You'd receive an Integrity Letter instead.

Why expose this to subscribers?

Because the gates are the difference between *"trust me, this is a good model"* and *"here are the four specific failure modes we check for every week, and here's the result."*

You don't have to take the methodology on faith. The governance block makes it auditable.