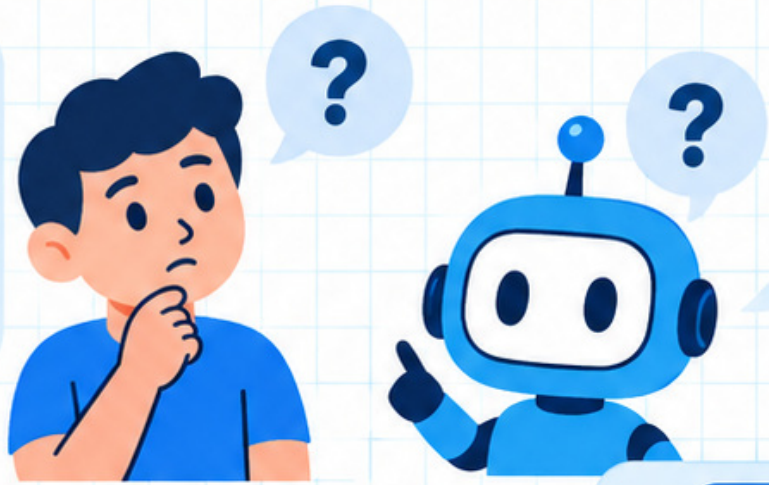
SAME FEATURE NAME. SAME MEANING?

Reproducible features. Same column name can hide different definitions.

“Orders in the last seven days” sounds clear.

But which orders, which clock, and which seven days?



Today: design reproducible features for an ML service that an application or agent can use.

E-COMMERCE APP

Orders in the last seven days

Feature value

?

Might mean paid customer orders, using the customer’s local time, for the last seven calendar days.

FEATURE DEFINITION

- **Which orders?**
- **Which clock?**
- **Which seven days?**

SUPPORT AGENT

Orders in the last seven days

Feature value

?

Might mean all orders (including canceled), using UTC time, for the last seven rolling 24 hours.

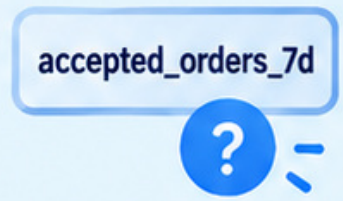
A FEATURE IS A DEFINED MODEL INPUT

A feature represents information supplied to a model.

Example: the number of accepted orders for a store during a defined seven-day window.

Reproducible means we can reconstruct the value from the recorded inputs and rules.

A **descriptive** column name helps, but it does not specify the calculation.

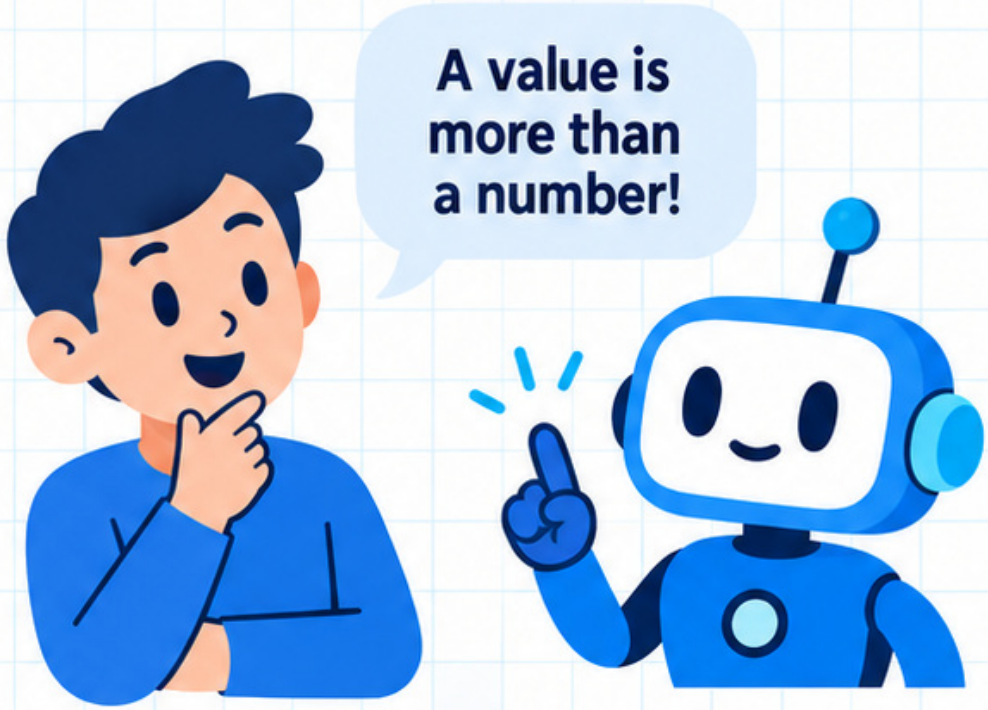


GIVE EACH VALUE AN IDENTITY

A feature value belongs to an entity and a time.

Entity: store S12.
Prediction time: 09:00 UTC on 8 June.
Feature: accepted orders in the preceding 168 hours.

“S12 has 140 orders” is incomplete without the window, availability rule, and feature-definition version.



Entity
Store S12.

Prediction time
09:00 UTC on 8 June.

Feature
Accepted orders in the preceding 168 hours.

Feature-definition version
Which definition?

ILLUSTRATIVE EXAMPLE

Entity (store)	S12
Prediction timestamp	09:00 UTC on 8 June
Feature	Accepted orders in the preceding 168 hours.
Feature-definition version	
Value	**140** *(illustrative)*

WRITE A FEATURE CONTRACT

For orders_7d, define:

- **Entity:** store_id
- **Count:** distinct accepted order IDs
- **Window:** 168 hours before prediction
- **Timezone:** UTC
- **Availability:** records usable by prediction time
- **Missing-source behavior:** unavailable, not zero
- **Version:** orders_7d_v1

SEPARATE EVENT TIME FROM AVAILABILITY TIME

Event time: when an order happened.

Availability time: when the feature system could use its record.

An order happens at 08:55 but arrives at 09:05.

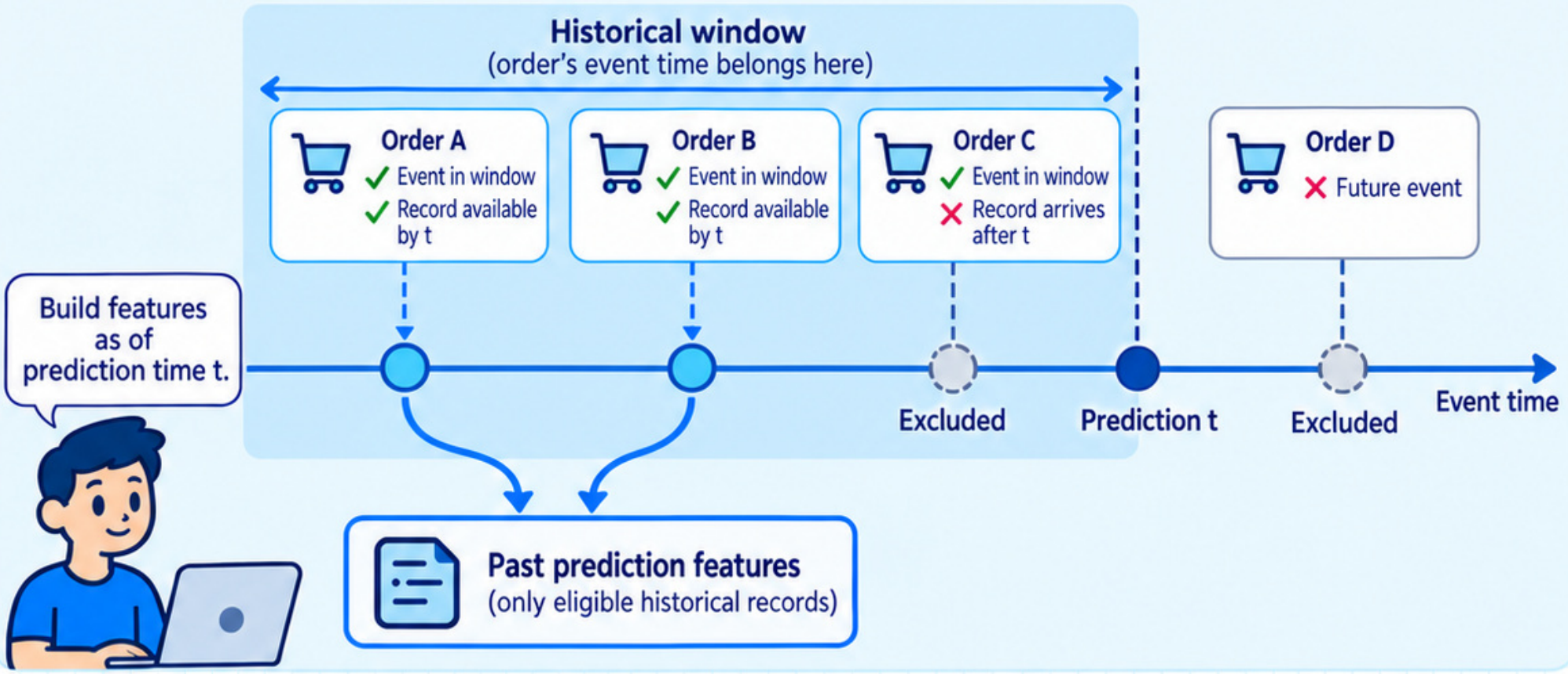
A forecast made at 09:00 cannot use that record, even though its event time is earlier than the prediction.

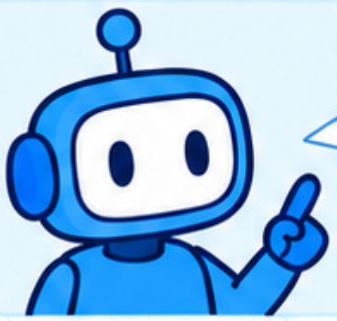
REBUILD WHAT WAS KNOWABLE THEN

Point-in-time correctness means using feature values valid for each historical prediction moment.

For our replay, include an order only when its event time belongs to the window and its record was available by prediction time.

A past prediction joins eligible historical records





Joining today's corrected totals onto old predictions can give the model information it never had.

Do not use current totals for past features.

DEFINE THE WINDOW EDGES

For prediction time t, this lesson uses:

$$[t - 168 \text{ hours}, t)$$

The opening bracket includes the start.
The closing parenthesis excludes t.

At 09:00 on 8 June, the window starts at 09:00 on 1 June.

Seven calendar days is a different definition; state which one you mean.

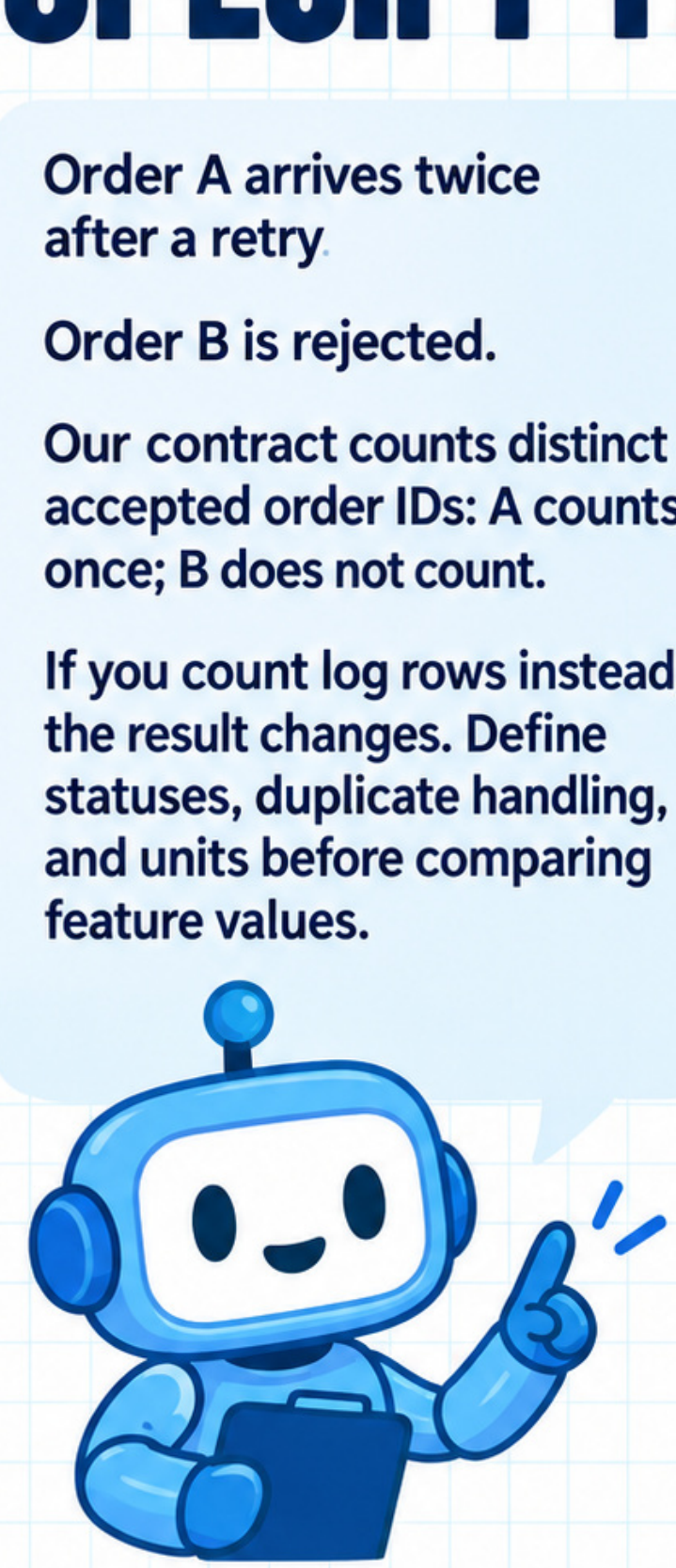
SPECIFY THE COUNTING RULE

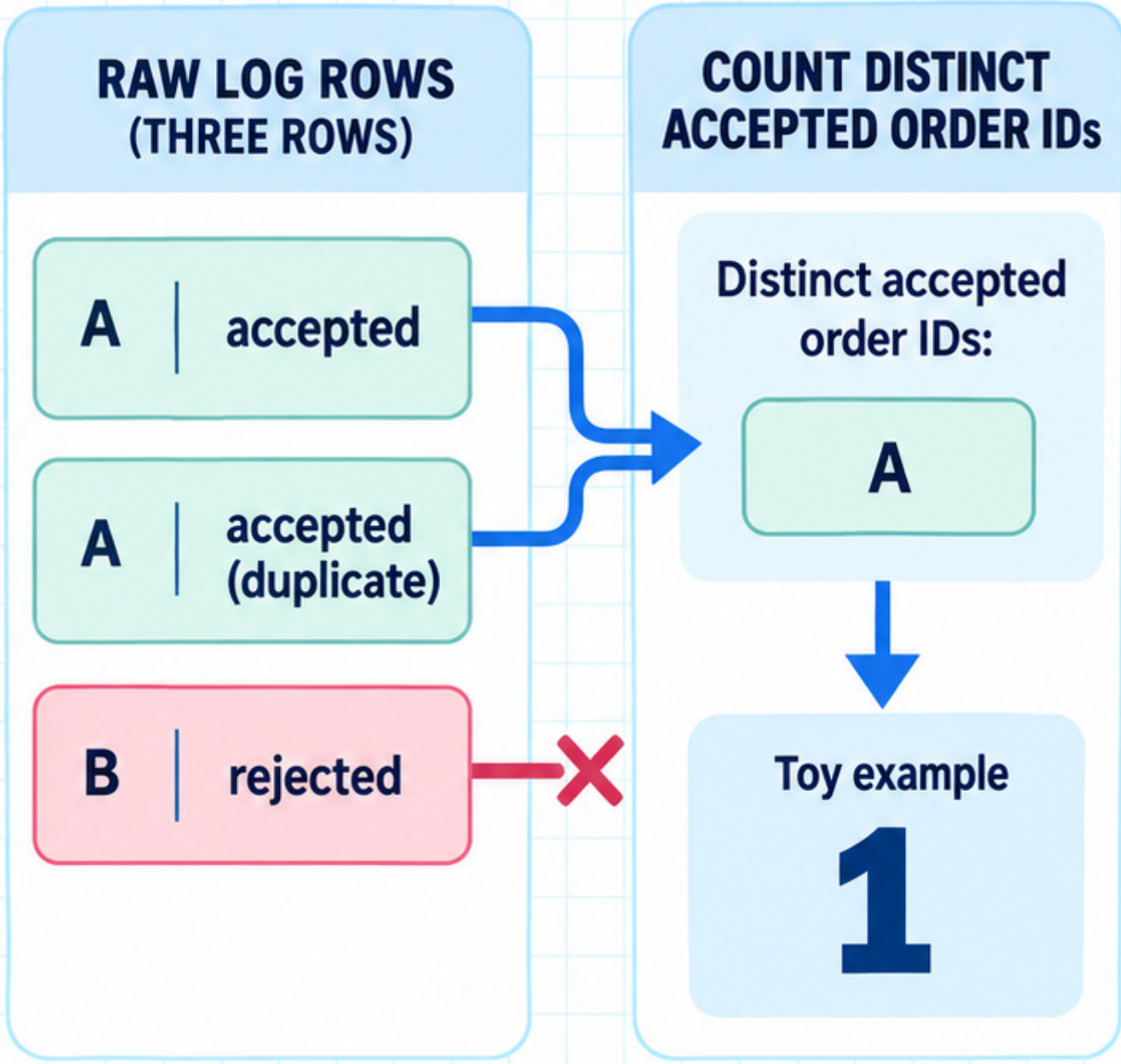
Order A arrives twice after a retry.

Order B is rejected.

Our contract counts distinct accepted order IDs: A counts once; B does not count.

If you count log rows instead, the result changes. Define statuses, duplicate handling, and units before comparing feature values.





ZERO AND MISSING ARE DIFFERENT

Healthy store feed
(complete source)

Unavailable store feed
(source or value is unavailable)

Zero
the source is complete and there were no qualifying orders.

Missing
the source or value is unavailable.

Replacing both with zero tells the model they mean the same thing.

Choose a documented policy: a trained imputation rule, a missingness indicator, or a fallback prediction path.

a trained imputation rule

a missingness indicator

a fallback prediction path.

CATEGORIES NEED CONSISTENT RULES TOO

A model may receive a store category such as “urban.”

The serving system must use the category mapping learned during training.

Define behavior for a new category before release: use the trained unknown-category path or an explicit fallback.

Do not silently reorder encoded columns when new categories appear.

Historical training rows
(from training data only)

store_category
urban
suburban
rural
...

Live request input
(with unseen category)

store_category
coastal

Saved training category mapping
(learned from training data only)

urban
suburban
rural
...
<unknown>

Reused to transform both historical training rows and live request inputs.

Encoded training rows
(using saved mapping)

			...

Encoded live request
(using saved mapping)

			...

New category uses the declared unknown-category path or explicit fallback.

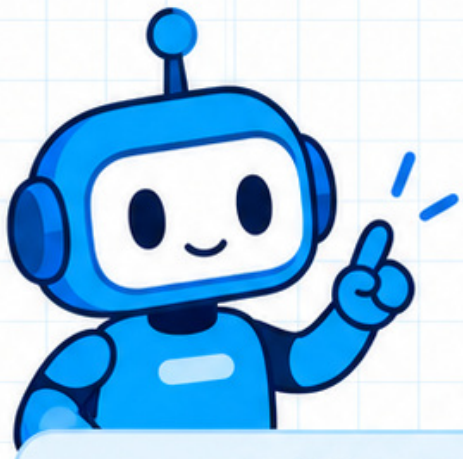
No silently re-ordered columns.

SHARE THE DEFINITION ACROSS PATHS

Offline computation prepares historical inputs for training or batch evaluation.

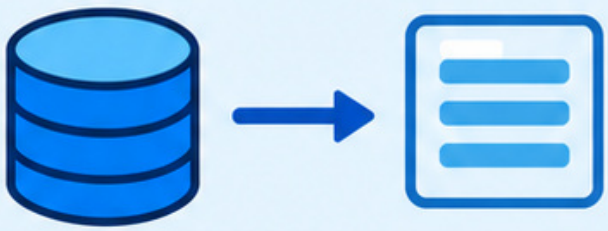
Online computation or retrieval supplies inputs during live requests.

Both should implement the same feature contract.



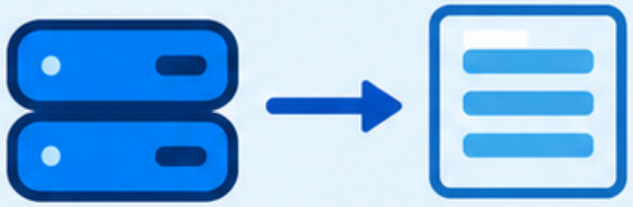
Feature Contract (versioned)

Offline Computation



Prepares historical inputs for training or batch evaluation.

Online Computation or Retrieval



Supplies inputs during live requests.

TRAINING-SERVING SKEW

If training counts accepted orders while serving counts all attempts, the model receives a different signal.
This is training-serving skew.

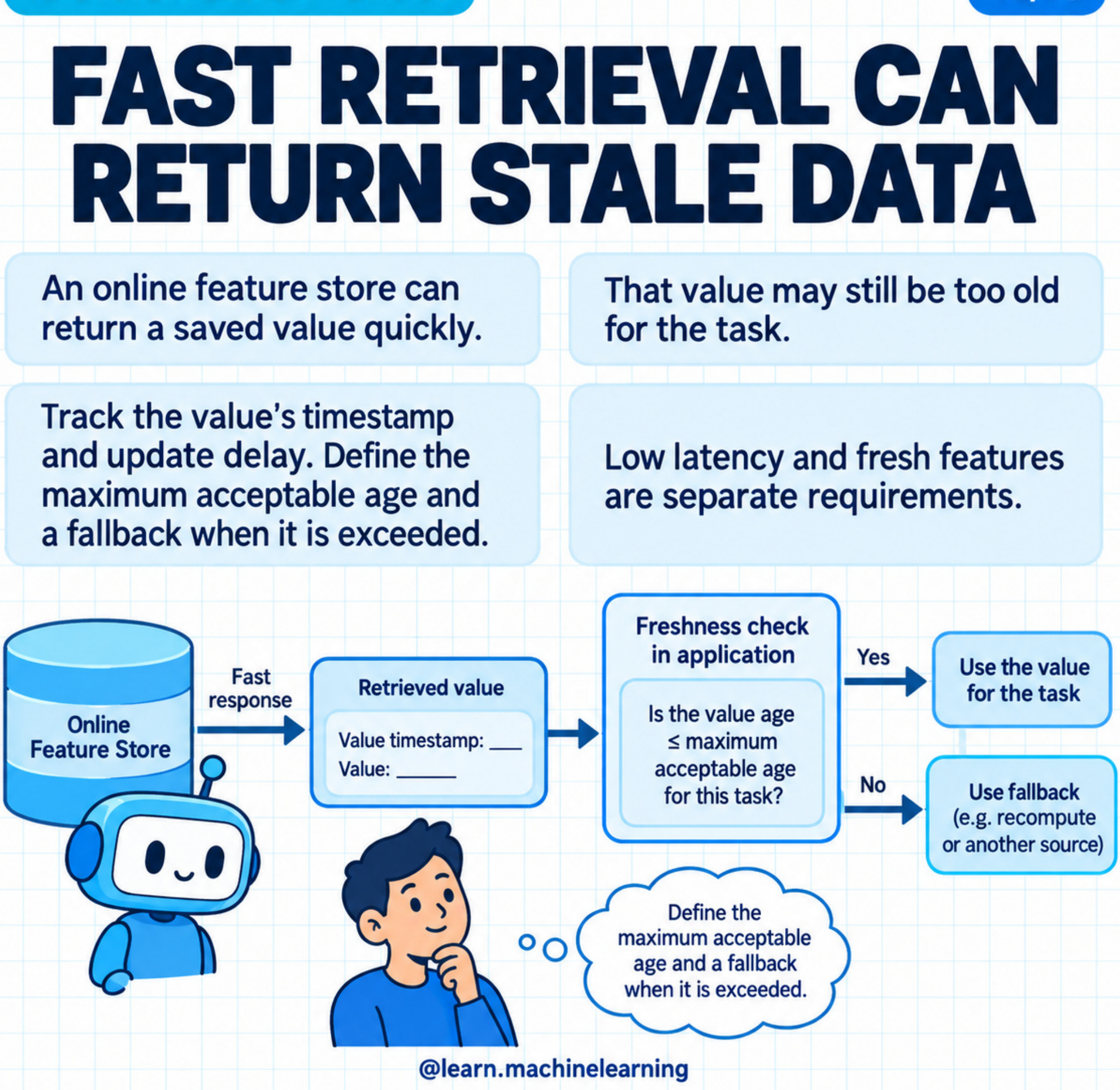
FAST RETRIEVAL CAN RETURN STALE DATA

An online feature store can return a saved value quickly.

That value may still be too old for the task.

Track the value's timestamp and update delay. Define the maximum acceptable age and a fallback when it is exceeded.

Low latency and fresh features are separate requirements.



CORRECTIONS NEED HISTORY

A late order or corrected status can change a historical aggregate.

A backfill recomputes older feature values after data or logic changes.

Same entity, multiple revisions

Original value
Available at timestamp T1

Corrected value
Available at timestamp T2

No overwritten past truth.
Both revisions are kept with their own availability timestamps.

T1
(original available)

T2
(correction available)

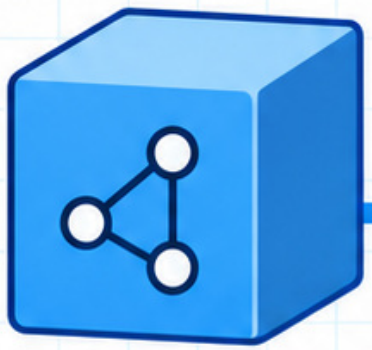
Preserve which revision was available when. Otherwise, a historical replay may use corrected knowledge that arrived later.

Keep the original serving record when the goal is to reproduce a past decision.

VERSION MORE THAN THE MODEL

Record:

- Feature definition and code version
- Input data snapshot or revision
- Window and timezone rules
- Saved preprocessing parameters
- Model version and run configuration



Model Artifact

v1.0.0

Model Manifest

- Feature definition and code version
- Input data snapshot or revision
- Window and timezone rules
- Saved preprocessing parameters
- Model version and run configuration

Changing accepted orders to completed orders changes the feature. Give that change a new version and evaluate the model with the matching inputs.

WHERE FEAST CAN HELP

Feast is one option for organizing and retrieving features.

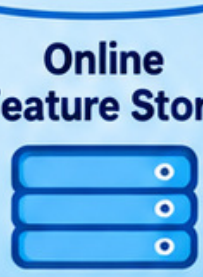
Feature View

Feature views describe groups of timestamped feature data.

Historical retrieval joins features for training examples.

Historical Store

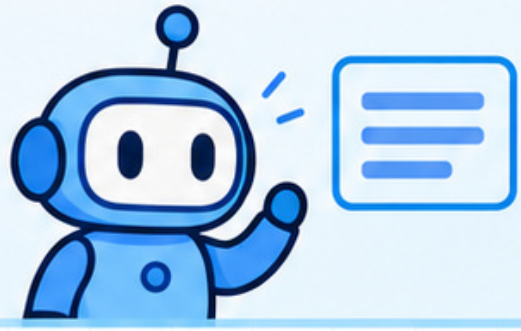
Online Feature Store



Online retrieval supplies stored values to live predictions.

Training Pipeline

Live Prediction Service



You still define correct timestamps, transformations, availability rules, and fallbacks.
A feature store is optional; a precise contract is essential.

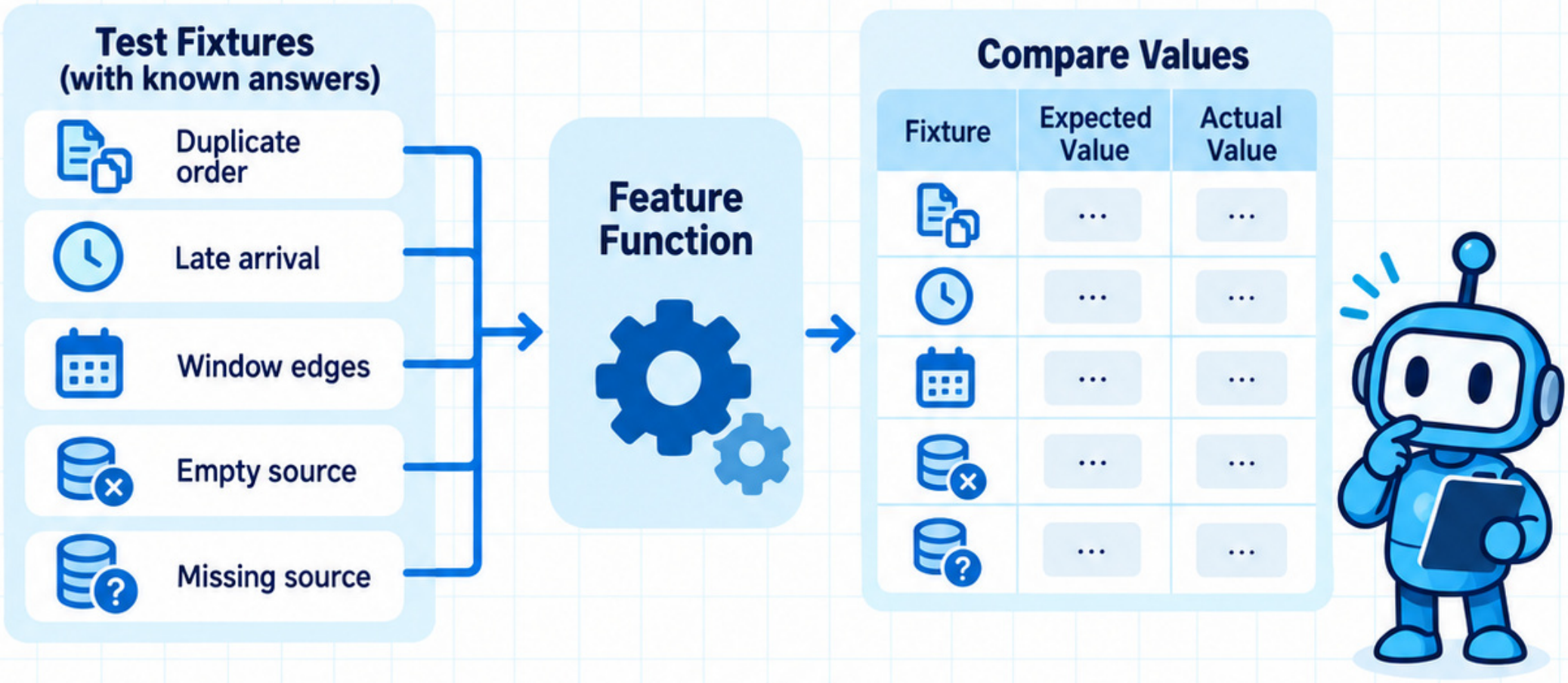
TEST THE FEATURE, NOT JUST THE MODEL

Use small records with known answers.

Check a duplicate order, a late arrival, both window edges, an empty source, and a missing source.

For the same entity and prediction time, compare offline replay with the recorded online value.

Investigate mismatches before interpreting a model-quality change.



WHICH ORDERS COUNT?

Prediction: 09:00. Window: previous 168 hours.

Assume all event dates are the prediction date, in UTC.

Order A

accepted at
08:55

available at
08:56

Order B

accepted at
08:57

available at
09:05

Order C

rejected at
08:58

available at
08:59

Duplicate A

A appears twice with the same order ID.

Under our contract, how many orders count? Explain each exclusion.

WRITE A FEATURE YOU CAN REPRODUCE

Define one input for a demand forecast.

Specify its entity, calculation, time window, availability rule, missing-value behavior, freshness limit, and version.

Add two small test cases with expected results.

Feature Contract

Feature name:

Entity:

Calculation:

Time window:

Availability rule:

Missing-value behavior:

Freshness limit:

Version:

Test Case 1

Input:

Expected result:

Test Case 2

Input:

Expected result:

DAY 11

DAY 11: Baselines and thresholds

Save the contract. Your model *depends* on it.