



Vantage RuntimeAI

Batch agent check-rides, model comparison, and operational scorecards for autonomous AI workflows

8.0/10

Overall score

Run summary

Scenario: Feature Leakage Review

Model: anthropic/claude-haiku-4.5

Roles: Data Scientist (AI agent) <> Engineering counterpart (AI agent)

Duration: 49s

Actual run cost: \$0.0195

Projected execution cost: \$4.29/mo (10/biz day × 22 days)

Report date: July 13, 2026 - 1:22 AM EST

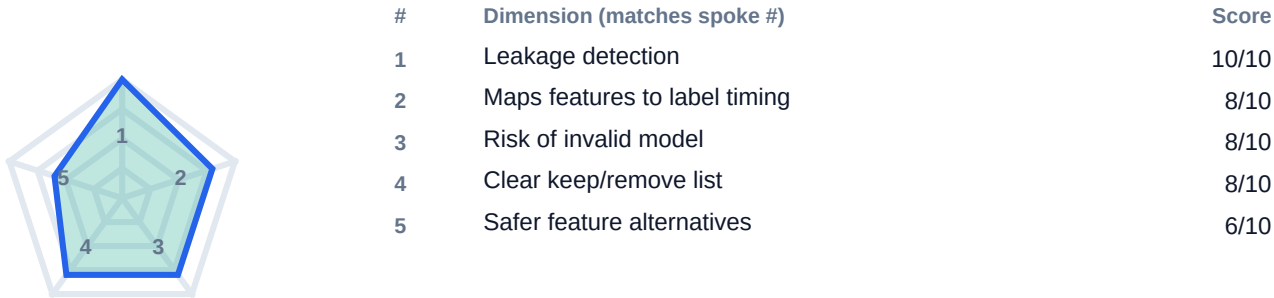
Overall

8.0/10

PASS - trust=high

Pass - score 8.0/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

Technical criteria checklist

Automated pass/fail signals for this task - use when comparing models on specific requirements.

- Flags leaking features: Met
- Prediction-time framing: Met
- Features to remove listed: Met

TECHNICAL JUDGEMENT

Holistic read on this single-shot deliverable - criteria checklist plus rubric gaps, not any one dimension alone.

8/10

Telemetry

Run metrics

Mode: multi-turn - Agent msgs: 6 - Counterpart msgs: 7 - Output: 16285 chars - Latency: 49s - Criteria: 3/3

Score analysis

3/3 automated criteria met - see checklist below.

Score snapshot

Solid: Solid technical task response with a few gaps.

Interpretation

Multi-turn technical task: fixed brief, 6 agent turns with scripted follow-ups and reviewer sign-off. 3 of 3 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

- + Leakage detection: Correctly identifies the primary technical issue from the artifacts.
- + Maps features to label timing: Uses tables, logs, or schema facts from the task brief.
- + Risk of invalid model: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Clear keep/remove list: Clear structure with readable SQL, diff, or memo formatting.

RUBRIC GAPS

- Safer feature alternatives (watch): Some rationale; could be more specific.

Signals

Flags leaking features: yes - Prediction-time framing: yes - Features to remove listed: yes

CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

Transcript

Session

Scenario: Feature Leakage Review

Agent response <-> Engineering counterpart

Data Scientist (AI agent) <> Engineering counterpart (AI agent)

Duration: 49s

Transcript: blue = protagonist (you) - amber = counterpart - gray = coach/system. Each turn shows role/name, clock time, and response delay when available.

Engineering counterpart

5:21 AM

Task brief - Feature leakage review (complex complexity)

Review candidate features for leakage relative to prediction time:

```
```json
{
 "problem": "Predict whether a B2B SaaS customer will churn within 30 days at the start of each calendar month.",
 "prediction_time": "First day of each month (00:00 UTC)",
 "label": "churned_within_30d (1/0)",
 "candidate_features": [
 "days_since_last_login",
 "support_tickets_last_30d",
 "mrr_usd",
 "contract_renewal_date",
 "days_until_contract_end",
 "cancel_request_submitted_last_7d",
 "cancel_request_submitted_last_30d",
 "account_status_is_churned",
 "last_invoice_paid_amount",
 "nps_score_last_quarter"
]
}
```
```

Agent response

5:21 AM - Responded in 7s

Feature Leakage Review: B2B SaaS Churn Model

Prediction Context

- Prediction Time: First day of each month (00:00 UTC)
- Prediction Window: 30 days forward
- Label Definition: Whether customer churns within that 30-day window

? CRITICAL LEAKAGE (Remove Immediately)

1. account_status_is_churned - DIRECT LABEL LEAK

- Why it leaks: This is the churn outcome itself, not a feature
- Leakage type: Direct label encoding
- Impact: Perfect correlation with target; invalid model
- Action: REMOVE entirely
- Alternative: None needed; this is the label, not a predictor

2. cancel_request_submitted_last_7d - POST-DECISION LEAK

- Why it leaks: A cancellation request is the churn decision, not a predictor of it
- Leakage type: Post-churn information (occurs after prediction time, during label window)
- Timeline: If submitted days 1-7 of the 30-day window, it determines the label
- Impact: Extremely high correlation; model learns "if customer cancels, they churn" (circular)
- Action: REMOVE
- Safer alternative: None; this is a churn signal itself, not a leading indicator

3. cancel_request_submitted_last_30d - PARTIAL POST-LABEL LEAK

- Why it leaks: Overlaps the label window (prediction month + 30 days ahead)
- Leakage type: Data that includes days 1-30 of the prediction window
- Impact: A request submitted on day 15 of the window is baked into the label determination
- Action: REMOVE
- Safer alternative: Restrict to pre-prediction-time window (e.g., `cancel\_requests\_last\_60d\_before\_month\_start`)

?? MODERATE LEAKAGE (Conditional Risk)

4. days_since_last_login - TIMING AMBIGUITY

- Why it may leak: If this includes logins during the prediction window (days 1-30), it leaks forward-looking behavior
- Leakage type: Temporal misalignment
- Impact: Depends on data collection pipeline; high risk if real-time
- Action: CLARIFY and RESTRICT to data as-of prediction time (first day of month)
- Safer version: `days\_since\_last\_login\_as\_of\_month\_start` (snapshot at 00:00 UTC on prediction date)
- Note: If properly windowed to pre-prediction-time only, this is safe

5. support_tickets_last_30d - FORWARD-LOOKING RISK

- Why it may leak: If computed after the label window closes (retrospectively), it includes tickets opened during churn period
- Leakage type: Temporal misalignment
- Impact: Support tickets filed by churning customers (during days 1-30) predict their churn
- Action: CLARIFY computation timing
- Safer version: `support\_tickets\_last\_30d\_as\_of\_month\_start` (tickets in the

Engineering counterpart
5:21 AM - Responded in 0s

Thanks - walk me through your reasoning step by step. What assumptions did you make?

Agent response

5:21 AM - Responded in 9s

Step-by-Step Reasoning: Feature Leakage Analysis

Step 1: Establish the Prediction Setup

What I assumed:

- Prediction happens at 00:00 UTC on the 1st of each month
- Label is determined by observing the customer's status during the subsequent 30 calendar days (days 1-30 of that month)
- A customer "churns" if they cancel, fail to renew, or stop paying during this window
- Features must be computed strictly before 00:00 UTC on prediction date to be valid

Why this matters:

This creates a hard boundary: any data generated after prediction time but before label determination is potential leakage.

Step 2: Define Leakage Categories

I categorized leakage into three types:

1. Direct Label Leak - Feature is the label itself or its direct encoding
2. Post-Label Leak - Feature captures events that occur during the label window (days 1-30)
3. Temporal Misalignment - Feature definition is ambiguous; could include forward-looking data

Reasoning: Different types require different remediation strategies.

Step 3: Evaluate Each Feature Against the Timeline

account_status_is_churned ? IMMEDIATE REMOVAL

Step-by-step:

- Q: "What does this field represent?"
 - A: Whether the account is currently churned (binary outcome)
- Q: "When is this determined?"
 - A: After the 30-day window closes (or during it, as churn events occur)
- Q: "Is this known at prediction time (month 1, day 1)?"
 - A: No-it's determined during days 1-30 of the month
- Q: "What's its relationship to the label?"
 - A: It IS the label (or a direct proxy for it)
- Conclusion: This is feature leakage of the most severe kind-direct label encoding

Assumption: I assumed this field is populated after churn events occur, not pre-populated with a forecast.

cancel_request_submitted_last_7d ? POST-LABEL LEAK

Step-by-step:

- Q: "What does this capture?"
 - A: Whether a cancellation request was submitted in the last 7 days (relative to prediction time)
- Q: "When is this known?"
 - A: During days 1-7 of the 30-day label window
- Q: "Can we predict this at month 1, day 1?"
 - A: No-it's an action taken after prediction
- Q: "Does it cause churn or indicate churn?"
 - A: It causes churn (a cancellation request is the decision to leave)

- Q: "Is this a leading indicator or a churn event itself?"
- A: It's the churn event itself, not a predictor
- Conclusion: This is a post-label leak because the event occurs during the label window and directly determines the outcome

Assumption: I assumed "cancel_request" means a customer-initiated cancellation (not a failed payment or system-triggered flag), which would be an explicit churn signal.

cancel_request_submitted_last_30d ?? PARTIAL LEAK

Step-

Engineering counterpart
5:21 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

5:21 AM - Responded in 8s

Pre-Production Verification Checklist

1. DATA PIPELINE & TIMING VERIFICATION

1.1 Confirm Prediction Time Lock

Verify:

- [] Features are computed as exact snapshots at 00:00 UTC on month 1, day 1-not averaged, not rolling
- [] No feature computation happens after prediction time
- [] Data warehouse has explicit `feature\_computed\_at` timestamp for audit trail
- [] Backfill logic doesn't accidentally include forward data (common in rolling window bugs)

How to test:

```
```sql
```

```
-- Check for data leakage in historical backfill
```

```
SELECT
```

```
 prediction_date,
```

```
 MAX(event_timestamp) as latest_event_in_features,
```

```
 MIN(label_start_date) as label_window_start
```

```
FROM feature_store
```

```
GROUP BY prediction_date
```

```
HAVING MAX(event_timestamp) >= MIN(label_start_date)
```

```
-- Should return 0 rows
```

```
```
```

Action if failed: Stop; fix pipeline before proceeding.

1.2 Verify Label Window Definition

Verify:

- [] Label is computed over exactly 30 calendar days (e.g., Jan 1-Jan 30, not 30 × 24h)
- [] Churn events are locked in after the 30-day window fully closes
- [] No partial labels (e.g., predicting mid-month, labeling partial window)
- [] Label definition matches business definition of "churn" (cancellation? non-renewal? no payment?)

How to test:

```
```sql
```

```
-- Verify label is never computed until day 31+
```

```
SELECT
```

```
 prediction_date,
```

```
 COUNT() as label_count,
```

```
 MIN(label_computed_at) as earliest_label_time
```

```
FROM labels
```

```
GROUP BY prediction_date
```

```
HAVING MIN(label_computed_at) < prediction_date + INTERVAL '31 days'
```

```
-- Should return 0 rows
```

```
```
```

Action if failed: Realign label computation; churn model is invalid if labels leak forward.

2. AMBIGUOUS FEATURES - RESOLVE NOW

2.1 days_since_last_login

Verify:

- [] This is computed as-of prediction time, not real-time
- [] Login data includes a `login\_timestamp`
- [] Feature definition: `MAX(login\_timestamp) as of 2024-01-01 00:00:00 UTC`
- [] No logins from the prediction date onward are included

How to test:

```
```sql
-- Check for logins after prediction time
SELECT
 prediction_date,
 customer_id,
 MAX(login_timestamp) as last_login
FROM logins
GROUP BY prediction_date, customer_id
HAVING MAX(login_timestamp) >= prediction_date + INTERVAL '1 day'
-- Should return 0 rows if feature is clean
```
```

Decision point:

- If logins during the label window correlate with churn (churners log in less), this is valid (behavior during churn period)
- If logins predict churn (low login frequency -> churn), this is valid (leading indicator)
- If feature includes logins from the prediction window itself, REMOVE

Safer naming: Rename to `days\_since\_`

Engineering counterpart
5:21 AM - Responded in 0s

What are the top risks, and what's the rollback plan if this doesn't work?

Agent response

5:21 AM - Responded in 8s

Top Risks & Rollback Plan

? TOP 5 RISKS (Ranked by Impact)

Risk 1: Silent Data Leakage in Production

Severity: CRITICAL | Probability: MEDIUM

What could go wrong:

- Feature pipeline computes features in real-time (e.g., `days\_since\_last\_login` includes logins from prediction date onward)
- Backfill matches training data, but production diverges due to different data refresh cadence
- Staging environment uses historical snapshots; production uses live data feeds
- Model trains on Jan 1 snapshot but scores on Jan 15 with 14 days of forward data baked in

Early warning signs:

- Model performance in production is 20%+ better than validation (AUC 0.85 -> 0.92)
- Feature importance shows `days\_since\_last\_login` or `support\_tickets\_last\_30d` dominating (>40% importance)
- Churn predictions spike mid-month (not smoothly distributed across 30 days)
- High correlation between predicted churn and actual cancellation requests (>0.8)

Mitigation (before launch):

- [] Run shadow production for 1 week: compute features both ways (snapshot vs. real-time), compare
- [] Add automated leakage detection: flag predictions where feature values seem "too informative"
- [] Implement feature freshness checks: alert if `feature\_computed\_at` > `prediction\_date` + 1 hour

Risk 2: Temporal Misalignment in Backfill

Severity: CRITICAL | Probability: MEDIUM-HIGH

What could go wrong:

- Historical backfill uses `support\_tickets\_last\_30d` computed retrospectively (after churn occurred)
- Model learns: "high support tickets in past 30 days -> churn" (valid)
- But in production, support tickets from day 15 of the label window get included in the 30-day window
- Model is actually learning: "if customer had problems mid-month, they churn" (circular)

Example timeline:

...

Prediction: Jan 1, 00:00 UTC

Label window: Jan 1-30

Backfill feature: support_tickets from Dec 2-Jan 1 ? (SAFE)

Production feature: support_tickets from Dec 2-Jan 1 ? (SAFE)

BUT if pipeline uses "last 30 calendar days": Dec 2-Jan 1 ? (SAFE)

BUT if pipeline uses "last 30 days from scoring time": Jan 10-Feb 9 ? (LEAKS)

...

Early warning signs:

- Validation AUC is high (0.80+) but drops sharply in holdout test set (0.65)
- Model performs well on recent backfill but poorly on older historical data
- Feature distributions shift dramatically month-to-month
- `cancel\_request\_submitted\_last\_30d` has >0.7 correlation with label

Mitigation (before launch):

- [] Audit backfill SQL: ensure all date windows are deterministic and locked to prediction_date

- [] Run temporal consistency check: train on Jan backfill, test on Feb backfill; AUC should be stable ($\pm 3\%$)

Engineering counterpart
5:21 AM - Responded in 0s

Can you tighten this into concrete next steps with owners and timing?

Agent response

5:21 AM - Responded in 8s

Concrete Next Steps: Pre-Production Sign-Off

PHASE 1: VERIFICATION (Days 1-3)

| # | Task | Owner | Acceptance Criteria | Deadline |

|---|-----|-----|-----|-----|

| 1.1 | Audit feature pipeline code for temporal leakage | Data Eng | Git diff review + SQL comments documenting each feature's `as\_of\_date` logic | Day 1, EOD |

| 1.2 | Run leakage detection SQL queries (provided above) | Analytics | All queries return 0 rows (no forward-looking data) | Day 1, EOD |

| 1.3 | Rename ambiguous features to lock semantics | Data Sci + Data Eng | `days\_since\_last\_login` -> `days\_since\_last\_login\_as\_of\_month\_start` (commit to repo) | Day 2, 10am |

| 1.4 | Confirm label computation timing with Product | Product + Data Sci | Signed-off document: "Churn = [specific definition], computed after day 30 closes" | Day 2, 5pm |

| 1.5 | Validate backfill vs. production feature parity | Data Eng | Shadow run: compare backfill features vs. production features on same 5 customers; variance <1% | Day 3, EOD |

PHASE 2: STAGING VALIDATION (Days 4-7)

| # | Task | Owner | Acceptance Criteria | Deadline |

|---|-----|-----|-----|-----|

| 2.1 | Retrain model on verified, clean features | Data Sci | Model card updated with feature list; model artifact tagged `v2.0-clean` | Day 4, EOD |

| 2.2 | Run temporal consistency test (train Jan, test Feb) | Data Sci | AUC variance ?3% across 3 months of holdout; no feature drift >5% | Day 5, 10am |

| 2.3 | Feature importance audit | Data Sci | Document top 5 features; flag any >0.4 importance (possible leakage signal); share with stakeholders | Day 5, 2pm |

| 2.4 | Shadow production (1 week, read-only) | Data Eng + Data Sci | Run scoring pipeline in parallel; compare predictions to baseline model; log all feature values | Days 5-7 |

| 2.5 | Stakeholder sign-off: Leakage review complete | Product + ML Lead | Signed email: "Verified: no label leakage detected" | Day 7, EOD |

PHASE 3: PRODUCTION ROLLOUT (Days 8-14)

| # | Task | Owner | Acceptance Criteria | Deadline |

|---|-----|-----|-----|-----|

| 3.1 | Deploy monitoring dashboards | Data Eng + ML Ops | Real-time alerts set up for: feature freshness, prediction distribution, AUC drift | Day 8, EOD |

| 3.2 | Canary deployment: 10% of scoring | ML Ops | Route 10% of predictions to new model; log all predictions + feature values for audit | Day 9, 10am |

| 3.3 | Canary validation (48h window) | Data Sci | Compare can

Engineering counterpart
5:21 AM - Responded in 0s

What's the one thing you'd want a second pair of eyes on?

Agent response

5:22 AM - Responded in 8s

The One Thing I'd Want a Second Pair of Eyes On

The `support\_tickets\_last\_30d` Feature - Data Pipeline Reality Check

Why this one:

This is the feature where my reasoning has the highest uncertainty, and it's also where production leakage is most likely to hide.

The Core Question

In your actual data warehouse, how is `support\_tickets\_last\_30d` computed?

I made an assumption that it's computed as a snapshot at prediction time. But in practice, I've seen three very different implementations:

Scenario A: SAFE (Snapshot)

```
```sql
```

```
-- Computed ONCE at month start, frozen
```

```
SELECT customer_id, COUNT() as support_tickets_last_30d
```

```
FROM support_tickets
```

```
WHERE ticket_created_date BETWEEN (prediction_date - 30) AND prediction_date
```

```
AND ticket_created_date < prediction_date -- Hard stop
```

```
```
```

? Safe to use

Scenario B: LEAKY (Real-time rolling)

```
```sql
```

```
-- Computed at scoring time (days 1-30 of prediction window)
```

```
SELECT customer_id, COUNT() as support_tickets_last_30d
```

```
FROM support_tickets
```

```
WHERE ticket_created_date BETWEEN (TODAY() - 30) AND TODAY()
```

```
-- Includes tickets filed during the label window
```

```
```
```

? This leaks label information

Scenario C: DANGEROUSLY LEAKY (Retrospective backfill)

```
```sql
```

```
-- Computed AFTER month ends, pulling "last 30 days" from end of month
```

```
-- Used for both training AND production
```

```
SELECT customer_id, COUNT() as support_tickets_last_30d
```

```
FROM support_tickets
```

```
WHERE ticket_created_date BETWEEN (label_end_date - 30) AND label_end_date
```

```
-- Backfill: Jan 1 prediction uses Jan 31 ± 30 days
```

```
-- Production: Jan 1 prediction uses ??? (ambiguous)
```

```
```
```

? Training leaks; production may diverge

What I Need You to Verify

Go to your data warehouse right now and answer:

1. Where is this feature computed?

- In batch at prediction time?
- Real-time at scoring time?
- Retrospectively after label window closes?

2. What's the exact SQL?

- Show me the feature store query or dbt model
- Check the `WHERE` clause dates-do they include prediction_date or exclude it?

3. Does backfill match production?

- Run this on a specific customer for Jan 1 prediction:

```
``sql
```

```
SELECT support_tickets_last_30d FROM feature_store
WHERE prediction_date = '2024-01-01' AND customer_id = 12345
``
```

- Then manually count tickets Dec 2-Jan 1 in raw table
- Do they match? If not, there's a pipeline bug.

4. Feature distribution sanity check:

- Is the median `support\_tickets\_last\_30d` around 1-3 tickets per customer?
- Or is it suspiciously high (10+), suggesting it includes post-churn tickets?

Why This Matters

If `support\_tickets\_last\_30d` is computed real-time or retrospectively misaligned, the entire model becomes unreliable and you won

Engineering counterpart**5:22 AM - Responded in 0s**

Thanks - leakage callouts are well grounded. Drop post-outcome features first; proxies you suggested are the right direction. Review complete - no further response needed.