



Vantage RuntimeAI

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

7.2/10

Overall score

Run summary

Scenario: Feature Leakage Review

Model: google/gemini-2.5-flash

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

Duration: 34s

Actual run cost: \$0.007950

Projected execution cost: \$1.75/mo (10/biz day × 22 days)

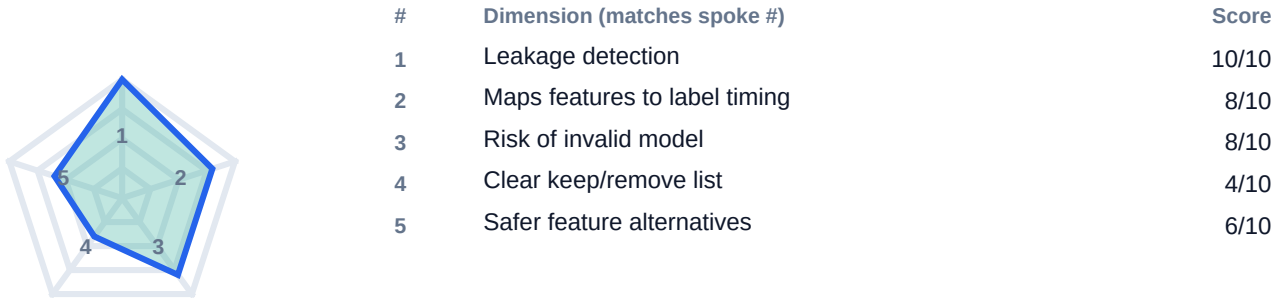
Report date: July 13, 2026 - 11:30 PM EST

Overall

7.2/10

PASS - trust=high
Pass - score 7.2/10 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: Not met

TECHNICAL JUDGEMENT

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

6/10

Telemetry

Run metrics

Mode: multi-turn - Agent msgs: 6 - Counterpart msgs: 7 - Output: 33660 chars - Latency: 34s - Criteria: 2/3

Score analysis

2/3 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. 2 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.

RUBRIC GAPS

- Clear keep/remove list (critical gap): Hard to follow or lacks concrete deliverable formatting.
- Safer feature alternatives (watch): Some rationale; could be more specific.

Signals

Flags leaking features: yes - Prediction-time framing: yes - Features to remove listed: no

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: 34s

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

Engineering counterpart

3:30 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"
]
}
```
```

3:30 AM - Responded in 5s

Problem: Predict whether a B2B SaaS customer will churn within 30 days at the start of each calendar month.

Label: `churned\_within\_30d` (1/0) - This means the customer churned between the prediction time and the next 30 days.

...

Here's a breakdown of each candidate feature, identifying potential leakage, explaining why, and suggesting alternatives.

| Safer Alternatives (if any)

[illegible]

| | | |
|-------------------------|------|---|
| `days_since_last_login` | Safe | This feature measures user activity leading up to the prediction time. A customer's last login could have been today (the first of the month) or earlier. It does not incorporate any information from after the prediction time. |
|-------------------------|------|---|

1

| Feature Name | Safe | Description |
|----------------------------|------|--|
| `support_tickets_last_30d` | Safe | This feature counts support tickets opened in the 30 days prior to the prediction time. It does not include tickets opened after the prediction time, so it's a valid historical signal. |

| Field | Description |
|------------------------|--|
| <code>`mrr_usd`</code> | Safe Assumed to be the current Monthly Recurring Revenue at the prediction time. This is a point-in-time value reflecting the customer's contract status at the start of the month, not future changes or churn. |

|

|

| `contract\_renewal\_date` | Safe | This is a fixed date in the future (or past, if the contract is already overdue for renewal). It's known at the prediction time and does not leak future churn information. It's a strong indicator for churn risk around that date.

|

|

| `days\_until\_contract\_end` | Safe | Derived from `contract\_renewal\_date` and the prediction time. This value is known at the prediction time and reflects the current state of the contract. It's a critical feature for churn.

Engineering counterpart
3:30 AM - Responded in 0s

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

Agent response

3:30 AM - Responded in 5s

You're welcome! Let's break down the reasoning step-by-step for each feature, explicitly stating the assumptions made.

General Assumptions for all Features:

1. Prediction Time: All feature values are captured at or before the `prediction\_time` (First day of each month, 00:00 UTC). Any information that becomes known after this exact moment, but before the 30-day churn window ends, is considered leakage if it informs the feature value.
2. Label Definition: `churned\_within\_30d` means the customer transitioned to a churned state at any point from `prediction\_time` + 1 minute up to `prediction\_time` + 30 days.
3. Feature Value Freshness: We assume feature values are "freshest" at the prediction time. For instance, `days\_since\_last\_login` reflects the state right at the start of the month, not an average over the past month.
4. No Future Knowledge Implied: Unless explicitly stated, we assume features are derived using only information available up to and including the prediction time.

Step-by-Step Reasoning for Each Feature:

1. `days\_since\_last\_login`

What it represents: The number of days that have passed since the customer's last interaction (login) with the service.

When is it known? This value is calculated based on historical login data up to the `prediction\_time`. If a customer logged in yesterday, `days\_since\_last\_login` would be 1. If they logged in one minute before the prediction time on the first of the month, it would be 0 (or a very small fraction).

Does it use future information? No. The "last login" event must have occurred before or at the `prediction\_time`. It cannot incorporate any logins that happen after the prediction time and within the 30-day churn window, because those events are in the future relative to when we are making the prediction.

Leakage? No.

Conclusion: Safe.

2. `support\_tickets\_last\_30d`

What it represents: The count of support tickets opened by the customer in the 30 days immediately preceding the `prediction\_time`.

When is it known? This is a historical count. To calculate this feature at the `prediction\_time` (e.g., Jan 1st), you would count tickets opened between Dec 2nd and Jan 1st.

Does it use future information? No. It explicitly refers to the "last 30 days," which, when evaluated at `prediction\_time`, means the 30-day period ending at `prediction\_time`. It does not include any tickets opened after `prediction\_time` (e.g., on Jan 5th or Jan 15th), which might occur in the 30-day churn window.

Leakage? No.

Conclusion: Safe.

3. `mrr\_usd`

What it represents: Monthly Recurring Revenue in USD.

When is it known? This is typically a value associated with the current active contract or subscription at a given point in time. At the `prediction\_time`, we would know the `mrr\_usd` value that is currently in effect for the customer.

Does it use future information? No. We assume `mrr\_usd`

Engineering counterpart
3:30 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

3:30 AM - Responded in 7s

Okay, before shipping these features and the churn prediction model to production, a thorough verification process is crucial. My primary goal during this phase is to ensure the model's reliability, fairness, and continued performance, as well as to guard against any lingering or new forms of leakage that might not have been caught in the initial review.

Here's what I would verify, categorized for clarity:

I. Data Integrity & Feature Engineering Verification

1. Strict Time-Series Validation (Re-emphasis on Leakage):

Backtesting Infrastructure Check: I'd meticulously review the code and data pipeline used to generate the features for historical training data. The most common source of production leakage is an improperly constructed backtesting environment.

"Time Travel" Test: For a sample of customers and prediction times (e.g., first of Jan, Feb, Mar 2023), I would manually inspect the raw data used to construct each feature value. I'd verify that no data point created after the specific `prediction\_time` for that row (e.g., a `cancel\_request\_submitted\_last\_7d` entry from Jan 5th for a Jan 1st prediction) inadvertently influenced any feature. This often involves looking at database timestamps.

Feature Refresh Rates: Confirm that the scheduling for feature generation in production aligns with the prediction time. If features are only updated weekly, but we predict daily, we need to understand the implications.

2. Feature Definition and Consistency:

Ambiguity Resolution: Clarify any remaining ambiguities. For `mrr\_usd`, for example, is it the standard MRR, or does it include one-off charges? Is it based on the current month's billing or the previous month's? This needs to be consistent between training and production.

Edge Case Handling: How are features calculated for new customers with less than 30 days of data? Or customers with very old contracts? Ensure these edge cases are handled consistently (e.g., `NaN`, 0, default values) and that the model is robust to them.

Data Type and Format: Confirm that feature data types and formats (e.g., integer vs. float, date format) are consistent between training, validation, and production environments.

3. Missing Value Strategy:

Imputation Consistency: Verify that missing value imputation strategies (e.g., mean, median, zero, or a specific "missing" category) are applied identically in training, validation, and the production inference pipeline.

Impact of Missingness: Analyze cases where features might be legitimately missing (e.g., `nps\_score\_last\_quarter` for a new customer). Does the imputation strategy make sense, and how does the model handle these cases?

II. Model Performance & Robustness Verification

1. Performance Monitoring Plan:

Metrics Definition: Define clear production monitoring metrics (e.g., AUC, Precision, Recall, F1-score for churn class, calibration curves).

Alerting Thresholds: Establish thresholds for these metrics that would trigger an alert for model degradation.

Drift Detection: Plan for monitoring data drift (changes in feature distributions) and concept drift (changes in the relationship between features and the label).

Feedback Loop: How will actual churn outcomes be fed back into the system to evaluate model performance over time and retrain?

2. Model Calibration:

Probability Interpretation: If the model outputs probabilities, verify that these probabilities are well-calibrated (e.g., using Platt Scaling, Isotonic

Engineering counterpart
3:30 AM - Responded in 0s

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

Agent response

3:30 AM - Responded in 6s

Excellent questions! Identifying top risks and having a solid rollback plan are critical for any production deployment.

Top Risks for Churn Model Deployment

When deploying a churn prediction model, especially one where feature leakage is a primary concern, the risks often boil down to the model not performing as expected in the real world, and potentially causing negative business outcomes.

Here are the top risks:

1. Undetected Feature Leakage (The "Silent Killer"):

Description: Despite rigorous review, a subtle form of leakage might still exist, leading to an artificially inflated performance during development and testing. This is the most insidious risk because the model appears to work well, but fails in production.

Impact: The model will perform poorly in production, making inaccurate predictions. This leads to wasted resources (e.g., offering retention incentives to customers who were never going to churn, or failing to identify truly at-risk customers), loss of trust in the model, and potentially critical business decisions based on faulty information.

Why it's a top risk: It's hard to detect without strong time-series validation and often only surfaces when exposed to live, unseen data.

2. Data Drift and Concept Drift:

Description:

Data Drift: The distribution of input features changes over time (e.g., a new product launch significantly alters user behavior, or a change in marketing strategy brings in a different customer demographic).

Concept Drift: The relationship between features and the churn label changes over time (e.g., economic downturns make customers more sensitive to price, or a competitor's new offering changes the "churn-triggering" behaviors).

Impact: The model, trained on historical data, becomes less relevant and accurate as the underlying patterns change, leading to degraded performance.

Why it's a top risk: Churn behavior is dynamic, influenced by market, product, and customer-specific factors. Models degrade over time without maintenance.

3. Actionability and Operational Integration Challenges:

Description: The model produces predictions, but the business teams struggle to interpret them, act on them effectively, or integrate them into existing workflows. This isn't a model performance issue, but a systemic failure to leverage the model.

Impact: The model becomes a "shelfware" project, providing no real business value despite potentially accurate predictions. Resources spent on development are wasted.

Why it's a top risk: A model is only as good as its ability to drive positive action. Without clear integration and buy-in, even a perfect model is useless.

4. Resource Overload / False Positives:

Description: The model predicts a high number of customers to be "at-risk" (false positives), leading to churn prevention teams being overwhelmed and having to inefficiently allocate resources.

Impact: Increased operational costs, frustration among retention teams, and a dilution of effort, making it harder to focus on the truly at-risk customers.

Why it's a top risk: A model needs to be not just accurate, but also efficient in identifying at-risk customers at a scale that the business can handle.

Rollback Plan

A robust rollback plan provides a safety net, allowing you to quickly revert to a stable state if the new model introduces problems.

Goal: Minimize business disruption and potential negative impact on customer relationships or revenue.

Phases of Rollback:

1. Pre-Deployment Readiness (Prevention & Preparation):
Baseline Performance: Clearly define and document the current

Engineering counterpart
3:30 AM - Responded in 0s

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

3:30 AM - Responded in 6s

Overall Goal: Deploy the Churn Prediction Model to Production Safely and Effectively.

Target Deployment Date: [Insert Desired Date, e.g., End of Q3 2024]

...

Phase 1: Pre-Deployment Readiness & Verification (2-3 Weeks)

Owner: Data Science Lead (with support from Data Engineer, MLOps, Business Analyst)

| # | Task | Description |
|---|------|-------------|
|---|------|-------------|

[illegible]

| 1 | Strict Time-Series Validation (Leakage Audit) | Re-audit the full feature pipeline with an independent data scientist. Manually inspect a sample of 10-20 customer-prediction time rows from historical data:

| | | | | |
|--|------------------------|---|--|--|
| | Data Scientist (Audit) | Day 1-5 (Parallel) | | |
| | | - Verify that every feature value for that specific `prediction_time` uses data strictly prior to or at that `prediction_time`. | | |

| | | - Pay special attention to features identified as high-risk
(`cancel\_request\_submitted\_last\_7d`, `cancel\_request\_submitted\_last\_30d`, `account\_status\_is\_churned`). Trace their
data sources to ensure no post-prediction information is used.

- Output: Detailed report of findings, any identified leakage points, and proposed fixes.

| | |
| 2 | Define & Implement Production Monitoring Metrics | Work with Stakeholders to identify key business metrics (e.g., number of at-risk customers identified, conversion rate of retention efforts) and technical metrics (AUC, Precision@K, Recall@K, Calibration).

| | | Data Science Lead, MLOps | Day 1-7 | |
| | | - Set up dashboards/alerts for model performance degradation, data drift, and concept drift.

Engineer | Day 8-14 | | | MLOps
| | | - Output: Monitoring dashboards, alert configurations, and a clear trigger for investigation/rollback (e

Engineering counterpart

3:30 AM - Responded in 0s

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

Agent response

3:30 AM - Responded in 5s

The one thing I would absolutely want a second (independent) pair of eyes on is the "Strict Time-Series Validation (Leakage Audit)" task, specifically for the features:

```
`cancel_request_submitted_last_7d`  
`cancel_request_submitted_last_30d`  
`account_status_is_churned`  
`days_until_contract_end` (specifically its interaction with cancellation requests)
```

Why this, above all else?

1. Highest Risk of Subtle Leakage: These features are inherently "closer" to the churn event itself. ``cancel_request_submitted_last_X_d`` directly signals an intent to churn, and ``account_status_is_churned`` is the churn event. While my initial assessment identified potential leakage risks and suggested careful handling, the implementation details in the actual data pipeline are where subtle leakage can still creep in.
2. Devastating Impact of Leakage: If leakage exists in these features, the model will appear to perform exceptionally well during training and validation, but will completely fall apart in production. This leads to false confidence, wasted resources (acting on "at-risk" customers who have already decided to leave or are already gone), and a significant loss of trust in the data science team. It's the "silent killer" of predictive models.
3. Complexity of Time-Series Data: Ensuring that all data points for these features are truly recorded before the prediction timestamp, and that no future event (like a cancellation request made after the prediction time but before the end of the 30-day forecast window) inadvertently influences them, requires meticulous and often manual verification of timestamps and data sources.
4. Assumptions vs. Reality: I've made assumptions about how these features are derived (e.g., "Assumed this is purely historical up to prediction time"). A second pair of eyes can challenge these assumptions against the actual data engineering implementation.

What I'd ask the second person to do:

"I need you to act as our 'leakage skeptic.' Take our current backtesting dataset, pick 5-10 random customer-prediction_time rows, especially some where churn did occur within 30 days and some where it didn't. For each of the high-risk features identified (``cancel_request_submitted_last_7d``, ``cancel_request_submitted_last_30d``, ``account_status_is_churned``, and how ``days_until_contract_end`` behaves around a cancellation), you need to manually trace the exact data sources and timestamps that generated those feature values. Prove to me that for a prediction made on YYYY-MM-DD HH:MM:SS, absolutely no information from YYYY-MM-DD HH:MM:SS + 1 second onwards influenced that feature value. I want to see the database query, the join conditions, and the timestamp filters. Be ruthless."

Engineering counterpart

3:30 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.