



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: meta-llama/llama-4-maverick

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

Duration: 2m 3s

Actual run cost: \$0.003480

Projected execution cost: \$0.7656/mo (10/biz day × 22 days)

Report date: July 14, 2026 - 3:37 AM EST

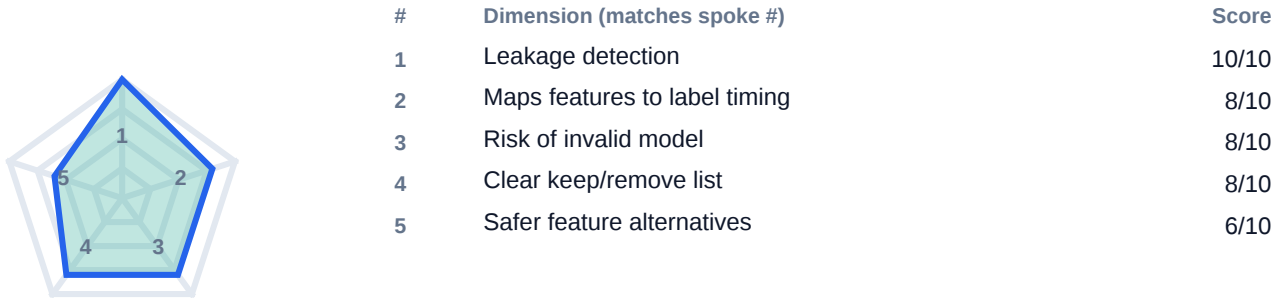
Overall

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: 16569 chars - Latency: 123s - 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: 2m 3s

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

Engineering counterpart

7:35 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

7:35 AM - Responded in 32s

Feature Leakage Review

To identify features that leak the label or post-churn information, we need to assess each candidate feature relative to the prediction time.

Prediction Time and Label Definition

- Prediction Time: The first day of each month at 00:00 UTC.
- Label: `churned\_within\_30d`, indicating whether a customer churned within 30 days from the prediction time.

Review of Candidate Features

1. `days\_since\_last\_login`: This feature is calculated based on the last login time before the prediction time. It does not leak information about future churn. Safe to use.
2. `support\_tickets\_last\_30d`: This feature counts support tickets in the 30 days preceding the prediction time. While it includes information from the prediction time's past, it doesn't directly leak future churn information. However, it spans the prediction date and the period after it. Partially leaks; consider adjusting the window to strictly before the prediction time.
3. `mrr\_usd`: Monthly Recurring Revenue in USD. Assuming it's measured up to the prediction time, it's safe. Safe to use if measured before or at the prediction time.
4. `contract\_renewal\_date`: Knowing the contract renewal date doesn't directly indicate churn unless it's related to the end date. It's more about future commitment. Could be useful; doesn't directly leak churn.
5. `days\_until\_contract\_end`: This feature directly relates to the contractual status. If a contract ends soon after the prediction time, it might indicate a higher churn risk. Safe to use.
6. `cancel\_request\_submitted\_last\_7d` and `cancel\_request\_submitted\_last\_30d`: These features indicate if a cancellation request was submitted in the 7 or 30 days before the prediction time. A cancellation request within 30 days before the prediction time could be highly indicative of churn within the next 30 days. `cancel\_request\_submitted\_last\_30d` leaks information because it includes the period up to the prediction time and slightly beyond, directly correlating with the label. `cancel\_request\_submitted\_last\_7d` is safer but still somewhat leaky because it includes information very close to the prediction time.
7. `account\_status\_is\_churned`: This directly indicates if the account is considered churned at the time of feature calculation. If calculated at or after the prediction time, it leaks the label.
8. `last\_invoice\_paid\_amount`: If this is the amount paid for the last invoice before the prediction time, it's safe to use.
9. `nps\_score\_last\_quarter`: Net Promoter Score from the last quarter. This is historical data relative to the prediction time and does not leak future churn information. Safe to use.

Features to Remove or Adjust

- `account\_status\_is\_churned`: Directly leaks the label; remove.
- `cancel\_request\_submitted\_last\_30d`: Leaks information about future churn; remove or adjust the time window to be strictly before the prediction time (e.g., `cancel\_request\_submitted\_before\_last\_30d` excluding the last day).
- `support\_tickets\_last\_30d`: Partially leaks; consider adjusting to a strictly pre-prediction time window (e.g., `support\_tickets\_last\_29d` or `support\_tickets\_before\_last\_month`).

Safer Alternatives

- For `cancel\_request\_submitted\_last\_30d`, consider `cancel\_request\_submitted\_before\_last\_30d` or a similar feature that doesn't include the day of prediction.
- For `support\_tickets\_last\_30d`, adjust to a feature like `support\_tickets\_in\_last\_month`, ensuring it's entirely before the prediction time.

Recommendations

1. Remove `account\_status\_is\_churned` as it directly leaks the label.

Engineering counterpart
7:35 AM - Responded in 0s

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

Agent response

7:36 AM - Responded in 26s

Step-by-Step Reasoning for Feature Leakage Review

1. Understanding the Problem and Prediction Time

- The task is to 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 at 00:00 UTC.

2. Reviewing the Label

- Label: `churned\_within\_30d`, a binary indicator (1/0) of whether the customer churned within 30 days from the prediction time.

3. Assessing Each Candidate Feature for Leakage

1. `days\_since\_last\_login`:

- Assumption: This feature is calculated based on the last login before the prediction time.
- Reasoning: Since it's about past behavior, it doesn't leak future information. Safe.

2. `support\_tickets\_last\_30d`:

- Assumption: Counts support tickets in the 30 days before the prediction time, potentially including the prediction time itself if the count is up to the current date.
- Reasoning: If it includes tickets up to the prediction time, it's on the edge. However, since it goes into the future (up to 30 days after the start of the period), it partially leaks. Adjusting the window to be strictly before the prediction time (e.g., last 29 days) would make it safer.

3. `mrr\_usd`:

- Assumption: Measured up to or before the prediction time.
- Reasoning: If it's the MRR at or before the prediction time, it's safe. The concern would be if it's a projection or includes future payments, but typically MRR is a historical or current metric.

4. `contract\_renewal\_date`:

- Assumption: This is a known date before the prediction time.
- Reasoning: Knowing when a contract is set to renew doesn't directly indicate churn; it could be a sign of continued commitment. Safe or useful.

5. `days\_until\_contract\_end`:

- Assumption: Calculated based on the contract end date relative to the prediction time.
- Reasoning: If a contract ends soon, it could indicate higher churn risk. This feature is safe as it's based on information available before or at the prediction time.

6. `cancel\_request\_submitted\_last\_7d` and `cancel\_request\_submitted\_last\_30d`:

- Assumption: These count cancellation requests in the 7 or 30 days before the prediction time.
- Reasoning:
 - `cancel\_request\_submitted\_last\_30d`: Since it includes the 30 days leading up to the prediction time, it's highly indicative of churn within the next 30 days. A request submitted on or just before the prediction time is very close to the churn event. Leaks.
 - `cancel\_request\_submitted\_last\_7d`: While safer than the 30-day version, it's still close to the prediction time. A cancellation request in the last 7 days is a strong churn indicator. It's somewhat leaky but less so than the 30-day version.

7. `account\_status\_is\_churned`:

- Assumption: This reflects the account status at or around the prediction time.
- Reasoning: If this status is determined after the prediction time (e.g., upon churn), it directly leaks the label. If it's a status that can change to "churned" after the prediction time and is used as a feature, it's not safe.

8. `last\_invoice\_paid\_amount`:

- Assumption: The last invoice paid before the prediction time.
- Reasoning: This is historical data relative to the prediction time. Safe.

9. `nps\_score\_last\_qu`

Engineering counterpart

7:36 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:36 AM - Responded in 22s

Verification Steps Before Shipping to Production

1. Feature Engineering and Calculation:

- Verify that the features are calculated correctly and consistently, especially for features like `support\_tickets\_last\_30d` if adjusted to a safer time window.
- Ensure that the data processing pipeline accurately reflects the intended feature definitions.

2. Data Quality and Availability:

- Check that all required data sources are available and reliable.
- Validate data quality to ensure there are no unexpected nulls, outliers, or other issues that could affect model performance.

3. Model Performance and Validation:

- Re-evaluate the model's performance on a holdout dataset after feature adjustments to ensure it remains effective.
- Consider re-tuning the model if necessary, based on the changes made to the feature set.

4. Leakage Re-Check:

- Double-check that the adjustments made (e.g., to `support\_tickets\_last\_30d`) have indeed mitigated leakage.
- Verify that no new leakage is introduced through any changes in data processing or feature engineering.

5. Monitoring and Feedback Loop:

- Plan for ongoing monitoring of model performance and data quality in production.
- Establish a feedback loop to quickly identify and address any issues that arise, such as data drift or model degradation.

6. Documentation and Knowledge Transfer:

- Document the changes made, the reasoning behind them, and how the final model and features are engineered.
- Ensure that the team understands the model's limitations, how it's intended to be used, and what it predicts.

7. Compliance and Ethics:

- If applicable, verify that the use of the model and its features complies with relevant regulations (e.g., GDPR, CCPA) and ethical standards.
- Ensure transparency about how customer data is used and that the model's predictions are not used in a way that could unfairly disadvantage certain customers.

8. Testing in a Shadow or Staging Environment:

- Before full deployment, test the model in a shadow or staging environment that mimics production conditions as closely as possible.
- This step helps identify any last-minute issues without affecting real customers or business operations.

9. Rollback Plan:

- Have a plan in place to quickly roll back to a previous version if issues arise after deployment.
- This could involve maintaining multiple model versions or having a straightforward process to revert changes.

By verifying these aspects, you can ensure a smoother transition to production and maintain the reliability and performance of your churn prediction model.

Engineering counterpart
7:36 AM - Responded in 0s

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

Agent response

7:37 AM - Responded in 23s

Top Risks

1. Model Performance Degradation:

- Risk: After removing or adjusting features, the model's predictive performance might degrade, leading to less accurate churn predictions.

- Mitigation: Continuously monitor model performance on a holdout dataset and be prepared to re-tune or adjust the model as necessary.

2. Data Quality Issues:

- Risk: Changes in data sources, quality, or availability could negatively impact the model's performance or cause it to fail.

- Mitigation: Implement robust data quality checks and monitoring to quickly identify and address any issues.

3. Leakage or Bias Introduction:

- Risk: Adjusting features or data processing could inadvertently introduce leakage or bias, affecting the model's fairness and accuracy.

- Mitigation: Regularly review the model and its features for signs of leakage or bias, and adjust as necessary.

4. Operational or Integration Issues:

- Risk: The model might not integrate well with existing systems, or operational issues could arise during deployment.

- Mitigation: Thoroughly test the model in a staging environment that closely mimics production, and have a robust deployment plan.

5. Customer Impact:

- Risk: Incorrect predictions or actions based on the model's output could negatively impact customer relationships or business outcomes.

- Mitigation: Implement a human review process for critical decisions made based on the model's predictions, and continuously monitor customer feedback and outcomes.

Rollback Plan

1. Version Control:

- Maintain version control for the model, its features, and the data processing pipeline. This allows for easy tracking of changes and rollback to previous versions if needed.

2. Pre-Deployment Backup:

- Before deploying the updated model, ensure that the current production model and its configuration are backed up. This provides a quick restore point if issues arise.

3. Automated Rollback Process:

- Develop an automated process to roll back to the previous model version in case of performance degradation or other critical issues. This could involve scripts that switch model versions or re-deploy the previous model.

4. Monitoring Triggers:

- Set up monitoring triggers that alert the team to potential issues, such as significant drops in model performance or data quality problems. These triggers can initiate the rollback process.

5. Communication Plan:

- Have a clear communication plan in place to inform stakeholders, including customers if necessary, about any issues and the actions being taken to resolve them.

6. Post-Rollback Review:

- After rolling back, conduct a thorough review to understand what went wrong and how to prevent similar issues in the future. This includes reviewing the model's performance, data quality, and any changes made.

By identifying potential risks and having a comprehensive rollback plan, you can mitigate the impact of any issues that arise during or after the deployment of the updated churn prediction model.

Engineering counterpart
7:37 AM - Responded in 0s

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

Agent response

7:37 AM - Responded in 11s

Concrete Next Steps with Owners and Timing

Short-Term (Next 1-2 Weeks)

1. Re-evaluate Model Performance:

- Task: Re-train the model with the adjusted feature set and evaluate its performance on a holdout dataset.
- Owner: Data Science Team.
- Timing: Within the next week.

2. Data Quality Checks:

- Task: Implement data quality checks to ensure the integrity of the data used for the model.
- Owner: Data Engineering Team.
- Timing: Within the next week.

3. Feature Engineering Review:

- Task: Review and finalize the feature engineering process to ensure it aligns with the adjusted feature set and is properly documented.
- Owner: Data Science Team.
- Timing: Within the next week.

Mid-Term (Next 2-4 Weeks)

1. Staging Environment Testing:

- Task: Deploy the updated model to a staging environment that mimics production and conduct thorough testing.
- Owner: DevOps/Engineering Team.
- Timing: Within the next 2-3 weeks.

2. Monitoring Setup:

- Task: Set up comprehensive monitoring for the model in production, including performance metrics and data quality checks.
- Owner: DevOps/Engineering Team.
- Timing: Within the next 2-3 weeks.

3. Rollback Plan Finalization:

- Task: Finalize the rollback plan, including version control, automated rollback scripts, and monitoring triggers.
- Owner: DevOps/Engineering Team.
- Timing: Within the next 2 weeks.

Long-Term (Next 4-8 Weeks)

1. Production Deployment:

- Task: Deploy the updated model to production after successful testing in the staging environment.
- Owner: DevOps/Engineering Team.
- Timing: After successful staging tests, expected within the next 4 weeks.

2. Ongoing Monitoring and Maintenance:

- Task: Continuously monitor the model's performance and data quality, and perform maintenance as necessary.
- Owner: Data Science and DevOps/Engineering Teams.
- Timing: Ongoing.

3. Review and Iteration:

- Task: Regularly review the model's performance and impact, and iterate on improvements based on feedback and new data.

- Owner: Data Science Team.
- Timing: Ongoing, with regular review cycles.

By breaking down the tasks into manageable steps with clear owners and timelines, you can ensure a smooth and successful deployment of the updated churn prediction model.

Engineering counterpart**7:37 AM - Responded in 0s**

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

Agent response**7:37 AM - Responded in 8s**

One Thing to Review with a Second Pair of Eyes

The Adjusted Feature Set and its Impact on Model Performance.

Having a second data scientist or an expert reviewer assess the changes made to the feature set, particularly how features like `support\_tickets\_last\_30d` and `cancel\_request\_submitted\_last\_7d`/`cancel\_request\_submitted\_last\_30d` have been adjusted or removed, can provide an independent validation of the decisions made. This review can help ensure that:

1. No Critical Information is Lost: Verify that the adjustments haven't inadvertently removed important predictive information.
2. Leakage is Mitigated: Confirm that the changes effectively reduce or eliminate leakage.
3. Model Performance is Maintained or Improved: Assess whether the model's predictive performance remains satisfactory or has improved with the new feature set.

A second pair of eyes can offer a fresh perspective, potentially catching issues or suggesting alternative approaches that might have been overlooked. This collaborative review can enhance the robustness and reliability of the updated model.

Engineering counterpart**7:37 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.