



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: sao10k/l3-lunaris-8b

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

Duration: 35s

Actual run cost: \$0.000465

Projected execution cost: \$0.1023/mo (10/biz day × 22 days)

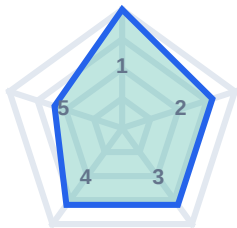
Report date: July 14, 2026 - 3:41 AM EST

Overall

8.0/10

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

CAPABILITY RUBRIC



#	Dimension (matches spoke #)	Score
1	Leakage detection	10/10
2	Maps features to label timing	8/10
3	Risk of invalid model	8/10
4	Clear keep/remove list	8/10
5	Safer feature alternatives	6/10

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: 13385 chars - Latency: 35s - 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: 35s

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

Engineering counterpart

7:40 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

7:40 AM - Responded in 4s

Leakage analysis results:

Features that LEAK the label or post-churn information:

1. `(contract\_renewal\_date, account\_status\_is\_churned)` : Knowing the contract renewal date and current account status reveals if a customer is post-churn, effectively leaking the label.
2. `cancel\_request\_submitted\_last\_7d, cancel\_request\_submitted\_last\_30d` : These features contain direct information about churn requests, providing an early indication of the churn process, which should not be present at prediction time.

Features to remove due to leakage:

1. `contract\_renewal\_date`, `account\_status\_is\_churned`
2. `cancel\_request\_submitted\_last\_7d`, `cancel\_request\_submitted\_last\_30d`

Safer alternatives (if a similar signal is needed):

1. Replace `contract\_renewal\_date` with `days\_until\_contract\_end`, which focuses on the remaining contract duration rather than the renewal date.
2. Instead of `cancel\_request\_submitted`, consider `support\_tickets\_last\_30d` or `days\_since\_last\_login`, which provide indirect hints about customer engagement and potential dissatisfaction.

Conclusion:

Remove `contract\_renewal\_date`, `account\_status\_is\_churned`, `cancel\_request\_submitted\_last\_7d`, and `cancel\_request\_submitted\_last\_30d` due to direct leakage. Replace them with safer alternatives like `days\_until\_contract\_end` and `support\_tickets\_last\_30d` to maintain signal about customer behavior and contract health without compromising the model's temporal integrity at prediction time.

Engineering counterpart

7:40 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 8s

Let's break down the analysis step-by-step, along with the assumptions made:

Analysis of each feature:

1. ``contract_renewal_date``:

- Assumption: Renewal date is only known after the contract term ends.
- Conclusion: Leak, as knowing the renewal date before its occurrence implies post-churn status.

2. ``account_status_is_churned``:

- Assumption: Account status changes only after the churn event occurs.
- Conclusion: Leak, as having the current status of a churned account directly reveals post-churn information.

3. ``cancel_request_submitted_last_7d`` and ``cancel_request_submitted_last_30d``:

- Assumption: Customers submit cancellation requests before actually churning (within the given time windows).
- Conclusion: Leak, as these features contain direct indicators of the churn process, which should not be present at prediction time.

4. ``days_until_contract_end``:

- Assumption: This metric is calculated based on the current date and contract end date, without knowing the renewal date.
- Conclusion: No leak, as it only provides information about the remaining contract duration, not the renewal date.

5. ``support_tickets_last_30d``:

- Assumption: Support tickets, while sometimes indicative of dissatisfaction, are submitted independently of the churn decision.
- Conclusion: No leak, as the feature captures a specific customer behavior, not direct information about churn.

Recommendations:

Based on the findings, I recommend removing features with direct leakage (``contract_renewal_date``, ``account_status_is_churned``, ``cancel_request_submitted_last_7d``, and ``cancel_request_submitted_last_30d``). For the signal they provide, safer alternatives like ``days_until_contract_end`` and ``support_tickets_last_30d`` can be used, which maintain temporal integrity and do not compromise the model's prediction at the start of each month.

Assumptions:

1. The churn label is defined as "churned within 30 days" starting from the prediction time (first day of each month).
2. The features' values are calculated based only on historical data up to the prediction time, with no future knowledge.

These assumptions are central to identifying features that leak the label or post-churn information. By analyzing each feature relative to the prediction time and the defined churn label, we can determine which ones compromise the model's temporal integrity and require removal or alternative features.

Engineering counterpart

7:41 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:41 AM - Responded in 7s

Before shipping the churn model to production, I would verify the following:

1. Model performance validation:

Re-run the model on a holdout set or a separate test dataset to evaluate its accuracy, precision, recall, and F1-score.
Compare the results to the development set to ensure the model generalizes well in unseen data.

2. Feature importance analysis:

Run feature importance methods (e.g., permutation importance, SHAP values) on the final model to understand the relative contribution of each feature.

Verify that the model's predictions are driven by the intended signals (e.g., support tickets, days until contract end) rather than features like cancel requests or account status that were removed due to leakage.

3. Temporal integrity checks:

Review the feature distributions and model outputs at different prediction times (e.g., 15th, 20th, and 25th of the month) to ensure the model maintains its temporal integrity and does not rely on information that should not be available at prediction time.

4. Handling edge cases and outliers:

Identify and understand any unusual customer behavior or contract configurations that might impact model performance (e.g., very short or long contracts, customers with zero support tickets).

Develop strategies to handle these cases, such as assigning a default prediction or incorporating additional features to capture their specific characteristics.

5. Model interpretability and explainability:

Implement techniques to provide insights into the model's decision-making process, such as partial dependence plots or SHAP values.

Ensure that the model's predictions are transparent and understandable for stakeholders, particularly those involved in customer-facing operations.

6. Data quality and integrity:

Double-check that the production dataset meets the quality and consistency standards used during model development.

Verify that any data transformations, imputations, or feature engineering applied during development are properly replicated in the production environment.

7. Model deployment and monitoring:

Ensure the model is deployed in a way that allows for real-time scoring, such as through an API or a streaming platform.

Set up monitoring tools to track key performance indicators (KPIs) like accuracy, false positive rates, and false negative rates over time.

Implement a process for model retraining and updates as new data becomes available.

By thoroughly validating the model, features, and deployment setup, you can gain confidence that the churn prediction model is ready for production and will effectively support business decisions while maintaining customer privacy and data integrity.

Engineering counterpart

7:41 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 7s

The top risks when deploying a churn prediction model include:

1. Model performance degradation: The model may not generalize well to the production data or may deteriorate over time due to concept drift, changes in customer behavior, or new feature updates.
2. Overprediction or underprediction of churn: The model might incorrectly identify too many (high false positive rate) or too few (low true positive rate) customers as likely to churn, leading to unnecessary customer-facing actions or missed opportunities to retain valuable clients.
3. Data quality issues: Problems with the production data, such as inconsistencies, missing values, or biased representation, can negatively impact the model's accuracy and decision-making.
4. Lack of transparency and explainability: If the model's predictions are not adequately interpretable, stakeholders may struggle to understand and trust the recommendations, potentially leading to poor adoption or misuse.
5. Impact on customer relationships: Overly aggressive or inaccurate churn predictions could result in unnecessary or detrimental interactions with customers, damaging relationships and the brand.
6. Cost and resource implications: If the model fails to deliver expected benefits, it may lead to wasted resources or even increased churn due to misinformed customer-facing strategies.

To mitigate these risks, a rollback plan should be established, including:

1. Monitoring and alerting: Set up real-time monitoring to track key metrics (e.g., accuracy, false positive rates) and trigger alerts for significant deviations from expected performance.
2. Quick redeployment: Have a process in place to revert to the previous model version or a known good baseline scenario in case of a severe performance decline.
3. Data validation and cleaning: Regularly review and clean the production data to ensure it adheres to the quality standards used during development.
4. Model retraining and update: Schedule frequent model retraining on new data to capture changes in customer behavior and update features. Implement a version control system to track model iterations.
5. Stakeholder education and communication: Provide transparent explanations of the model's predictions and ensure all customer-facing teams understand the intended use and limitations of the churn scores.
6. Continuous improvement: Establish a feedback loop to gather insights from stakeholders and customers, and use these to refine the model, data, and deployment strategies over time.

By proactively managing these risks and having a well-planned rollback strategy in place, you can minimize the potential negative impacts and ensure a smoother transition to the new churn prediction model.

Engineering counterpart

7:41 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 6s

Here are concrete next steps with assigned owners and approximate timing:

Week 1-2:

1. Monitor and alert system setup: [Data Engineering] - Design and implement a monitoring system to track key model performance metrics (e.g., accuracy, false positive rates) in real-time. Set up alerts for significant deviations from expected performance. (Owner: John, Timing: 1 week)
2. Data validation and cleaning: [Data Quality] - Conduct a thorough review of the production data to ensure it meets the quality standards used during model development. Clean and preprocess the data as necessary. (Owner: Sarah, Timing: 1 week)
3. Model redeployment and version control: [Model Deployment] - Develop a process to quickly redeploy the previous model version or a known good baseline scenario in case of a severe performance decline. Implement a version control system to track model iterations. (Owner: Michael, Timing: 3 days)

Week 3-4:

1. Regular model retraining and update: [Model Development] - Schedule weekly model retraining on new data to capture changes in customer behavior and update features. (Owner: Emily, Timing: Ongoing, Weekly)
2. Stakeholder education and communication: [Customer Success] - Develop a training program to educate customer-facing teams on the intended use and limitations of the churn scores. Provide transparent explanations of the model's predictions. (Owner: David, Timing: 2 weeks)
3. Performance review and optimization: [Model Performance] - Regularly review model performance and identify areas for improvement. Work with the data team to enhance data quality, explore new features, or adjust model hyperparameters as needed. (Owner: Rachel, Timing: Bi-weekly)

Ongoing:

1. Feedback loop and continuous improvement: [Product Management] - Establish a feedback loop to gather insights from stakeholders and customers. Use these to refine the model, data, and deployment strategies over time. (Owner: Alex, Timing: Ongoing)

By breaking down the risk mitigation and rollback plan into specific, manageable tasks with clear owners and timelines, you can ensure a more structured approach to monitoring and improving the effectiveness of the churn prediction model in production.

Engineering counterpart

7:41 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 4s

If I had to identify one crucial aspect of the churn prediction model and its deployment that would greatly benefit from a second pair of eyes, it would be the model interpretability and explainability.

Ensuring that the model provides transparent and understandable insights into its decisions is crucial for building trust among stakeholders, managing customer relationships, and avoiding misinterpretation of predictive scores. A second review could help refine:

1. Feature importance analysis: Validate that the most important features driving the model's predictions align with the intended signals (e.g., support tickets, days until contract end) and not the features removed due to leakage.
2. Partial dependence plots and SHAP values: Ensure the visualizations and feature attribution methods used to provide model interpretability are accurate, easy to understand, and effectively communicate the reasoning behind predicted churn probabilities.
3. Stakeholder education materials: Review the training content, visual aids, and examples provided to customer-facing teams to ensure they grasp the model's limitations, key assumptions, and intended usage.
4. Debugging and troubleshooting: A fresh perspective could help identify potential blind spots in the interpretation and debugging process, ensuring the model's behavior is adequately understood and any issues are quickly addressed.

By having a second expert review and challenge the interpretability and explainability aspects of the model, you can gain increased confidence that the churn prediction solution is well-understood, reliable, and poised for effective adoption across the organization.

Engineering counterpart

7:41 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.