



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-3.3-70b-instruct

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

Duration: 3m 52s

Actual run cost: \$0.002010

Projected execution cost: \$0.4422/mo (10/biz day × 22 days)

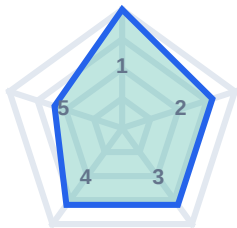
Report date: July 14, 2026 - 3:37 AM EST

Overall

8.0/10

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

CAPABILITY RUBRIC



#	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: 16517 chars - Latency: 232s - 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: 3m 52s

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

Engineering counterpart

7:33 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

7:34 AM - Responded in 1m 1s

Feature Leakage Review

To ensure the accuracy and reliability of the churn model, it's essential to identify and remove features that leak information about the label or post-churn events. Based on the provided candidate features, the following features are suspected to leak information:

Leaking Features

1. `account_status_is_churned`: This feature directly indicates whether an account has been churned, which is the target label. At the prediction time (first day of each month), this feature would not be available for the current month, as it would be based on future information.
2. `cancel_request_submitted_last_7d` and `cancel_request_submitted_last_30d`: Although these features seem relevant, they might leak information about the churn event. If a customer submits a cancellation request, it's likely that they will churn soon. However, if these features are based on data from the past 7 or 30 days, they might include requests submitted after the prediction time, which would be leaking information about the label.
3. `days_until_contract_end`: While this feature seems relevant, it might leak information if the contract end date is used as a proxy for churn. If a contract is about to end, and the customer hasn't renewed, it could indicate a higher likelihood of churn. However, if the contract end date is in the future (beyond the 30-day prediction window), it might not be a strong indicator of churn.

Recommended Removal

Based on the analysis, it's recommended to remove the following features to avoid leakage:

`account_status_is_churned`
`cancel_request_submitted_last_7d` and `cancel_request_submitted_last_30d` (or modify them to only include requests submitted before the prediction time)

Safer Alternatives

To replace the removed features, consider the following safer alternatives:

Instead of `cancel_request_submitted_last_7d` and `cancel_request_submitted_last_30d`, use `cancel_request_submitted_before_prediction_time` to capture cancellation requests submitted before the prediction time.

For `days_until_contract_end`, consider using `contract_renewal_status` (e.g., "renewed", "pending", "not renewed") or `days_since_contract_renewal` to capture information about the contract renewal process without leaking information about the contract end date.

Updated Feature List

After removing the leaking features, the updated candidate feature list would be:

```
```json
[
 "days_since_last_login",
 "support_tickets_last_30d",
 "mrr_usd",
 "contract_renewal_date",
 "last_invoice_paid_amount",
 "nps_score_last_quarter",
 "cancel_request_submitted_before_prediction_time",
 "contract_renewal_status",
 "days_since_contract_renewal"
]
```

] ...

By removing the leaking features and using safer alternatives, the model can focus on predicting churn based on relevant, non-leaking information, ensuring a more accurate and reliable prediction.

**Engineering counterpart**  
**7:34 AM - Responded in 0s**

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

## Agent response

7:35 AM - Responded in 28s

Here's a step-by-step breakdown of my reasoning:

### Step 1: Understanding the problem and prediction time

The goal is to predict whether a B2B SaaS customer will churn within 30 days, starting from the first day of each calendar month (00:00 UTC).

I assumed that the prediction model will be trained on historical data and will make predictions for future time periods, using features that are available up to the prediction time.

### Step 2: Identifying potential leakage

I examined each feature in the `candidate\_features` list to determine whether it could potentially leak information about the label (`churned\_within\_30d`) or post-churn events.

I considered the following types of leakage:

- + Label leakage: Features that directly indicate whether a customer has churned.
- + Temporal leakage: Features that include information from the future, which would not be available at the prediction time.

### Step 3: Analyzing each feature for leakage

#### 1. account\_status\_is\_churned:

I assumed that this feature is updated in real-time, reflecting the current status of the account.

Since the prediction time is the first day of each month, this feature would likely include information about churn events that have already occurred, making it a label leakage.

Conclusion: This feature leaks information about the label and should be removed.

#### 2. cancel\_request\_submitted\_last\_7d and cancel\_request\_submitted\_last\_30d:

I assumed that these features capture cancellation requests submitted within the specified time windows (7 days or 30 days).

If these features include requests submitted after the prediction time, they would leak information about the label.

Conclusion: These features may leak information about the label, depending on how they are implemented. To avoid leakage, they should be modified or removed.

#### 3. days\_until\_contract\_end:

I assumed that this feature reflects the number of days until the contract is scheduled to end.

If the contract end date is in the future (beyond the 30-day prediction window), this feature might not be a strong indicator of churn.

However, if the contract end date is used as a proxy for churn, this feature could potentially leak information about the label.

Conclusion: This feature may leak information about the label, depending on how it is used. To avoid leakage, it's recommended to use alternative features that capture contract renewal status or history.

### Step 4: Recommending removal and safer alternatives

Based on the analysis, I recommended removing features that leak information about the label or post-churn events.

I suggested safer alternatives, such as:

- + cancel\_request\_submitted\_before\_prediction\_time: This feature captures cancellation requests submitted before the prediction time, avoiding temporal leakage.
- + contract\_renewal\_status or days\_since\_contract\_renewal: These features capture information about the contract renewal process without leaking information about the contract end date.

### Assumptions made

I assumed that the prediction model will be trained on historical data and will make predictions for future time periods.

I assumed that the features are updated in real-time, reflecting the current status of the accounts.

I assumed that the contract end date is used as a proxy for churn, which might not always be the case.



I assumed that the cancellation request features include requests submitted after the prediction time, which might not always be the case.

By walking through each feature and considering potential leakage, I aimed to provide a thorough analysis and recommendations for removing leaking features and using safer alternatives.

What would you verify before shipping this to production?

**Agent response**

**7:35 AM - Responded in 8s**

Before shipping the updated feature set to production, I would verify the following:

### Data Quality and Integrity

1. Feature computation: Verify that the computation of each feature is correct and consistent with the expected behavior.
2. Data types and formats: Ensure that the data types and formats of the features match the expected types and formats.
3. Missing values: Check for missing values in the features and ensure that the imputation strategy is correct and consistent.
4. Data distribution: Verify that the distribution of the features is as expected and that there are no unexpected outliers or anomalies.

### Leakage and Temporal Consistency

1. Feature leakage: Re-verify that the updated feature set does not leak information about the label or post-churn events.
2. Temporal consistency: Ensure that the features are computed using only data that is available up to the prediction time (first day of each month).
3. Time-zone consistency: Verify that all date and time-related features are computed using a consistent time zone (e.g., UTC).

### Model Performance and Fairness

1. Model re-training: Re-train the model using the updated feature set and verify that the performance metrics (e.g., accuracy, precision, recall, F1-score) are acceptable.
2. Model fairness: Evaluate the model's fairness and ensure that it does not exhibit any unexpected biases or disparities.
3. Feature importance: Analyze the feature importance and ensure that the updated feature set is contributing to the model's performance as expected.

### Deployment and Monitoring

1. Deployment pipeline: Verify that the deployment pipeline is correct and that the updated feature set is being used in production.
2. Monitoring and logging: Ensure that the model's performance and feature computation are being monitored and logged correctly.
3. Alerting and notification: Set up alerting and notification mechanisms to detect any unexpected changes in the model's performance or feature computation.

### Stakeholder Sign-off

1. Business stakeholder review: Obtain review and approval from business stakeholders to ensure that the updated feature set aligns with business requirements and expectations.
2. Data science team review: Obtain review and approval from the data science team to ensure that the updated feature set is correct and consistent with data science best practices.

By verifying these aspects, you can ensure that the updated feature set is correct, consistent, and fair, and that the model is performing as expected in production.

**Engineering counterpart**  
**7:35 AM - Responded in 0s**

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

## Agent response

7:35 AM - Responded in 34s

### ### Top Risks

1. Model performance degradation: The updated feature set may not perform as well as the original feature set, leading to a decrease in model accuracy and effectiveness.
2. Leakage reintroduction: The updated feature set may still contain leakage, which could lead to overfitting and poor model generalization.
3. Data quality issues: The updated feature set may be affected by data quality issues, such as missing values, outliers, or incorrect data types.
4. Unintended consequences: The updated feature set may have unintended consequences, such as changes to the model's fairness or interpretability.
5. Deployment issues: The deployment of the updated feature set may be problematic, leading to errors, downtime, or other issues.

### ### Rollback Plan

If the updated feature set does not work as expected, the following rollback plan can be executed:

1. Monitoring and detection: Continuously monitor the model's performance and feature computation to detect any issues or degradation.
2. Root cause analysis: Perform a root cause analysis to identify the source of the problem and determine the best course of action.
3. Reversion to previous feature set: Revert to the previous feature set and re-deploy the model with the original features.
4. Feature set refinement: Refine the updated feature set to address any issues or problems that were identified during the root cause analysis.
5. Re-deployment: Re-deploy the model with the refined feature set and continue to monitor its performance.

### ### Rollback Triggers

The following triggers can be used to determine when to roll back to the previous feature set:

1. Model performance threshold: If the model's performance falls below a certain threshold (e.g., 5% decrease in accuracy).
2. Error rate threshold: If the error rate exceeds a certain threshold (e.g., 10% increase in errors).
3. Data quality issues: If data quality issues are detected, such as missing values, outliers, or incorrect data types.
4. Unintended consequences: If unintended consequences are detected, such as changes to the model's fairness or interpretability.
5. Stakeholder feedback: If stakeholders provide feedback indicating that the updated feature set is not meeting their needs or expectations.

### ### Rollback Timeline

The rollback plan can be executed on the following timeline:

1. Immediate rollback: If a critical issue is detected, the rollback can be executed immediately.
2. Short-term rollback: If a non-critical issue is detected, the rollback can be executed within a short-term timeframe (e.g., 1-2 weeks).
3. Long-term refinement: If the issue requires refinement of the feature set, the rollback can be executed within a longer-term timeframe (e.g., 2-6 weeks).

By having a clear rollback plan in place, you can minimize the risks associated with deploying an updated feature set and ensure that the model continues to perform effectively.

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

#### Agent response

**7:37 AM - Responded in 1m 38s**

Here are the concrete next steps with owners and timing:

##### ### Next Steps

1. Verify feature computation and data quality (Owner: Data Scientist, Timing: 2 days)  
Review feature computation code and verify that it is correct and consistent with the expected behavior.  
Check for missing values, outliers, and incorrect data types in the feature data.
2. Re-train and reEvaluate the model (Owner: Data Scientist, Timing: 3 days)  
Re-train the model using the updated feature set and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score.  
Compare the performance of the updated model with the original model to ensure that it is not degraded.
3. Conduct root cause analysis and refine the feature set (if necessary) (Owner: Data Scientist, Timing: 2-5 days)  
If the updated model's performance is degraded, conduct a root cause analysis to identify the source of the problem.  
Refine the feature set to address any issues or problems that were identified during the root cause analysis.
4. Obtain stakeholder review and approval (Owner: Product Owner, Timing: 2 days)  
Obtain review and approval from business stakeholders to ensure that the updated feature set aligns with business requirements and expectations.
5. Deploy the updated model to production (Owner: Engineering Team, Timing: 2 days)  
Deploy the updated model to production and configure it to use the updated feature set.  
Monitor the model's performance and feature computation to detect any issues or errors.
6. Monitor and evaluate the updated model's performance (Owner: Data Scientist, Timing: Ongoing)  
Continuously monitor the updated model's performance and feature computation to detect any issues or errors.  
Evaluate the updated model's performance using metrics such as accuracy, precision, recall, and F1-score, and compare it with the original model's performance.

##### ### Rollback Plan

If the updated model's performance is degraded or if issues are detected, the following rollback plan will be executed:

1. Immediate rollback (Owner: Engineering Team, Timing: 1 day)  
If a critical issue is detected, the updated model will be rolled back to the original model immediately.
2. Short-term rollback (Owner: Engineering Team, Timing: 2-3 days)  
If a non-critical issue is detected, the updated model will be rolled back to the original model within a short-term timeframe (e.g., 2-3 days).
3. Long-term refinement (Owner: Data Scientist, Timing: 2-6 weeks)  
If the issue requires refinement of the feature set, the updated model will be rolled back to the original model, and the feature set will be refined within a longer-term timeframe (e.g., 2-6 weeks).

##### ### Communication Plan

The following communication plan will be used to keep stakeholders informed:

1. Regular updates (Owner: Data Scientist, Timing: Weekly)  
Provide regular updates to stakeholders on the progress of the updated feature set and model deployment.
2. Issue escalation (Owner: Data Scientist, Timing: As needed)  
Escalate any issues or errors detected during the deployment and monitoring of the updated model to stakeholders.
3. Stakeholder review and approval (Owner: Product Owner, Timing: As needed)  
Obtain review and approval from stakeholders at each stage of the deployment and monitoring process.

**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 3s**

If I had to choose one thing that I'd want a second pair of eyes on, it would be the feature computation code and data quality checks.

Specifically, I'd want someone to review the code that computes the new features, such as ``cancel_request_submitted_before_prediction_time`` and ``contract_renewal_status``, to ensure that:

1. The logic is correct and consistent with the expected behavior.
2. The code is handling edge cases and potential issues, such as missing values or outliers.
3. The data quality checks are thorough and effective in detecting any issues with the feature data.

Having a second pair of eyes on this code would help catch any potential errors or oversights, and ensure that the features are being computed correctly and reliably. This would give me greater confidence in the accuracy and effectiveness 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.