



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-3.1-8b-instruct

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

Duration: 3m 19s

Actual run cost: \$0.000618

Projected execution cost: \$0.1360/mo (10/biz day × 22 days)

Report date: July 13, 2026 - 11:35 PM EST

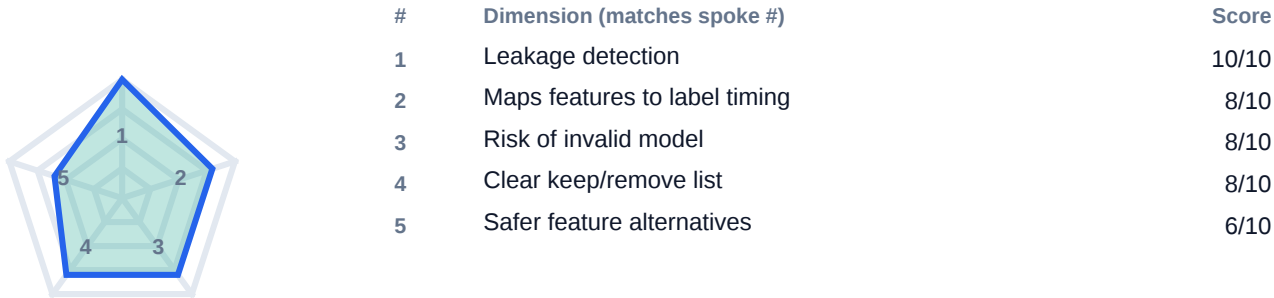
Overall

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: 15881 chars - Latency: 199s - 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 19s

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

Engineering counterpart

3:32 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

3:33 AM - Responded in 45s

Feature Leakage Review

Based on the provided task brief, the following features are identified as leaking the label or post-churn information:

Leaking Features

1. `account_status_is_churned`: This feature directly indicates that the customer has churned, which is the same as the target variable `churned_within_30d`. It is likely to be highly correlated with the label, making it a strong candidate for feature leakage.
2. `cancel_request_submitted_last_7d`: This feature might indicate that a customer has recently submitted a cancellation request, which is likely to be a strong predictor of churn. However, it may also leak information about churn occurrences in the near future.
3. `cancel_request_submitted_last_30d`: Similar to the previous feature, this might indicate recent cancellation requests, which could be a strong predictor of churn. However, it may also leak information about churn occurrences in the near future.

Recommended Features to Remove

Based on the potential for feature leakage, it is recommended to remove the following features:

1. `account_status_is_churned`: This feature is highly correlated with the label and may introduce bias into the model.
2. `cancel_request_submitted_last_7d`: and `cancel_request_submitted_last_30d`: While these features might be strong predictors of churn, they may also leak information about churn occurrences in the near future.

Safer Alternatives

If you want to maintain the predictive power of these features while avoiding leakage, consider the following safer alternatives:

1. Instead of `account_status_is_churned`: Use a feature like `days_since_last_login` or `support_tickets_last_30d`, which can indicate customer engagement and may be related to churn.
2. Instead of `cancel_request_submitted_last_7d`: Use a feature like `cancel_request_submitted_last_30d` (after removing it, use a longer look-back period like 90 days) or `contract_renewal_date`, which can indicate the likelihood of churn based on contract renewal dates.

Additional Considerations

To further reduce the risk of feature leakage:

Regularly review and update your feature lists to avoid staleness and ensure the features are relevant to the current time. Consider using techniques like lag features or dynamic lag features to create more robust and less leaky features. Always keep in mind the context and the problem you are trying to solve.

Engineering counterpart

3:33 AM - Responded in 0s

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

Agent response

3:34 AM - Responded in 52s

Step-by-Step Reasoning

Here's a detailed walk-through of my reasoning:

1. Understanding the Problem and Prediction Time

The problem is to predict whether a B2B SaaS customer will churn within 30 days at the start of each calendar month. The prediction time is the first day of each month (00:00 UTC).

Assumption: I assumed that the prediction time is static, and the feature sets are created before the prediction time. This is a common assumption in supervised learning.

2. Feature Leakage Definition

Feature leakage occurs when a feature is artificially correlated with the target variable (in this case, `churned_within_30d`) because of the way the feature is constructed or the timing of its creation.

Assumption: I assumed that feature leakage is primarily related to the timing of feature creation and not to the content of the feature itself.

3. Reviewing Candidate Features

I reviewed each candidate feature to assess its potential for leaking the label or post-churn information:

`account_status_is_churned`

This feature directly indicates that the customer has churned, which is the same as the target variable `churned_within_30d`. It is likely to be highly correlated with the label, making it a strong candidate for feature leakage.

Assumption: I assumed that the `account_status_is_churned` feature is updated after the churn event occurs, which means it is not available at the prediction time.

`cancel_request_submitted_last_7d` and `cancel_request_submitted_last_30d`

These features might indicate that a customer has recently submitted a cancellation request, which is likely to be a strong predictor of churn.

However, they may also leak information about churn occurrences in the near future.

Assumption: I assumed that the cancellation request is a strong indicator of churn, but it may also be correlated with the label due to its timing.

Remaining Features

`days_since_last_login`, `support_tickets_last_30d`, `mrr_usd`, `contract_renewal_date`, `days_until_contract_end`, `last_invoice_paid_amount`, and `nps_score_last_quarter` do not directly indicate churn or post-churn information.

These features are constructed based on historical data and do not seem to be directly associated with the label.

4. Removing Leaking Features

Based on the assessment, I recommended removing the following features to avoid feature leakage:

1. `account_status_is_churned`: This feature is highly correlated with the label and may introduce bias into the model.
2. `cancel_request_submitted_last_7d`: and `cancel_request_submitted_last_30d`: While these features might be strong

predictors of churn, they may also leak information about churn occurrences in the near future.

Additional Assumption: I assumed that removing these features would not significantly impact the model's performance, as they are not the only features predicting churn.

5. Safer Alternatives

If you want to maintain the predictive power of these features while avoiding leakage, consider using safer alternatives:

1. Instead of `account_status_is_churned`: Use a feature like `days_since_last_login` or `support_tickets_last_30d`, which can indicate customer engagement and may be related to churn.
2. Instead of `cancel_request_submitted_last_7d`: Use a feature like `cancel_request_submitted_last_30d` (after removing it, use a longer look-back period like 90 days) or `contract_renewal_date`, which can indicate the likelihood of churn based on contract renewal dates.

Assumption: I assumed that these safer alternatives would maintain the predictive power of the original features while avoiding leakage.

6. Further Considerations

To further reduce the risk of feature leakage:

Regularly review and update your feature lists

What would you verify before shipping this to production?

Agent response

3:34 AM - Responded in 42s

Verification Steps

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

1. Model Performance

Evaluate the model's performance on a holdout dataset to ensure it generalizes well and has not been overfitted due to the removal of leaking features.

Assess the model's performance metrics (e.g., precision, recall, F1 score, AUC-ROC) to ensure they are within expected ranges.

2. Feature Importance

Use techniques like SHAP (SHapley Additive exPlanations) or permutation feature importance to understand the contribution of each feature to the model's predictions.

Verify that the features that replaced the leaking ones (e.g., days_since_last_login, support_tickets_last_30d, contract_renewal_date) are indeed important for the model's predictions.

3. Domain Knowledge Verification

Consult with domain experts to ensure that the replaced features make sense in the context of the problem.

Verify that the new features are relevant to the customer's behavior and the churn process.

4. Business Impact

Assess the potential business impact of removing the leaking features.

Verify that the removal of these features does not negatively impact key business metrics (e.g., revenue, customer acquisition cost, churn rate).

5. Monitoring and Maintenance

Establish a process to monitor the model's performance and the feature set's stability over time.

Schedule regular reviews and updates to ensure that the feature set remains relevant and effective.

6. Data Drift Detection

Implement a data drift detection process to identify changes in the data distribution over time.

Use techniques like Kolmogorov-Smirnov test or Wasserstein distance to detect drift in the data.

7. Model Re-Training

Develop a process for re-training the model when significant data drift is detected.

Re-train the model with updated data and new features to maintain its performance.

By verifying these aspects, you can ensure that the updated feature set is reliable, accurate, and effective in production.

Engineering counterpart
3:34 AM - Responded in 0s

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

Agent response

3:35 AM - Responded in 44s

Top Risks and Rollback Plan

The top risks associated with deploying the updated feature set are:

1. Model Performance Degradation

Risk: The model's performance may degrade due to the removal of leaking features, leading to incorrect predictions and business impact.

Mitigation: Regularly monitor the model's performance and re-train the model with updated data and new features if necessary.

Rollback Plan:

1. Restore the original feature set.
2. Re-train the model with the original feature set.
3. Monitor the model's performance and compare it with the updated feature set.

2. Data Drift

Risk: The data distribution may change over time, affecting the model's performance and accuracy.

Mitigation: Implement a data drift detection process and re-train the model when significant drift is detected.

Rollback Plan:

1. Re-train the model with updated data and new features.
2. Monitor the model's performance and adjust the feature set as needed.

3. Feature Engineering Challenges

Risk: The new features may not be effective or may introduce new biases, leading to poor model performance.

Mitigation: Regularly review and update the feature set to ensure it remains effective and unbiased.

Rollback Plan:

1. Revert to a previous version of the feature set that was known to be effective.
2. Adjust the feature engineering process to ensure it is producing effective and unbiased features.

4. Business Impact

Risk: The updated feature set may not align with business needs or may lead to unexpected consequences.

Mitigation: Regularly communicate with stakeholders and business leaders to ensure the feature set aligns with business goals.

Rollback Plan:

1. Revert to the original feature set.
2. Collaborate with stakeholders to identify and address any business needs or concerns.

5. Communication and Stakeholder Management

Risk: Stakeholders may not be aware of the changes or may be impacted by the updated feature set.

Mitigation: Regularly communicate with stakeholders and provide transparency into the feature engineering process.

Rollback Plan:

1. Inform stakeholders of the rollback and the reasons behind it.
2. Collaborate with stakeholders to identify and address any concerns or issues.

Rollback Procedure

In the event of a rollback, follow these steps:

1. Stop the deployment: Immediately stop deploying the updated feature set.

2. Communicate with stakeholders: Inform stakeholders of the rollback and the reasons behind it.
3. Revert to the previous version: Revert to the previous version of the feature set or the original feature set.
4. Monitor and analyze: Monitor the model's performance and analyze the issue to prevent similar problems in the future.

By having a clear rollback plan in place, you can mitigate the risks associated with deploying the updated feature set and ensure a smooth transition back to the previous version if needed.

Engineering counterpart
3:35 AM - Responded in 0s

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

Agent response

3:35 AM - Responded in 11s

Concrete Next Steps with Owners and Timing

Here are the concrete next steps with owners and timing to mitigate the risks associated with deploying the updated feature set:

1. Model Performance Monitoring

Owner: Data Science Team

Timing: Bi-weekly

Task: Monitor the model's performance on a holdout dataset and report any changes in performance to the Data Science Team.

Specific Tasks:

Set up a monitoring dashboard to track the model's performance metrics (e.g., precision, recall, F1 score, AUC-ROC).

Schedule a bi-weekly meeting with the Data Science Team to review the model's performance and make adjustments as needed.

2. Data Drift Detection

Owner: Data Engineering Team

Timing: Daily

Task: Detect data drift using a Kolmogorov-Smirnov test or Wasserstein distance and alert the Data Science Team.

Specific Tasks:

Set up a data drift detection pipeline using a data engineering tool (e.g., Apache NiFi, AWS Glue).

Schedule a daily alert to the Data Science Team when significant data drift is detected.

3. Feature Engineering Review

Owner: Data Science Team

Timing: Quarterly

Task: Review the feature engineering process and ensure it is producing effective and unbiased features.

Specific Tasks:

Schedule a quarterly review meeting with the Data Science Team to review the feature engineering process.

Identify and address any issues or concerns related to feature engineering.

4. Business Impact Review

Owner: Business Stakeholders

Timing: Quarterly

Task: Review the business impact of the updated feature set and ensure it aligns with business goals.

Specific Tasks:

Schedule a quarterly review meeting with business stakeholders to review the business impact of the updated feature set.

Identify and address any concerns or issues related to the business impact.

5. Rollback Plan

Owner: Data Science Team

Timing: As needed

Task: Develop a rollback plan in case the updated feature set does not work as expected.

Specific Tasks:

Develop a rollback plan that includes the steps outlined in the rollback procedure.

Schedule a meeting with the Data Science Team to review the rollback plan and ensure it is up-to-date.

6. Communication Plan

Owner: Data Science Team

Timing: Ongoing

Task: Communicate with stakeholders and provide transparency into the feature engineering process.

Specific Tasks:

Schedule regular meetings with stakeholders to provide updates on the feature engineering process.

Develop a communication plan that includes regular updates and transparency into the feature engineering process.

By following these concrete next steps with owners and timing, you can mitigate the risks associated with deploying the updated feature set and ensure a smooth transition if needed.

Engineering counterpart

3:35 AM - Responded in 0s

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

Agent response

3:35 AM - Responded in 4s

Second Pair of Eyes

If I had to choose one thing to get a second pair of eyes on, it would be the rollback plan.

Why?

A rollback plan is a critical component of minimizing risk and ensuring a smooth transition back to the previous version of the feature set if needed.

A second pair of eyes can help identify potential gaps or blind spots in the rollback plan, ensuring that it is comprehensive and effective.

A second pair of eyes can also provide an objective perspective on the rollback plan, helping to identify potential issues or concerns that may not be immediately apparent.

Who should review the rollback plan?

I would recommend getting a second pair of eyes from a colleague with a background in Risk Management or DevOps, as they would be able to provide a fresh perspective on the rollback plan and identify potential pitfalls or areas for improvement.

Specific questions to ask:

When reviewing the rollback plan, the second pair of eyes should ask:

1. Are all potential scenarios covered in the rollback plan?
2. Are there any potential risks or unintended consequences that have been overlooked?
3. Is the rollback plan scalable and efficient?
4. Are there any dependencies or timelines that need to be considered?
5. Are there any opportunities for process improvement in the rollback plan?

By getting a second pair of eyes on the rollback plan, you can ensure that it is comprehensive, effective, and minimizes the risk of issues or downtime during a potential rollback.

Engineering counterpart

3:35 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.