



Vantage RuntimeAI

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

6.8/10

Overall score

Run summary

Scenario: Feature Leakage Review

Model: mistralai/mistral-nemo

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

Duration: 1m 34s

Actual run cost: \$0.000234

Projected execution cost: \$0.0515/mo (10/biz day × 22 days)

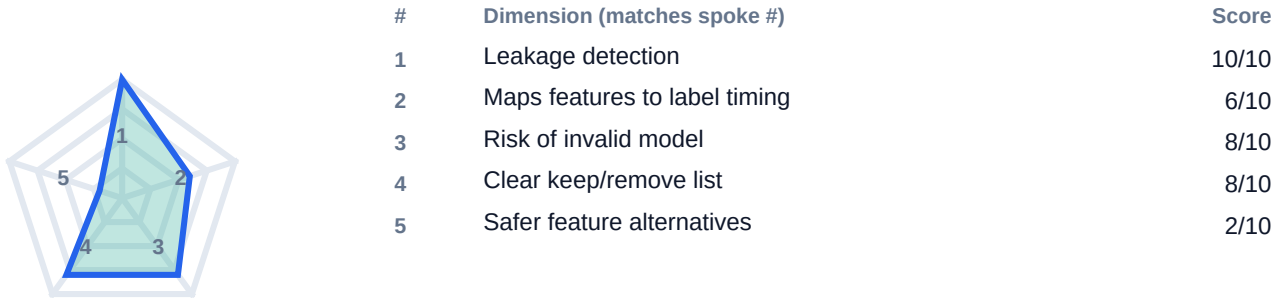
Report date: July 13, 2026 - 11:36 PM EST

Overall

6.8/10

FAIL (decision gate) - below_pass_line - trust=high
Fail - score 6.8/10 below pass line 7.0.

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: 19768 chars - Latency: 94s - Criteria: 3/3

Score analysis

3/3 automated criteria met - see checklist below.

Score snapshot

Developing: Mixed technical task quality - several criteria only partially met.

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.
- + 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

- Maps features to label timing (watch): References some artifacts but misses key inputs.
- Safer feature alternatives (critical gap): Weak or missing rationale / next steps.

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: 1m 34s

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

Engineering counterpart

3:35 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

3:35 AM - Responded in 9s

Based on the provided candidate features and the prediction time, here are the features that leak label or post-churn information, the reasons why, and safer alternatives if available:

1. Features that leak label or post-churn information:

- ``cancel_request_submitted_last_7d`` and ``cancel_request_submitted_last_30d``: These features indicate that a cancellation request was made in the last 7 or 30 days, which implies that the churn event has already occurred or is imminent. This is post-churn information and leaks the label.
- ``account_status_is_churned``: This feature directly indicates whether an account is churned, which is the same information as the target label. Using this feature would trivialize the model, as it contains the same information we're trying to predict.
- ``days_until_contract_end`` (to an extent): While not a direct leak, this feature might indirectly leak information because if a customer is close to contract end and hasn't renewed, it's a strong indicator of churn. However, this feature might still be useful for early churn prediction if used carefully.

2. Reasons for leakage relative to prediction time:

- The prediction time is the first day of each month. Features like ``cancel_request_submitted_last_7d`` and ``cancel_request_submitted_last_30d`` contain information that occurs after the prediction time, meaning these features are not available at the time of prediction.
- ``account_status_is_churned`` and ``days_until_contract_end`` contain information that is available at the time of prediction, but they are too strongly correlated with the target variable, making them less useful for building a predictive model.

3. Safer alternatives (if any):

- Instead of ``cancel_request_submitted_last_7d`` and ``cancel_request_submitted_last_30d``, consider using features that indicate cancellation request trends over time, such as the number of cancellation requests in the previous quarters or the ratio of cancellation requests to total requests.
- Instead of ``account_status_is_churned``, consider using features that lead up to churn, such as support ticket volume, payment delays, or changes in usage patterns.
- For ``days_until_contract_end``, consider using it in combination with other features to build a more comprehensive churn prediction model, but be cautious about its predictive power and potential leakage.

Recommended features to remove:

- ``cancel_request_submitted_last_7d``
- ``cancel_request_submitted_last_30d``
- ``account_status_is_churned`` (remove or use with caution)

These features should be removed or used with extreme caution, as they can lead to overly optimistic model performance estimates and may not generalize well to new, unseen data. Always validate your model using appropriate techniques, such as time series cross-validation, to ensure its robustness and generalizability.

Engineering counterpart

3:35 AM - Responded in 0s

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

Agent response

3:35 AM - Responded in 14s

I'm glad you found the initial response helpful! Here's a step-by-step walkthrough of the reasoning process, including the assumptions made:

1. Understand the task and the prediction time:

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

2. Identify features with post-churn information:

- Assumption: Features containing information that occurs after the prediction time (first day of the month) leak the label, as they are not available at the time of prediction.

- Reasoning:

- ``cancel_request_submitted_last_7d`` and ``cancel_request_submitted_last_30d``: These features indicate that a cancellation request was made in the last 7 or 30 days. Since the prediction time is the first day of the month, this information is not available at the time of prediction.

- Conclusion: These features leak post-churn information and should be removed.

3. Identify features with strong correlation to the target variable:

- Assumption: Features that are strongly correlated with the target variable (``churned_within_30d``) may trivialize the model, as they contain the same information we're trying to predict.

- Reasoning:

- ``account_status_is_churned``: This feature directly indicates whether an account is churned, which is the same information as the target label. Using this feature would make the model overly simplistic and might not capture other important factors leading to churn.

- ``days_until_contract_end`` (to an extent): While not a direct leak, if a customer is close to contract end and hasn't renewed, it's a strong indicator of churn. However, this feature might still be useful for early churn prediction if used carefully.

- Conclusion: ``account_status_is_churned`` should be removed, or used with caution. ``days_until_contract_end`` might still be useful, but its predictive power should be assessed carefully.

4. Consider safer alternatives for removed features:

- Reasoning:

- Instead of ``cancel_request_submitted_last_7d`` and ``cancel_request_submitted_last_30d``, consider using features that indicate cancellation request trends over time (e.g., number of cancellation requests in previous quarters, ratio of cancellation requests to total requests), as these might provide useful insights into churn probabilities.

- Instead of ``account_status_is_churned``, consider using features that lead up to churn, such as support ticket volume, payment delays, or changes in usage patterns, which might capture the underlying reasons for churn.

- For ``days_until_contract_end``, consider using it in combination with other features to build a more comprehensive churn prediction model, but be cautious about its predictive power and potential leakage.

- Conclusion: These alternatives might provide valuable information for predicting churn while minimizing leakage issues.

5. Ensure robust model validation:

- Assumption: Always validate your model using appropriate techniques, such as time series cross-validation, to ensure its robustness and generalizability to new, unseen data.

- Reasoning: Removing leaked features might improve the model's performance on the training data, but it's crucial to validate the model's performance on independent test data or using cross-validation to ensure its reliability.

- Conclusion: Conduct proper model validation to assess the generalizability and robustness of the churn prediction model.

By following these steps and assumptions, we can identify and address feature leakage issues in the given churn prediction task.

Engineering counterpart
3:35 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

3:35 AM - Responded in 10s

Before shipping a churn prediction model to production, it's essential to verify several aspects to ensure its reliability, robustness, and business value. Here's a checklist of items to verify:

1. Data quality and consistency:

- Ensure that the data used for training and validation is representative of the data that will be encountered in production.
- Validate that the data preprocessing steps (e.g., handling missing values, encoding categorical variables, feature scaling) work correctly and consistently in the production environment.
- Verify that the data access and retrieval processes are efficient and reliable.

2. Model performance:

- Evaluate the model's performance using appropriate metrics (e.g., AUC-ROC, precision, recall, F1-score) on an independent test dataset or through time series cross-validation.
- Assess the model's performance across different customer segments or cohorts to ensure its generalizability.
- Compare the performance of the new model with any existing models or baselines to ensure improvements.

3. Feature importance and interpretability:

- Investigate the most important features contributing to the model's predictions to ensure they make business sense.
- Validate that the model's predictions align with domain knowledge and business understanding of churn drivers.
- Consider using SHAP, LIME, or other interpretability techniques to explain the model's predictions and gain insights into feature importance.

4. Model fairness and bias:

- Evaluate the model's fairness by assessing if it treats all customer groups equally, regardless of their demographic characteristics (e.g., industry, company size, location).
- Identify and address any biases in the model's predictions, such as disproportionately predicting churn for specific customer groups.

5. Model robustness and stability:

- Assess the model's performance on edge cases and extreme data points to ensure it handles them appropriately.
- Validate that the model's performance is consistent over time and doesn't degrade rapidly due to concept drift or changing customer behavior.

6. Integration and deployment:

- Verify that the model can be integrated seamlessly into the existing tech stack and data pipelines.
- Test the model's API or integration points with other systems to ensure they function correctly and handle errors gracefully.
- Validate that the model can be easily updated or retrained when new data becomes available.

7. Monitoring and alerting:

- Implement monitoring and alerting mechanisms to track the model's performance, data quality, and other relevant metrics in production.
- Set up alerts to notify stakeholders when specific performance thresholds are breached, indicating potential issues with the model or data.

8. Business value and impact:

- Quantify the potential business impact of the churn prediction model, such as the number of churns prevented or the additional revenue generated.
- Define success metrics and key performance indicators (KPIs) to continuously evaluate the model's performance and impact on the business.

9. Feedback loop and continuous improvement:

- Establish a feedback loop to collect user feedback, monitor performance, and identify areas for improvement.
- Regularly retrain and update the model with fresh data to maintain its predictive power and adapt to changing customer

behavior.

By verifying these aspects, you can increase the likelihood of a successful production deployment and ensure that the churn prediction model delivers business value and remains reliable over time.

Engineering counterpart
3:35 AM - Responded in 0s

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

Agent response

3:36 AM - Responded in 26s

When deploying a churn prediction model to production, there are several potential risks to consider. Here are the top risks and a rollback plan if the model doesn't work as expected:

Top risks:

1. Model performance degradation:

The model's performance degrades over time due to concept drift or changing customer behavior.

The model fails to capture new trends or patterns in the data, leading to inaccurate predictions.

Mitigation: Regularly monitor model performance, retrain the model with fresh data, and implement feedback loops to adapt to changes.

2. Data quality issues:

Data inconsistencies, missing values, or errors affect the model's performance.

Changes in data collection or processing pipelines introduce biases or noise into the model's inputs.

Mitigation: Implement data quality checks, monitor data pipelines, and maintain documentation on data sources and preprocessing steps.

3. Model interpretability and explainability:

Stakeholders struggle to understand the model's predictions or the underlying reasons for churn.

Transparency and trust in the model's decisions are compromised, leading to user resistance or lack of buy-in.

Mitigation: Use interpretability techniques (e.g., SHAP, LIME) to explain predictions, document feature importance, and ensure model transparency.

4. Unintended consequences or biases:

The model inadvertently discriminates against certain customer groups based on biased or correlated features.

Churn mitigation strategies based on the model's predictions have unintended consequences, such as increased customer dissatisfaction or revenue loss.

Mitigation: Regularly evaluate the model's fairness, assess the impact of churn mitigation strategies, and address any identified biases.

5. Integration and compatibility issues:

The model fails to integrate seamlessly with existing systems or data pipelines.

Errors or delays in the model's API or integration points hinder its usability and effectiveness.

Mitigation: Thoroughly test the model's integration with other systems, document API endpoints, and maintain open communication with technical stakeholders.

6. Communication and stakeholder management:

Poor communication of the model's purpose, capabilities, and limitations leads to unrealistic expectations or user resistance.

Failure to engage key stakeholders, such as data scientists, business users, and decision-makers, results in a lack of buy-in or support.

Mitigation: Clearly communicate the model's objectives, capabilities, and limitations, engage stakeholders throughout the development process, and foster a culture of collaboration and transparency.

Rollback plan if the model doesn't work as expected:

1. Quickly identify and diagnose the issue:

Monitor model performance and key metrics closely during the initial deployment phase.

Set up alerts to notify stakeholders when specific performance thresholds are breached.

Gather data on user feedback and model behavior to pinpoint the problem.

2. Temporarily disable the model (if necessary):

If the model's performance is severely compromised, consider temporarily disabling it to minimize the impact on users and business operations.

Enable a fallback mechanism, such as a rule-based or heuristic-based churn prediction method, to maintain business continuity.

3. Conduct a thorough root cause analysis:

Investigate the underlying reason for the model's poor performance, such as data quality issues, concept drift, or model limitations.

Gather evidence and data to support your findings and inform the next steps.

4. Communicate the issue and rollback plan:

Transparently communicate the problem, its impact, and the planned course of action to affected stakeholders.

Maintain open lines of communication and provide regular updates on the progress and timeline for resolution.

5. Develop and implement a plan to fix the issue:

Based on the root cause analysis, develop a plan to resolve the problem, such as retraining the model, addressing data quality issues

Engineering counterpart
3:36 AM - Responded in 0s

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

Agent response

3:36 AM - Responded in 27s

Based on the identified risks and rollback plan, here are concrete next steps with owners and suggested timelines to ensure a successful production deployment and address potential issues:

Pre-deployment activities (Within 2 weeks):

1. Data quality and integration:

Owner: Data Engineer

Tasks:

- + Validate data quality and consistency in the production environment.
- + Test data retrieval and preprocessing pipelines.
- + Ensure seamless integration of the model with existing systems and data pipelines.

Timeline:

- Data quality checks: 3 days
- Pipeline testing: 4 days
- Integration testing: 3 days

2. Model performance evaluation:

Owner: Data Scientist

Tasks:

- + Conduct final model performance evaluation on an independent test dataset or using time series cross-validation.
- + Assess feature importance and interpretability.
- + Define baseline metrics and performance thresholds for monitoring.

Timeline:

- Performance evaluation: 3 days
- Feature importance analysis: 2 days
- Baseline metrics definition: 1 day

3. Stakeholder communication and alignment:

Owner: Cross-functional team lead (e.g., Product Manager or Data Science Manager)

Tasks:

- + Communicate the model's objectives, capabilities, and limitations to key stakeholders.
- + Gather feedback and address any concerns.
- + Ensure alignment and buy-in from business users, decision-makers, and technical stakeholders.

Timeline:

- Initial communication: 1 day
- Feedback collection and addressing concerns: 2 days
- Alignment and buy-in: 2 days

Post-deployment monitoring and maintenance (Ongoing):

1. Model performance monitoring:

Owner: Data Scientist or Data Engineer

Tasks:

- + Monitor model performance and key metrics regularly (e.g., daily, weekly).
- + Set up alerts for significant performance degradation or unexpected behavior.
- + Conduct periodic model retraining and updates with fresh data.

Timeline:

- Daily/weekly monitoring: Ongoing
- Alert configuration: 2 days
- Model retraining and updates: Quarterly or as needed

2. Data quality monitoring:

Owner: Data Engineer

Tasks:

- + Monitor data quality and consistency in the production environment.
- + addressing any data-related issues that may impact model performance.

Timeline:

- Ongoing data quality monitoring: Daily/weekly

3. User feedback collection and analysis:

Owner: Product Manager or Customer Success Manager

Tasks:

- + Collect user feedback on the model's predictions and churn mitigation strategies.
- + Analyze feedback to identify trends, common issues, or areas for improvement.
- + Communicate findings to the data science team and inform model updates.

Timeline:

- Ongoing feedback collection and analysis: Quarterly or as needed

4. Regular model reviews and continuous improvement:

Owner: Cross-functional team lead (e.g., Product Manager or Data Science Manager)

Tasks:

- + Schedule regular model reviews to assess performance, gather feedback, and discuss potential improvements.
- + Facilitate collaboration between data scientists, business users, and technical stakeholders to drive continuous improvement.

Timeline:

- Model reviews: Quarterly or as needed

Rollback plan (If performance issues are detected):

1. Quick issue identification and diagnosis (Within 1-2 days):

Owner: Data Scientist or Data Engineer

Tasks:

- + Investigate and diagnose the root cause of the performance issue.
- + Gather evidence and data to support findings.

Timeline:

Engineering counterpart

3:36 AM - Responded in 0s

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

Agent response

3:36 AM - Responded in 9s

The one thing I'd want a second pair of eyes on is the model performance and interpretation to ensure unbiased and accurate evaluation. This includes:

1. Performance metrics: Review and validate the selected performance metrics (e.g., AUC-ROC, precision, recall, F1-score) to ensure they align with the business problem and objectives. Assess the model's performance across different customer segments, edge cases, and time horizons to ensure its generalizability and robustness.
2. Feature importance: Evaluate the most important features driving the model's predictions. Investigate if the top features make business sense and capture the underlying reasons for churn. Use feature importance visualizations, SHAP, LIME, or other interpretability techniques to gain insights into the model's decision-making process.
3. Fairness and bias: Assess the model's fairness by checking if it treats all customer groups equally, regardless of their demographic characteristics (e.g., industry, company size, location). Identify and address any biases in the model's predictions that could lead to unfair outcomes or unintended consequences.

By having a second pair of eyes review these aspects, you can:

Ensure an unbiased and thorough evaluation of the model's performance and interpretability.

Identify potential blind spots or assumptions that might have been overlooked during model development.

Build confidence in the model's predictions and its ability to drive business decisions.

Foster a culture of collaboration and continuous improvement within the data science team and other stakeholders.

Requesting a second pair of eyes for these critical aspects will help ensure that the churn prediction model is well-validated, reliable, and fair before it's deployed to production.

Engineering counterpart

3:36 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.