



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: amazon/nova-micro-v1

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

Duration: 15s

Actual run cost: \$0.000609

Projected execution cost: \$0.1340/mo (10/biz day × 22 days)

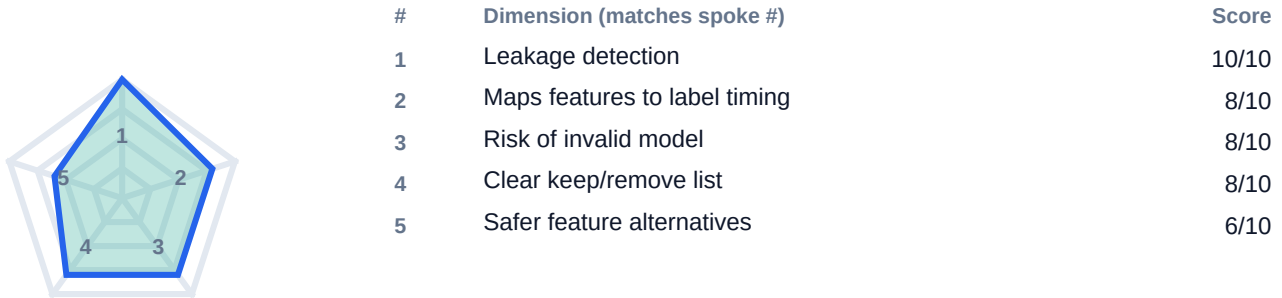
Report date: July 14, 2026 - 3:33 AM EST

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: 19179 chars - Latency: 15s - 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: 15s

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

7:33 AM - Responded in 3s

When reviewing candidate features for leakage relative to prediction time, we need to ensure that no feature contains information that has only become available after the label (i.e., churned_within_30d) was determined. This would compromise the model's ability to generalize and thus reduce its effectiveness. Here's an analysis of the leakage potential for each feature:

Leakage Analysis:

1. days_since_last_login

- Leakage Potential: Low if the last login date is known at prediction time.
- Explanation: This feature might have leakage if we know the current date at prediction time. However, if the last login date is as of the first day of the month, it is safe.

2. support_tickets_last_30d

- Leakage Potential: Low if the tickets are counted as of the first day of each month.
- Explanation: If support tickets are counted from the first day of the month, there should be no leakage.

3. mrr_usd

- Leakage Potential: Low if MRR is based on available historical data up to the start of the month.
- Explanation: MRR calculated up to the first day of the month should be safe.

4. contract_renewal_date

- Leakage Potential: None.
- Explanation: This date is determined by the contract terms and is not dependent on any post-prediction events.

5. days_until_contract_end

- Leakage Potential: None.
- Explanation: This is derived from the fixed contract renewal date and does not rely on post-prediction information.

6. cancel_request_submitted_last_7d

- Leakage Potential: High.
- Explanation: This feature will leak information if we know the date today. To avoid leakage, we must count up to the first day of the month.

7. cancel_request_submitted_last_30d

- Leakage Potential: Low if counted from the first day of the month.
- Explanation: As long as the count starts from the first day, this feature should be safe.

8. account_status_is_churned

- Leakage Potential: High.
- Explanation: This feature directly represents the label we are trying to predict and thus leaks information.

9. last_invoice_paid_amount

- Leakage Potential: Low if the amount is known for the current period.
- Explanation: This feature should be based on available historical data.

10. nps_score_last_quarter

- Leakage Potential: Low if known as of the start of the month.
- Explanation: If NPS score is based on the previous quarter and known by the first day of each month, it should be safe.

Recommendations for Removal and Alternatives:

1. cancel_request_submitted_last_7d

- Remove: Yes

- Reason: This feature leaks information as it depends on current data beyond the first day of the month.
- Safer Alternative: Change the timeframe to `cancel\_request\_submitted\_last\_30d` if counted from the first day of each month.

2. account_status_is_churned

- Remove: Yes
- Reason: This feature directly indicates the label, thus leaking information.
- Safer Alternative: Avoid this feature entirely, as it undermines the model's predictive nature.

Summary:

- Remove `cancel\_request\_submitted\_last\_7d` unless timeframe is adjusted to last 30

Engineering counterpart
7:33 AM - Responded in 0s

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

Agent response

7:33 AM - Responded in 2s

Certainly, let's go through the reasoning for each feature step-by-step and the assumptions we made to determine whether they leak information. Here's a detailed breakdown:

General Assumptions:

1. Prediction Time Frame: The prediction is made at the start of each calendar month, specifically at 00:00 UTC on the first day of the month.
2. Availability of Data: We assume that all feature values are calculated up to and including the first day of the month.
3. Interpretation of Date-Based Features: Date-based features need to be evaluated based on the context in which the current date is known.

Feature Analysis and Assumptions:

1. days_since_last_login

- Reasoning: This feature calculates the number of days since the customer last logged in.
- Assumption: If the last login date is known as of the first day of the month, the feature will provide a historical count. This does not leak future information as it's based on past data.
- Conclusion: Safe to include if calculated up to the first day of the month.

2. support_tickets_last_30d

- Reasoning: This feature counts the support tickets opened in the last 30 days.
- Assumption: If the count starts from the first day of the month, it will not include tickets that were opened post the prediction time.
- Conclusion: Safe to include if calculated up to the first day of the month.

3. mrr_usd

- Reasoning: This represents the monthly recurring revenue.
- Assumption: MRR is calculated based on historical data up to the start of the month.
- Conclusion: Safe to include if MRR is based on data cutoff at the first day of the month.

4. contract_renewal_date

- Reasoning: This is a fixed date derived from the contract terms.
- Assumption: This date doesn't change and doesn't depend on current or future events.
- Conclusion: Safe to include.

5. days_until_contract_end

- Reasoning: This feature calculates the number of days until the contract ends.
- Assumption: Calculated based on the fixed contract renewal date.
- Conclusion: Safe to include.

6. cancel_request_submitted_last_7d

- Reasoning: This feature counts the cancel requests submitted in the last 7 days.
- Assumption: If the count includes today's date, it will include information post-prediction time.
- Leakage Potential: High because it depends on knowing the exact current date.
- Conclusion: Remove or adjust to count from the first day of the month to ensure no leakage if the count adjusts to last 30 days instead.

7. cancel_request_submitted_last_30d

- Reasoning: This feature counts cancel requests submitted in the last 30 days.
- Assumption: Counting from the first day of the month ensures it doesn't include requests made post-prediction time.
- Conclusion: Safe to include if the count is established from the first day of each month.

8. account_status_is_churned

- Reasoning: This feature directly indicates if the account has churned.

- Assumption: This feature represents the label (churned_within_30d) we are trying to predict.
- Leakage Potential: High because it directly represents the target variable.
- Conclusion: Remove. It leaks the target information.

9. last_invoice_paid_amount

- Reasoning: This feature contains the last

Engineering counterpart
7:33 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:33 AM - Responded in 2s

Before shipping any change to production, especially a data model or code modifications that involve features like churn prediction, you should conduct a thorough set of verifications and validations. Here's a detailed checklist to ensure the integrity and effectiveness of the feature:

1. Data Integrity and Availability

- Feature Data Availability: Ensure all required features are available for the prediction at the correct time (first day of each month).
- Data Pre-aggregation: Verify that data needed for any feature is pre-aggregated before the first day of each month to avoid post-prediction data leakage.
- Historical Data Consistency: Check that historical data used for features is consistent and correctly aligned with the prediction time frame.

2. Feature Calculation Logic

- Correct Time Frame Calculation: Validate that features like `days\_since\_last\_login` and `support\_tickets\_last\_30d` are calculated correctly using the appropriate time frame based on the first day of each month.
- No Future Data Leakage: Confirm that none of the features include future data or have been adjusted based on knowledge post the prediction time.

3. Model Validation

- Cross-Validation: Rerun cross-validation to ensure the model's performance hasn't degraded with the new features.
- Feature Importance: Recheck feature importance to ensure new features are contributing positively and not overfitting.
- Performance Metrics: Verify key performance metrics such as accuracy, precision, recall, and AUC-ROC to ensure they meet or exceed the acceptance criteria.

4. Integration Testing

- System Integration: Ensure that the feature integration into the existing system works seamlessly.
- Data Pipeline Reliability: Test the end-to-end data pipeline to make sure it reliably delivers the necessary data to feature engineers and the model by the start of each month.

5. Code and Configuration Verification

- Code Reviews: Conduct thorough reviews to ensure there are no logical errors or vulnerabilities in the code implementing the new features.
- Configuration Management: Verify that configurations for feature calculation and model training/inference are correctly set up.

6. Error Handling and Monitoring

- Error Handling: Ensure robust error handling is in place to deal with missing or incorrect data.
- Monitoring Setup: Implement robust monitoring to catch any anomalies or performance drop post-deployment.

7. Compliance and Security

- Data Compliance: Ensure that all data handling complies with relevant regulations like GDPR, CCPA, etc.
- Security Checks: Validate that no new security vulnerabilities have been introduced.

8. User Acceptance Testing (UAT) for Business Logic

- Stakeholder Review: Get feedback from business stakeholders to ensure the logic and expected outputs align with business requirements.
- Business Scenario Tests: Test to ensure that the model behaves as expected under various business scenarios.

By following these steps, you can ensure that the churn prediction model, including its features, is robust, reliable, and free from data leakage issues, thereby maintaining the integrity of the model's predictions when deployed in production.

Engineering counterpart
7:33 AM - Responded in 0s

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

Agent response

7:33 AM - Responded in 3s

When deploying a new model or set of features into a production environment, especially those related to critical business operations like churn prediction, it's essential to identify potential risks upfront and have a rollback plan if things don't go as planned. Here are some top risks to consider and a proposed rollback plan:

Top Risks:

1. Data Leakage:

- Risk: Information leakage could happen if any of the features inadvertently use data that's only available post-prediction time, leading to unreliable predictions.
- Mitigation: Implement robust checks on feature calculation logic to ensure no future data is used at the prediction time.

2. Feature Overfitting or Underfitting:

- Risk: New features might lead to overfitting the model to the current dataset, diminishing its ability to generalize and underfitting due to insufficient complexity.
- Mitigation: Conduct extensive cross-validation and hyperparameter tuning.

3. Integration Issues:

- Risk: The new features or model integration could break existing functionalities or lead to system instability.
- Mitigation: Perform thorough integration testing and have a staging environment to test before going live.

4. Performance Degradation:

- Risk: Performance metrics such as prediction accuracy, response time, and resource consumption might degrade.
- Mitigation: Monitor key performance indicators (KPIs) closely post-deployment.

5. Model Drift:

- Risk: Changes in business dynamics or user behavior could lead to model drift over time, impacting prediction accuracy.
- Mitigation: Implement automatic model re-training and monitoring for data and model drift.

6. Operational Failures:

- Risk: System errors including failures in data retrieval, processing, or model inference.
- Mitigation: Utilize robust error handling and monitoring systems.

Rollback Plan:

1. Immediate Rollback Mechanism:

- Activate: Implement a mechanism that can quickly disable or revert the model or feature set in case a serious issue requires immediate attention.
- Execution: This should include a switch/flag that can turn off the new features/model, restoring the system to its pre-deployment state.

2. Data Consistency Checks:

- Activate: Before any rollback, ensure that all data processes are stopped to maintain data consistency during the rollback process.
- Execution: Implement locking mechanisms to prevent any new data from being processed during the rollback.

3. Communication Plan:

- Activate: Inform all relevant stakeholders (IT, business units, and customers if applicable) about the rollback plan and potential downtime.
- Execution: Utilize existing communication channels to ensure transparency.

4. Testing and Verification:

- Activate: After a rollback, perform extensive testing to verify that the system has returned to its pre-deployment state.
- Execution: Run the same set of tests used during pre-deployment checks to confirm stability and functionality.

5. Review and Analysis:

- Activate: Conduct a review meeting to analyze what went wrong and how to prevent it in the future.
- Execution: Document the issues encountered, the rollback process, and insights gained to refine the approach for any future deployments.

By methodically addressing these risks and having a clear rollback plan in place, you can minimize the potential negative impacts of deploying new features or models into a production environment.

Engineering counterpart
7:33 AM - Responded in 0s

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

Agent response

7:33 AM - Responded in 3s

Certainly! Here's a more concrete plan with steps, owners, and estimated timing for both identifying the top risks and preparing the rollback plan for your churn prediction model deployment:

Identification of Top Risks

1. Data Leakage

- Owner: Data Engineer
- Timing: Week 1
- Task: Review feature calculation logic to confirm no future data use. Implement checks to log any potential leakage incidents.

2. Feature Overfitting/Underfitting

- Owner: Data Scientist
- Timing: Week 2
- Task: Conduct cross-validation and hyperparameter tuning. Set up tests to monitor model performance against baseline.

3. Integration Issues

- Owner: DevOps Team Lead
- Timing: Week 3
- Task: Perform full integration testing on staging environment, focusing on data pipeline, model serving, and system interactions.

4. Performance Degradation

- Owner: Data Engineer & Performance Analyst
- Timing: Week 4
- Task: Define and set up monitoring for key performance indicators (KPIs) such as response time, error rates, and resource consumption.

5. Model Drift

- Owner: Data Scientist
- Timing: Ongoing
- Task: Implement automatic monitoring for model drift using statistical methods and set thresholds for triggering re-training.

6. Operational Failures

- Owner: DevOps Team Lead
- Timing: All Weeks
- Task: Put in place robust error handling and monitoring systems. Test these systems under simulated failure scenarios.

Rollback Plan

Immediate Rollback Mechanism

- Owner: DevOps Team Lead
- Timing: Concurrent with risk identification
- Task: Develop and implement a toggle or switch mechanism that can disable the model/features. Ensure it can be activated by a designated administrator.

Data Consistency Checks

- Owner: Data Engineer
- Timing: Concurrent with rollback mechanism activation
- Task: Implement locking mechanisms to halt any new data processes during rollback to maintain data consistency.

Communication Plan

- Owner: Communication Lead (Internal) & Customer Service Lead (if applicable)
- Timing: Day of Deployment & Day of Rollback
- Task: Prepare and distribute communication to all relevant stakeholders about the rollback plan and associated downtime.

Testing and Verification

- Owner: QA Team Lead
- Timing: Day post-rollback
- Task: Run a comprehensive set of tests to ensure the system has returned to its pre-deployment state. Verify all functionality and performance metrics.

Review and Analysis

- Owner: Project Manager & Data Scientist
- Timing: Day following successful verification
- Task: Conduct a meeting to review incidents, the rollback process, and any insights gained to refine future deployments.

This structured approach assigns clear ownership, sets a timeline for each critical task, and ensures that potential risks are addressed and that there is a well-defined process for reverting to the previous state in case of deployment failure.

Engineering counterpart

7:33 AM - Responded in 0s

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

Agent response

7:33 AM - Responded in 2s

The most critical component to have a second pair of eyes on is the Immediate Rollback Mechanism. Here's why:

Importance of the Immediate Rollback Mechanism

1. Criticality: If the new model or features go awry in production, an immediate rollback is essential to mitigate damage quickly. The effectiveness of this rollback mechanism determines how swiftly you can restore system functionality and minimize potential risks.
2. Complexity and Urgency: Implementing and testing an instant rollback requires careful coordination and execution to ensure that it doesn't inadvertently impact other systems or processes. The stakes are high, and any mistake in this process can have significant repercussions.
3. Verification: It's not enough to merely implement the mechanism; it also needs thorough verification tests under both normal and failure scenarios to ensure it works as intended when most needed.

Who Should Review It?

- DevOps Lead: Given their expertise in system infrastructure and deployment, they can validate that the rollback mechanism integrates seamlessly with the existing systems.
- System Architect: To ensure the rollback has minimal impact on overall system architecture and performance.

Specific Points to Verify:

- Activation Mechanism: Ensure that the toggle/switch can be reliably and securely activated by authorized personnel.
- Deactivation Scope: Verify that deactivating the mechanism only impacts the intended components and doesn't inadvertently shut down the entire system or affect other critical processes.
- Consistent Rollback: Under test conditions, confirm that rolling back the model/features reverts the system to its exact pre-deployment state in all aspects.
- Fail-safes and Backups: Check that there are no single points of failure and that sufficient backups are in place to ensure that in case of an error, recovery can be efficiently executed.
- Communication: Confirm that all necessary stakeholders are notified immediately upon activation of the rollback mechanism.

By having a second pair of eyes (and thereby a second set of expert hands) review and test this critical element, you greatly enhance the robustness and reliability of your churn prediction model's deployment process.

Engineering counterpart

7:33 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.