



## 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: openai/gpt-4o-mini

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

Duration: 53s

Actual run cost: \$0.002610

Projected execution cost: \$0.5742/mo (10/biz day × 22 days)

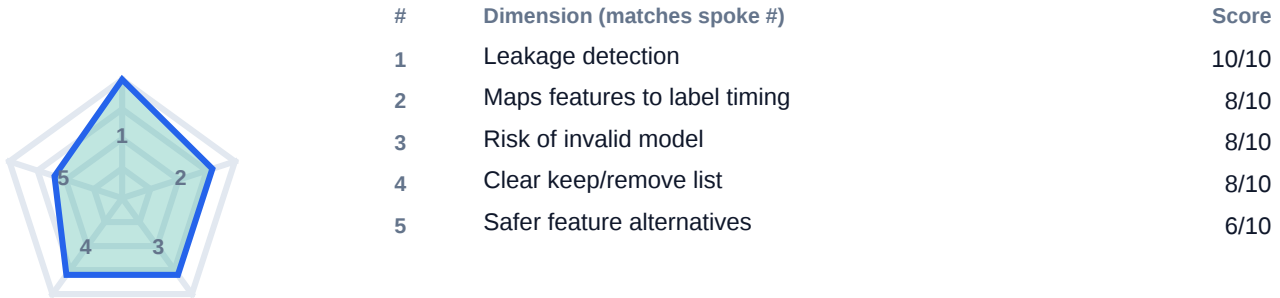
Report date: July 13, 2026 - 11:37 PM EST

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: 19956 chars - Latency: 53s - 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: 53s

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

## Engineering counterpart

3:36 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

3:37 AM - Responded in 18s

### Feature Leakage Review

### 1. days\_since\_last\_login

- Leakage: No immediate label leakage. It reflects user engagement prior to the prediction time.
- Recommendation: Keep this feature as it is a relevant engagement metric.

### 2. support\_tickets\_last\_30d

- Leakage: No immediate label leakage. It captures customer support interactions leading up to the prediction time.
- Recommendation: Keep this feature as it indicates customer satisfaction and potential issues.

### 3. mrr\_usd (Monthly Recurring Revenue)

- Leakage: No immediate label leakage. It provides information on the financial status of the account.
- Recommendation: Keep this feature as it is critical for understanding customer value.

### 4. contract\_renewal\_date

- Leakage: No immediate label leakage. This informs about the contract timelines, but does not include post-churn information. However, it may indirectly suggest churn risk if renewal is imminent.
- Recommendation: Keep, but be cautious about using it as it might reflect customer intention.

### 5. days\_until\_contract\_end

- Leakage: This may lead to leakage if it indicates a contract ending soon (e.g., if the customer is approaching their end date, it could correlate with higher churn risk).
- Recommendation: Remove or replace with features that do not directly relate to contract end dates.

### 6. cancel\_request\_submitted\_last\_7d

- Leakage: This feature leaks label information as a cancellation request submitted within 7 days predicts churn almost directly.
- Recommendation: Remove this feature as it directly indicates intent to churn.

### 7. cancel\_request\_submitted\_last\_30d

- Leakage: Similar to the last feature, it directly indicates churn intent, as a cancellation request submitted within 30 days almost guarantees churn.
- Recommendation: Remove this feature for similar reasons.

### 8. account\_status\_is\_churned

- Leakage: This feature leaks the label directly, as it literally states whether an account is churned.
- Recommendation: Remove this feature as it is not useful for predicting future churn.

### 9. last\_invoice\_paid\_amount

- Leakage: No immediate label leakage. It is a standard financial metric that can be indicative of account health.
- Recommendation: Keep this feature as it may provide insights into customer commitment.

### 10. nps\_score\_last\_quarter

- Leakage: No immediate label leakage. NPS scores from the last quarter reflect customer sentiment prior to the prediction time.
- Recommendation: Keep this feature as it is useful in predicting churn based on customer satisfaction.

### Summary of Recommended Removals

- Remove:

- `days\_until\_contract\_end`
- `cancel\_request\_submitted\_last\_7d`
- `cancel\_request\_submitted\_last\_30d`
- `account\_status\_is\_churned`

#### ### Safer Alternatives

- For contract-related features, consider using a derived metric that captures engagement trends or renewal history without directly indicating if a contract is near its end or in default.
- Instead of cancellation requests, consider customer engagement metrics from support tickets or login patterns that may indicate intent to churn without being direct indicators.

**Engineering counterpart**  
**3:37 AM - Responded in 0s**

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

## Agent response

3:37 AM - Responded in 8s

Sure! Let's walk through the reasoning for the feature leakage review step by step, including the assumptions made at each stage.

### ### Step-by-Step Reasoning

#### 1. Understanding the Problem Statement:

- The objective is to predict whether a customer will churn within 30 days at the start of each month.
- The prediction is made on the first day of the month, meaning that features used for prediction should reflect the state of the customer up to that point, avoiding any information that would suggest churn has already occurred.

#### 2. Identifying Feature Leakage:

- Feature leakage occurs when a feature provides information that would not have been available at the time of prediction, or it directly indicates the outcome (in this case, churn).
- I analyzed each candidate feature to determine whether it could provide insights into a customer's behavior leading to churn while also considering if it could either directly or indirectly indicate a churn status.

#### 3. Evaluating Each Feature:

- No Leakage: For features such as `days\_since\_last\_login`, `support\_tickets\_last\_30d`, `mrr\_usd`, `last\_invoice\_paid\_amount`, and `nps\_score\_last\_quarter`, I assumed that they reflect past behavior or sentiment without directly indicating if a customer has already churned.
- Potential Leakage: Features like `days\_until\_contract\_end` and `contract\_renewal\_date` were treated cautiously as they may hint at churn risks due to nearing contract expiration. I reasoned that while they don't outright indicate churn, they may correlate with risk, thus warranting careful consideration.
- Direct Leakage: Features such as `cancel\_request\_submitted\_last\_7d`, `cancel\_request\_submitted\_last\_30d`, and `account\_status\_is\_churned` were considered clear instances of leakage. The assumption here was that if a customer has submitted a cancellation request or is marked as churned, this is concrete evidence that they have already churned.

#### 4. Making Recommendations:

- The recommendations to remove features that directly leak the label were clear based on the objective of avoiding any indicators of churn that would not be available at prediction time.
- Features that were retained needed to be relevant and indicative of customer behavior but not directly tied to the label.

#### 5. Concluding with Safer Alternatives:

- The alternatives were based on the idea that while certain features may be removed due to leakage, other features could provide valuable insights. The assumption was that indirect indicators of customer engagement could still serve as useful predictors of churn without leaking the outcome.

### ### Assumptions Made

1. Temporal Relevance: It is assumed that features should only reflect the state of the account before the prediction time (i.e., everything is up to date as of the first day of the month).
2. Direct vs. Indirect Indicators: Assumed that some features could represent indirect signals of customer behavior, while others could directly reflect churn status.
3. Impact of Financial Features: I took into account that financial metrics like `mrr\_usd` or `last\_invoice\_paid\_amount` could be indicative of churn risk but do not provide direct evidence of churn.
4. Customer Behavior Interpretation: Assumed that customer engagement metrics such as support tickets and login frequency would generally lead to insights about customer satisfaction and the likelihood of churn.

Through this structured approach, the goal was to ensure that the finalized feature set is robust against leakage while still retaining informative metrics that can help in predicting churn effectively.

**Engineering counterpart**  
**3:37 AM - Responded in 0s**

What would you verify before shipping this to production?

## Agent response

3:37 AM - Responded in 8s

Before deploying a churn prediction model to production, it's crucial to verify several key aspects to ensure the model is robust, reliable, and functions as intended. Here's what I would focus on:

### ### 1. Data Quality Validation

- Data Completeness: Ensure there are no missing values in critical features. If missing values are present, strategies such as imputation or removal need to be defined.
- Data Consistency: Verify that data formats are consistent across the dataset. For example, check that date fields are in the same format and that numerical fields don't have unexpected characters.
- Data Accuracy: Cross-check a sample of the records to ensure that they accurately reflect the real state of customer engagement and financial metrics.

### ### 2. Feature Engineering Verification

- Leakage Checks: Confirm that all features selected for the model pass the leakage tests. Reassess features that are borderline to ensure no indirect leakage exists.
- Feature Importance Analysis: Use techniques such as feature importance scores (e.g., from tree-based models) to validate that the most predictive features are aligned with business logic and expectations.
- Correlation Analysis: Check for multicollinearity among features using correlation matrices. Highly correlated features could distort the model performance.

### ### 3. Model Training and Evaluation

- Cross-Validation: Employ k-fold cross-validation to assess model robustness and ensure that it generalizes well to unseen data.
- Performance Metrics Validation: Check key evaluation metrics like precision, recall, F1 score, and AUC-ROC to ensure the model meets performance benchmarks for business objectives.
- Comparison with Baseline: Compare the model performance against a baseline (e.g., a simple logistic regression or even a naive prediction) to ensure the new model offers a meaningful improvement.

### ### 4. Testing Against Edge Cases

- Simulated Scenarios: Run simulations or test cases against specific scenarios that may trigger churn, such as significant drops in usage or recent support tickets. Evaluate how the model responds to these edge cases.
- Bias and Fairness Checks: Assess the model for potential biases that could affect specific customer segments to ensure equitable treatment of all customers.

### ### 5. Production Readiness

- Monitoring Plan: Develop a plan for monitoring model performance post-deployment, including setting up alerts for drift in feature distributions or significant drops in prediction accuracy.
- Rollback Strategy: Prepare a rollback plan in case the model performs poorly in production. This should include clearly defined criteria for performance failure.
- User Feedback Loop: Establish a mechanism for collecting feedback from end-users on the model's predictions, which can be used for future iterations and improvements.

### ### 6. Documentation and Communication

- Clear Documentation: Document the model development process, feature selection rationale, and performance outcomes for transparency and future reference.
- Stakeholder Communication: Engage with stakeholders to ensure they understand the model's workings, potential limitations, and how to appropriately interpret results.

### ### 7. Ethical Considerations

- Privacy Compliance: Review compliance with data privacy laws (e.g., GDPR, CCPA) to ensure customer data is handled responsibly and ethically.
- Impact Assessment: Consider the societal implications of the churn prediction model and how it might affect customer relationships and business practices.

Verifying these aspects before production can significantly enhance the likelihood of deploying a successful model that delivers real value while minimizing risks.

**Engineering counterpart**  
**3:37 AM - Responded in 0s**

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

## Agent response

3:37 AM - Responded in 8s

When deploying a churn prediction model to production, several risks may arise that could affect its effectiveness, reliability, and overall business impact. Below are the top risks associated with deploying the model, along with a clear rollback plan in case the deployment does not work as intended.

### ### Top Risks

#### 1. Model Performance Issues:

- Risk: The model may underperform in terms of accuracy, precision, recall, or other key metrics when exposed to real-world data.
- Impact: Poor predictions could lead to ineffective retention strategies, wasted resources, and dissatisfied customers.

#### 2. Feature Drift:

- Risk: Changes in customer behavior or business operations could lead to a shift in the distributions of the input features over time.
- Impact: The model may become less relevant or accurate, decreasing its predictive ability.

#### 3. Data Quality Problems:

- Risk: Data quality issues, such as missing values or inaccuracies, could become more pronounced in a production environment.
- Impact: Poor data quality can lead to unreliable predictions and erode trust in the model.

#### 4. User Misinterpretation:

- Risk: Stakeholders may misinterpret the model's predictions, leading to inappropriate actions based on the output.
- Impact: Misguided business decisions could harm customer relationships and reduce retention efforts.

#### 5. Regulatory and Compliance Issues:

- Risk: Non-compliance with data protection regulations (e.g., GDPR, CCPA) may arise if customer data is not handled properly.
- Impact: Potential legal consequences and damage to the company's reputation.

#### 6. Bias and Fairness Concerns:

- Risk: The model may inadvertently favor or disadvantage specific customer segments.
- Impact: This could lead to ethical concerns and negative public perception.

### ### Rollback Plan

In the event that the churn prediction model does not perform as expected after deployment, the following rollback plan can be implemented:

#### 1. Pre-Deployment Backup:

- Ensure that a backup of the previous version of the prediction model (if applicable) and any related processes is securely stored and ready to be restored.

#### 2. Immediate Monitoring:

- Set up monitoring to track key performance metrics in real-time. Define clear thresholds for success and failure that trigger the rollback process.

#### 3. Evaluation Period:

- After deployment, establish an evaluation period (e.g., one month) to closely monitor model performance. If metrics fall below the defined thresholds during this period, initiate the rollback.

#### 4. Reverting to Previous Model:

- If performance issues are detected, revert to the previous model version by deploying the backup. This ensures

continuity in churn predictions while the issues with the new model are addressed.

5. Analysis and Debugging:

- Conduct a thorough analysis to identify the root causes of the model's poor performance. This may involve:
  - Examining data quality issues.
  - Analyzing feature distribution changes.
  - Reviewing user feedback and interpreting errors.

6. Model Refinement:

- Based on the analysis, refine the model by exploring alternative algorithms, adjusting hyperparameters, or incorporating additional features that may improve performance.

7. Re-deployment Testing:

- Before re-deploying any revised model, conduct extensive testing using a validation set that simulates real-world conditions. Ensure improvements are robust and reliable.

8. Documentation and Communication:

- Document all findings, adjustments made, and updates to the model. Communicate with stakeholders to keep them informed about any changes and the reasons for rollback or adjustments.

9. Ongoing Monitoring and Feedback:

- After re-deployment, continue to monitor performance closely and establish a feedback loop to collect insights from stakeholders and users regarding model performance and predictions.

By proactively addressing these risks and having a

**Engineering counterpart**  
**3:37 AM - Responded in 0s**

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

## Agent response

3:37 AM - Responded in 8s

Certainly! Below are concrete next steps for deploying the churn prediction model, along with assigned owners and a timeline for each step.

### ### Deployment Next Steps

| Step | Action Item                                                                                                            | Owner                           | Timing                                          |
|------|------------------------------------------------------------------------------------------------------------------------|---------------------------------|-------------------------------------------------|
| 1.   | Pre-Deployment Backup   Prepare and store a backup of the previous model and processes.                                | Data Engineering Team           | 1 week before deployment                        |
| 2.   | Final Data Quality Check   Conduct a thorough review of data quality (missing values, consistency).                    | Data Quality Analyst            | 3 days before deployment                        |
| 3.   | Monitoring Setup   Set up monitoring dashboards for key performance metrics (accuracy, precision, recall).             | Data Scientist                  | 3 days before deployment                        |
| 4.   | Stakeholder Training   Train stakeholders on how to interpret model predictions and expected outcomes.                 | Project Manager                 | 2 days before deployment                        |
| 5.   | Model Deployment   Deploy the churn prediction model to production.                                                    | Machine Learning Engineer       | 1 day after training                            |
| 6.   | Immediate Performance Monitoring   Monitor model performance metrics in real-time for the first month.                 | Data Scientist                  | Start on deployment day and ongoing for 1 month |
| 7.   | Evaluation Period   Establish an evaluation period. Assess if metrics meet success criteria.                           | Data Scientist                  | 1 month post-deployment                         |
| 8.   | Rollback Criteria   Define clear thresholds for when to initiate a rollback (e.g., performance metrics drop below X%). | Project Manager                 | 1 week before deployment                        |
| 9.   | Rollback Execution (if needed)   If performance issues arise, revert to the previous model version.                    | Machine Learning Engineer       | As needed during evaluation period              |
| 10.  | Root Cause Analysis   Analyze performance issues and identify root causes if the rollback occurs.                      | Data Scientist and Data Analyst | Within 1 week of rollback                       |
| 11.  | Model Refinement   Refine the model based on findings. Explore alternative algorithms or feature adjustments.          | Data Scientist                  | 2 weeks after rollback                          |
| 12.  | Re-deployment Testing   Test the refined model thoroughly before redeployment.                                         | Data Scientist                  | 1 week after refinement                         |
| 13.  | Re-deployment   Deploy the updated model after successful testing.                                                     | Machine Learning Engineer       | 1 day after testing completion                  |
| 14.  | Ongoing Monitoring and Feedback   Continue monitoring performance and collect feedback for continuous improvement.     | Data Scientist                  | Ongoing after re-deployment                     |

### ### Summary of Owners and Timing

- Data Engineering Team: Pre-deployment backup (1 week before deployment).
- Data Quality Analyst: Final data quality check (3 days before deployment).
- Data Scientist: Monitoring setup (3 days before deployment) and performance monitoring (ongoing).
- Project Manager: Stakeholder training (2 days before deployment) and rollback criteria definition (1 week before deployment).
- Machine Learning Engineer: Model deployment (1 day after training), rollback execution (if needed), and re-deployment (1 day after testing completion).
- Data Scientist and Data Analyst: Root cause analysis (within 1 week of rollback) and model refinement (2 weeks after rollback).

This structured plan ensures all tasks are assigned, timelines are defined, and the team is prepared for a successful deployment and potential rollback process if needed.

**Engineering counterpart****3:37 AM - Responded in 0s**

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

**Agent response****3:37 AM - Responded in 3s**

The one thing I'd want a second pair of eyes on is the data quality assessment and validation process before deploying the churn prediction model.

### Rationale:

- Data is Fundamental: The accuracy and performance of the churn model heavily depend on the quality of the input data. Any issues related to missing values, inaccurate records, or inconsistencies can severely impact the model's predictive capabilities.
- Comprehensive Review: It's crucial to ensure that every relevant feature is scrutinized for quality-including checking for outliers, distributions, and unexpected patterns that could indicate data entry errors or changes in user behavior.
- Potential Blind Spots: A fresh perspective may catch errors or inconsistencies that the original team might overlook, especially if they are too close to the data or the model.

### Specific Areas for Review:

1. Completeness: Are there any missing values in the critical features?
2. Consistency: Are the data formats uniform across all records and features?
3. Accuracy: Is there a process in place to validate the factual accuracy of critical data points (like customer engagement metrics)?
4. Relevance: Are all included features genuinely relevant and correctly calculated?

Engaging a second opinion on this aspect can help bolster confidence that the model is built on a strong foundation and minimize risks related to data quality once the model is in production.

**Engineering counterpart****3:37 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.