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: qwen/qwen-2.5-7b-instruct
Roles: Data Scientist (AI agent) <> Engineering counterpart (AI agent)
Duration: 48s
Actual run cost: $0.001320
Projected execution cost: $0.2904/mo (10/biz day × 22 days)
Report date: July 14, 2026 - 3:39 AM EST

Session: sample-lb-rankings-seed-50034c0e-ebc8-4f1e-b1e3-c081989603aa

ds_feature_leakage_v1 - agent - openrouter - qwen/qwen-2.5-7b-instruct - ended

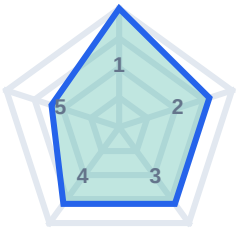
Overall

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: 20484 chars - Latency: 47s - 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: 48s

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

Engineering counterpart
7:38 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
7:38 AM - Responded in 7s

Feature Leakage Review

To ensure our churn model is robust and reliable, we need to identify and remove features that could leak post-churn information. Leakage typically occurs when features contain information known only after the time of prediction, which can lead to overly optimistic performance during cross-validation.

Candidate Features:

```json
{
  "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"
}
```

Leakage Analysis:

1. `days_since_last_login`
   - Issue: If the customer has churned, then `days_since_last_login` could be very high or undefined if the account is already closed.
   - Fix: Ensure this feature is only populated for active accounts and does not contain information about churned accounts.

2. `support_tickets_last_30d`
   - Issue: This feature includes the number of support tickets in the last 30 days, which might be high for churned accounts who might have had frequent issues.
   - Fix: Use this feature but ensure it is aggregated for active accounts only.

3. `mrr_usd`
   - Issue: This feature represents the current monthly recurring revenue. If a customer churns, `mrr_usd` would typically drop.
   - Fix: Use this feature but ensure it is only used for active accounts.

4. `contract_renewal_date`
   - Issue: The contract renewal date is only known for active accounts, and if the contract has not been renewed, it might indicate churn.
   - Fix: Use this feature but ensure it is only used for active accounts.

5. `days_until_contract_end`
   - Issue: Similar to `contract_renewal_date`, the days until contract end is only meaningful for active accounts.
   - Fix: Use this feature but ensure it is only used for active accounts.

6. `cancel_request_submitted_last_7d`
   - Issue: This indicates if a cancel request was submitted within the last 7 days, which strongly correlates with churn.
   - Fix: Use this feature but ensure it is only used for active accounts.

7. `cancel_request_submitted_last_30d`

- Issue: This indicates if a cancel request was submitted within the last 30 days.
- Fix: Use this feature but ensure it is only used for active accounts.

8. `account_status_is_churned`
   - Issue: This is the label we are trying to predict. Using it as a feature will lead to perfect predictions.
   - Fix: REMOVE this feature entirely.

9. `last_invoice_paid_amount`
   - Issue: If the customer has churned, they might not have made a recent payment.
   - Fix: Use this feature but ensure it is only used for active accounts.

10. `nps_score_last_quarter`
    - Issue: The Net Promoter Score (NPS) might be low for churned accounts but should be considered for active accounts.
    - Fix: Use this feature but ensure it is only used for active accounts.

Recommendations for Safe Features:

- `days_since_last_login`: Ensure it is only populated for active accounts.
- `support_tickets_last_30d`: Ensure it is only populated for active accounts.
- `mrr_usd`: Ensure it is only populated for active accounts.
- `contract

Engineering counterpart
7:38 AM - Responded in 0s

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

Agent response
7:39 AM - Responded in 7s

Certainly! Let's walk through the reasoning step by step.

Problem Context
We are building a churn prediction model for a B2B SaaS company, where the goal is to predict whether a customer will churn within the next 30 days. The model will be trained and evaluated using data from the first day of each month.

Candidate Features
```json
{
  "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"
}
```

Assumptions
1. Prediction Time: The model will predict churn starting from the first day of each month.
2. Label: `churned_within_30d` indicates whether a customer churned within the next 30 days.
3. Features: Each feature is evaluated based on whether it could contain information known only after the prediction time.

Feature Analysis

1. `days_since_last_login`
- Issue: If the customer has churned, `days_since_last_login` could be very high or undefined if the account is already closed.
- Fix: Use this feature but ensure it is only populated for active accounts.

2. `support_tickets_last_30d`
- Issue: This feature includes the number of support tickets in the last 30 days, which might be high for churned accounts who might have had frequent issues.
- Fix: Use this feature but ensure it is only aggregated for active accounts.

3. `mrr_usd`
- Issue: This feature represents the current monthly recurring revenue. If a customer churns, `mrr_usd` would typically drop.
- Fix: Use this feature but ensure it is only populated for active accounts.

4. `contract_renewal_date`
- Issue: The contract renewal date is only known for active accounts, and if the contract has not been renewed, it might indicate churn.
- Fix: Use this feature but ensure it is only used for active accounts.

5. `days_until_contract_end`
- Issue: Similar to `contract_renewal_date`, the days until contract end is only meaningful for active accounts.
- Fix: Use this feature but ensure it is only used for active accounts.

6. `cancel_request_submitted_last_7d`

- Issue: This indicates if a cancel request was submitted within the last 7 days, which strongly correlates with churn.
- Fix: Use this feature but ensure it is only used for active accounts.

7. `cancel_request_submitted_last_30d`
- Issue: This indicates if a cancel request was submitted within the last 30 days.
- Fix: Use this feature but ensure it is only used for active accounts.

8. `account_status_is_churned`
- Issue: This is the label we are trying to predict. Using it as a feature will lead to perfect predictions.
- Fix: REMOVE this feature entirely.

9. `last_invoice_paid_amount`
- Issue: If the customer has churned, they might not have made a recent payment.
- Fix: Use this feature but ensure it is only used for active accounts.

10. `nps_score_last_quarter`
- Issue: The Net Promoter Score (NPS) might be low for churned accounts but should be considered for active accounts.
- Fix: Use this feature but ensure it is only used for active accounts.

Summary of Recommendations

Engineering counterpart
7:39 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response
7:39 AM - Responded in 11s

Before shipping the churn model to production, it's crucial to perform a series of rigorous checks to ensure its robustness, reliability, and fairness. Here's what you should verify:

1. Feature Leakage
- Recheck: Ensure that all features that are relevant to churn (like `days_since_last_login`, `support_tickets_last_30d`, etc.) are only used for active accounts in the training and testing datasets.
- Validation: Use techniques like time-series cross-validation to simulate the prediction time and ensure that no post-churn information is used.

2. Model Performance
- Training and Validation: Evaluate the model using various metrics (e.g., AUC-ROC, precision, recall, F1-score) on a holdout dataset.
- Cross-Validation: Use k-fold cross-validation to ensure the model performs consistently across different subsets of the data.

3. Feature Importance
- Analysis: Check the feature importance scores to understand which features are driving the model's predictions. Ensure that critical features are indeed important and not just surrogates for churn status.
- Validation: Verify that the top features still hold importance when the model is retrained with the cleaned features.

4. Model Interpretability
- Explainability: Use techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to understand how the model makes predictions.
- Documentation: Document the decision-making process and the rationale behind feature selection and model choices.

5. Edge Cases and Anomalies
- Edge Cases: Identify and handle edge cases, such as customers with extremely long `days_since_last_login` or very high `support_tickets_last_30d`.
- Anomalies: Detect and handle any anomalies in the data that could skew the model's performance.

6. Performance in Production
- A/B Testing: Conduct A/B testing to compare the performance of the new model against the existing model in a controlled environment.
- Monitoring: Set up monitoring and alerting systems to track the model's performance in real-time. Monitor key metrics such as prediction accuracy, false positives, and false negatives.

7. Regulatory and Ethical Considerations
- Compliance: Ensure that the model complies with any relevant regulatory requirements and ethical guidelines.
- Bias: Perform bias audits to ensure that the model does not unfairly discriminate against any particular group of customers.

8. Documentation and Training
- Documentation: Document the entire process, including data preprocessing, feature engineering, model training, and validation steps.
- Training: Train relevant stakeholders on how to use and interpret the model, including how to handle false positives and false negatives.

9. Feedback Loop
- Continuous Improvement: Implement a feedback loop to collect and incorporate customer feedback to continuously improve the model.
- User Education: Educate customers on the potential impact of churn predictions and how the model is used.

Example Verification Steps

Feature Leakage Verification
```python
# Example code for time-series cross-validation
from sklearn.model_selection import TimeSeriesSplit

tscv = TimeSeriesSplit(n_splits=5)
for train_index, test_index in tscv.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    # Ensure features are correctly filtered for active accounts
    active_mask = X_train[:, feature_indices].mean(axis=1) > 0  # Assuming feature_indices are the indices of active account features
    X_train_active = X_train[active_mask]
    X_test_active = X_test[~active_mask]
```

Engineering counterpart
7:39 AM - Responded in 0s

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

Agent response
7:39 AM - Responded in 8s

Top Risks and Rollback Plan

Top Risks

1. Feature Leakage
   - Risk: The model might incorrectly predict churn based on post-churn information, leading to overly optimistic performance.
   - Impact: The model won't perform well in real-world applications, leading to incorrect customer actions and potential customer dissatisfaction.

2. Poor Model Performance
   - Risk: The model might not perform well on new data, leading to high false positive or false negative rates.
   - Impact: Incorrect churn predictions could lead to wasted resources and potential loss of customers.

3. Unfair Bias
   - Risk: The model might exhibit bias against certain groups of customers, leading to unfair treatment.
   - Impact: This could damage the company's reputation and lead to legal and ethical issues.

4. Overfitting
   - Risk: The model might overfit to the training data, leading to poor generalization on unseen data.
   - Impact: The model won't perform well when deployed in production.

5. Insufficient Monitoring
   - Risk: Lack of proper monitoring and alerting systems could lead to missed issues and delays in addressing them.
   - Impact: Performance degradation could go unnoticed, leading to customer churn.

Rollback Plan

1. Immediate Action
   - Stop Using the Model: Temporarily stop using the model for churn prediction.
   - Notify Stakeholders: Inform the relevant stakeholders about the issue and the steps being taken to address it.

2. Data Reassessment
   - Re-evaluate Features: Ensure that all features are correctly filtered for active accounts and re-run the time-series cross-validation to verify feature leakage.
   - Retrain Model: Retrain the model with the corrected features and re-evaluate its performance.

3. Model Evaluation
   - Performance Metrics: Use a variety of performance metrics (e.g., AUC-ROC, precision, recall) to evaluate the model's performance on a holdout dataset.
   - Cross-Validation: Perform k-fold cross-validation to ensure the model's robustness.

4. Feature Importance Validation
   - Shapley Values: Use SHAP values to understand which features are driving the model's predictions.
   - Interpretability: Ensure that the top features are indeed critical and not just surrogates for churn status.

5. Bias Audit
   - Bias Detection: Use techniques like bias detection tools to ensure the model is fair and unbiased.
   - Mitigation Measures: Implement mitigation measures if bias is detected.

6. Continuous Monitoring
   - Set Up Monitoring: Implement real-time monitoring and alerting systems to track the model's performance.
   - Log Metrics: Log key metrics and performance indicators to continuously monitor the model's behavior.

7. Stakeholder Communication
   - Regular Updates: Provide regular updates to stakeholders on the progress and any issues encountered.
   - Feedback Loop: Gather feedback from stakeholders and incorporate it into the model improvement process.

8. Documentation
   - Document Changes: Document all changes made to the model, including any fixes and improvements.
   - Change Management: Implement a change management process to ensure that all changes are properly documented and reviewed.

9. Reintroduce the Model
   - Gradual Rollout: Gradually reintroduce the model in production, starting with a small subset of customers.
   - Monitor Closely: Closely monitor the model's performance and continuously collect feedback.

Example Rollback and Reintroduction Steps

Immediate Action
```python
# Stop using the current model for churn prediction
model_in_production = False

# Notify stakeholders
notify_stakeholders("Churn prediction model has been suspended.")

# Re-evaluate features and retrain the model
```

Retrain and Re-

Engineering counterpart
7:39 AM - Responded in 0s

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

Agent response
7:39 AM - Responded in 8s

Concrete Next Steps with Owners and Timing

Immediate Action (0-1 Day)
1. Notify Stakeholders
   - Owner: Data Science Lead
   - Action: Send a detailed email to the data science team, product managers, and relevant stakeholders.
   - Timing: Immediate

2. Stop Using the Model
   - Owner: Engineering Team (Model Deployment)
   - Action: Temporarily stop using the model for churn prediction.
   - Timing: Immediate

Feature Reassessment and Retraining (1-3 Days)
3. Re-evaluate Features
   - Owner: Data Scientist
   - Action: Re-evaluate the features, ensuring they are correctly filtered for active accounts.
   - Timing: Day 1

4. Retrain Model
   - Owner: Data Scientist
   - Action: Retrain the model with the corrected features.
   - Timing: Day 1-2

5. Performance Validation
   - Owner: Data Scientist
   - Action: Use time-series cross-validation to validate the model's performance.
   - Timing: Day 1-2

Model Evaluation (2-4 Days)
6. Evaluate Performance Metrics
   - Owner: Data Scientist
   - Action: Evaluate the model using AUC-ROC, precision, recall, etc.
   - Timing: Day 2

7. Cross-Validation
   - Owner: Data Scientist
   - Action: Perform k-fold cross-validation to ensure robustness.
   - Timing: Day 2-3

Feature Importance Validation (3-5 Days)
8. Shapley Values
   - Owner: Data Scientist
   - Action: Use SHAP values to understand feature importance.
   - Timing: Day 3

9. Interpretability
   - Owner: Data Scientist
   - Action: Ensure the top features are critical and not surrogates for churn.
   - Timing: Day 3

Bias Audit (4-6 Days)
10. Bias Detection

- Owner: Data Scientist
- Action: Use bias detection tools to identify any unfair bias.
- Timing: Day 4

11. Mitigation Measures
- Owner: Data Scientist
- Action: Implement any necessary mitigation measures.
- Timing: Day 4

Continuous Monitoring Setup (6-7 Days)
12. Set Up Monitoring
- Owner: Engineering Team (Monitoring)
- Action: Implement real-time monitoring and alerting systems.
- Timing: Day 4-5

13. Log Metrics
- Owner: Engineering Team (Monitoring)
- Action: Log key metrics and performance indicators.
- Timing: Day 4-5

Gradual Rollout (7-10 Days)
14. Gradual Rollout
- Owner: Engineering Team (Model Deployment)
- Action: Gradually reintroduce the model in production, starting with a small subset of customers.
- Timing: Day 6-7

15. Closely Monitor
- Owner: Monitoring Team
- Action: Closely monitor the model's performance and continuously collect feedback.
- Timing: Day 6-7

Key Owners and Responsibilities

- Data Science Lead: Overall coordination and stakeholder communication.
- Data Scientist: Feature re-evaluation, model retraining, performance validation, feature importance validation, bias audit.
- Engineering Team (Model Deployment): Stopping and reintroducing the model, setting up monitoring.
- Monitoring Team: Continuous monitoring and alerting.

Timeline

| Day |

Engineering counterpart
7:39 AM - Responded in 0s

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

Agent response
7:39 AM - Responded in 5s

One Thing to Get a Second Pair of Eyes On

Feature Filtering for Active Accounts

Why This is Critical
Filtering features for active accounts to avoid leakage is crucial because using post-churn information in the feature set can lead to overly optimistic performance and poor generalization in production. This issue can cause the model to perform well during validation but poorly in real-world scenarios.

What to Check
1. Correct Filtering Logic: Ensure that the logic used to filter features for active accounts is correct.
2. Data Consistency: Verify that the data filtering logic is consistent across different datasets (e.g., training, validation, and test sets).
3. Feature Importance Validation: Ensure that important features remain significant after filtering for active accounts.

Example Code for Feature Filtering

```python
import pandas as pd

def filter_active_accounts(df):
    """
    Filter features for active accounts.
    Active accounts are those without churned status.
    """
    active_mask = df['account_status_is_churned'] == 0
    return df[active_mask]

# Load data
data = pd.read_csv('data.csv')

# Filter features for active accounts
filtered_data = filter_active_accounts(data)

# Ensure features are correctly filtered
active_features = filtered_data.drop(columns=['account_status_is_churned', 'label'])
```

Steps to Verify

1. Check Filter Logic: Ensure the logic in `filter_active_accounts` correctly filters out churned accounts.
2. Validate Feature Importance: Use SHAP values or other interpretability techniques to ensure the importance of key features remains consistent after filtering.
3. Cross-Validation: Use time-series cross-validation to ensure the filtered features lead to robust performance.

Example Verification Steps

```python
from sklearn.model_selection import TimeSeriesSplit

# TimeSeriesSplit for cross-validation
tscv = TimeSeriesSplit(n_splits=5)
```

```
for train_index, test_index in tscv.split(filtered_data):
    X_train, X_test = filtered_data.iloc[train_index][active_features.columns], filtered_data.iloc[test_index][active_features.columns]
    y_train, y_test = filtered_data.iloc[train_index]['label'], filtered_data.iloc[test_index]['label']

    # Train and evaluate the model
    model = ...  # Train the model
    y_pred = model.predict(X_test)

    # Evaluate performance
    auc_roc = roc_auc_score(y_test, y_pred)
    print(f"AUC-ROC: {auc_roc}")
```

By ensuring the features are correctly filtered for active accounts, you can avoid leakage and ensure the model performs well in production. A second pair of eyes can help catch any potential issues in the filtering logic and ensure the model is robust.

Engineering counterpart
7:39 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.