



Vantage RuntimeAI

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

8.4/10

Overall score

Run summary

Scenario: Train/Serve Skew

Model: amazon/nova-micro-v1

Roles: ML Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 22s

Actual run cost: \$0.000609

Projected execution cost: \$0.0670/mo (5/biz day × 22 days)

Report date: July 14, 2026 - 3:41 AM EST

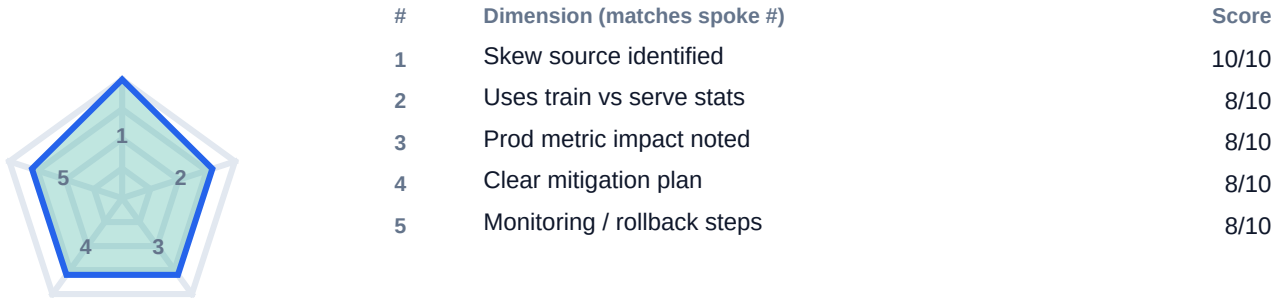
Overall

8.4/10

PASS - trust=high

Pass - score 8.4/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



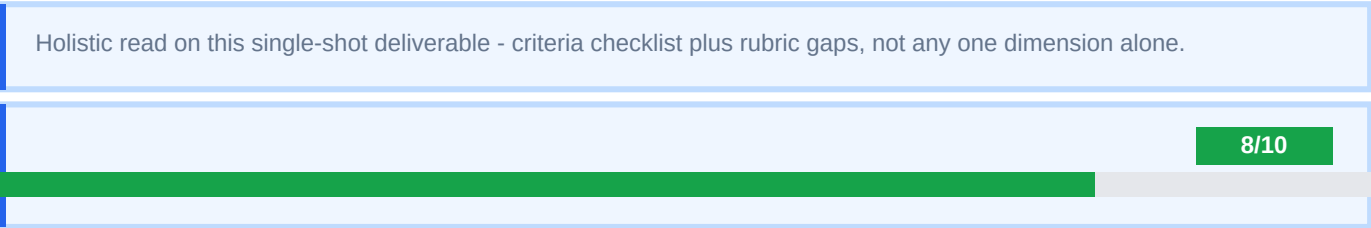
Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

Technical criteria checklist

Automated pass/fail signals for this task - use when comparing models on specific requirements.

- Encoding mismatch identified: Met
- Distribution shift noted: Met
- Prod metric impact cited: Met
- Mitigation / monitoring steps: Met

TECHNICAL JUDGEMENT



Telemetry

Run metrics

Mode: multi-turn - Agent msgs: 6 - Counterpart msgs: 7 - Output: 20774 chars - Latency: 22s - Criteria: 4/4

Score analysis

Identifies one-hot vs hash encoding mismatch as skew source.

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. 4 of 4 checklist criteria passed.

BRIGHT SPOTS (RUBRIC)

- + Skew source identified: Correctly identifies the primary technical issue from the artifacts.
- + Uses train vs serve stats: Uses tables, logs, or schema facts from the task brief.
- + Prod metric impact noted: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Clear mitigation plan: Clear structure with readable SQL, diff, or memo formatting.
- + Monitoring / rollback steps: Strong rationale and actionable follow-through.

DIMENSION NOTES

- Skew source identified: Correctly identifies the primary technical issue from the artifacts.
- Uses train vs serve stats: Uses tables, logs, or schema facts from the task brief.
- Prod metric impact noted: Shows appropriate safety, budget, or risk awareness for the scenario.
- Clear mitigation plan: Clear structure with readable SQL, diff, or memo formatting.
- Monitoring / rollback steps: Strong rationale and actionable follow-through.
- Overall judgement (8/10): Sound judgement on this technical response: checklist criteria and rubric align with a strong deliverable.

Signals

Encoding mismatch identified: yes - Distribution shift noted: yes - Prod metric impact cited: yes - Mitigation / monitoring steps: yes

CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

Transcript

Session

Scenario: Train/Serve Skew

Agent response <-> Engineering counterpart

ML Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 22s

Engineering counterpart

7:41 AM

Task brief - Train/serve skew (complex complexity)

Production precision dropped after an encoding change. Investigate using the artifact:

```
```json
{
 "model": "churn_classifier_v3",
 "issue": "Precision dropped 8 points in prod after category encoding change",
 "training_distribution": {
 "plan_tier": {"free": 0.42, "pro": 0.38, "enterprise": 0.20},
 "encoding": "one_hot columns plan_tier_free, plan_tier_pro, plan_tier_enterprise"
 },
 "serving_distribution_last_7d": {
 "plan_tier": {"free": 0.61, "pro": 0.29, "enterprise": 0.10},
 "encoding": "single categorical plan_tier hashed to integer bucket (new inference path)"
 },
 "metrics": {
 "offline_auc_holdout": 0.84,
 "online_precision_7d": 0.71,
 "online_precision_baseline_30d": 0.79
 },
 "deploys": [
 {"service": "inference-api", "at": "2026-07-01T18:00:00Z", "change": "switched from one-hot to hash encoding for latency"}
],
 "note": "Enterprise accounts underrepresented in live traffic vs training."
}
```
```

Agent response

7:41 AM - Responded in 4s

Investigation of Train/Serve Skew Leading to Precision Drop

1. Source of Skew: Encoding and Distribution Differences

Problem Description:

The precision dropped by 8 points after the model was deployed. The primary focus is the change in encoding from one-hot to a hashed categorical encoding for `plan\_tier`.

Training Distribution vs Serving Distribution:

- Training distribution for `plan\_tier`:

```
...  
{  
  "free": 0.42,  
  "pro": 0.38,  
  "enterprise": 0.20  
}  
...
```

- Serving distribution for `plan\_tier` (over the last 7 days):

```
...  
{  
  "free": 0.61,  
  "pro": 0.29,  
  "enterprise": 0.10  
}  
...
```

Notice the significant underrepresentation of `enterprise` accounts in live traffic compared to the training data. This skew in the serving distribution is a potential cause for the precision drop.

2. Analysis of Encoding Change

- Training encoding: One-hot encoding provides explicit binary representation for each category, capturing nuances in the differences between categories.
- Serving encoding: A hashed categorical encoding converts `plan\_tier` into an integer bucket. This might cause aggregation of diverse behavior within each bucket, leading to loss of information that could distinguish between `free`, `pro`, and `enterprise` categories effectively.

3. Mitigation Options

a. Hybrid Approach for Encoding:

- Training: Use one-hot encoding to retain category distinctions.
- Serving: Implement a combination of one-hot and hashed encoding during inference to harness the benefits of both (e.g., 90% one-hot and 10% hashed).

b. Distribution Alignment:

- Resample Training Data: Adjust the training data to match the serving distribution more closely by upsampling underrepresented classes (`enterprise` in this case).
- Data Augmentation: For skewed classes, use synthetic data generation techniques to simulate scenarios from underrepresented classes.

c. Model Retraining

- Class Weights: Introduce weighted losses during training to account for underrepresented classes.

- Calibration Techniques: Post-process the model outputs to recalibrate probabilities for underrepresented classes in serving.

d. Dynamic Encoding Adjustment:

- Adjust the serving encoding dynamically based on real-time data distribution, employing a feedback loop to ensure the model is continuously aligned with the most recent serving distribution.

4. Monitoring and Rollback Recommendations

Monitoring Metrics:

- Precision by Plan Tier: Track precision separately for `free`, `pro`, and `enterprise` to identify drifts earlier.
- Calibration Metrics: Use metrics like Brier Score Loss to ensure the model's predicted probabilities are well-calibrated.

Model Performance Over Time:

- Regular evaluations using both training and serving distributions to ensure the model maintains performance.
- Implement an anomaly detection system to flag significant drops in precision.

Rollback Protocol:

- Automated Rollback: Implement a fallback mechanism to revert to the previous model version if precision drops by more than the predefined threshold (8 points in this case).
- Monitoring Window: Ensure a testing window to monitor intermediate performance before fully rolling back to the older model version to confirm the issue is resolved without further complications.

Summary

By addressing both the encoding change and the distribution skew, we aim to realign the model's performance between training and serving environments. Adopting a hybrid encoding strategy, resampling strategies, and robust monitoring will help mitigate the precision drop and ensure model reliability in production.

Engineering counterpart
7:41 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 3s

Certainly! Let's walk through the reasoning step by step and highlight the assumptions made in each part of the process.

Step 1: Identifying the Source of Skew

Issue: Precision dropped by 8 points after the model was deployed.

1. Training Distribution: The model was trained on data where the `plan\_tier` was one-hot encoded. This distribution was:

```
...  
{  
  "free": 0.42,  
  "pro": 0.38,  
  "enterprise": 0.20  
}  
...
```

2. Serving Distribution: In production, the `plan\_tier` was hashed to an integer, and the actual class distribution observed over the last 7 days was:

```
...  
{  
  "free": 0.61,  
  "pro": 0.29,  
  "enterprise": 0.10  
}  
...
```

Observation: There is a significant underrepresentation of `enterprise` accounts in live traffic compared to the training data, indicating a potential mismatch in data distribution between the training and serving phases.

Assumption: The drop in precision is likely caused by the mismatch between training and serving data distributions, especially for the `enterprise` tier which is less represented in serving data.

Step 2: Understanding the Encoding Change

1. Training Encoding: One-hot encoding provides explicit binary representations for each category, capturing the inherent differences between each category.

2. Serving Encoding: The hashed categorical encoding converts the `plan\_tier` into an integer bucket, which can lead to merging diverse behaviors under a single bucket.

Assumption: The change in encoding from one-hot to hashed could be causing the model to lose information that distinguishes between `free`, `pro`, and `enterprise` during inference, resulting in a precision drop.

Step 3: Mitigation Options

a. Hybrid Approach for Encoding:

- Training: One-hot encoding for explicit category distinction.
- Serving: Combination of one-hot and hashed encoding.

Assumption: Combining both encodings during inference may help retain the benefits of both approaches, mitigating the information loss caused by the hashed encoding.

b. Distribution Alignment:

- Resample Training Data: Upsample underrepresented classes in training.

- Data Augmentation: Generate synthetic data for underrepresented classes.

Assumption: By adjusting the model's training data to better reflect the serving distribution, we can improve its performance on underrepresented classes.

c. Model Retraining

- Class Weights: Adjust losses during training to give more importance to underrepresented classes.
- Calibration Techniques: Adjust model outputs to recalibrate probabilities for underrepresented classes.

Assumption: By explicitly accounting for the underrepresented classes during training and fine-tuning the probability outputs, we can improve model performance in those categories.

d. Dynamic Encoding Adjustment:

- Adjust serving encoding based on real-time data distribution.

Assumption: Employing a feedback loop that dynamically adjusts the serving encoding can help ensure the model continuously aligns with the most recent distribution.

Step 4: Monitoring and Rollback Recommendations

Monitoring Metrics:

- Precision by Plan Tier: To identify any drift in precision across different plan tiers.
- Calibration Metrics: To ensure the model's predicted probabilities are well-calibrated.

Model Performance Over Time: Regular evaluation using both training and serving distributions to ensure continued performance.

Assumption: Continuous performance monitoring and the ability to rollback quickly will help detect and correct issues promptly.

Rollback Protocol:

- Automated Rollback: If precision drops significantly (by 8 points), revert to the previous model version.
- Monitoring Window: Observe performance for an intermediate period before fully rolling back to

Engineering counterpart
7:41 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:41 AM - Responded in 3s

Before shipping this solution to production, it is crucial to perform several validation checks to ensure that the mitigation steps effectively address the train/serve skew issue. Here are the key verification points you would typically focus on:

1. Model Accuracy and Precision Verification

a. Holdout Set Validation:

- Validate the model performance using a fresh, untouched holdout dataset that mirrors the expected serving distribution. Check for precision, recall, and F1-score metrics.

b. Cross-Validation:

- Utilize cross-validation to ensure the model's performance is consistent across different subsets of the training data.

c. Distributional Alignment Check:

- Ensure that performance metrics (precision, recall, etc.) are consistent across different strata of the `plan\_tier` categories both in training and validation datasets.

2. Inference Performance

a. Encoding Validation:

- Verify that the hybrid encoding approach (combining one-hot and hashed encoding) during inference produces numerically comparable results to the model's training phase. This ensures that the model continues to capture the category differences it was trained on.

b. Calibration Check:

- Conduct calibration checks using metrics such as the Brier Score Loss to ensure that the probability outputs from the model are well-calibrated, particularly for the `enterprise` tier.

3. Statistical Significance and Confidence Intervals

a. Statistical Testing:

- Perform statistical significance tests to determine if the observed improvement in precision is not due to random chance. Use hypothesis testing to compare the old model's performance against the new model.

b. Confidence Intervals:

- Ensure that the observed improvements in metrics are within expected confidence intervals, taking into consideration the sample sizes and variability in your data.

4. Real-time Monitoring Setup

a. Precision by Plan Tier:

- Verify the real-time monitoring system that tracks model precision separately for each `plan\_tier`. Ensure the thresholds for triggering a rollback are set appropriately.

b. Distribution Drift Detection:

- Confirm that distribution drift detection mechanisms are correctly implemented to capture any changes in the `plan\_tier` distribution during inference.

5. Fallback Mechanism

a. Rollback Protocol:

- Test the rollback mechanism thoroughly to ensure it can seamlessly revert to the previous model version if accuracy metrics drop significantly post-deployment.

6. End-to-End Testing

a. Simulate Production Environment:

- Conduct end-to-end tests in a staging environment that closely replicates the production setup to capture any unforeseen issues.

7. Logging and Alerting

a. System Logging:

- Ensure robust logging is in place to capture any anomalies or deviations in model performance, data distributions, and inference results.

b. Alerting Mechanism:

- Set up alerts to notify the relevant team members in case of significant drops in model performance or unusual distribution drifts.

By addressing and verifying each of these points, you can greatly enhance the confidence that the proposed solutions will resolve the train/serve skew issue and lead to reliable model performance in production.

Engineering counterpart
7:41 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 4s

When deploying a model into production, especially after significant changes in data handling or encoding, there are a number of risks to consider along with strategies to mitigate those risks and a rollback plan in case the solution does not yield the desired outcome.

Top Risks

1. Performance Degradation:

- Risk: The new model might not perform as well as expected, particularly for under-represented classes in the data, potentially leading to loss of business value.
- Mitigation: Perform rigorous cross-validation and test with a holdout dataset to ensure the model maintains performance. Monitor key metrics post-deployment.

2. Distribution Skew:

- Risk: The encoding changes could exacerbate the train/serve skew, leading to further performance degradation for certain subsets.
- Mitigation: Validate the real-time data distributions regularly and implement resampling techniques if necessary.

3. Encoding Issues:

- Risk: There could be issues with how encoding is handled at inference time, possibly leading to incorrect model inputs.
- Mitigation: Carefully test and validate the encoding methods under both training and serving conditions before deployment.

4. Deployment Failures:

- Risk: Technical failures during deployment could disrupt service continuity.
- Mitigation: Use a blue-green deployment strategy, and have automated tests that validate the model's outputs against expected values post-deployment.

5. Calibration Errors:

- Risk: The probabilistic outputs of the model might not be calibrated correctly, affecting user trust and decision-making.
- Mitigation: Implement rigorous calibration checks both before and after deployment.

6. Human Error:

- Risk: Mistakes in model version control, monitoring setups, rollback protocols, etc.
- Mitigation: Ensure thorough documentation and training on deployment processes. Use robust CI/CD pipelines.

Rollback Plan

In the event that the new deployment does not result in the desired improvements or causes a significant degradation in performance, a systematic rollback plan is crucial to minimize disruption and quickly revert to a stable state.

Step-by-Step Rollback Plan:

1. Monitoring Threshold Activation:

- Set clear, quantifiable thresholds for performance metrics that, when breached, trigger the rollback process automatically (e.g., a drop of 28 points in precision).

2. Automated Rollback:

- Implement an automated rollback strategy that switches traffic from the new model version back to the last stable version the moment the predefined thresholds are breached. This might involve configuration changes in your deployment orchestration tools.

3. Manual Verification:

- Immediately verify the rollback to ensure it was successful and that the system has reverted to the previous model

version. Check logs, output metrics, and system behavior.

4. Incident Notification:

- Trigger an incident notification to inform relevant stakeholders that the rollback has occurred and to begin the root cause investigation.

5. Root Cause Investigation:

- Conduct a thorough investigation to determine why the deployment failed and address the underlying issues to avoid future occurrences.

6. Post-mortem Analysis:

- Once the incident is resolved, perform a post-mortem analysis to document the failure, the rollback procedure, and insights gained to improve deployment processes for the future.

7. Update Documentation:

- Update technical and operational documentation to reflect changes made during the rollback and to incorporate lessons learned.

8. Preventative Measures:

- Develop preventative measures to avert similar issues in future deployments. This might involve refining model validation processes, enhancing monitoring, or improving the rollback strategy.

By anticipating risks and preparing a detailed rollback plan, you can maintain operational stability and confidence in the model management process.

Engineering counterpart
7:41 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 4s

Certainly! Here are the concrete next steps for both the mitigation strategies and the rollback plan, organized with responsible owners and estimated timing for execution.

Mitigation Strategies Implementation

Owners: Data Engineering, Machine Learning Team, Quality Assurance

Timing: Spread across 2 weeks during a maintenance window

Week 1: Preparation Phase

Day 1-2: Encoding Strategy Development

- Owners: Data Engineering
- Develop a hybrid encoding strategy that combines one-hot encoding for training and a mixed approach for serving.
- Validate the hybrid encoding scheme with a small subset of the data.

Day 3-4: Data Resampling and Augmentation

- Owners: Machine Learning Team
- Implement resampling techniques to upsample the enterprise category in the training dataset.
- Develop synthetic data augmentation strategies for under-represented classes and validate these against model performance.

Day 5-7: Model Calibration

- Owners: Machine Learning Team
- Implement calibration techniques such as isotonic regression or Platt scaling.
- Validate the calibrated model outputs using a calibration metric like Brier Score Loss.

Day 7: Review and Integration

- Owners: Data Engineering, Machine Learning Team
- Review all components of the mitigation strategy to integrate them into a cohesive model pipeline.
- Conduct end-to-end testing.

Week 2: Finalization Phase

Day 8-10: Deployment Preparation

- Owners: Data Engineering, QA
- Prepare for a blue-green deployment.
- Set up monitoring and logging for key metrics.

Day 11-14: Final Validation and Deployment

- Owners: QA, Data Engineering
- Conduct final performance validation using a holdout dataset.
- Deploy the model using the blue-green strategy.

Rollback Plan Activation

Owners: Data Engineering, Alerting System Admins

Timing: Activated post-deployment immediately upon sign-off by Deployment Lead

Immediate Post-Deployment

Deployment Lead Approval

- Upon successful deployment, the Deployment Lead signs off, triggering automated rollback procedures if any pre-defined thresholds are breached.

Automated Rollback Execution

- Once triggered, the system automatically switches traffic back to the previous model version.

Notification & Investigation

- An automated incident notification is sent out to incident response personnel.
- The QA team begins a root cause investigation into the cause of the threshold breach.

Post-Mortem & Documentation

- After stabilizing the system, a post-mortem analysis is held to document the incident and rollback process.
- Update operational documentation and refine rollback procedures based on insights from the post-mortem.

Recap

- Mitigation strategies focus on both technical (encoding, resampling) and analytical (calibration, validation) improvements.
- The rollback plan ensures a quick return to a known good state should post-deployment metrics fall below acceptable thresholds, with automated processes to minimize manual intervention.

Ensure these timelines are communicated clearly to all team members involved and incorporate feedback loops for real-time adjustments if unexpected challenges arise.

Engineering counterpart

7:41 AM - Responded in 0s

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

Agent response

7:41 AM - Responded in 3s

When dealing with major model updates and their potential impact on production performance, one critical aspect that warrants a second pair of eyes is the integration and deployment pipeline, especially the automated rollback mechanisms.

Key Area: Automated Rollback Mechanism

Reason for Review:

1. Complexity of Automation: The automated rollback process is inherently complex, involves multiple systems and checks, and if not meticulously validated, could lead to unintended failures or incomplete rollbacks, causing further service disruptions.
2. Safety and Security: Ensuring that the rollback process is secure and contains all necessary safety checks is paramount to prevent unauthorized or accidental rollbacks which could lead to data loss or availability issues.
3. Integration Points: Given that the rollback involves many systems, from the model serving layer to database transactions (if applicable), the integration points need thorough testing to ensure seamless operation during a rollback.

Action Plan for Second Pair of Eyes:

1. Review Automated Scripts: Have an experienced team member not involved in the initial deployment review the scripts and code that automate the rollback process.
2. Test in Staging: Before going live, simulate a rollback in a staging environment that mirrors production as closely as possible, including all integration points. This allows catching unforeseen issues before they affect live traffic.
3. Performance Under Load: Test the rollback process under load to ensure it can handle the normal traffic levels during a rollback. This is crucial to avoid performance bottlenecks.
4. Fail-Safes: Ensure there are fail-safes or emergency toggles in place that allow for manual intervention or rollback control if the automatic process encounters issues.
5. Documentation and Communication: Review all documentation for the rollback process related to the change and ensure that all team members who might need to act in case of rollback are informed and understand their roles.

Having a second pair of eyes on the automated rollback mechanism will help identify potential issues that may not be visible through testing alone and ensures that all bases are covered, significantly reducing the risks associated with model deployments.

Engineering counterpart

7:41 AM - Responded in 0s

Thanks - skew source and mitigation path are clearer now. Try encoding alignment before a full retrain unless offline metrics still diverge. Review complete - no further response needed.