



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: meta-llama/llama-3.3-70b-instruct

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

Duration: 2m 49s

Actual run cost: \$0.002010

Projected execution cost: \$0.2211/mo (5/biz day × 22 days)

Report date: July 14, 2026 - 3:45 AM EST

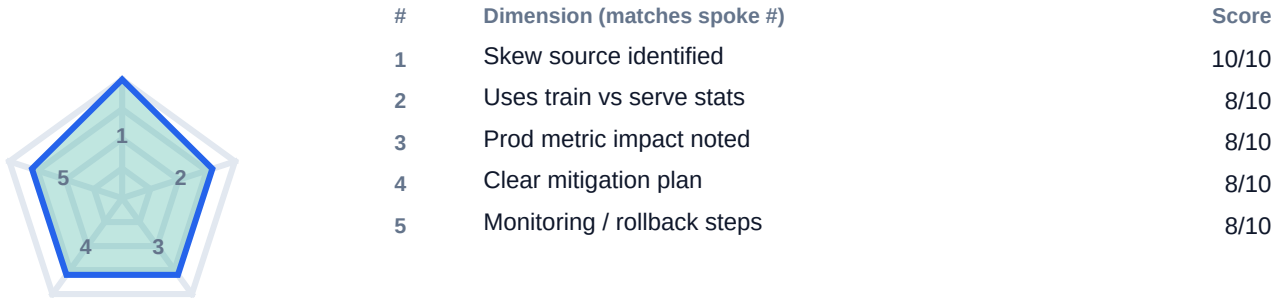
Overall

8.4/10

PASS - trust=high

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

CAPABILITY RUBRIC



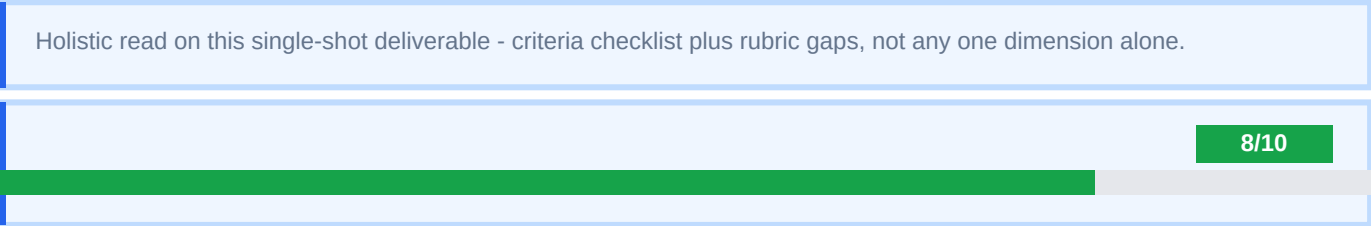
Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

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: 16390 chars - Latency: 169s - 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: 2m 49s

Engineering counterpart

7:42 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

7:44 AM - Responded in 1m 25s

Investigation Report: Train/Serve Skew

Introduction

The production precision of the `churn\_classifier\_v3` model has dropped by 8 points after a category encoding change. This report aims to identify the source of the train/serve skew, provide mitigation options, and recommend monitoring and rollback strategies.

Skew Source

The skew source is attributed to two main factors:

1. **Distribution Shift:** The training distribution of `plan\_tier` is significantly different from the serving distribution. The training data has a distribution of `free`: 0.42, `pro`: 0.38, and `enterprise`: 0.20, whereas the serving distribution is `free`: 0.61, `pro`: 0.29, and `enterprise`: 0.10. This shift in distribution is likely causing the model to perform poorly on the underrepresented `enterprise` accounts.
2. **Encoding Change:** The encoding scheme was changed from one-hot encoding to a single categorical hash encoding, which may be introducing additional noise or bias into the model.

Mitigation Options

To address the train/serve skew, consider the following mitigation options:

1. **Rebalance Training Data:** Rebalance the training data to match the serving distribution. This can be achieved by oversampling the underrepresented `enterprise` accounts or using techniques like SMOTE (Synthetic Minority Over-sampling Technique).
2. **Use Domain-Adaptation Techniques:** Apply domain-adaptation techniques, such as adversarial training or multi-task learning, to help the model generalize better to the serving distribution.
3. **Revert Encoding Change:** Revert the encoding change to the original one-hot encoding scheme to eliminate any potential issues introduced by the new encoding.
4. **Use More Robust Encoding Schemes:** Explore more robust encoding schemes, such as label encoding or binary encoding, that can better handle the distribution shift.

Monitoring and Rollback Recommendations

To monitor the situation and determine the best course of action, recommend the following:

1. **Monitor Online Precision:** Continuously monitor the online precision metric to track any changes or improvements.
2. **Compare to Baseline:** Compare the current online precision to the baseline precision (0.79) to determine if the mitigation strategies are effective.
3. **Rollback Encoding Change:** If the online precision does not improve after implementing the mitigation strategies, consider rolling back the encoding change to the original one-hot encoding scheme.
4. **Re-Deploy Model:** If the mitigation strategies are effective, re-deploy the model with the updated encoding scheme and monitor its performance.

Action Plan

1. Implement data rebalancing techniques to match the serving distribution.
2. Explore domain-adaptation techniques to improve model generalization.
3. Revert the encoding change to the original one-hot encoding scheme.
4. Monitor online precision and compare to baseline.
5. Rollback encoding change if necessary.
6. Re-deploy model with updated encoding scheme if mitigation strategies are effective.

By following this action plan, we can identify and address the root causes of the train/serve skew, improve the model's performance, and restore the production precision to its original level.

Engineering counterpart
7:44 AM - Responded in 0s

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

Agent response

7:44 AM - Responded in 4s

I'll break down my reasoning step by step, highlighting the assumptions I made and the information I used from the provided artifact.

Step 1: Understand the Problem

Given information: The production precision of the `churn\_classifier\_v3` model dropped by 8 points after a category encoding change.

Assumptions:

- + The encoding change is the primary cause of the precision drop.
- + The model was performing well before the encoding change.

Initial thoughts: The encoding change might have introduced some issue that's affecting the model's performance.

Step 2: Analyze the Encoding Change

Given information: The encoding scheme was changed from one-hot encoding to a single categorical hash encoding.

Assumptions:

- + The one-hot encoding scheme was working well before the change.
- + The hash encoding scheme might be introducing some noise or bias into the model.

Thoughts: The change in encoding scheme could be causing the model to behave differently, potentially leading to the precision drop.

Step 3: Examine the Distribution Shift

Given information: The training distribution of `plan\_tier` is different from the serving distribution.

Assumptions:

- + The distribution shift is significant enough to affect the model's performance.
- + The model is not robust to changes in the distribution of the input data.

Thoughts: The distribution shift could be causing the model to perform poorly on the underrepresented `enterprise` accounts, contributing to the precision drop.

Step 4: Consider the Metrics

Given information: The offline AUC (0.84) and online precision (0.71) metrics are provided.

Assumptions:

- + The offline AUC is a good indicator of the model's performance on the training data.
- + The online precision is a good indicator of the model's performance on the serving data.

Thoughts: The discrepancy between the offline AUC and online precision might indicate that the model is not generalizing well to the serving distribution.

Step 5: Develop Mitigation Strategies

Given information: The encoding change, distribution shift, and metrics are considered.

Assumptions:

- + Rebalancing the training data or using domain-adaptation techniques can help improve the model's performance on the underrepresented `enterprise` accounts.
- + Reverting the encoding change or using a more robust encoding scheme can help eliminate any issues introduced by the new encoding.

Thoughts: A combination of these strategies might be necessary to address the precision drop and improve the model's performance.

Assumptions Summary

1. The encoding change is the primary cause of the precision drop.

2. The one-hot encoding scheme was working well before the change.
3. The hash encoding scheme might be introducing some noise or bias into the model.
4. The distribution shift is significant enough to affect the model's performance.
5. The model is not robust to changes in the distribution of the input data.
6. The offline AUC and online precision metrics are good indicators of the model's performance.
7. Rebalancing the training data or using domain-adaptation techniques can help improve the model's performance.
8. Reverting the encoding change or using a more robust encoding scheme can help eliminate any issues introduced by the new encoding.

These assumptions were used to guide my reasoning and develop the mitigation strategies. However, it's essential to validate these assumptions and consider additional factors to ensure the effectiveness of the proposed solutions.

Engineering counterpart
7:44 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:44 AM - Responded in 24s

Before shipping the updated model to production, I would verify the following:

Model Performance

1. Offline metrics: Re-run the offline evaluation metrics (e.g., AUC, precision, recall) on the updated model to ensure it performs well on the training data.
2. Online metrics: Run A/B testing or online experiments to evaluate the updated model's performance on a subset of the production traffic.
3. Comparison to baseline: Compare the updated model's performance to the baseline model's performance (before the encoding change) to ensure it meets or exceeds the previous standards.

Encoding Scheme

1. Consistency: Verify that the encoding scheme is consistently applied across all features and models.
2. Robustness: Test the encoding scheme's robustness to different input scenarios, such as missing values, out-of-vocabulary terms, or edge cases.
3. Interpretability: Ensure that the encoding scheme is interpretable and easy to understand, allowing for straightforward debugging and analysis.

Distribution Shift

1. Monitoring: Set up monitoring tools to track the distribution of the input data and detect any significant shifts.
2. Rebalancing: Regularly rebalance the training data to ensure it remains representative of the serving distribution.
3. Adaptation: Implement mechanisms for adapting to changes in the distribution, such as online learning or transfer learning.

Deployment and Rollback

1. Deployment strategy: Ensure a smooth deployment process, with minimal downtime and a clear rollback plan in case of issues.
2. Rollback testing: Test the rollback process to ensure it can be executed quickly and reliably.
3. Monitoring and alerting: Set up monitoring and alerting systems to detect any issues or anomalies after deployment.

Additional Checks

1. Data quality: Verify that the data used for training and testing is accurate, complete, and consistent.
2. Model explainability: Ensure that the model is explainable and provides insights into its decision-making process.
3. Fairness and bias: Test the model for fairness and bias, ensuring it treats all groups equally and without discrimination.

By verifying these aspects, you can ensure that the updated model is reliable, performs well, and meets the required standards before shipping it to production.

Engineering counterpart
7:44 AM - Responded in 0s

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

Agent response

7:44 AM - Responded in 14s

Top Risks:

1. Model performance degradation: The updated model may not perform as well as expected, leading to a further decline in precision.
2. Encoding scheme issues: The new encoding scheme may introduce unexpected behavior or errors, affecting the model's performance.
3. Distribution shift: The serving distribution may continue to shift, rendering the model's training data less relevant and affecting its performance.
4. Technical debt: The changes made to the model and encoding scheme may introduce technical debt, making it more challenging to maintain or update the model in the future.
5. Overfitting or underfitting: The model may overfit or underfit the training data, leading to poor performance on new, unseen data.

Rollback Plan:

1. Monitoring and alerting: Set up monitoring and alerting systems to detect any issues or anomalies after deployment.
2. Baseline comparison: Continuously compare the updated model's performance to the baseline model's performance (before the encoding change).
3. Rollback criteria: Establish clear criteria for rolling back the changes, such as:
 - Precision drops below a certain threshold (e.g., 5% below the baseline).
 - Model performance degrades significantly (e.g., AUC decreases by 10%).
 - Errors or issues arise that cannot be resolved promptly.
4. Rollback procedure:
 - Revert the encoding scheme to the previous version (one-hot encoding).
 - Re-deploy the previous model version (before the encoding change).
 - Update the training data and re-train the model if necessary.
5. Post-rollback analysis: Conduct a thorough analysis to identify the root cause of the issue and implement changes to prevent similar problems in the future.

Contingency Planning:

1. Temporary fixes: Develop temporary fixes or workarounds to mitigate the issue until a more permanent solution can be implemented.
2. Emergency re-deployment: Establish a process for emergency re-deployment of the previous model version or a temporary fix.
3. Communication plan: Develop a communication plan to inform stakeholders, including customers, developers, and management, about the issue and the rollback plan.

By having a clear understanding of the top risks and a well-defined rollback plan, you can minimize the potential impact of any issues that may arise and ensure a smooth recovery process.

Engineering counterpart

7:44 AM - Responded in 0s

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

Agent response

7:45 AM - Responded in 30s

Concrete Next Steps:

1. Verify Updated Model Performance (Owner: ML Engineer, Timing: 2 days)
Re-run offline evaluation metrics (AUC, precision, recall) on the updated model.
Compare results to the baseline model's performance.
2. Test Encoding Scheme Robustness (Owner: ML Engineer, Timing: 3 days)
Test the encoding scheme's robustness to different input scenarios (missing values, out-of-vocabulary terms, edge cases).
Verify that the encoding scheme is consistently applied across all features and models.
3. Monitor Distribution Shift (Owner: Data Engineer, Timing: Ongoing)
Set up monitoring tools to track the distribution of the input data.
Detect any significant shifts in the distribution and alert the ML Engineer.
4. Develop Rollback Plan (Owner: ML Engineer, Timing: 2 days)
Establish clear criteria for rolling back the changes.
Define the rollback procedure (reverting encoding scheme, re-deploying previous model version).
5. Conduct A/B Testing (Owner: ML Engineer, Timing: 5 days)
Run A/B testing or online experiments to evaluate the updated model's performance on a subset of the production traffic.
Compare results to the baseline model's performance.
6. Review and Refine (Owner: ML Engineer, Timing: 2 days)
Review the results of the A/B testing and refine the updated model as needed.
Verify that the updated model meets the required standards.

Rollback Plan:

1. Revert Encoding Scheme (Owner: ML Engineer, Timing: 1 day)
Revert the encoding scheme to the previous version (one-hot encoding).
2. Re-deploy Previous Model (Owner: DevOps Engineer, Timing: 1 day)
Re-deploy the previous model version (before the encoding change).
3. Update Training Data (Owner: Data Engineer, Timing: 2 days)
Update the training data to reflect any changes in the distribution.
Re-train the model if necessary.

Contingency Planning:

1. Develop Temporary Fix (Owner: ML Engineer, Timing: 2 days)
Develop a temporary fix or workaround to mitigate any issues that may arise.
2. Establish Emergency Re-deployment Process (Owner: DevOps Engineer, Timing: 1 day)
Establish a process for emergency re-deployment of the previous model version or temporary fix.
3. Communicate with Stakeholders (Owner: Project Manager, Timing: Ongoing)
Develop a communication plan to inform stakeholders (customers, developers, management) about any issues or changes.

Timeline:

Day 1-2: Verify updated model performance, test encoding scheme robustness

Day 3-5: Monitor distribution shift, develop rollback plan, conduct A/B testing

Day 6-8: Review and refine updated model, revert encoding scheme (if necessary)

Day 9-10: Re-deploy previous model (if necessary), update training data (if necessary)

Ongoing: Monitor distribution shift, communicate with stakeholders

By following this plan, you can ensure a smooth deployment process and minimize the potential impact of any issues that may arise.

Engineering counterpart**7:45 AM - Responded in 0s**

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

Agent response**7:45 AM - Responded in 12s**

If I had to choose one thing to get a second pair of eyes on, it would be the encoding scheme change. Specifically, I'd want to double-check that the new encoding scheme is correctly implemented and that it doesn't introduce any unexpected behavior or biases.

Here's why:

1. Encoding scheme complexity: The new encoding scheme involves hashing categorical variables, which can be tricky to implement correctly.
2. Potential for bias: If the encoding scheme is not properly designed, it can introduce biases into the model, leading to poor performance or unfair outcomes.
3. Impact on model performance: The encoding scheme change has already had a significant impact on the model's performance, so it's crucial to ensure that the new scheme is correct and optimal.

Having a second pair of eyes on the encoding scheme change would provide an added layer of assurance that the implementation is correct and that the model is not introducing any unintended consequences. This would involve reviewing the code, testing the encoding scheme with different inputs, and verifying that the results are as expected.

Some specific questions to ask when getting a second pair of eyes on the encoding scheme change include:

Is the hashing function correctly implemented?

Are the categorical variables properly normalized and scaled?

Are there any potential biases or issues with the encoding scheme that could affect the model's performance?

Have we thoroughly tested the encoding scheme with different inputs and edge cases?

By getting a second pair of eyes on the encoding scheme change, we can increase confidence in the implementation and reduce the risk of introducing unintended consequences into the model.

Engineering counterpart**7:45 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.