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: openai/gpt-4.1-nano
Roles: ML Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 25s
Actual run cost: $0.001740
Projected execution cost: $0.1914/mo (5/biz day × 22 days)
Report date: July 14, 2026 - 3:46 AM EST

Session: sample-lb-rankings-seed-7a1a1413-3515-4320-907a-30985f1cd72a

ds_train_serve_skew_v1 - agent - openrouter - openai/gpt-4.1-nano - ended

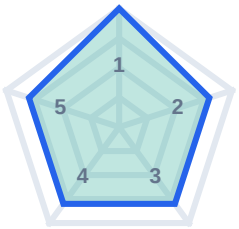
Overall

8.4/10

PASS - trust=high

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

CAPABILITY RUBRIC



#	Dimension (matches spoke #)	Score
1	Skew source identified	10/10
2	Uses train vs serve stats	8/10
3	Prod metric impact noted	8/10
4	Clear mitigation plan	8/10
5	Monitoring / rollback steps	8/10

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.

- Encoding mismatch identified: Met
- Distribution shift noted: Met
- Prod metric impact cited: Met
- Mitigation / monitoring steps: 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: 18037 chars - Latency: 25s - 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: 25s

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

Engineering counterpart
7:45 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
7:45 AM - Responded in 4s

The production precision drop correlates strongly with an encoding change and distribution shift in the `plan_tier` feature. Here's a detailed investigation and mitigation plan:

---

1. Identify the Skew Source

a. Encoding Shift:
- Training: `plan_tier` was encoded with one-hot encoding (`plan_tier_free`, `plan_tier_pro`, `plan_tier_enterprise`).
- Serving (last 7 days): Now encoded as a hashed categorical bucket (single integer), which abstracts the original categories and can introduce distributional differences if bucketing isn't uniform or if certain categories dominate.

b. Distribution Shift:
- Training distribution:
  - Free: 42%
  - Pro: 38%
  - Enterprise: 20%
- Serving distribution (last 7 days):
  - Free: 61%
  - Pro: 29%
  - Enterprise: 10%

Key Observation:
- Enterprise accounts are underrepresented in live traffic compared to training, leading to potential model extrapolation issues.

c. Impact of Change:
- The switch to hashed encoding may have diminished the model's ability to distinguish between categories, especially for underrepresented enterprise accounts, worsening precision.
- The distribution shift amplifies this, as the model has less exposure to enterprise-like data in production.

---

2. Mitigation Options

a. Address Encoding Skew
- Restore or emulate one-hot encoding during inference for categorical features where interpretability and stability matter, especially for minority classes like "enterprise."
- Use consistent encoding techniques:
  - Apply target encoding or frequency encoding if suitable, ensuring the encoding is consistent between train and inference.
  - Alternatively, train with the same encoding method as inference to reduce discrepancy.

b. Improve Distribution Alignment
- Reweight training samples or collect more data from underrepresented categories (enterprise) to balance the training dataset.
- Implement stratified sampling during training to preserve class proportions.
- Deploy an online data collection pipeline to monitor and capture the distribution of features in real-time, aiding future re-training.

c. Model and Feature Engineering
- Incorporate category-aware features that explicitly encode the `plan_tier` rather than relying solely on hashed buckets.
- Use embedding layers trained jointly with the model for categorical features, which can provide robustness against distribution shifts.

d. Model Updating
- Re-train the model with the new encoding method and distribution.
- Consider ensemble models combining models trained on different encodings for robustness.

---

3. Monitoring & Rollback Recommendations

a. Monitoring
- Track per-category precision, recall, and AUC in production.
- Set up drift detection on `plan_tier` distribution, focusing on the enterprise segment.
- Monitor online metrics in real-time (every few hours/days) to catch further degradation early.

b. Rollback Strategy
- If the encoding change is identified as the primary cause, rollback to the previous one-hot encoding.
- Use A/B testing for a subset of traffic with the previous encoding in parallel, validating if precision improves before a full rollout.
- Maintain versioning of models and encodings for quick rollback if needed.

Summary:

Aspect	Findings & Recommendations
Skew Source	Distribution shift + encoding change, underrepresentation of enterprise accounts
Encoding Issue	Hashing introduces ambiguity, losing category-specific info; revert or align encoding method
Distribution Shift	Underrepresented enterprise accounts in live traffic; re-sample/train

Engineering counterpart
7:45 AM - Responded in 0s

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

Agent response
7:45 AM - Responded in 5s

Certainly! Here's a detailed step-by-step walkthrough of my reasoning process, including the assumptions made at each stage:

Step 1: Analyzing the Reported Issue

Observation:
- Precision dropped by ~8 points after a recent change.

Assumption:
- The primary cause of the drop is the recent change (either encoding or traffic distribution).

Step 2: Comparing Training and Serving Data

Key Data:

Feature	Training Distribution	Serving Distribution (last 7d)
`plan_tier`	Free: 42%, Pro: 38%, Enterprise: 20%	Free: 61%, Pro: 29%, Enterprise: 10%

Reasoning:
- There is a clear distribution shift, especially the underrepresentation of the "Enterprise" segment in production.
- Underrepresented classes are often more sensitive to model performance issues, especially if the model wasn't trained with sufficient data to generalize well on these.

Assumption:
- The model may have overfit or become less accurate on underrepresented classes because it saw fewer examples during training, especially with a different encoding.

Step 3: Analyzing the Encoding Change

- Training encoding: One-hot for each `plan_tier` category.
- Inference encoding: Hashed categorical, a single integer bucket.

Reasoning:
- Hashing reduces the explicit category information - the model cannot directly associate the bucket with specific plan tiers.
- For minority classes like "Enterprise," this mapping may be weak or inconsistent, especially if the hash function scatters them unevenly.

Assumption:
- The switch to hashed encoding was primarily motivated by latency improvements, but this may have compromised the model's capacity to distinguish categorical nuances, particularly for underrepresented classes.

Step 4: Linking the Encoding Shift to Performance Drop

- The precision drop coincides temporally with the encoding change.

- The underrepresentation of enterprise accounts in live data likely worsens this, since the model's learned parameters for these classes are less stable or absent in the new encoding.

Assumption:
- The performance degradation is primarily due to the change in encoding combined with the distribution shift, rather than other factors like feature drift or model errors.

Step 5: Identifying the Source of Skew & Its Impact

- The underrepresented enterprise accounts in production likely cause the model to perform poorly when those accounts are involved.
- The hashing approach may obscure the meaning of these categories, especially for small counts, leading to worse predictions, especially for minority classes.

Assumption:
- Reintroducing or maintaining explicit category encoding (like one-hot or similar) aligned with training will improve performance for those classes.

Step 6: Formulating Mitigation Strategies

Options considered:
- Revert to previous encoding (one-hot).
- Use consistent encoding across training and inference.
- Collect more data for underrepresented classes.
- Use embedding layers to handle categorical features more robustly.
- Re-train the model with the new encoding, possibly with class rebalancing.

Assumption:
- Consistency between training and inference encodings is critical to prevent skew-induced performance issues.
- Increasing sample sizes or rebalancing will improve model robustness for minority classes.

Step 7: Monitoring and Rollback

- Set up per-category performance metrics to detect issues early.
- Use drift detection on the distribution of `plan_tier`.
- Implement a rollback plan to previous encoding if necessary.

Assumption:
- That the previous encoding was more stable and better aligned with training data, and that reverting it will immediately improve performance.

Overall Assumptions Summary:

- The primary cause of the precision drop is the combination of encoding change and distribution shift

Engineering counterpart
7:45 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response
7:45 AM - Responded in 7s

Before deploying any mitigation strategies or rolling back, I would verify the following to ensure stability and effectiveness:

---

1. Data Consistency & Distribution

- Confirm the distribution shift persists:
  - Monitor the last 7+ days of traffic to see if `plan_tier` distribution remains skewed or has stabilized.
- Check sampling and data collection pipelines:
  - Ensure that the data collection process accurately reflects production traffic, including minority classes like enterprise.

---

2. Encoding Consistency & Compatibility

- Verify implementation of the encoding used in the model deployment:
  - Confirm that the inference pipeline applies the correct encoding (either restored one-hot or consistent hashing).
- Test encoding mappings:
  - For the "enterprise" category, verify that the hash bucket aligns with the model's expectations or that the one-hot encoding is correctly reconstructed.

---

3. Model Performance on a Controlled Subset

- Perform A/B testing:
  - Deploy a subset of traffic with the previous encoding (if reverting) or an improved encoding.
  - Compare offline metrics (precision, recall, AUC) and online metrics in real-time.
- Collect per-class metrics:
  - Specifically monitor performance on enterprise vs. non-enterprise accounts.

---

4. Offline & Online Validation

- Offline reruns on recent data:
  - Evaluate the model with recent production-like data, ensuring that the new encoding and distribution are accounted for.
- Simulate production traffic:
  - Run shadow tests or canary deployments to validate the impact before full rollout.

---

5. Model Failures & Edge Cases

- Check error injection or failure modes:
  - Ensure no encoding errors (e.g., missing mappings) occur during inference.
- Monitor latency and throughput:
  - Confirm that returning to previous encoding does not reintroduce latency issues that prompted the switch.

---

6. Long-term Monitoring & Rollback Plan

- Establish alerts:
  - Set up real-time alerts on key metrics (precision, class-wise performance, distribution drift).
- Have a rollback plan ready:
  - Prepare scripts and procedures to revert to previous stable models if unforeseen issues arise post-deployment.

---

7. Documentation & Communication

- Document all changes:
  - Record encoding methods, data observations, and performance metrics.
- Communicate with stakeholders:
  - Share findings, expected impacts, and contingency plans with relevant teams.

In summary:
Verification involves ensuring data integrity, consistent encoding practices, thorough offline and online validation, and a clear rollback and monitoring plan before making any live changes.

Engineering counterpart
7:45 AM - Responded in 0s

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

Agent response
7:45 AM - Responded in 4s

Top Risks

1. Further Precision Degradation or Bias for Minority Classes (Enterprise Accounts):
   - Underrepresented categories may perform worse if the encoding isn't properly aligned, leading to higher false positives or negatives.

2. Model Instability Due to Encoding or Distribution Shift:
   - Reverting to previous encoding might cause inconsistencies if other parts of the pipeline were also changed, or if model parameters are no longer aligned with the data.

3. Latency or Throughput Regressions:
   - Reintroducing previous encoding methods (e.g., one-hot) could impact inference latency or resource consumption, especially if the new encoding was optimized for speed.

4. Operational Overhead or Data Quality Issues:
   - Incorrect mappings, missing categories, or encoding bugs could cause inference errors or degraded model outputs.

5. User Impact & Business Risks:
   - Poor performance on critical segments might lead to customer dissatisfaction, increased churn, or missing revenue opportunities in major segments like enterprise.

Rollback Plan

Objective: Quickly restore known stable conditions if the current deployment causes unexpected or unacceptable issues.

Steps:

1. Pre-deploy Backup & Versioning:
   - Ensure previous model versions and encoding pipelines are stored and accessible.
   - Label current production setup clearly to allow rapid reversion.

2. Implement Immediate Rollback:
   - Switch traffic back to the previous, well-performing model version with the original encoding (e.g., one-hot encoding).
   - Use feature flags or deployment tools that allow quick toggling.

3. Validate Post-Rollback:
   - Monitor key metrics (precision, recall, distribution stability) in real-time to confirm recovery.
   - Re-run offline validation on recent data with the previous version to confirm restored performance.

4. Troubleshoot & Isolate Issues:
   - Diagnose whether the encoding, data, or other pipeline components caused the failure.
   - Address underlying issues before planning a safer re-deployment.

5. Communicate & Document:
   - Notify stakeholders of the rollback and reasons.
   - Document the incident, root cause, and lessons learned for future prevention.

Contingency Summary

- If the new approach (e.g., hashed encoding) fails or causes performance issues, immediately revert to the last known stable model with the original encoding.
- Use automation (deployment flags, version controls) for rapid rollback.
- Continue monitoring and testing to understand the root cause before attempting re-deployment with improvements.

In essence:
Proactively prepare a rollback plan involving quick switch-back to previous stable models, thorough monitoring, and communication. This minimizes downtime and mitigates risk of operational or business impact.

Engineering counterpart
7:45 AM - Responded in 0s

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

Agent response
7:46 AM - Responded in 4s

Certainly! Here's a concise, actionable plan with clearly defined next steps, owners, and timing:

Immediate Next Steps (Within 24 Hours)

Step	Action	Owner	Timing	Notes
1	Verify current encoding implementation in inference pipeline	ML Infrastructure Engineer	Within 4 hours	Ensure the encoding (hash vs. one-hot) is correctly applied and test with sample data.
2	Deploy a quick validation batch or A/B test comparing previous encoding (one-hot) vs. current	MLOps Engineer / Data Scientist	Within 8 hours	Validate model performance offline and in a controlled production environment.
3	Monitor live metrics (precision, recall, class-wise performance, distribution drift)	Data/ML Ops Team	Continuous	Set up dashboards and alerts for real-time monitoring.

Short-term (Next 1-2 Days)

Step	Action	Owner	Timing	Notes
4	Prepare rollback scripts and deployment switch	DevOps / MLOps Engineer	Within 24 hours	Ensure a one-click revert to the previous stable model/version.
5	Confirm data consistency and distribution	Data Engineering	Within 24 hours	Verify recent traffic reflects expected distributions, especially for minority classes.
6	Communicate plan to stakeholders	Product Lead / ML Lead	Within 24 hours	Share current status, risks, and next steps.

Follow-up (Next 2-3 Days)

Step	Action	Owner	Timing	Notes
7	Execute rollback if performance deteriorates	DevOps / MLOps	Immediate upon issue detection	Switch traffic back to previous version, monitor re-performance.
8	Investigate root cause of performance issues	Data Scientist / Data Engineer	Within 48 hours after rollback	Check encoding integrity, distribution shifts, and data quality.
9	Plan re-deployment with improved encoding/robust features	ML Engineer / Data Scientist	Within 3 days	Incorporate learnings, re-train if needed, and test thoroughly before redeploy.

Contingency & Monitoring

- Set up alerts on critical metrics (precision drop, distribution anomalies).
- Document all actions and findings for post-incident review.
- Schedule periodic review of data distributions and model performance.

Summary:
- Immediate: verify encoding, run validation, prepare rollback tools (by end of day).
- Short-term: monitor closely, implement a rollback plan if needed, communicate with stakeholders.
- Long-term: root cause analysis, iterative improvement, and re-deployment plan.

This structure ensures clear ownership, accountability, and timely response to mitigate risks effectively.

Engineering counterpart
7:46 AM - Responded in 0s
What's the one thing you'd want a second pair of eyes on?

Agent response
7:46 AM - Responded in 2s
The one thing I'd want a second pair of eyes on is the consistency and correctness of the encoding implementation in the inference pipeline, specifically:

- How the `plan_tier` feature is being encoded during inference (hashed buckets vs. one-hot).
- Whether the recent deployment has inadvertently introduced encoding mismatches or bugs-for example, a mismatch between training and inference mappings, or incorrect hash seed/configuration.

Why?
Because encoding mismatches are often subtle but critical. If the inference pipeline is not correctly aligned with training, it can cause significant performance issues, especially for underrepresented categories. Confirming this ensures that any performance decline isn't due to an implementation bug rather than data or distribution issues.

Actionable items for review:
- Share the current inference code/configuration.
- Verify the hash seed, bucket count, and how categories map to buckets.
- Cross-check with training preprocessing code.
- Test with sample data to confirm consistent encoding.

Summary:
A second review here reduces the risk of silent bugs causing the observed precision drop, ensuring that the root cause analysis and subsequent mitigation are based on a correct understanding of the data transformations.

Engineering counterpart
7:46 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.