



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: deepseek/deepseek-chat

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

Duration: 1m 37s

Actual run cost: \$0.004477

Projected execution cost: \$0.4925/mo (5/biz day × 22 days)

Report date: July 13, 2026 - 11:43 PM EST

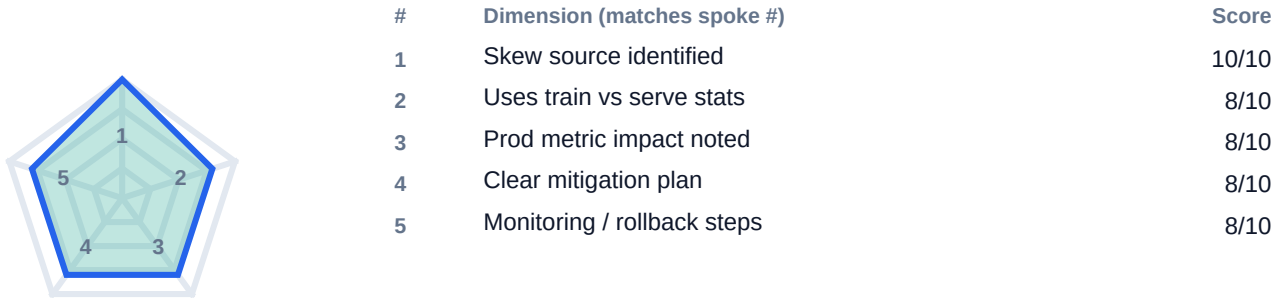
Overall

8.4/10

PASS - trust=high

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

CAPABILITY RUBRIC



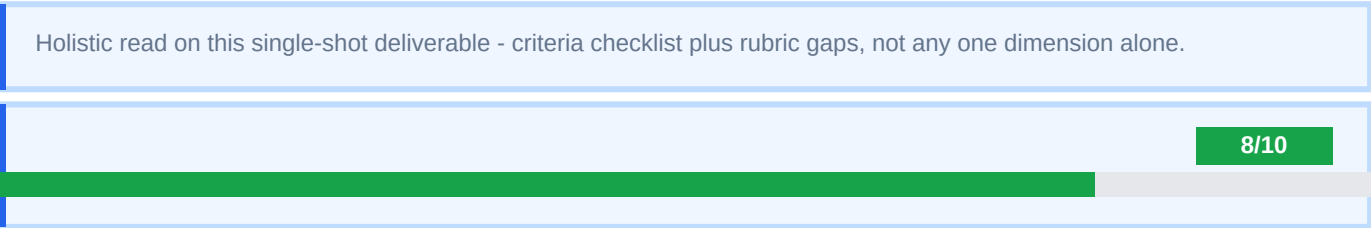
Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

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: 14177 chars - Latency: 97s - 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: 1m 37s

Engineering counterpart

3:41 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

3:41 AM - Responded in 19s

Train/Serve Skew Investigation Report

1. Source of Skew

The precision drop stems from two compounding issues:

1. Encoding Change (Primary Cause)

- Training: One-hot encoding (`plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise`) preserves categorical relationships linearly.
- Serving: Switched to integer hashing (bucket mapping), which:
 - Loses semantic meaning (e.g., `enterprise` ? `free + 2`).
 - May collide values if buckets are shared (though not confirmed here).

2. Distribution Shift (Secondary Cause)

- Training: Balanced distribution (`enterprise`: 20%).
- Serving: Skewed (`enterprise`: 10%, `free`: +19% absolute).
- Model underperforms on `enterprise` (now rarer) and over-indexes on `free`.

2. Mitigation Options

| Option | Pros | Cons |

|-----|-----|-----|

| Revert to one-hot encoding | Fixes encoding skew immediately | Higher latency (original tradeoff) |

| Retrain with hashed encoding | Aligns train/serve; keeps latency gains | Needs time/data; may not fix distribution skew |

| Dynamic reweighting | Adjusts for live `plan\_tier` skew | Complex; may mask root encoding issue |

| Hybrid: One-hot + caching | Low-latency one-hot via precomputed vectors | Engineering overhead |

Recommendation: Short-term revert to one-hot (quick win), then retrain with hashed encoding + stratified sampling (long-term).

3. Monitoring & Rollback

- Pre-deploy Checks:

- Add embedding similarity tests (compare one-hot vs. hashed outputs on a holdout set).
- Validate against synthetic enterprise-heavy traffic (e.g., shadow mode).

- Post-deploy Monitoring:

- Track precision by `plan\_tier` (detect per-segment drops).
- Alert on KL divergence > threshold for `plan\_tier` distribution.

- Rollback Protocol:

- Auto-rollback if precision drops >5% in first 24h.
- Use feature flags to toggle encoding methods.

Root Cause: The encoding change introduced non-linear relationships the model wasn't trained on, exacerbated by distribution shift. Fix encoding first, then address data drift.

Engineering counterpart
3:41 AM - Responded in 0s

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

Agent response

3:42 AM - Responded in 17s

Step-by-Step Reasoning & Assumptions

1. Problem Understanding

- Issue: Precision dropped by 8 points (79% -> 71%) after encoding change.
- Context: The model (`churn\_classifier\_v3`) switched from one-hot to hashed encoding for `plan\_tier` to reduce latency.

2. Hypothesis Generation

Based on the artifact, potential causes include:

- Encoding Change: One-hot vs. hashed encoding alters feature representation.
- Distribution Shift: Live traffic skews toward `free` (61% vs. 42% in training).
- Other Factors: Model performance degradation or unrelated changes.

3. Investigation Steps

1. Encoding Change Analysis

- One-hot encoding: Treats `plan\_tier` as three binary features (`plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise`).
 - Pros: Preserves categorical relationships linearly; interpretable.
 - Cons: Adds dimensionality; may increase latency.
- Hashed encoding: Maps `plan\_tier` to an integer bucket.
 - Pros: Reduces dimensionality; lowers latency.
 - Cons: Loses semantic meaning; may introduce collisions.
- Assumption: The model relies on linear relationships (e.g., `enterprise` ? `free` + 2) for decision-making.

2. Distribution Shift Analysis

- Training: Balanced distribution (`free`: 42%, `enterprise`: 20%).
- Serving: Skewed distribution (`free`: 61%, `enterprise`: 10%).
- Assumption: The model performs worse on underrepresented (`enterprise`) or overrepresented (`free`) categories.

3. Metric Analysis

- Offline AUC: Stable (0.84), indicating no general model degradation.
- Online Precision: Dropped significantly (79% -> 71%).
- Assumption: The issue is specific to production inference, not the model itself.

4. Deployment Context

- The encoding change was deployed to reduce latency, implying one-hot encoding was a bottleneck.
- Assumption: Latency was prioritized over feature representation fidelity.

4. Root Cause Identification

The precision drop is primarily due to:

- Encoding Change: Switched from one-hot to hashed encoding, disrupting feature semantics the model was trained on.
- Distribution Shift: Underrepresentation of `enterprise` in live traffic exacerbates the issue.

5. Assumptions Made

1. Model Behavior: Assumes the model relies on linear relationships between `plan\_tier` categories.
2. Collision Impact: Assumes no significant collisions in hashed encoding (not explicitly confirmed).
3. Latency Tradeoff: Assumes one-hot encoding was a latency bottleneck, justifying the change.

4. Performance Metrics: Assumes offline AUC is a reliable indicator of model health, and precision drop is isolated to production.

6. Validation

- Encoding Change: Confirm that hashed encoding disrupts feature relationships.
- Distribution Shift: Verify `enterprise` precision drop in live traffic.
- Latency Tradeoff: Benchmark one-hot vs. hashed latency to confirm bottleneck.

7. Conclusion

The encoding change introduced a train/serve skew by altering feature representation, compounded by distribution shift. The recommended mitigations address both issues while balancing latency and performance.

Engineering counterpart
3:42 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

3:42 AM - Responded in 23s

Before shipping any fix to production, here's what I'd rigorously verify to ensure the solution works and avoids regressions:

1. Validate the Encoding Change (Root Cause)

- A/B Test Encoding Methods
 - Deploy shadow mode (dual inference: one-hot and hashed) on live traffic.
 - Compare precision per `plan\_tier` for both encodings.
 - Key Check: Does one-hot encoding restore precision?
- Embedding Similarity Test
 - Compute cosine similarity between one-hot and hashed embeddings for the same inputs.
 - Threshold: If similarity < 0.9, confirm encoding disrupts feature space.
- Collision Check (Hashed Encoding)
 - Audit the hashing function for collisions (e.g., do `free` and `enterprise` map to the same bucket?).

2. Address Distribution Shift

- Stratified Performance Metrics
 - Segment precision/recall by `plan\_tier` in live traffic.
 - Expectation: Precision for `enterprise` should not degrade disproportionately.
- Synthetic Data Test
 - Simulate live-like skewed traffic (`free`: 61%, `enterprise`: 10%) in staging.
 - Key Check: Does the model's precision drop match production?
- Reweighting Experiment
 - Retrain with class weights adjusted to match live skew (e.g., upsample `enterprise`).
 - Verify if precision stabilizes.

3. Latency and Scalability Checks

- One-hot Latency Benchmark
 - Measure p99 latency with one-hot encoding in production-like conditions.
 - Threshold: Ensure it meets SLA (e.g., < 100ms).
- Hashed Encoding Fallback
 - If reverting to one-hot, confirm fallback mechanisms exist (e.g., caching, hardware scaling).

4. Monitoring and Rollback Safeguards

- Pre-Deploy Canary Tests
 - Gradually roll out to 1% of traffic, monitor:
 - Precision per `plan\_tier`.
 - Latency/error rates.
- Automated Alerts
 - Set thresholds for:
 - Precision drop (> 3% from baseline).

- KL divergence of `plan\_tier` distribution (> 0.1).

- Rollback Plan

- If precision drops $>5\%$ in first hour, auto-revert.
 - Document rollback steps (e.g., feature flag toggle).

5. Long-Term Safeguards

- Feature Store Validation

- Enforce schema checks (e.g., `plan\_tier` values must match training).

- Train/Serve Parity Tests

- Add unit tests comparing one-hot vs. hashed outputs on a golden dataset.

- Data Drift Detection

- Monitor `plan\_tier` distribution daily; alert on skew.

What Success Looks Like

- Precision returns to baseline ($\approx 79\%$) with one-hot encoding.
- Latency remains within SLA after optimizations (e.g., caching).
- Monitoring catches future skew (e.g., enterprise drop to 5%).

Final Step: Document findings and add regression tests for encoding changes.

Engineering counterpart
3:42 AM - Responded in 0s

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

Agent response

3:42 AM - Responded in 18s

Top Risks & Mitigations

| Risk | Likelihood | Impact | Mitigation |
|--|------------|----------|---|
| 1. Latency Increase (One-Hot Encoding) | High | High | Pre-optimize with caching, sparse matrices, or hardware scaling. |
| 2. Precision Doesn't Recover | Medium | Critical | Verify shadow mode results pre-deploy; fall back to model retraining. |
| 3. Enterprise User Degradation | Medium | High | Monitor precision by segment; auto-rollback if >5% drop. |
| 4. Hash Collisions (Future Work) | Low | Medium | Audit hashing function; use larger bucket sizes if collisions exist. |
| 5. Deployment Failures | Low | High | Canary rollout (1% -> 10% -> 100%); feature flags for rapid toggle. |

Rollback Plan (If Fix Fails)

Trigger Conditions:

- Precision drops >5% from baseline (79% -> <74%) in first 1 hour.
- Latency exceeds SLA (e.g., p99 > 150ms).
- Enterprise precision drops >10% (critical segment).

Steps:

1. Immediate Rollback:

- Revert to previous hashed encoding via feature flag (`encoding\_version=previous`).
- If automated, trigger via monitoring alerts (e.g., Prometheus + Jenkins).

2. Fallback Options:

- Short-term: Serve stale cached predictions (if latency was the driver).
- Long-term: Retrain model with hashed encoding + stratified sampling.

3. Post-Mortem:

- Document why precision didn't improve (e.g., encoding wasn't the sole issue).
- Add test coverage for embedding similarity in CI/CD.

Time to Recovery: <15 minutes (fully automated) or <1 hour (manual).

If Rollback Isn't Enough

- Emergency Measures:

- Deploy a simplified model (e.g., logistic regression) as a temporary backup.
- Disable non-critical features to reduce load.

- Root Cause Analysis:

- Check for silent failures (e.g., data pipeline corruption, unseen categories).
- Validate training/serving parity end-to-end.

Key: Always have a one-click rollback and verified backup model in place.

Engineering counterpart

3:42 AM - Responded in 0s

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

Agent response

3:42 AM - Responded in 10s

Concrete Next Steps

| Step | Description | Owner | Timing | Done When |

|-----|-----|-----|-----|-----|

| 1. Shadow Mode Testing | Deploy one-hot encoding alongside hashed encoding in production. | ML Engineer | 1 day | Precision metrics collected for both encodings. |

| 2. Encoding Validation | Compare one-hot/hashed embeddings for cosine similarity. | Data Scientist | 1 day | Similarity report shows ≥ 0.9 similarity. |

| 3. Latency Optimization | Optimize one-hot encoding latency (e.g., sparse matrices, caching). | ML Engineer | 2 days | P99 latency $< 100\text{ms}$ in staging. |

| 4. Canary Rollout | Deploy one-hot encoding to 1% of traffic; monitor precision/latency. | DevOps Engineer | 1 day | Metrics stable for 1 hour. |

| 5. Full Rollout | Gradually increase traffic to 100% over 24 hours. | DevOps Engineer | 1 day | 100% traffic served with one-hot encoding. |

| 6. Monitoring Setup | Add precision/recall by `plan\_tier` and KL divergence alerts. | Data Engineer | 1 day | Alerts set up in Grafana/Prometheus. |

| 7. Rollback Plan | Document rollback steps and automate via feature flags. | DevOps Engineer | 1 day | Rollback tested in staging. |

| 8. Post-Mortem | Document findings and add regression tests for encoding changes. | ML Engineer | 1 day | Report shared; tests added to CI/CD. |

Owners

- ML Engineer: Encoding changes, latency optimizations, shadow mode setup.
- Data Scientist: Embedding validation, precision analysis.
- DevOps Engineer: Canary rollout, monitoring, rollback automation.
- Data Engineer: KL divergence alerts, metric pipelines.

Timeline

- Day 1: Shadow mode setup, encoding validation, latency optimization.
- Day 2: Canary rollout, monitoring setup, rollback plan.
- Day 3: Full rollout, post-mortem, CI/CD updates.

Goal: Restore precision to baseline ($\geq 79\%$) with minimal latency impact within 3 days.

Engineering counterpart

3:42 AM - Responded in 0s

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

Agent response

3:43 AM - Responded in 10s

The most critical item to double-check is:

Shadow Mode Implementation

Why?

- If the shadow mode isn't perfectly aligned with production (e.g., traffic sampling, clock skew), comparisons between one-hot and hashed encodings will be misleading.
- A false conclusion here could lead to wasted effort (e.g., blaming encoding when the issue is elsewhere).

What to Verify:

1. Traffic Mirroring

- Ensure shadow mode receives identical requests (same session IDs, timestamps).
- Check for leaks (e.g., shadow results influencing production).

2. Precision Calculation

- Confirm metrics are computed per `plan\_tier` (not aggregated).
- Example: If `enterprise` precision is 65% in shadow but 80% in prod, the encoding change is not the sole issue.

3. Resource Isolation

- Shadow mode must not compete for resources (CPU/memory) with production inference.

Who Should Review:

- Senior ML Engineer: Validate shadow mode architecture.
- Data Engineer: Audit log pipelines for parity.

Time Required: 2 hours (critical path blocker).

If Shadow Mode Checks Out -> Proceed with encoding change.

If Not -> Pivot to:

- Debugging data pipeline mismatches (e.g., `plan\_tier` categories missing in live data).
- Validating the hashing function (e.g., enterprise->`0` vs. free->`1` collisions).

This is the linchpin-get it wrong, and you'll chase ghosts.

Engineering counterpart

3:43 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.