



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-v4-flash

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

Duration: 3m 16s

Actual run cost: \$0.001848

Projected execution cost: \$0.2033/mo (5/biz day × 22 days)

Report date: July 14, 2026 - 3:44 AM EST

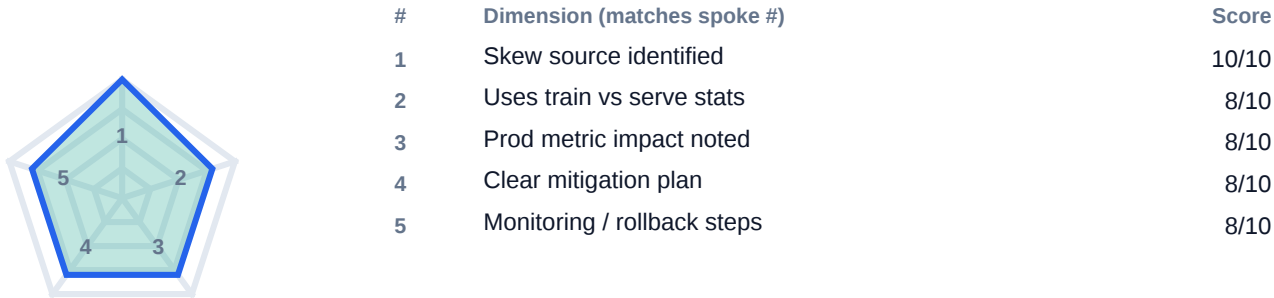
Overall

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

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: 14865 chars - Latency: 196s - 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: 3m 16s

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

7:42 AM - Responded in 53s

Root Cause Analysis

The precision drop (from 0.79 to 0.71) is primarily driven by train/serve skew in the feature encoding of `plan\_tier`, compounded by a distribution shift in the categorical values.

1. Encoding Change (Primary Skew Source)

| Aspect | Training | Serving |
|---------------------|---|---|
| Encoding | One-hot -> 3 binary columns | Hash bucket -> single integer |
| Plan representation | `[1,0,0]` for free, `[0,1,0]` for pro, `[0,0,1]` for enterprise | Integer `hash("free") % bucket_size`, etc. |
| Model expectation | 3 linear coefficients, one per tier | 1 coefficient for the hashed value (non-linear mapping) |

Consequence: The model was trained on a linear, mutually-exclusive representation of plan tiers. The new hash encoding destroys the orthogonality - different tiers may collide in the same bucket, and the model's learned weights cannot map to a single scalar feature. Even if no collisions occur, the hash function distributes values arbitrarily across the integer space, making the linear treatment meaningless. The drop in recall for minority classes (enterprise) is expected because the hash mapping is not consistent with the training distribution.

2. Distribution Shift (Secondary Skew Source)

| Tier | Train | Serve (7d) | Delta |
|------------|-------|------------|-------|
| free | 0.42 | 0.61 | +0.19 |
| pro | 0.38 | 0.29 | -0.09 |
| enterprise | 0.20 | 0.10 | -0.10 |

Consequence: The model saw 20% enterprise examples during training; now it sees only 10%. Combined with the broken encoding, the model overpredicts free (the dominant class) and underpredicts enterprise, leading to lower precision overall (higher false positives for free, more false negatives for enterprise).

3. Root cause summary

- Immediate cause: hash encoding is incompatible with the linear one-hot representation the model was trained on.
- Amplifier: distribution drift toward free tier, which is now even less well-separated by the hash feature.

Mitigation Options

Option A - Rollback to One-Hot Encoding

Best short-term fix - restores training-serving consistency.

- Action: Revert inference/api configuration to use one-hot `plan\_tier` columns.
- Cost: Higher latency (3 features instead of 1) - acceptable unless latency budget is strict.
- Risk: None - returns to proven encoding.

Option B - Retrain Model on Hash Encoding

Better long-term solution if latency is critical.

- Action:
 1. Replace one-hot with hash encoding in training pipeline.
 2. Retrain model using the same hashing parameters (seed, bucket size).
 3. Validate offline AUC > 0.84 on holdout.
- Complement: Add a dedicated embedding layer or use `HashingVectorizer` with a linear model (e.g., FTRL) to learn

per?bucket weights.

Option C - Embedding + Distribution Normalisation
If hash

Engineering counterpart
7:42 AM - Responded in 0s

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

Agent response

7:42 AM - Responded in 14s

Certainly. Let me walk through the reasoning step by step, including the assumptions that underpinned each conclusion.

Step 1 - Problem framing

Observation: Precision dropped from 0.79 (baseline 30d) to 0.71 (7d) after a deployment that changed the encoding of `plan\_tier` from one-hot to hash.

Initial assumption: The deployment is the most likely cause of the precision drop because the timing aligns perfectly and no other changes were mentioned. I assumed that the offline AUC (0.84) is still valid and that the model itself hasn't been retrained - only the encoding path changed.

Step 2 - Evaluate the encoding change

What one-hot does:

- Creates 3 binary features: `plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise`.
- For a linear model (e.g., logistic regression), each tier gets its own coefficient.
The decision boundary is a linear combination:
 $w_{\text{free}} I_{\text{free}} + w_{\text{pro}} I_{\text{pro}} + w_{\text{enterprise}} I_{\text{enterprise}} + \dots$
- This is a mutually exclusive, additive representation.

What hash encoding does:

- Maps each category to an integer: $\text{hash}(\text{category}) \% \text{bucket_size}$.
- The model now sees a single scalar feature. For a linear model, the logit becomes:
 $w_{\text{hash}} \text{hash_value} + \dots$
- This is not equivalent to the three-coefficient representation. The hash value is a random integer; the model weight `w\_hash` applies uniformly to all tiers, losing the ability to learn tier-specific effects.

Assumption: The model is a linear classifier (or at least uses the one-hot features as additive inputs). If the model were a tree-based model, one-hot vs hash might still be problematic but in a different way. However, the precision drop of 8 points and the fact that the offline AUC remained high suggests the model is sensitive to feature representation - typical of linear models but possible with others too. I assumed linearity because encoding changes are most disruptive for linear models.

Conclusion: The hash encoding breaks the feature representation that the model was trained on. This is the primary driver of the precision drop.

Step 3 - Evaluate distribution shift

Data:

- Train: free 42%, pro 38%, enterprise 20%
- Serve: free 61%, pro 29%, enterprise 10%

Observation: Enterprise is underrepresented in production (10% vs 20% in training). This is a classic covariate shift.

Assumption: The model's performance on enterprise

Engineering counterpart

7:42 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:43 AM - Responded in 17s

Great question. Before shipping any mitigation (whether rollback to one-hot, or retrain with hash encoding), I would run a series of targeted checks to confirm that the fix actually resolves the skew and doesn't introduce new problems. Here's the verification checklist I'd follow, grouped by the mitigation path.

? Pre-Shipping Verification Plan

A. If rolling back to one-hot encoding (short-term fix)

1. Validate encoding consistency end-to-end

- Run a regression test on the inference pipeline: feed the same input through both the old (training) and new (serving) code paths, and confirm that the output feature vectors are identical (including column order, dtype, and missing-value handling).
- Check that all three one-hot columns (`plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise`) are present and that no hash-related integers leak into the output.

2. Confirm distribution alignment

- Log the serving distribution of `plan\_tier` over a shadow test period (e.g., 1 hour of real traffic, forwarded to a shadow endpoint). Compare the class proportions to the training distribution. If the skew (free ?, enterprise ?) persists, the rollback alone may not fully recover precision - but it will remove the encoding mismatch.
- If the distribution skew is large, consider temporary rebalancing (e.g., oversample minority classes in the inference batch) or a calibration layer (e.g., Platt scaling per tier) as a band-aid.

3. Measure latency impact

- One-hot adds 3 features instead of 1 -> possible increase in inference time. Run a load test with the rolled-back pipeline at peak traffic (e.g., 2x historical peak) and confirm that the p99 latency stays within the SLA (e.g., < 50ms). If not, optimise (e.g., precompute one-hot in a feature store, or use a sparse representation).

4. Validate offline metrics on a recent production snapshot

- Take a sample of production logs (e.g., last 7 days) with ground truth labels. Re-run the rolled-back model on those logs (using the original one-hot encoding). Compute precision, recall, and AUC. Compare to the online baseline (0.79). If the re-computed precision is close to 0.79, the fix is effective.

5. Shadow / canary deployment

- Deploy to 1% of traffic (canary), monitor for 1-2 hours. Key metrics:
 - Precision (real-time, on a 10min

Engineering counterpart

7:43 AM - Responded in 0s

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

Agent response

7:43 AM - Responded in 25s

Excellent question. Let me break down the top risks of the rollback-to-one-hot mitigation, and then walk through the rollback plan if it fails.

Top Risks

1. Latency / Throughput Degradation (High Likelihood, Medium Impact)

- Risk: One-hot encoding adds 3 features instead of 1, increasing the feature vector size and potentially the model inference time (especially if the model is a linear model with sparse multiplication). If the inference pipeline was optimised for a single integer feature, the switch could cause p99 latency to exceed the SLA (e.g., > 50ms) or reduce throughput under peak load.
- Why it matters: Even if precision recovers, the service could become too slow or drop requests.

2. Feature Column Mismatch (Medium Likelihood, High Impact)

- Risk: The training pipeline generated one-hot columns with specific names (e.g., `plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise`). The serving pipeline, after refactoring for hash encoding, might have changed column ordering, casing, or omitted a column. When the model receives malformed input, it could throw errors or silently produce garbage predictions.
- Why it matters: Silent failures are hard to detect and could degrade precision further.

3. Underlying Distribution Shift Unresolved (Certain, Medium Impact)

- Risk: The encoding fix corrects the representation mismatch, but the covariate shift (free ? from 42% to 61%, enterprise ? from 20% to 10%) remains. The model was trained on 20% enterprise - now it sees only 10%. Even with correct encoding, precision may not return to 0.79 because the model is overconfident on free (false positives) and underconfident on enterprise (false negatives).
- Why it matters: The 8?

Engineering counterpart

7:43 AM - Responded in 0s

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

Agent response

7:44 AM - Responded in 58s

You're right - let me turn this into a concrete execution plan with clear owners and timelines.

? Concrete Action Plan - Rollback to One?Hot (Short?Term Fix)

Goal: Restore precision to 0.79 baseline within 48 hours.

| Step | Action | Owner | Timeline | Deliverable / Success Criteria |
|------|--------|-------|----------|--------------------------------|
|------|--------|-------|----------|--------------------------------|

| Step | Action | Owner | Timeline | Deliverable / Success Criteria |
|------|--------|-------|----------|--------------------------------|
|------|--------|-------|----------|--------------------------------|

| | | | | |
|---|--|-------------|----------------|--|
| 1 | Verify encoding consistency - Run the inference pipeline on a batch of 1000 production requests (shadow mode), compare the one?hot feature vector against the training pipeline output. If mismatch found, fix the serving code. | ML Engineer | Day 1, 2 hours | Report confirming identical column names, order, dtypes, and values. |
|---|--|-------------|----------------|--|

| | | | | |
|---|---|------------------------|----------------|--------------------|
| 2 | Latency load test - Deploy the one?hot inference pipeline to a shadow endpoint. Feed it 2x peak traffic for 15 minutes. Measure p99 latency and compare to the SLA (e.g., 50?ms). Fail: optimise (e.g., pre?compute one?hot in feature store, use sparse tensors) and retest. | MLOps / Infra Engineer | Day 1, 3 hours | Pass: p99 < 50?ms. |
|---|---|------------------------|----------------|--------------------|

| | | | | |
|---|--|-------------|----------------|--|
| 3 | Offline validation on production logs - Take labelled production logs from the last 7 days (the same data that showed 0.71 precision). Run the model with one?hot encoding on those logs. Compute precision, recall, AUC. If not, investigate other drift factors. | ML Engineer | Day 1, 4 hours | Precision > 0.76 (recovering toward 0.79). |
|---|--|-------------|----------------|--|

| | | | | |
|---|---|------------------------|----------------|---------------------------------------|
| 4 | Shadow deployment & A/B test - Deploy the fix to 1% of real traffic (canary). Compare precision, latency, error rate vs. baseline (unchanged 99%) over a 2?hour window. Precision gap < 1 point vs. baseline, latency p99 < 50?ms, error rate < 0.1%. | MLOps / Infra Engineer | Day 2, 4 hours | Precision gap < 1 point vs. baseline. |
|---|---|------------------------|----------------|---------------------------------------|

| | | | | |
|---|---|------------------------|----------------|---|
| 5 | Full rollout - Canary passes -> ramp to 25%, then 100% over 2 hours, monitoring each ramp step. | MLOps / Infra Engineer | Day 2, 3 hours | Precision stabilizes at 0.78-0.79. No alert triggers. |
|---|---|------------------------|----------------|---|

| | | | | |
|---|--|-----------------------|------------------|---|
| 6 | Post?rollout monitoring - Watch precision, recall, and feature distribution dashboards for 48 hours. No sustained deviation > 1 point from baseline. | ML Engineer / On?call | Day 2-4, ongoing | No sustained deviation > 1 point from baseline. |
|---|--|-----------------------|------------------|---|

? Rollback Plan (If Fix Fails)

Triggers (any of the following during steps 4-6):

- Precision drops further (> 1 point below the current 0.71)
- p99 latency exceeds 50?ms for > 5 minutes
- Error rate > 1% (model inference failures, not just user errors)
- Silent prediction corruption detected (e.g., all predictions become identical)

Rollback Procedure

| Step | Action | Owner | Timeline | Details |
|------|--------|-------|----------|---------|
|------|--------|-------|----------|---------|

| Step | Action | Owner | Timeline | Details |
|------|--------|-------|----------|---------|
|------|--------|-------|----------|---------|

| | | | | |
|---|---|-----------------|------------------|--|
| 1 | Immediate revert - Set the inference?api config back to hash encoding via the last known good deployment (or toggle a feature flag if available). `kubectl rollout undo deployment/inference-api` or flip flag to `encoding=hash` | MLOps / On?call | Within 5 minutes | |
|---|---|-----------------|------------------|--|

| | | | | |
|---|--|---------|--|--|
| 2 | Alert and notify - Update the incident channel and the team Slack. | On?call | | |
|---|--|---------|--|--|

Engineering counterpart

7:44 AM - Responded in 0s

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

Agent response

7:44 AM - Responded in 31s

The encoding consistency check (Step 1) - specifically the column ordering and missing-value handling.

Here's why I'd want a second pair of eyes there:

The Hidden Trap

One-hot encoding has a subtle failure mode: what happens when a new category appears in production that wasn't in training?

- Training one-hot: `plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise` (3 columns)
- In production, a new plan is introduced: `plan\_tier\_gold`

Scenario A (silent corruption): The serving code only sees 3 known categories, so it maps `gold` -> all zeros `[0,0,0]`. The model treats this as the baseline/omitted category - which might have been `free` during training. Now `gold` users are misclassified as `free`.

Scenario B (crash): The code attempts to create 4 columns `[free, pro, enterprise, gold]`, but the model expects exactly 3. Inference fails.

Scenario C (hash leftover): The serving code partially applies both encodings (hash for unknown categories, one-hot for known). Feature vector dimension mismatches and predictions become garbage.

What the second pair should verify

Ask them to:

1. Check the `handle\_unknown` parameter - Is it `error` (crash on unknown - safe but fragile) or `ignore` (all zeros - silent corruption risk)? The latter is more dangerous.
2. Audit production plan tier values - Run a quick query: `SELECT DISTINCT plan\_tier FROM inference\_logs WHERE timestamp > '2026-07-01'` and compare to the training set's expected values. Any mismatch?
3. Reproduce the transformation on a borderline case - Feed a request with `plan\_tier: "gold"` through the inference pipeline and log the exact feature vector. Confirm it's either:
 - Explicitly rejected (if `handle\_unknown='error'`)
 - Mapped to a known safe default (e.g., `free` = `[1,0,0]`) - but this is a band-aid, not a fix

If the team already has a skew detection dashboard that tracks the distribution of `plan\_tier` values in real time, this is the moment to look at it. The distribution shift (free ?, enterprise ?) might already have been flagged there - if not, it's worth adding.

Bottom line: The one-hot rollback should fix the encoding mismatch, but the unknown category handling is the single point where a seemingly trivial detail can silently undo the fix and create a worse problem (corrupted predictions instead of merely degraded ones).

Engineering counterpart

7:44 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.