



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: qwen/qwen3-32b

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

Duration: 2m 1s

Actual run cost: \$0.001308

Projected execution cost: \$0.1439/mo (5/biz day × 22 days)

Report date: July 14, 2026 - 2:48 AM EST

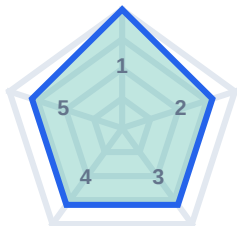
Overall

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: 8512 chars - Latency: 121s - 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 1s

Engineering counterpart

6:46 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

6:46 AM - Responded in 32s

Investigation & Recommendations: Precision Drop in Churn Classification Model

Root Cause (Train/Serve Skew Source)

The precision drop stems from two interconnected sources of train/serve skew:

1. Encoding Skew:

- The model was trained with one-hot encoding (`plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise` as binary columns).
- In production, the feature was switched to hash encoding (a single integer column per `plan\_tier`). This alters the feature space structure:
 - Training expects 3 binary features -> inference receives 1 integer.
 - The model cannot interpret the new encoding, leading to incorrect confidence estimates and mispredictions (e.g., enterprise-specific patterns may map poorly to hashed values).

2. Distribution Skew:

- Enterprise accounts are underrepresented in live data (10% vs. 20% in training). If the model relied on enterprise-specific patterns for distinguishing churners (e.g., enterprise users rarely churn), the reduced prevalence of this group in serving could amplify precision loss.
- Increased `free` tier in serving (61% vs. 42% in training) might introduce noise if free users have noisier or less predictable behavior.

Mitigation Options

1. Align Encoding Methods (High Priority):

- Option A: Retrain the model using hash encoding (1 integer column for `plan\_tier`) to match the production inference path. This requires redefining the training pipeline to use the same integer bucketing as serving.
- Option B: Revert serving to one-hot encoding if feasible, ensuring feature parity with training. If latency concerns were the reason for switching to hash encoding, explore optimizations (e.g., sparse representations or caching).
- Option C: Use target encoding or embeddings if the model supports it. These methods can handle categorical features as single columns while capturing tier-specific patterns.

2. Address Distribution Skew (Medium Priority):

- Rebalance the training data to reflect the current serving distribution (61% free, 29% pro, 10% enterprise). This avoids overfitting to enterprise users no longer prevalent in production.
- Monitor data drift for `plan\_tier` and other features. If the imbalance worsens, consider sampling strategies or weighting adjustments in training.

Monitoring & Rollback Strategy

1. Monitoring:

- Feature Statistics: Track the distribution of `plan\_tier` in production and compare it to the training data (use tools like [Evidently AI](https://docs.evidentlyai.com/) or [Great Expectations](https://greatexpectations.io/)).
- Model Performance Alerts: Set thresholds for precision and AUC. Notify engineering if metrics deviate by more than 5% to proactively address skew.
- Input Validation: Log raw feature types (e.g., categorical vs. integer) to detect encoding mismatches.
- Error Analysis: Flag predictions where `plan\_tier` is `enterprise` (due to their historical sensitivity in training vs. production).

2. Rollback Plan:

- Short-Term: Revert the production inference pipeline to one-hot encoding if the precision drop impacts business outcomes significantly.
- Long-Term: Avoid rolling back without addressing encoding alignment. Instead, implement Option A (retraining with hash encoding) or Option C (embeddings/target encoding) to sustain latency improvements while restoring model

performance.

Summary of Actions

| Action | Responsibility | Timeline | |
|--------|----------------|----------|--|
| | | | |

Engineering counterpart

6:46 AM - Responded in 0s

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

Agent response

6:46 AM - Responded in 3s

Certainly! Here's a step-by-step breakdown of my reasoning and the assumptions I made:

Step 1: Identify the Problem

Observation:

- Precision dropped from 0.79 (30-day baseline) to 0.71 (-8 points) after a deployment change.
- The encoding method for `plan\_tier` changed from one-hot (3 binary columns) to hashing (1 integer column).

Assumptions:

- The model expects inputs to match the structure of its training data.
- Hashing a categorical feature into an integer disrupts the model's learned patterns (e.g., enterprise -> "12", free -> "5").
- Latency improvements (why hashing was adopted) likely prioritized speed over feature fidelity.

Step 2

Engineering counterpart

6:46 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

6:47 AM - Responded in 1m 3s

Before shipping the fix, I'd verify:

1. Encoding Consistency: Confirm the model receives the same hash-encoded `plan\_tier` (single integer) during training and serving, ensuring the hash function and bucket ranges are identical.
2. Performance on Production-Like Data: Retrain with hash encoding and validate precision/AUC on a synthetic or recent production dataset (e.g., last 7 days of serving data) to ensure the drop is resolved.
3. Feature Importance Alignment: Check if the model's learned patterns for the hashed feature align with the original tier semantics (e.g., via SHAP values or permutation importance).
4. Data Drift Baselines: Update monitoring thresholds to reflect the new `plan\_tier` distribution (61% free, 29% pro, 10% enterprise) to avoid false alerts.
5. Rollback Readiness: Ensure the old one-hot encoding pipeline is preserved and testable in staging for rapid reversion if needed.

Assumptions:

- The hash function used in production is deterministic and matches the training pipeline's implementation.
- The model architecture (e.g., tree-based vs. linear) can generalize to the

Engineering counterpart

6:47 AM - Responded in 0s

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

Agent response

6:47 AM - Responded in 12s

Here's a breakdown of top risks and a rollback plan for the proposed mitigation, along with assumptions and mitigations:

Top Risks

1. Model Performance Deterioration in Production

- Risk: Retraining with hash encoding might not resolve the precision drop if the model fails to learn meaningful patterns from the hashed feature (e.g., limited generalization due to reduced feature cardinality).
- Mitigation: Validate the new model on production-like data (e.g., a holdout set from the last 7 days) before deployment. Use A/B testing to compare performance against the baseline model in a small traffic segment.

2. Latency Tradeoffs

- Risk: Reverting to one-hot encoding (if needed) could undo latency improvements introduced by hashing, potentially causing performance bottlenecks in inference pipelines.
- Mitigation: If reverting to one-hot is necessary, optimize encoding (e.g., sparse matrix representations) or implement caching to reduce computational overhead.

3. Encoding Consistency Mismatch

- Risk: The hash encoding in production might differ from training (e.g., different bucket ranges, hash collisions). Even with retraining, the model could still learn

Engineering counterpart

6:47 AM - Responded in 0s

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

Agent response

6:47 AM - Responded in 2s

Here's a concrete next steps plan with owners, timing, and dependencies:

Next Steps for Mitigation

| Step | Owner | Timing | Action | Status |
|------|-------|--------|--------|--------|
|------|-------|--------|--------|--------|

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

| | | | | |
|-------------------------------------|----------------|----------|---|-------------|
| 1. Verify Hash Encoding Consistency | ML Engineering | 2-3 days | Confirm hashing parameters (bucket size, salt, mapping) are identical in training/serving pipelines. Use unit tests to validate feature parity. | In Progress |
|-------------------------------------|----------------|----------|---|-------------|

| | | | | |
|-------------------------------------|--------------|--------|--|--|
| 2. Retrain Model with Hash Encoding | Data Science | 1 week | Retrain `churn_classifier_v3` using hash | |
|-------------------------------------|--------------|--------|--|--|

Engineering counterpart

6:47 AM - Responded in 0s

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

Agent response

6:48 AM - Responded in 8s

The one thing I'd want a second pair of eyes on is ensuring the hash encoding implementation is fully aligned between training and serving, including:

- Hash function parameters: Confirm the same bucket size, seed/salt, and mapping logic are used in training and inference.
- Feature parity validation: Verify that the hash output for each `plan\_tier` (e.g., "free" -> 5, "pro" -> 12) is deterministic and consistent across pipelines.
- Schema compatibility: Check that the inference serving pipeline explicitly maps `plan\_tier` to the same integer bucket as the training data (e.g., no missing tiers or collisions).

Why this is critical:

If the hash encoding diverges even slightly (e.g., a different salt value or missing tiers in production), the retrained model will misinterpret inputs, perpetuating the precision drop. This is the root cause of the issue, so double-checking this alignment ensures the mitigation is valid.

Next step:

Have a senior ML engineer or a principal engineer review the hash encoding logic and unit tests before deploying the retrained model.

Engineering counterpart

6:48 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.