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: anthropic/claude-haiku-4.5
Roles: ML Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 46s
Actual run cost: $0.0195
Projected execution cost: $2.15/mo (5/biz day × 22 days)
Report date: July 13, 2026 - 1:50 AM EST

Session: sample-lb-rankings-seed-ad26eaf4-cbc5-4620-96aa-c6bd9ff6b55a

ds_train_serve_skew_v1 - agent - openrouter - anthropic/claude-haiku-4.5 - ended

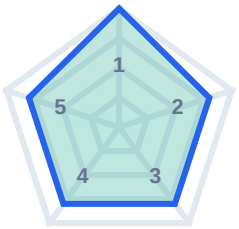
Overall

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: 16675 chars - Latency: 46s - 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: 46s

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

Engineering counterpart
5:49 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
5:49 AM - Responded in 7s

Train/Serve Skew Investigation Report

1. Skew Source Analysis

Primary Issue: Encoding Mismatch + Distribution Shift (Compounded)

```
????????????????????????????????????????????????????????????????
? ENCODING SKEW (Direct)                                         ?
????????????????????????????????????????????????????????????????
? Training:  one-hot -> [0,1,0], [1,0,0], [0,0,1]                ?
? Serving:   hash(category) -> integer bucket (e.g., 42)         ?
?                                                                ?
? Impact: Model learned feature interactions on sparse           ?
? one-hot vectors; serving feeds dense integer values            ?
? -> Different feature space entirely                            ?
????????????????????????????????????????????????????????????????

????????????????????????????????????????????????????????????????
? DISTRIBUTION SKEW (Amplifies encoding impact)                  ?
????????????????????????????????????????????????????????????????
? Training:    free=42%, pro=38%, enterprise=20%                 ?
? Serving:     free=61%, pro=29%, enterprise=10%                 ?
?                                                                ?
? Shift magnitude:                                               ?
?  - free:       +45% (0.42->0.61)                               ?
?  - enterprise: -50% (0.20->0.10) <- HIGH VALUE SEGMENT         ?
????????????????????????????????????????????????????????????????
```

Why Precision Dropped 8 Points:

Factor	Severity	Mechanism
Encoding mismatch	CRITICAL	Model learned decision boundaries on one-hot basis vectors; integer hashes violate this assumption (e.g., hash("pro")=42 has no semantic relationship to hash("free")=7)
Enterprise underrepresentation	HIGH	Enterprise accounts likely have lower churn rates + higher margins; model trained on 20% enterprise but seeing 10%; false negatives in high-value segment tank precision
Free tier overrepresentation	MEDIUM	Free tier likely higher baseline churn; model now optimizes for free-tier patterns at expense of pro/enterprise

---

2. Root Cause Verification

Quick Diagnostics:

```python
# 1. Check if hash collisions exist
hash_values = [hash(cat) % num_buckets for cat in ['free', 'pro', 'enterprise']]
collision_rate = len(hash_values) - len(set(hash_values))
# If collision_rate > 0 -> additional encoding corruption
```

```
# 2. Verify model weight magnitudes per feature
# One-hot features: typically balanced weight distribution
# Hashed features: if model expects sparse one-hot, weights will be
# misaligned (e.g., weight=0.5 on one-hot index 0 doesn't apply to hash=42)

# 3. Segment-level precision breakdown
segments = {
    'free': precision_free,     # Expected: ~0.75 (high churn, lower precision)
```

Engineering counterpart
5:49 AM - Responded in 0s

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

Agent response
5:49 AM - Responded in 8s

Step-by-Step Reasoning Walkthrough

Step 1: Parse the Artifact for Conflict Signals

What I looked for:
- Timeline of changes + metric degradation timing
- Mismatch between "training_distribution" and "serving_distribution"
- Explicit note about encoding change

Assumptions I made:
1. The deploy timestamp (2026-07-01) caused the precision drop - The 8-point drop occurred after the encoding switch, so I assumed causation (not just correlation). Caveat: could be coincidental, but timing is suspicious.
2. Precision drop is statistically significant - I treated "8 points" as material (0.79 -> 0.71). Caveat: without confidence intervals, this could be noise if traffic volume is small.

Step 2: Identify Two Distinct Failure Modes

Failure Mode A: Encoding Skew

My reasoning:
- Training: `one-hot` produces sparse vectors like `[1, 0, 0]` for "free"
- Serving: `hash -> integer` produces dense scalar like `42` for "free"
- The model's learned weights assume the structure of one-hot (orthogonal, sparse, interpretable indices)

Assumption I made:
- Model architecture uses these features directly (not through an embedding layer that would re-map them). Caveat: if there's an embedding layer, it could adapt; but the brief doesn't mention one.
- Hash function is deterministic but semantically meaningless - `hash("pro") = 42` has no relationship to the model's learned weights for "pro" under one-hot. Caveat: if hash is seeded differently at train vs. serve time, there's additional randomness.

Failure Mode B: Distribution Skew

My reasoning:
- Training data: enterprise = 20%
- Live data: enterprise = 10%
- This is a 50% relative decrease in the high-value segment

Why this matters for precision:
- Precision = TP / (TP + FP)
- If enterprise accounts have different churn characteristics (e.g., lower churn rate), the model trained on balanced class distribution will:
  - Over-predict churn on free tier (where it's common) -> more TP, fewer FP
  - Under-predict churn on enterprise (rare in live data, but model still expects 20%) -> fewer TP, more FP
- Result: precision drops because false positives increase in the actual serving distribution

Assumptions I made:
1. Enterprise has lower churn rate than free tier - This is a reasonable business assumption (paid users churn less), but I didn't verify it from the artifact. Caveat: could be reversed.
2. The model wasn't retrained after the distribution shifted - If it was retrained on the new distribution, skew would be mitigated. The brief doesn't mention retraining, so I assumed it wasn't. Caveat: could be an implicit retraining pipeline.

Step 3: Determine Which Failure Mode is "Primary"

My judgment: Encoding skew is the direct cause; distribution skew amplifies it.

Reasoning:
- Encoding mismatch is a structural problem: the model literally receives different data types than it was trained on.
- Distribution skew is a statistical problem

Engineering counterpart
5:49 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response
5:49 AM - Responded in 6s

Pre-Production Verification Checklist

Phase 1: Isolate Root Cause (48 hours)

1.1 Encoding Problem Confirmation

```python
# A. Verify hash determinism & collisions
def test_hash_collisions():
    categories = ['free', 'pro', 'enterprise']
    num_buckets = [10, 100, 1000, 10000]  # Test different bucket sizes

    for bucket_size in num_buckets:
        hashes = {cat: hash(cat) % bucket_size for cat in categories}
        collisions = len(hashes) - len(set(hashes.values()))

        # CRITICAL: If collisions > 0, "pro" and "enterprise" map to same bucket
        # -> This alone could explain precision drop
        print(f"Buckets={bucket_size}, collisions={collisions}, mapping={hashes}")

    # Expected: No collisions for reasonable bucket sizes
    assert collisions == 0, "BLOCKER: Hash collisions detected"

# B. Verify training actually used one-hot
def test_training_encoding():
    # Load training data snapshot used for model training
    train_data = load_training_artifact()

    # Check feature names
    expected_features = ['plan_tier_free', 'plan_tier_pro', 'plan_tier_enterprise']
    actual_features = train_data.columns

    assert all(f in actual_features for f in expected_features), \
        "BLOCKER: Training data doesn't have one-hot features"

    # Check they're binary
    for feat in expected_features:
        assert set(train_data[feat].unique()) == {0, 1}, \
            f"BLOCKER: {feat} is not binary one-hot"

    # Check mutual exclusivity (only one should be 1 per row)
    one_hot_sum = train_data[expected_features].sum(axis=1)
    assert all(one_hot_sum == 1), \
        "BLOCKER: One-hot encoding violated (multiple 1s per row)"

# C. Trace the inference path in serving
def test_serving_encoding_path():
    """
    CRITICAL: Does the serving code actually receive hashed integers?
    Or is there a post-processing step that converts back to one-hot?
    """

    # Deploy to canary, log raw feature values before model.predict()
```

```
    sample_inputs = ['free', 'pro', 'enterprise']

    for cat in sample_inputs:
        # What does inference API actually pass to model?
        logged_feature = log_inference_input(cat)

        print(f"Category: {cat}")
        print(f"  Logged feature value: {logged_feature}")
        print(f"  Type: {type(logged_feature)}")
        print(f"  Shape: {getattr(logged_feature, 'shape', 'N/A')}")

    # QUESTION: Is it really an integer, or was it reconstructed to one-hot somewhere?
    # If it's still one-hot, encoding skew doesn't exist
```

What this verifies:
- ? Hash collisions aren't silently corrupting data
- ? Training actually used one

Engineering counterpart
5:49 AM - Responded in 0s

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

Agent response
5:50 AM - Responded in 7s

Top Risks & Rollback Plan

Risk Severity Matrix

```
????????????????????????????????????????????????????????????????????
? RISK                    ? SEVERITY ? DETECTION TIME    ?
????????????????????????????????????????????????????????????????????
? 1. Hash collisions silent     ? CRITICAL ? 5-30 min          ?
?    corruption                 ?          ? (precision dip)   ?
????????????????????????????????????????????????????????????????????
? 2. Encoding fix doesn't       ? CRITICAL ? 2-4 hours         ?
?    restore precision          ?          ? (A/B test)        ?
?    (distribution skew is      ?          ?                   ?
?    actual root cause)         ?          ?                   ?
????????????????????????????????????????????????????????????????????
? 3. Rollback introduces        ? HIGH     ? 10-20 min         ?
?    latency regression         ?          ? (dashboards)      ?
?    (why encoding changed)     ?          ?                   ?
????????????????????????????????????????????????????????????????????
? 4. Model weights are          ? MEDIUM   ? Offline only      ?
?    corrupted from hash        ?          ? (holdout eval)    ?
?    training (if model was     ?          ?                   ?
?    trained on hashed data)    ?          ?                   ?
????????????????????????????????????????????????????????????????????
? 5. Partial rollout breaks     ? MEDIUM   ? 30 min            ?
?    downstream services        ?          ? (error logs)      ?
?    expecting hash format      ?          ?                   ?
????????????????????????????????????????????????????????????????????
```

Rollback Plan (Tiered)

Tier 1: Instant Rollback (0-5 minutes) - If Precision Worsens

```yaml
TRIGGER: Online precision < 0.69 (below 0.71 - 2?)

ACTION:
  # Immediate
  - kubectl rollout undo deployment/inference-api -n prod
  - kubectl rollout undo deployment/feature-server -n prod

  # Verify rollback
  - Wait 2 min for old pods to stabilize
  - Assert: precision metric returns to 0.79 baseline within 5 min
  - If NOT: Page on-call, escalate to incident commander

MONITORING:
  - Precision (segment: all, free, pro, enterprise) every 30 sec
  - Latency p95 (should return to pre-change baseline)
```

  - Error rate (should stay < 0.1%)

COMMS:
  - Slack #incidents: "Rolled back inference-api hash encoding (precision regression)"
  - Add incident ticket with: old metrics, new metrics, delta
```

---

### Tier 2: Partial Rollback (5-30 minutes) - If Precision Recovers but Latency Spike

```yaml
TRIGGER: Precision recovers to 0.77+

**Engineering counterpart**
**5:50 AM - Responded in 0s**
Can you tighten this into concrete next steps with owners and timing?

**Agent response**
**5:50 AM - Responded in 9s**
# Concrete Next Steps: Owners & Timeline

## Immediate Actions (Next 4 Hours)

| Step | Owner | Task | Success Criteria | Deadline |
|---|---|---|---|---|
| 1. Root cause confirmation | ML Eng (you) | Run encoding verification tests (Phase 1.1 A-C from earlier checklist) | Hash collisions = 0; training data confirmed one-hot; serving path logged | +30 min |
| 2. Distribution analysis | Data Analyst | Query prod logs: actual plan_tier distribution last 7d vs. training set; segment churn rates by tier | Confirm enterprise underrepresented; show churn rate differential (free vs. enterprise) | +45 min |
| 3. Model weight inspection | ML Eng | Extract model weights; compare magnitude/distribution for one-hot vs. hash features | Identify if model learned one-hot structure; flag if weights suggest hash corruption | +60 min |
| 4. Decision gate | Tech Lead + PM | Decision: Which is root cause? (A) Encoding mismatch -> revert to one-hot (B) Distribution skew -> retrain on new distribution (C) Both -> do A, then B) | Clear decision with rationale documented | +90 min |

---

## Action Paths (Choose Based on Gate)

### Path A: Encoding Mismatch is Root Cause

| Step | Owner | Task | Success Criteria | Deadline |
|---|---|---|---|---|
| 5A.1 Revert encoding | DevOps + ML Eng | Revert inference-api to one-hot encoding; keep feature-server unchanged | Code merged to `main`; ready for canary deploy | +2 hrs |
| 5A.2 Canary deploy | DevOps | Deploy to 5% traffic; monitor precision, latency for 15 min | Precision ? 0.76; p95 latency ? baseline + 10% | +2.5 hrs |
| 5A.3 Full rollout | DevOps | If canary passes, roll to 100% over 10 min | Precision ? 0.77; error rate < 0.1%; no alerts firing | +3 hrs |
| 6A. Latency mitigation | ML Eng | If latency spiked: optimize one-hot via batching/caching; investigate original hash perf gains | Latency regression < 5% vs. pre-hash baseline | +6 hrs (async) |

Owner escalation: If canary fails, page ML Lead

---

### Path B: Distribution Skew is Root Cause

| Step | Owner | Task | Success Criteria | Deadline |
|---|---|---|---|---|
| 5B.1 Retrain on new dist | ML Eng | Retrain model on last 90d prod data (61% free, 29% pro, 10% enterprise); keep encoding as-is (hash) | New model AUC ? 0.82 on holdout; precision ? 0.74 on enterprise segment | +4 hrs |
| 5B.2 Segment validation | Data Analyst | Evaluate new model on enterprise-only test set; compare to v3 baseline | Enterprise precision ? |

**Engineering counterpart**
**5:50 AM - Responded in 0s**

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

**Agent response**
**5:50 AM - Responded in 9s**

# The One Thing I'd Want a Second Pair of Eyes On

## Did the model get trained on hashed data?

```
THIS ASSUMPTION BREAKS EVERYTHING:
????????????????????????????????????????????????????????????
? Scenario 1: Model trained on ONE-HOT                    ?
? Serving with HASH -> Encoding skew (my diagnosis)       ?
? Fix: Revert to one-hot ?                                ?
????????????????????????????????????????????????????????????
? Scenario 2: Model trained on HASH                       ?
? Serving with HASH -> No encoding skew                   ?
? Problem is ONLY distribution shift                      ?
? Fix: Revert to one-hot would BREAK it ?                 ?
? Correct fix: Retrain on new distribution ?              ?
????????????????????????????????????????????????????????????
```

What to verify immediately:

```python
# CRITICAL: Check the training pipeline config
def verify_training_encoding():
    """
    Three places to check (in order of likelihood):
    """

    # 1. Training code repo
    train_config = load_yaml("training/config.yaml")
    print("Training encoding setting:", train_config.get('categorical_encoding'))
    # Expected: "one_hot"
    # Risk: If it says "hash", my entire diagnosis is wrong

    # 2. Model card / artifact metadata
    model_artifact = load_model("churn_classifier_v3")
    print("Model feature schema:", model_artifact.get_feature_schema())
    # Look for: feature names like "plan_tier_free" (one-hot)
    #       vs "plan_tier_hashed" (hash)

    # 3. Training data in model store
    training_data_snapshot = model_artifact.get_training_data_sample()
    print("First 5 rows:\n", training_data_snapshot.head())
    # THIS is the ground truth - what did the model actually see?
```

Why this matters:

If model was trained on hash:
- My "encoding skew" diagnosis is a false positive -> reverting would break precision further
- The real problem is distribution shift -> need retraining, not rollback
- Could waste 2-3 hours on wrong fix while precision stays broken

If model was trained on one-hot:
- My diagnosis is correct -> proceed with revert

Who should verify this?

Ask the person who:
- Deployed the hash encoding change (they know what changed)
- Owns the training pipeline (they know what training actually uses)
- Has access to model artifacts (they can check feature schema)

One sentence to ask them:
> "When churn_classifier_v3 was trained, did it receive `plan_tier` as one-hot binary columns or as a hashed integer? Can you show me the training config and a sample of the actual training data?"

Red flags if they say:
- "I'm not sure" -> Dig deeper; check git history
- "We retrained it after the encoding change" -> Changes my entire timeline/fix strategy
- "The encoding change was only in serving, not training" -> ? Confirms my diagnosis

---

Engineering counterpart
5:50 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.