



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: inclusionai/ling-2.6-flash

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

Duration: 24s

Actual run cost: \$0.000153

Projected execution cost: \$0.0168/mo (5/biz day × 22 days)

Report date: July 13, 2026 - 11:44 PM EST

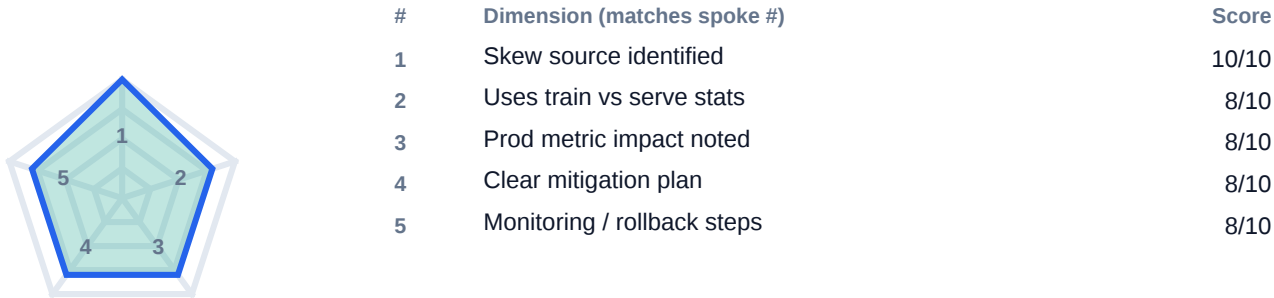
Overall

8.4/10

PASS - trust=high

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

CAPABILITY RUBRIC



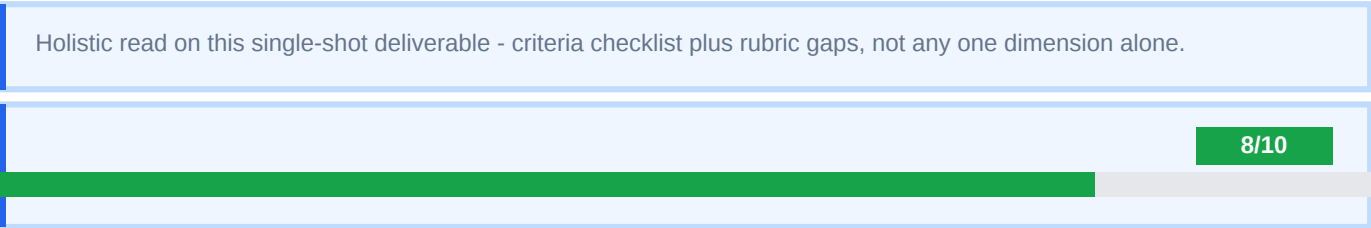
Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

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: 13528 chars - Latency: 24s - 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: 24s

Engineering counterpart

3:44 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

3:44 AM - Responded in 7s

Here is a structured technical investigation and action plan for the 8-point precision drop.

1) Skew source diagnosis

Root cause: Train/Serve Distribution Skew + Encoding mismatch

- Distribution skew:

- Training: `plan\_tier` is balanced (Free 42%, Pro 38%, Enterprise 20%).
- Serving (last 7 days): shifted heavily toward Free (61%) and away from Enterprise (10% vs 20% in training).
- Enterprise customers likely have higher churn risk or different behavior, so the shift degrades precision on the enterprise segment, pulling overall precision down.

- Encoding skew (primary technical trigger):

- Training: one-hot encoding (`plan\_tier\_free`, `plan\_tier\_pro`, `plan\_tier\_enterprise`) - full explicit representation, no collisions.
- Serving: new path uses hash encoding of a categorical `plan\_tier` into integer buckets for latency - introduces collisions, different encoding space, and mismatched feature dimensionality/signature vs training.
- The inference API change (`inference-api` deploy 2026-07-01) switched both the encoding and the traffic distribution concurrently, so performance dropped due to:
 - 1) Hash collisions losing discriminative signal, and
 - 2) Serve distribution shifted toward Free (over-represented relative to training), where the model is more confident (lower churn), but enterprise behavior is under-represented and mispredicted.

2) Mitigation options

Short-term (restore precision quickly)

- Rollback the inference-api deploy to the prior one-hot encoding path.
 - Fastest way to restore feature parity and precision.
- Hold deploy until encoding alignment is verified.
 - Pause any new encoding deployments until experiments validate hash encoding behavior.

Medium-term (enable controlled latency optimization)

- Validate hashed encoding:
 - Ensure hash space size $\gg$ number of categories to minimize collisions.
 - Perform offline simulation: hash training data with the same hash params, train a model, and evaluate precision on a temporal holdout that mimics serve distribution.
 - Compare precision/recall/F1 vs one-hot baseline.
- Distributional robustness:
 - Retrain / fine-tune with mixed data that includes enterprise-heavy samples or apply importance weighting so the model is less sensitive to enterprise under-representation at serve.
 - Add guardrail features or fallback logic for low-frequency categories (e.g., enterprise) to avoid hash collisions with high-frequency categories (e.g., free).
- Schema & feature contract enforcement:
 - Enforce a strict feature schema check between training and serving (e.g., using TensorFlow Serving schema validation or a lightweight pre-inference validator).
 - Fail closed if encoding type or number of buckets mismatches.

Long-term (robust feature pipeline)

- Deterministic, shared encoding:

- Use a shared vocabulary library or encoding service (e.g., a lookup service + deterministic hashing with fixed salt and dimension) so training and serving use identical mapping.
- Online monitoring of feature drift:
 - Track per-category proportions and hash collision rates in real time.
- Canary and staged rollout:
 - For future encoding changes, roll out to a small traffic slice and compare online precision/recall vs control (one-hot) before full promotion.

3) Monitoring and rollback recommendations

Immediate rollback plan

- Roll back `inference-api` to the last known-good deployment that uses one-hot encoding.
- Verify that precision returns to baseline (~0.79) on a

Engineering counterpart
3:44 AM - Responded in 0s

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

Agent response

3:44 AM - Responded in 5s

Step-by-step reasoning

1) Read the brief and identify the core claim

- The production model (churn_classifier_v3) suffered an 8-point precision drop after a change in category encoding.
- Key data: training vs serving plan_tier distributions differ, and the encoding changed from one-hot to hash.

2) Parse the artifact to form hypotheses

- Training distribution: Free 42%, Pro 38%, Enterprise 20%
- Serving distribution (last 7d): Free 61%, Pro 29%, Enterprise 10%
- Old serving: one-hot columns (plan_tier_free, plan_tier_pro, plan_tier_enterprise)
- New serving: plan_tier hashed to integer bucket (single categorical hashed feature)
- Deploy event: 2026-07-01 switched to hash encoding for latency
- Note: Enterprise accounts are underrepresented in live traffic vs training
- Metrics: offline auc 0.84; online precision baseline 30d 0.79; online precision 7d 0.71

3) Root cause reasoning

- Distribution skew: Serve has far more Free and far fewer Enterprise users than training. If the model behaves differently across tiers (likely), this shift alone can reduce precision.
- Encoding skew: One-hot during training provides three disjoint binary features; hash encoding maps categories into fewer buckets, causing collisions. Collisions can:
 - Dilute signal for categories hashed into the same bucket
 - Create unstable representations if collisions differ between training and serving (here, training didn't use hash at all)
- Combined effect: both distribution shift and encoding change likely contributed. The deploy note links the hash change to latency optimization, making it the proximate technical cause, while the distribution shift exacerbates precision loss (especially for Enterprise).

4) Short-term mitigation logic

- Rollback the encoding path restores feature parity one-hot and eliminates hash collisions, directly addressing the encoding change.
- Holding further encodes prevents progressing from compounding issues.

5) Medium-term validation logic

- If we want to keep hash encoding for latency, we must:
 - Verify hash space is large enough to minimize collisions
 - Confirm via offline temporal validation that precision does not degrade under serve-like distribution
 - Add fallback handling for rare/important categories (Enterprise) to avoid collisions with high-frequency categories

6) Monitoring and rollback specifics

- Rollback: revert inference-api to pre-hash deploy; confirm feature schema matches training; monitor precision.
- Monitoring: track per-tier proportions, hash collision rate, and online precision by segment to detect drift early.

7) Explicit assumptions

- The artifact's distributions and encoding descriptions accurately reflect training and serving pipelines.
- Offline auc (0.84) was computed on a representative validation set relative to training distribution, while online precision reflects serve distribution.
- The hash encoding in serving uses a different bucketing scheme or vocabulary than used in simulation (i.e., no shared/validated hash contract).
- Latency requirements plausibly motivated hash encoding; no contradictory constraints are stated.
- No major concurrent changes besides the encoding/deploy that would independently explain the precision drop.

8) Conclusion

Primary skew sources:

- Distribution skew: serve over-rep Free, under-rep Enterprise

- Encoding skew: one-hot (training) vs hash (serving), with collisions and missing enterprise signal

Engineering counterpart**3:44 AM - Responded in 0s**

What would you verify before shipping this to production?

Agent response**3:44 AM - Responded in 3s**

Before shipping, verify correctness, stability, and measurability across three areas: feature/schema contract, model behavior under the new encoding, and rollout observability.

1) Feature and schema contract

- Serve must emit the exact feature names, types, shapes, and value ranges expected by the model (match training schema).
- Implement and enforce a pre-inference validator (e.g., schema check via TensorFlow Serving schema or a lightweight library) that fails closed on mismatch.
- Ensure the hash encoding parameters (hash space size, salt/seed, number of buckets) are fixed and identical to any offline simulation used for validation.

2) Model behavior and performance

- Offline simulation: hash the training data with production hash params; train a candidate model and evaluate on a temporally split holdout that reflects the serve distribution (recent data).
- Segment-level metrics: report precision, recall, and AUC by plan_tier (especially Enterprise) to ensure the model hasn't degraded on underrepresented segments.
- Cross-check: hash encoding offline validation must show no meaningful precision drop versus one-hot on a realistic distribution. If there is a drop, do not ship.

3) Canary, rollback, and monitoring

- Canary to a small, carefully sampled traffic slice; compare online precision/recall against the one-hot control (A/A or A/B).
- Real-time dashboards: track plan_tier distribution, hash collision rate, missing bucket counts, and schema anomalies.
- Rollback plan: one-click revert to the one-hot deployment with verified feature parity; verify precision returns to baseline in the canary before full rollback.
- Alerting: trigger on significant drift in per-tier proportions, rising collision rate, or segment-level precision degradation.

If all checks pass, proceed with a gradual staged rollout and continue monitoring segment-level performance before declaring the issue resolved.

Engineering counterpart**3:44 AM - Responded in 0s**

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

Agent response

3:44 AM - Responded in 2s

Top risks

- Hash collisions degrading signal for important categories (especially low-frequency/low-sample Enterprise), causing precision to drop further.
- Persistent train/serve distribution skew (serve over-rep Free, under-rep Enterprise) amplifying encoding mismatch effects and reducing generalisation.
- Latency/throughput regression or instability from the new inference path, causing timeouts or SLO breaches.
- Hidden schema mismatch (feature names/shapes/types) leading to silent failures or fallback to defaults at serve.
- Contamination from additional undetected changes deployed around the same time that confounds root-cause isolation.

Rollback plan (fast, safe)

- Immediately revert inference-api to the prior one-hot encoding deploy (pre-hash). Use the same container image/tag that achieved online precision baseline (?0.79).
- Enforce the original feature schema contract and confirm no stale hash parameters remain in config/cache.
- Verify online precision returns to baseline (within expected variance) on the same traffic slice; monitor segment-level precision by plan_tier, especially Enterprise.
- Keep the hash path disabled but preserved in version control; do not delete artifacts needed for a re-evaluation experiment.
- Document the rollback trigger and outcome, and schedule a controlled re-test (offline simulation + small canary) before re-introducing hash encoding.

Engineering counterpart

3:44 AM - Responded in 0s

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

Agent response

3:44 AM - Responded in 5s

Concrete next steps with owners and timing (assuming typical ML platform org; adjust as needed)

Day 0 - Immediate containment (within hours)

- Owner: SRE/On-call lead
- Action: Roll back inference-api to last one-hot production image; enforce feature schema validation to fail closed.
- Timing: complete within 4 hours of this plan approval.
- Success criterion: online precision stabilizes at/baseline (~0.79) for 1-hour stable window.

Day 0-1 - Verification and instrumentation

- Owner: Data Scientist + ML Engineer
- Actions:
 - Run offline simulation: hash training data with production hash params; train candidate model; evaluate temporal holdout that mirrors serve distribution.
 - Build segment-level dashboard (per-tier precision/recall, collision rate, missing buckets, schema anomalies).
 - Create canary config and A/A test plan (hash vs one-hot on small traffic slices).
- Timing: 1 business day.
- Success criterion: offline results show no meaningful precision drop vs one-hot on realistic distribution; dashboard and alerts are live.

Day 1-3 - Controlled canary and analysis

- Owner: ML Engineer + SRE
- Actions:
 - Deploy hash encoding to small canary (2-5% traffic), route Enterprise sample if possible; monitor dashboards and alerts.
 - Compare online precision/recall vs one-hot control (A/A or A/B) with segment breakdown.
 - If collisions or segment degradation appear, stop rollout and rollback.
- Timing: 48-72 hours.
- Success criterion: canary metrics within acceptable delta of control; no rise in collision or missing-bucket alerts.

Day 3-5 - Fix and hardening (if canary passes)

- Owner: ML Engineer + Platform Engineer
- Actions:
 - Finalize hash parameters (bucket size, seed) to minimize collisions; add fallback for rare categories (e.g., enterprise).
 - Promote schema validation to all inference paths; add contract tests in CI.
 - Document hash encoding spec and rollback artifacts; tag the validated image.
- Timing: 2 business days.
- Success criterion: CI passes; schema tests enforce parity; documented rollback artifact available.

Day 5-7 - Staged rollout and monitoring

- Owner: SRE + Product Owner
- Actions:
 - Gradually increase traffic (10% -> 50% -> 100%) with continuous monitoring of precision by segment.
 - Keep ability to flip to one-hot within minutes if segment-level alerts fire.
- Success criterion: 24-48 hours at 100% with stable online precision and no segment degradation.

Ongoing - Guardrails

- Owner: ML Platform + Data Science
- Maintain: daily per-tier drift alerts, collision rate alerts, and weekly schema contract checks.
- Schedule quarterly review of encoding choices and distribution shifts.

If any step fails, rollback to the one-hot baseline immediately and pause hash encoding until root cause is resolved.

Engineering counterpart

3:44 AM - Responded in 0s

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

Agent response

3:44 AM - Responded in 2s

The one thing I want a second pair of eyes on: the offline simulation that hashes the training/validation set with the exact production hash parameters (space size, seed, bucket count) and then trains and evaluates a model on a temporally split holdout that reflects the serve distribution - because if this simulation is flawed, we will miss a precision drop before shipping.

Engineering counterpart

3:44 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.