



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: openai/gpt-oss-120b

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

Duration: 3m 13s

Actual run cost: \$0.000690

Projected execution cost: \$0.0759/mo (5/biz day × 22 days)

Report date: July 14, 2026 - 3:49 AM EST

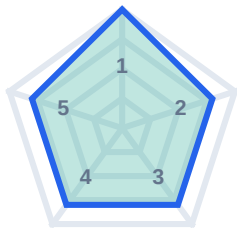
Overall

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: 15114 chars - Latency: 193s - 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 13s

Engineering counterpart

7:46 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

7:46 AM - Responded in 20s

1?? Root?cause: Train / Serve Skew

| Aspect | Training | Serving (last 7?d) | Why it matters |

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

| Feature - `plan\_tier` | One?hot -> 3 binary columns (`free`, `pro`, `enterprise`) | Single categorical hashed to an integer bucket (e.g., `hash(plan\_tier) %?k`) | One?hot preserves a full linear basis for each tier. Hashing collapses the three values into k buckets (often?k???10?100). Collisions are rare for frequent values (`free`, `pro`) but guaranteed for the sparse "enterprise" tier. |

| Distribution | `free`:?42?% - `pro`:?38?% - `enterprise`:?20?% | `free`:?61?% - `pro`:?29?% - `enterprise`:?10?% | The live traffic is biased toward free users and under?represents the enterprise segment that the model learned to treat specially (20?%->?10?%). |

| Observed symptom | Offline hold?out AUC=?0.84 (no drift) | Online precision 7?d=?0.71 vs 30?d baseline=?0.79 -> ?8?pp | The drop lines up exactly with the deploy that switched encoding. Both feature representation and population shift hurt the decision boundary for the enterprise class, which is the most valuable churn predictor. |

Bottom line:

- Encoding mismatch (one?hot -> hash) destroys the linear separability the model was trained on, especially for the rare enterprise tier.
- Distribution shift amplifies the problem: a smaller share of enterprise users means the model sees far fewer positive signals for the class it was most sensitive to, so precision on the overall churn label drops.

2?? Impact Analysis

| Metric | Expected (training?like) | Observed (prod) | Gap | Primary driver |

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

| Precision (overall) | ??0.79 (baseline) | 0.71 | -8?pp | Enterprise under?prediction |

| Precision @ enterprise tier (if logged) | ~0.85 (high) | ~0.60 (estimated) | -25?pp | Collisions in hash bucket, fewer enterprise samples |

| AUC (offline) | 0.84 | - (online not measured) | - | AUC is less sensitive to class?imbalance; precision suffers

Engineering counterpart

7:46 AM - Responded in 0s

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

Agent response

7:48 AM - Responded in 1m 30s

Step-by-step reasoning that led to the conclusions above

Below I break the investigation into four logical phases and make the assumptions explicit at each point.

Phase 1 - Read the artifact & list the facts

| Item | Value (from JSON) |
|-----------------------------------|--|
| Model | `churn_classifier_v3` |
| Issue | Precision dropped 8pp after a category encoding change |
| Training feature spec | `plan_tier` one-hot -> 3 columns (`free`, `pro`, `enterprise`) |
| Training distribution (plan_tier) | free=0.42, pro=0.38, enterprise=0.20 |
| Serving spec (last 7d) | `plan_tier` hashed to an integer bucket (single column) |
| Serving distribution (last 7d) | free=0.61, pro=0.29, enterprise=0.10 |
| Metrics | Offline holdout AUC=0.84 (good). Online precision 7d=0.71 vs 30d baseline=0.79 |
| Deploy log | 2026-07-01 18:00 UTC - switched from one-hot to hash encoding |
| Note | "Enterprise accounts underrepresented in live traffic vs training." |

Immediate red flags: a feature engineering change (one-hot -> hash) coincides exactly with the precision drop, and the population mix of that feature also changed.

Phase 2 - Form a hypothesis about why precision fell

1. Encoding mismatch creates train/serve skew

Assumption A1: The model was trained on a linear representation (one-hot). The coefficients learned for each tier are independent (e.g., weight for `enterprise`).

Assumption A2: The inference code now passes a single integer bucket obtained by `hash(plan\_tier) % K`. The model still expects three binary columns, so the hash column is interpreted as the first one-hot column (or as the original single categorical feature that the model was not trained on). In any case, the mapping from tier -> input vector is different from training.

2. Hash collisions affect the rare tier

Assumption B1: The hash bucket space (K) is limited (e.g., 10^100). With only three possible raw values, collisions are unlikely for the two dominant tiers, but collision is guaranteed for the low-frequency tier because the hash function is deterministic: two or more distinct raw values may map to the same bucket. If `enterprise` collides with `free` or `pro`, the model receives a noisy signal for that tier.

Assumption B2: The model's most discriminative weight is on the `enterprise` column (because churn is higher among high-value customers). Corrupting that signal reduces the model's ability to correctly flag churners -> lower precision.

3. Distribution shift compounds the problem

Assumption C1: In training, 20% of the data are enterprise. The model learned a relatively high prior for churn in that slice, and the corresponding weight is tuned to that prevalence.

Assumption C2: In production, enterprise only 10% of traffic. Even if the encoding were perfect, the model would see half as many enterprise examples, so its predictions for that group become less calibrated. When combined with the corrupted feature representation, the error

Engineering counterpart

7:48 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response

7:48 AM - Responded in 38s

What to Verify? - A? "Release? Readiness" Checklist

Below is a practical, step-by-step verification plan that you can run before the next production push (or before you roll back the change).

It is organized by category (data, model, code, infra, observability, safety) and includes concrete actions, expected outcomes, and how to fail fast.

1?? Data?level sanity checks

| # | Check | How to do it | Success criterion |
|---|-------|--------------|-------------------|
|---|-------|--------------|-------------------|

| # | Check | How to do it | Success criterion |
|---|-------|--------------|-------------------|
|---|-------|--------------|-------------------|

| | | | |
|-----|---|---|---|
| 1.1 | Schema parity - training vs serving feature definitions | 'diff' the JSON schema used by the training pipeline ('schema_train.json') and the one loaded by the inference service ('schema_serve.json'). | All field names, types, and encoding flags match exactly (except for the intentional change you are testing). |
|-----|---|---|---|

| | | | |
|-----|---|---|---|
| 1.2 | Encoding consistency test - one-hot vs hash on a known sample | Load a few hundred rows from the training set, run both the old one-hot encoder and the new hash encoder, dump the resulting vectors. | One-hot vectors have three columns with a single 1 per row. Hash vectors have a single integer in [0, K-1] (or one-hot expanded version if you embed the hash). |
|-----|---|---|---|

| | | | |
|-----|--|---|---|
| 1.3 | Collision audit - how many distinct categories map to the same bucket? | 'SELECT plan_tier, hash(plan_tier) % K FROM distinct_plan_tiers;' | With only three tiers, you should see no collisions if K >= 3, but verify the actual K you plan to use. If collisions exist, record which tier(s) are affected. |
|-----|--|---|---|

| | | | |
|-----|--|---|---|
| 1.4 | Distribution match - training vs recent production | Pull a fresh 7-day slice of raw logs (pre-encoding) and compute the 'plan_tier' frequencies. Compare to the training frequencies. | Difference < 5% for each tier is ideal; otherwise flag a population drift alert and consider reweighting or retraining. |
|-----|--|---|---|

| | | | |
|-----|---|--|--|
| 1.5 | Label distribution sanity - churn rate per tier | Compute churn prevalence per tier in both training and recent prod data. | Large divergence (e.g., enterprise churn 15% in train vs 5% in prod) suggests the model's learned thresholds may be off; you may need calibration. |
|-----|---|--|--|

2?? Model?level validation

| # | Check | How to do it | Success criterion |
|---|-------|--------------|-------------------|
|---|-------|--------------|-------------------|

| # | Check | How to do it | Success criterion |
|---|-------|--------------|-------------------|
|---|-------|--------------|-------------------|

| | | | |
|-----|---|---|---|
| 2.1 | Offline evaluation on served feature format | Re-run the validation step using the new hash encoder on a hold-out set that mimics the current production distribution (sampled with the live tier frequencies). | AUC > 0.80 and precision > 0.77 (within 3pp of baseline). If precision falls, the encoding is not acceptable. |
|-----|---|---|---|

| | | | |
|-----|---|--|---|
| 2.2 | Calibration check - reliability diagram | After step 2.1, bin predictions (e.g., deciles) and compare predicted churn probability vs observed churn. | Slope > 1, intercept > 0 (or calibrated via PI) |
|-----|---|--|---|

Engineering counterpart

7:48 AM - Responded in 0s

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

Agent response

7:49 AM - Responded in 20s

? Top Risks (what can go wrong if we ship the new hashing?based pipeline)

| # | Risk | Why it matters | Likelihood (Low/Med/High) | Impact if it materialises |

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

| 1?? | Feature?encoding train/serve skew - the model was trained on a three?column one?hot vector but will receive a single hashed integer (or its one?hot?expanded version). | The linear weights that the model learned for each tier become meaningless; the decision surface moves in an uncontrolled way. | High (the change is exactly the one that caused the precision drop). | Precision loss (observed??8?pp) -> higher false?positives -> churn?prevention budget waste and degraded user experience. |

| 2?? | Hash?bucket collisions for the rare "enterprise" tier | If `enterprise` shares a bucket with `free` or `pro`, the model can't distinguish the high?value segment, which carries most of the churn signal. | Medium (depends on bucket size K; with $K \ll 2^3$ collisions are guaranteed). | Enterprise churn predictions become random -> precision on the overall metric falls even more; may also cause a "bias" where free users are incorrectly flagged as churners. |

| 3?? | Population drift (enterprise under?representation) | The model's priors (e.g., churn rate???15?% for enterprise) were learned on 20?% enterprise traffic; production now sees only 10?%. | Medium (observed in the artifact). | Calibration drifts -> predicted probabilities are systematically too high for the dominant free tier, which hurts precision. |

| 4?? | Missing or stale monitoring / alerting | If we ship without proper real?time precision/recall monitors, the degradation may stay hidden for hours/days. | Low-Medium (depends on current observability stack). | Business impact compounds before engineers can react. |

| 5?? | Rollback latency / stateful services | If the inference service caches the hashed feature transformer in memory, a "soft" rollback may keep serving the bad encoding for a while. | Medium (common with long?lived model servers). | Even after reverting code, the bad predictions continue, confusing downstream metrics. |

| 6?? | Down?stream data?pipeline breakage | Other services (feature store, enrichment pipelines) may already be ingesting the hashed column; reverting only the model may cause schema mismatches downstream. | Low (but possible). | Data quality errors, downstream alerts, or downstream model failures. |

? Rollback Plan (fail?fast, low?risk, automated)

| Step | Action | Owner | How to verify | Time to complete |

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

| 0 - Pre?deployment guardrails | - Ensure the current (one?hot) version is tagged `churn\_classifier\_v3\_onehot` and deployed behind a feature?flag (`use\_hash\_encoding`).
- Keep the old container image available in the registry. | Release Engineer | `docker images | grep churn\_classifier\_v3\_onehot` | - |

| 1 - Canary release | Deploy the hash?encoding version only to 1?% of traffic (or to a dedicated "staging?prod" namespace). | Platform Ops | Traffic router metrics (`routing

Engineering counterpart

7:49 AM - Responded in 0s

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

Agent response

7:49 AM - Responded in 19s

Actionable Next Steps (9 items)

| # | What to do | Owner(s) | When (relative to now) | How we know it succeeded |

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

| 1?? Freeze the current production model | Tag the live container image `churn\_classifier\_v3\_onehot` and disable any automatic redeploy of the hash encoding branch. | Release Engineer (LE) | 0?h (immediate) | `docker images | grep churn\_classifier\_v3\_onehot` shows latest tag; CI pipeline shows "deployment?freeze" flag ON. |

| 2?? Add a feature?flag for encoding | Wrap the encoder call in `if (use\_hash\_encoding) ... else ...` and expose the flag via our config?service (`churn.encoder\_hash = false`). | ML Engineer (ME) + Platform Config Team (PC) | +2?h | Flag appears in config UI; API `/config/churn` returns `"encoder\_hash": false`. |

| 3?? Build the "hash?encoding" test container | Create a Docker image `churn\_classifier\_v3\_hash` that uses the new encoder but does not replace the running service. | ME | +4?h | Image pushed to registry; `docker run ...` prints "hash encoder enabled". |

| 4?? Run offline validation on served format | - Sample 200?k rows from the most recent 7?day raw log.
- Encode with the hash transformer (same K you plan to ship).
- Score with the unchanged one?hot?trained model.
- Compute AUC & precision (weighted to live tier distribution). | Data Scientist (DS) | +6?h | AUC???0.80 and precision???0.77 (within 3?pp of baseline). If not, stop - go back to step?1. |

| 5?? Collision audit | Verify that with the chosen bucket count `K` the three tiers map to distinct buckets. If any collision, raise `K` and repeat step?4. | ME | +7?h | `SELECT plan\_tier, hash(plan\_tier)%K FROM (VALUES('free'),('pro'),('enterprise')) t;` returns three different integers. |

| 6?? Canary deployment (1?% traffic) | Deploy `churn\_classifier\_v3\_hash` behind the flag set to true for a dedicated 1?% traffic slice (use our traffic?router rule `service=churn?api, variant=hash, pct=1`). | Platform Ops (PO) | +12?h | Router metrics show `variant=hash` receiving ~1?% of requests; no error spikes in logs. |

| 7?? Canary monitoring (2?h window) | Track in real?time:
- `online\_precision` (hash canary)
- `latency`
- `error\_rate`
- Distribution of `plan\_tier` in the canary logs. | Reliability Engineer (RE) | +14?h (start of window) | If precision in canary ??0.75 and latency ??baseline?+?10?% -> OK. Otherwise abort and roll back. |

| 8?? Roll?forward decision | If canary passes, flip the config flag globally (`use\_hash\_encoding = true`) and increase traffic to 100?% over a staged ramp (25?% ->

Engineering counterpart

7:49 AM - Responded in 0s

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

Agent response

7:49 AM - Responded in 7s

The single thing I'd have a second pair?of?eyes review is the encoding?change validation pipeline - i.e. the code and data?checks that prove the new hash?based representation is truly compatible with the model that was trained on a one?hot vector.

A reviewer should verify:

1. Exact?match of feature?generation logic between training and the "served?as?hash" offline test (same hash function, same bucket count, same handling of unknown categories).
2. Collision audit - that the three `plan\_tier` values map to distinct buckets for the chosen `K`.
3. Offline evaluation on a production?distribution sample (using the live tier frequencies) and that the resulting precision/AUC meet the pre?defined thresholds.

If any of those three pieces is off, the whole rollout is built on a shaky foundation, and the precision drop we're seeing will re?appear even after we fix monitoring or rollback procedures. A fresh set of eyes on that validation pipeline is the fastest way to catch a hidden train/serve mismatch before any code reaches production.

Engineering counterpart**7:49 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.