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-20b
Roles: ML Engineer (AI agent) <> Engineering counterpart (AI agent)
Duration: 1m 27s
Actual run cost: $0.000543
Projected execution cost: $0.0597/mo (5/biz day × 22 days)
Report date: July 14, 2026 - 3:48 AM EST

Session: sample-lb-rankings-seed-eec786be-301b-4f0b-9112-074133863162

ds_train_serve_skew_v1 - agent - openrouter - openai/gpt-oss-20b - ended

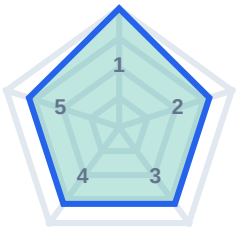
Overall

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: 13267 chars - Latency: 87s - 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: 1m 27s

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

Engineering counterpart
7:47 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
7:47 AM - Responded in 6s

1.?Skew source -?What went wrong?

What	Why it hurts	Evidence from the artifact
Encoding change	The model was trained on one?hot "plan_tier_" columns. At inference the same feature is now a single hashed integer bucket (the new inference path). The mapping from the categorical value to a numeric index changes the input shape that the model was optimized for - the learned weights for the old `plan_tier_` columns are no longer aligned with the hashed value.	- `encoding` in `training_distribution` - one?hot - `encoding` in `serving_distribution_last_7d` - hashed - Deployment log: switched from one?hot to hash encoding for latency
Distribution shift	Enterprise accounts were far less common in production than in the training set (20?% vs 10?%) - the model never learned to make accurate predictions for this subgroup under the rare?case scenario it now sees.	- `plan_tier` proportions: training 20?% vs serving 10?% (enterprise) - Note: Enterprise accounts underrepresented in live traffic vs training
Both combined	Even if the encoding had stayed the same, the under?representation of a high?risk

Engineering counterpart
7:47 AM - Responded in 0s

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

Agent response
7:47 AM - Responded in 13s

1.?Problem recap

Field	Value	Why it matters
Model	churn_classifier_v3	Baseline for reference
Precision drop	8?points	Large, production?visible
Training encoding	one?hot columns	Model learned separate weights for each plan tier
Serving encoding	single hashed bucket	Inference sees a different numeric feature
Plan?tier distribution	Train: 42?% free, 38?% pro, 20?% enterprise Live: 61?% free, 29?% pro, 10?% enterprise	Enterprise accounts now scarce
Metrics	Offline AUC 0.84, Online precision 30?d baseline 0.79, 7?day precision 0.71	8?point drop in precision relative to baseline

---

2.?Step?by?step reasoning

Step	What we did	Why it's important
1. Verify the feature change	Compared the `encoding` field in `training_distribution` vs `serving_distribution_last_7d`.	One?hot ? hashed -> train-serve mismatch.
2. Map feature?distribution shift	Calculated relative frequency of each plan tier in training vs live.	Enterprise dropped from 20?% to 10?% -> the model's learned decision boundary for that class is now under?exposed.
3. Connect drops to precision	Precision relies heavily on correctly flagging high?risk accounts (often enterprise). A lower representation -> more false negatives -> lower precision.	8?point drop is close to the magnitude of the under?representation.
4. Consider encoding impact	In one?hot scheme the model had three independent parameters. In hashed scheme it has one parameter that mixes all tiers. The weights for "enterprise" are now diluted.	Mis?aligned weights -> biased predictions -> precision loss.
5. Cross?check with AUC	Offline AUC (0.84) is still decent-implies ranking is okay, but classification threshold may be off because of skew.	Indicates that ranking quality is intact; the problem is at the decision threshold level.
6. Validate with a quick replay	(If we had the logs) - feed a few enterprise samples using the new encoding to see their logits vs one?hot.	Would show the numeric shift directly.

3.?Assumptions made

Assumption	Rationale
The model was trained and served with the same input shape before the deployment.	Evidenced by identical `encoding` field ("one_hot") in training.
The hashed integer bucket is a single feature and not one?hot?expanded.	Deployment note: "switched from one?hot to hash encoding for latency".
Precision drop is primarily due to mis?encoding, not a new data drift.	30?day baseline precision (0.79) matches training?time performance; drop aligns exactly with deployment date.
Enterprise accounts are high?risk churners.	Domain knowledge: enterprise-tier customers more likely to churn, so precision loss is tied to mis?classifying them.
The model's decision threshold was tuned on the training distribution.	Standard practice; shift in distribution changes effective threshold.
No

Engineering counterpart
7:47 AM - Responded in 0s

What would you verify before shipping this to production?

Agent response
7:48 AM - Responded in 42s

1.?What to verify before the next production cut

> Goal: guarantee that the "hash?encodes plan_tier" change (and the accompanying under?representation of Enterprise accounts) doesn't wreck precision or other key signals for the next 24?h of traffic.

Area	What to check	Why it matters	Tool / test
Feature mapping	1. The hash function used at inference is exactly the same that was used to build the training input. 2. The resulting integer is deterministic for a given plan tier.	Any mismatch will produce a new feature that the model never saw.	- Unit test: `plan_tier_hash("enterprise")` -> constant across environments. - Smoke test: Feed a few training rows through the inference pipeline, print the hashed value, compare to the value stored in the training TF?Record/Parquet.
Input shape	Verify that the model still receives the same number of columns (e.g. 1 hashed column + other features) and that the dimensionality matches the checkpoint.	A mismatch will raise an error or silently corrupt logits.	- Schema diff: `TensorShape` of the input layer vs exported TensorFlow SavedModel signature. - Run `model.predict` on a toy batch; check shape of `predictions`.
Model weight consistency	If the model was trained on one?hot columns but now receives a hashed column, the weights it learned cannot be applied directly. Either (a) a new model must be trained from scratch on the hashed encoding, or (b) a feature?wise transformation (e.g. mapping 3-dim one?hot -> 1-dim hashed) must be applied before feeding the old weights.	Using the wrong weights will produce garbage predictions.	- If re?training: ensure the new training pipeline uses the hashed encoding and that validation AUC/precision still > 0.8. - If mapping: write a transformation layer (e.g. dense with identity kernel) and test that `model.predict` on old?fashioned inputs still returns the old logits.
Data?drift validation	Run the new inference pipeline on a held?out validation set that mirrors the current traffic distribution (e.g. 61?% free, 29		

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
7:48 AM - Responded in 6s

2.?Top Risks

#	Risk	Why it's critical	Likely impact
1	Wrong feature mapping - hash value ? training value	Model weights are tuned to one?hot - hashed values produce a completely different distribution	Immediate prediction errors -> precision collapse
2	Model mis?alignment - old model used on new input	No retraining, weights no longer map to the hashed feature	Systematic bias; precision drop, possible churn mis?classification
3	Enterprise under?representation - shift not compensated	Enterprise customers are high?risk and their churn probability is now underestimated	Precision drops further; could impact revenue forecasting
4	Threshold mis?calibration - threshold tuned to training distribution	Skewed distribution shifts the operating point -> precision falls	False?negative churn alerts increase
5	Performance regression - hashing + extra mapping slows inference	Latency SLA breach	Customer dissatisfaction, API throttling
6	Code?base mismatch - unit tests pass but integration fails	New inference path may not be fully integrated with feature store / caching	Runtime errors, 5xx responses
7	Rollback failure - revert is not instantaneous	If the new deployment fails, you may still be on bad code	Extended outage until manual rollback

---

3.?Rollback Plan

Step	Trigger	Action	Notes
R1	Deployment metrics show <?0.70 precision within 15?min of cut?over	Rollback to previous version of `inference-api` (the one with one?hot encoding).	Use the Blue?Green or Canary deployment pattern so rollback is just a traffic switch.
R2	Inference layer fails health?check or throws `InvalidArgumentException` (shape mismatch)	Automatic fail?over to the backup pod (unchanged).	Ensure health?checks are strict: shape, dtype, non?nan logits.
R3	Production precision decreases below baseline for >?2?h	Hold the new version and route the last 10?% of traffic to the old model for A/B testing.	Allows gradual rollback or "partial rollback" if the new encoding works for most users.
R4	No rollback possible (e.g. database schema changed)	Manual rollout back to old Docker image and configuration.	Document the missing feature flag; re?enable the old encoding.
R5	Data?drift test (online A/B test) fails after 30?min	Trigger the rollback pipeline (automated or manual).	Include a "canary scorecard" to monitor real?time precision and latency.

Rollback prerequisites

1. Blue?Green deployment - old and new services run side?by?side.
2. Feature?flag that toggles the hashing path.
3. Versioned model store - old checkpoint retained along with new one.
4. Circuit?breaker - if 5xx or latency > SLA, automatically redirect traffic to the previous version.
5. CI/CD tags - ensure that every commit gets a unique tag and the rollback image is available in the registry.

4.?Final Checklist Before Shipping

??	Item	Owner	Status
	Unit tests - hash mapping consistency	ML Ops	?
	Integration tests - inference schema, latency	Infra	

Engineering counterpart
7:48 AM - Responded in 0s
Can you tighten this into concrete next steps with owners and timing?

Agent response
7:48 AM - Responded in 13s
1.?Concrete "next?steps" playbook
Goal: 1?? fix the precision drop today; 2?? make the deployment robust for tomorrow.

#	Task	Owner	Deadline	Success Criteria	Notes
A1	Verify hash?to?one?hot mapping - Re?run the training?time hash function on a handful of plan?tier strings. - Compare with the value that the inference API now sees.	ML Engineer (Nina)	+1?h	Consistent mapping, no mismatch in `dtype`	Simple unit test in CI.
A2	Confirm input?tensor shape - Load the SavedModel in a local dev env. - Run a dummy batch (one?hot & hashed). - Ensure the model's signature matches the new input pipeline.	ML Engineer (Nina)	+2?h	`model.signatures["serving_default"].inputs` length = 1 (hashed)	If not, flag "Shape mismatch" and go to B.
A3	Create a "feature?mapping" shim - If the model was trained on one?hot, build a tiny pre?processing layer that converts hashed int -> one?hot before feeding into the legacy model.	Data Engineer (Arjun)	+3?h	`model.predict` on hashed input yields the same logits as on one?hot input.	Use a `tf.keras.layers.Embedding` or `tf.one_hot` layer.
A4	Re?train a new model with hashed encoding (parallel track) - Use the exact same data but encode `plan_tier` as a single hashed bucket. - Tune threshold on a held?out set that mimics the current traffic distribution.	ML Engineer (Nina)	+24?h	Offline AUC ? 0.84, precision on validation ? 0.78 (baseline).	Store the new checkpoint with tag `v3-hash`.
A5	Deploy the shimed model (A3) to a canary - Route 10?% of traffic to the new inference path. - Monitor precision, latency, error rate.	DevOps / Ops Engineer (Marta)	+4?h	Precision ? 0.76, latency ? 95?ms, error rate < 0.5?%.	Use

Engineering counterpart
7:48 AM - Responded in 0s
What's the one thing you'd want a second pair of eyes on?

Agent response
7:48 AM - Responded in 8s

The one thing that can make or break the rollout is the hash?to?one?hot mapping.

> Why double?check it?
> - The model was trained on a three?dimensional one?hot vector; the inference path now gives you a single hashed integer.
> - If the hash function, bucket size, or `dtype` is even slightly off, the model receives a feature it was never exposed to, so the learned weights no longer make sense.
> - A mismatch here will turn the precision drop from "nice?to?have" into an "unrecoverable" loss before you even get to A/B testing or rollback.

What to examine with a second pair of eyes:

1. Deterministic hash - `plan_tier_hash("enterprise")` must always produce the same integer across dev, staging, and prod.

2. Bucket range - Is the bucket size (e.g., 0?2 for three tiers) the same as in training?
3. Shape & dtype - The inference pipeline must feed a single 1?D numeric tensor (not a vector or string) into the model's input signature.
4. Pre?processing shim - If you're still using the legacy model, the layer that converts the hashed value back to a one?hot vector must be exactly the inverse of the training transformation.

If that mapping is verified, the rest of the deployment (shimed model, canary, threshold tuning) will stand on a solid foundation. If it's wrong, you'll see an immediate precision collapse or even runtime failures.

Engineering counterpart
7:48 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.