Duopoly Pricing Agent – Business & Technical Summary

Introduction

When I designed this pricing agent, I didn't want it to just "work" under competition rules. I wanted something that a business manager could trust, that a data scientist could defend, and that an engineer could run without breaking constraints. Every choice was about finding balance: **profit vs. exploration**, **safety vs. adaptability**, and **simplicity vs. power**.

1. State Design

At first, I considered keeping all history so the agent could see every past event. But I realized this would blow up memory and make the agent slow. I settled on a **ring buffer (128 steps)** with incremental stats (EWMA means and variances).

Why 128? Large enough to capture meaningful patterns, but small enough to adapt quickly to market shifts.

Trade-off: Long-term seasonality beyond ~128 steps is lost.

Business impact: This guarantees the system won't "freeze" or slow down in production.

2. Demand Learning (Fast and Adaptive)

I wanted a demand model that adapts in real time. I tried full regressions at each step, but they were too heavy. Exponential smoothing was lighter, but ignored elasticity.

The compromise was a **decayed OLS regression**, updating running sums (Sx, Sy, Sxx, Sxy).

Why: Newer data is weighted more, so the agent adapts after shocks.

Drawback: It assumes a mostly linear demand curve.

Business impact: Keeps forecasts simple, interpretable, and quick enough to always meet the 0.2s runtime requirement.

3. Pricing Policy

I went through three phases:

- **Purely Greedy OLS:** stable but blind to new opportunities.
- **Purely UCB (bandit):** explored widely but wasted revenue on poor prices.
- **Hybrid (final choice):** OLS "favourite" price + UCB grid exploration.

Why these parameters?

- **W=80:** A window of 100 made the agent too slow to react; W=50 was too noisy. W=80 struck the best balance.
- **K=31:** K=21 felt coarse (missed sweet spots), K=41 slowed exploration with little added value.
- **ε=0.05:** 0.10 jittered too much, 0.02 missed shocks.
- **UCB_C=1.0:** 1.5 over-explored, 0.8 got conservative too early.

Business value: Fast learning early on, stable profit-seeking behavior later.

4. Cold Start & Safety Rails

I didn't want the agent to "gamble" blindly at the start, so it uses a safe mid-price until data arrives. Then I added rails:

- **Never below cost + buffer.**
- **Never too far above competitor.**
- **Always within legal price bounds.**

Business impact: Protects revenue, avoids losses, and builds trust with managers by never selling at a loss").

5. Development Journey

During testing, I faced a bug where the agent priced irrationally when the competitor had no capacity. Instead of ignoring it, I added a specific override check to handle that scenario.

Business angle: Identifies failure modes before they reach production.

6. Transparency (The Decision Card)

I realized a manager or analyst shouldn't need to be a data scientist to audit the agent. So I added a **Decision Card** in info_dump:

- OLS price, UCB price, chosen price
- Exploration rate ε, UCB bonus, constraints applied

Business impact: A product manager can explain pricing moves to leadership without needing to read the code.

7. What I Rejected

- **Unbounded history** → memory risk.

- **Deep RL / heavy optimizers** → too slow, disallowed, unpredictable.
- **Pure fixed rules** → safe but unresponsive to market changes.

8. Business Takeaway

This agent maximizes revenue safely under constraints. It adapts in real time, explores without being reckless. It consistently **maximizes revenue** while **protecting the bottom line**.

Appendix – Technical Terms Explained

- **EWMA (Exponentially Weighted Moving Average):** updates mean 'μ' with formula

$$\mu_t = \alpha x_t + (1 - \alpha)\mu_t - 1$$

  → gives more weight to recent sales.

- **Decayed OLS:** regression where old data is down-weighted so new data dominates.
- **UCB (Upper Confidence Bound):** bandit formula

$$P = \dot{r} + c\sqrt{\frac{lnt}{n}}$$

  → balances trying profitable prices vs. exploring less-tested ones.

- **ε-greedy:** small chance ε to pick a random price for discovery.