The Value of Information from a Forecast: A Step-by-Step Guide to Inventory Management under Demand Uncertainty with an Imperfect ML Signal

Companion Document to `inventory_voi.jl`

Kyumin Kim

May 24, 2026

Contents

Introduction: Decisions Made Before the Future Arrives

A great deal of economics, and almost all of operations, comes down to one uncomfortable fact: we have to commit to actions *before* we know how the world will turn out. A factory sets production today; demand reveals itself tomorrow. The decision and the resolution of uncertainty are separated in time, and that separation is where all the interesting structure lives.

This document studies that separation in its cleanest form. A firm must decide how much to produce *before* demand is realized. It can carry unsold goods forward as inventory, at a cost. And crucially, it has access to an imperfect forecast—think of a machine-learning classifier that predicts whether next period's demand will be high or low, but is sometimes wrong. The central question is deceptively simple to state and surprisingly rich to answer: **how much is that forecast worth?**

A short note on what this document is, and is not. The companion blog post for this project makes the broad case for the question and shows the headline results. This document is the technical counterpart: a careful, step-by-step record of *how* the model is actually set up and solved—the derivations, the timing logic, and the algorithm details that did not fit in the post. If the post is the "why," this is the "how."

The question sits at the intersection of three things that are usually taught and practiced separately: machine learning (which produces the forecast), decision-making under uncertainty (which prices it), and dynamic optimization (which links today's choices to tomorrow's state). Putting all three inside a single, fully worked model is the entire point of the exercise.

Why This Problem?

"ML for forecasting" and "optimization" are often treated as two separate specialties. But a forecast has no value in a vacuum—its value is defined entirely by the decision it improves. To price a forecast, you must embed it inside the optimization problem it feeds. This document works through exactly that embedding, end to end.

What Is This Document?

This document walks through an infinite-horizon inventory model with demand uncertainty and an imperfect demand signal, and explains, step by step, how value function iteration computes the value of that signal. It is written as a companion to the Julia implementation in `inventory_voi.jl`.

The goal is not just to explain the code, but to build intuition for *why* each step is necessary and how the pieces connect—the kind of detail a blog post has to skip. This document is deliberately text-only: the figures (production policies, value functions, the accuracy curve, the efficiency map) are generated by the companion script `inventory_voi.jl` and shown in the blog post; here we focus on the derivations and the algorithm.

The Problem: What Are We Trying to Do?

A monopolist lives forever and discounts the future. Each period it carries some inventory, observes a forecast of demand, produces, then sells once demand is realized. Leftover goods are carried forward at a convex holding cost. We want to:

- solve for the firm's optimal production and sales policy, and

- measure the **value of information** (VOI): how much higher the firm's value is when it has the forecast than when it does not, and how that compares to the value of a *perfect* forecast.

Why "Value of Information"?

A reduced-form approach might correlate forecast accuracy with profit across firms. Here we go structural: we model the firm as a forward-looking optimizer, solve its problem with and without the forecast, and take the difference in value. That difference is the forecast's worth, expressed in the same profit units the firm already cares about.

The Model

State, Actions, and Timing

The single dynamic state variable is inventory $S_t \geq 0$. Each period the firm chooses two controls: production $q_t \geq 0$ and sales $y_t \geq 0$. The timing *within* a period is what makes this an information problem, so it must be stated precisely:

1. The firm begins the period holding inventory S_t.
2. It observes a signal $\hat{\theta}_t \in \{\hat{H}, \hat{L}\}$ about demand. The true demand state is *not* yet known.
3. It chooses production $q_t \geq 0$ at constant unit cost c.
4. The true demand multiplier θ_t is realized.
5. It chooses sales y_t, subject to $0 \leq y_t \leq S_t + q_t$.
6. Leftover stock carries forward: $S_{t+1} = S_t + q_t - y_t$.

The Heart of the Model: Two Decisions, Two Information Sets

Production q_t is chosen *before* the demand state is known (the firm sees only the signal). Sales y_t are chosen *after* the state is known. The two decisions are made with different information. If they were made simultaneously with the same information, the forecast would have no role, and the value of information would collapse to zero. The whole problem is built on this asymmetry.

Demand: A Multiplicative Shock

Demand is high or low, drawn i.i.d. across periods:

$$\theta_t = \begin{cases} \theta_H & \text{with probability } \pi_H \\ \theta_L & \text{with probability } \pi_L = 1 - \pi_H \end{cases} \tag{1}$$

In the implementation $\theta_H = 1.3$ (a 30% demand boost) and $\theta_L = 0.8$ (a 20% drop), with $\pi_H = 0.5$.

Why i.i.d.?

Because demand is independent across periods, this period's signal carries no information about *next* period's demand. The consequence is enormous for tractability: the firm's belief never becomes a state variable. The only thing carried across periods is physical inventory S, so the value function is one-dimensional, $V(S)$. (If demand were Markov, the belief would enter the state and the value function would be $V(S, \text{belief})$—a genuinely harder, two-dimensional problem.)

Payoffs

The three primitive functions are deliberately simple:

$$p(y, \theta) = a\theta - by \qquad \text{(inverse demand)} \tag{2}$$

$$C(q) = c\,q \qquad \text{(production cost: constant marginal cost)} \tag{3}$$

$$h(S) = \tfrac{1}{2} h\, S^2 \qquad \text{(holding cost: convex)} \tag{4}$$

The inverse demand is linear, so marginal revenue is linear. The production cost is *linear*—this is exactly the constant-marginal-cost case implied by a constant-returns-to-scale Cobb–Douglas technology, where the unit cost c is pinned down by factor prices. The holding cost is *convex*, and that convexity is what keeps the inventory problem non-trivial: piling stock into the warehouse gets expensive at an increasing rate.

The Information Structure: From Confusion Matrix to Posterior

The ML classifier is summarized by two accuracy rates:

$$\alpha \equiv \mathrm{P}(\hat{H} \mid \theta_H) \;\text{ (true positive rate)}, \qquad \beta \equiv \mathrm{P}(\hat{L} \mid \theta_L) \;\text{ (true negative rate)}. \tag{5}$$

But the firm does not act on these directly. When it *sees* a signal, it needs the probability that the state is good *given* that signal—a posterior. The four joint probabilities $\mathrm{P}(\theta, \hat{\theta})$ come from the chain rule (prior $\times$ conditional accuracy):

$$\begin{aligned} &\mathrm{P}(\theta_H, \hat{H}) = \alpha\,\pi_H && \text{(true positive)} && \mathrm{P}(\theta_H, \hat{L}) = (1-\alpha)\,\pi_H && \text{(false negative)} \\ &\mathrm{P}(\theta_L, \hat{H}) = (1-\beta)\,\pi_L && \text{(false positive)} && \mathrm{P}(\theta_L, \hat{L}) = \beta\,\pi_L && \text{(true negative)} \end{aligned} \tag{6}$$

Summing each signal column gives its marginal probability; Bayes' rule (joint over marginal) gives the posteriors:

$$\pi_H^{\hat{H}} \equiv \mathrm{P}(\theta_H \mid \hat{H}) = \frac{\alpha\,\pi_H}{\alpha\,\pi_H + (1-\beta)\,\pi_L}, \qquad \pi_H^{\hat{L}} \equiv \mathrm{P}(\theta_H \mid \hat{L}) = \frac{(1-\alpha)\,\pi_H}{(1-\alpha)\,\pi_H + \beta\,\pi_L}. \tag{7}$$

This is the `posterior()` function.

The Classifier Quality Is a Dial

A perfect classifier ($\alpha = \beta = 1$) makes the posterior degenerate: the signal *is* the truth. A useless classifier (a coin flip, $\alpha = 1 - \beta$) makes the posterior collapse back to the prior: the signal tells you nothing. Every classifier in between places the posterior somewhere on the segment between "prior" and "truth." That position is precisely the lever that sets the value of information.

The Bellman Equation

From Sequence Problem to Recursion

The firm maximizes expected discounted profit:

$$\max_{\{q_t, y_t\}} \mathbb{E}_0 \sum_{t=0}^{\infty} \delta^t \left[(a\theta_t - b y_t)\, y_t - c\, q_t - \tfrac{h}{2} S_t^2 \right]. \tag{8}$$

The standard trick is to split the infinite sum into "this period" plus "δ times the same problem starting next period." Calling the optimized future value $V(S)$ turns the whole problem into a one-period decision plus a continuation value. Because demand is i.i.d., the only state is S.

Two Stages, by Backward Induction

The two controls are decided at different times, so we solve them in reverse: first the *later* decision (sales), then the *earlier* one (production).

Stage 2 (sales), after θ is realized. Given available stock $A = S + q$ and the realized state θ:

$$W(A, \theta) = \max_{0 \le y \le A} \Big\{ \underbrace{(a\theta - by)\, y}_{\text{current revenue}} + \underbrace{\delta\, V(A - y)}_{\text{discounted value of leftover stock}} \Big\}. \tag{9}$$

Stage 1 (production), after the signal but before θ. The firm picks q and takes the expectation over the true state under the posterior belief:

$$Q(S, \hat{\theta}) = \max_{q \ge 0} \Big\{ -c\, q + \mathbb{E}_{\theta|\hat{\theta}}\, W(S + q, \theta) \Big\}, \qquad \mathbb{E}_{\theta|\hat{\theta}}\, W = \pi_H^{\hat{\theta}} W(\cdot, \theta_H) + (1 - \pi_H^{\hat{\theta}}) W(\cdot, \theta_L). \tag{10}$$

Full value function. Averaging over which signal arrives and subtracting the holding cost:

$$\boxed{V(S) = \mathbb{E}_{\hat{\theta}} \big[\, Q(S, \hat{\theta}) \,\big] - \tfrac{h}{2} S^2 = \mathrm{P}(\hat{H})\, Q(S, \hat{H}) + \mathrm{P}(\hat{L})\, Q(S, \hat{L}) - \tfrac{h}{2} S^2.} \tag{11}$$

Two Nested Expectations

Equation (11) hides two distinct expectations. The *inner* one, $\mathbb{E}_{\theta|\hat{\theta}}$, averages over the true demand state given the signal—the firm has produced but does not yet know θ. The *outer* one, $\mathbb{E}_{\hat{\theta}}$, averages over which signal fires in the first place. The $\max_q$ sits between them: chosen after the signal, before the state. Reading equation (11) from the inside out reproduces the period timing exactly.

Why the Holding Cost Sits Outside

The current-stock holding cost $\frac{h}{2} S^2$ depends only on S, not on the choices q or y. It is therefore additive and can be subtracted once, outside both maximizations, without affecting the optimal policy.

Why a Bellman Equation Is Necessary

If the firm were myopic, it would just sell to maximize current revenue and never hold inventory deliberately. But it is forward-looking: unsold stock today has value tomorrow, and that value depends on tomorrow's optimal behavior, which depends on the day after, ad infinitum. Equation (9) captures this infinite recursion through the term $\delta V(A-y)$. The Bellman equation is not an object of interest in itself; it is the computational bottleneck between the model primitives and the value we want to measure.

The Three Information Regimes

The three regimes differ *only* in the belief used when choosing production. Stage 2 (the sales value W) is identical across all three.

1. **No information.** Ignore the signal; use the prior π_H:
$$V^{\text{no}}(S) = \max_{q\geq 0}\big\{-cq + \pi_H W(S+q,\theta_H) + \pi_L W(S+q,\theta_L)\big\} - \tfrac{h}{2}S^2.$$
2. **Signal.** Use the posterior $\pi_H^{\hat{\theta}}$ from equation (7); this is equation (11).
3. **Perfect information.** Choose q *knowing* θ, then average:
$$V^{\text{perf}}(S) = \pi_H \max_{q\geq 0}\big\{-cq + W(S+q,\theta_H)\big\} + \pi_L \max_{q\geq 0}\big\{-cq + W(S+q,\theta_L)\big\} - \tfrac{h}{2}S^2.$$

Why Perfect Information Is the Ceiling

Compare the production step under belief versus perfect information:

$$\text{belief: } \max_q \mathbb{E}_\theta[\,\cdot\,] \qquad \text{vs.} \qquad \text{perfect: } \mathbb{E}_\theta\big[\max_q(\,\cdot\,)\big].$$

The max and the expectation swap order. Under perfect information the firm tailors q to each state *before* averaging; under belief it must pick one q for the average state. By Jensen's inequality, $\mathbb{E}[\max] \geq \max[\mathbb{E}]$, so perfect information always weakly dominates. This is the formal reason it is an upper bound on VOI.

The Algorithm: Value Function Iteration

Why Iterate?

The value function appears on both sides of equation (11): V is on the left, and it is buried inside W on the right through the term $\delta V(A-y)$. This is a fixed-point equation $V = T[V]$, where T is the Bellman operator. It cannot be solved in closed form, so we iterate. Because $\delta < 1$, T is a **contraction mapping**, which guarantees convergence to the unique fixed point from *any* starting guess.

Handling a Continuous State

Inventory S is continuous, so V is stored on a grid and evaluated off-grid by **linear interpolation** (`make_V_interp()`). Production q and sales y are genuine *continuous* choices, each solved by a bounded scalar optimizer (Brent's method, via `Optim.jl`).

One Bellman Sweep, Step by Step

A single application of T (the `bellman()` function) does the following:

1. **Interpolate the current V.** Wrap the current value vector in a linear interpolant so it can be evaluated at any S' (`make_V_interp()`).
2. **Compute beliefs.** Convert (α, β) into posteriors via equation (7) (`posterior()`).
3. **Optimize production at every state.** For each S on the grid, choose q to maximize $-cq + \mathbb{E}_{\theta|\hat{\theta}} W(S+q, \theta)$ (`best_belief()` or `best_perfect()`). Each evaluation of $W(S+q, \theta)$ solves the sales problem of equation (9) *directly* via `sales_value()`—there is no grid over available stock A and no cached W. The quantity $A = S + q$ is purely intermediate, never a state variable.
4. **Combine by regime and subtract holding cost.** Average over signals (signal regime) or states (perfect regime), then subtract $\frac{h}{2}S^2$, producing the updated V_{new}.

Backward Induction: Time Runs One Way, Computation the Other

In the model, the firm produces *first* and sells *later*. But the code solves the sales value *first* (it evaluates W) and the production choice *later*. This is not a contradiction—it is backward induction. To choose production well, the firm must already know the value of selling the resulting stock. So we evaluate the later decision's value first, then step back to the earlier decision. (Solving W directly inside the production search is the simplest, exact choice; caching W on a grid of A values would trade a little accuracy for speed, but it is not necessary.)

The Outer Loop

The `solve_vfi()` function repeats the sweep until convergence:

1. Initialize $V^{(0)} = \mathbf{0}$ (a vector of zeros, one per grid point).
2. Compute $V^{(k+1)} = T[V^{(k)}]$ via one Bellman sweep.
3. Check convergence: $\|V^{(k+1)} - V^{(k)}\|_\infty < \texttt{tol}$.
4. If converged, return V^*. Otherwise, repeat.

The Guess Starts Wrong, and That Is Fine

The very first sweep evaluates W from $V^{(0)} = \mathbf{0}$—a completely wrong value function. But the contraction property means each sweep moves the guess closer to the truth: the sales values W get sharper as V improves, which sharpens the next V, and so on. The sup-norm gap shrinks geometrically (roughly by a factor of δ each sweep), so convergence is guaranteed regardless of the starting point.

Measuring the Value of Information

Solve all three regimes and read each value at $S = 0$ (an empty warehouse—a clean common reference point). Then (`compute_voi()`):

$$\text{VOI}_{\text{signal}} = V^{\text{signal}}(0) - V^{\text{no}}(0), \tag{12}$$

$$\text{VOI}_{\text{perfect}} = V^{\text{perfect}}(0) - V^{\text{no}}(0), \tag{13}$$

$$\text{efficiency} = \frac{\text{VOI}_{\text{signal}}}{\text{VOI}_{\text{perfect}}} \in [0, 1]. \tag{14}$$

Efficiency answers: "what share of the maximum attainable (perfect-information) value does this classifier capture?" The ordering $V^{\text{no}} \leq V^{\text{signal}} \leq V^{\text{perfect}}$ always holds—more information is weakly better.

A Unit Caution: VOI Is a Present Value, Not a Per-Period Flow

Each $V(S)$ is the discounted sum of *all* future profits, $\mathbb{E}\sum_{t\geq 0}\delta^t[\,\cdot\,]$, not a single period's profit. So every VOI number above is a present value—the total discounted worth of having the forecast, today. If a genuine per-period (flow-equivalent) figure were wanted, one would annuitize by multiplying the present value by $(1-\delta)$; the code does not do this, so the reported numbers and plots are present values throughout.

Results

A Good Forecast Moves Production at Every Inventory Level

The cleanest way to see the forecast act is the production policy $q^\star(S)$—how much the firm makes as a function of the inventory it begins the period with—traced separately for each information regime. Under the signal regime we plot the *ex-ante* policy: the production committed after seeing the signal but before demand is realized, averaged over which signal fires (weighted by $\Pr(\hat{H})$ and $\Pr(\hat{L})$). This is built in `extract_production_policies()` from the per-belief optimizers `q_opt_belief()` and `q_opt_theta()`, and plotted by `plot_production_policies()`.

The striking feature is not just that the three curves are ordered, but *which way*: the firm produces *less*, not more, as information improves. No-information production sits highest, the signal regime below it, and perfect information lowest, at every inventory level. The reason is precautionary overproduction. Without a forecast, the firm hedges against the chance of strong demand by building a buffer it might not sell; when demand then turns out weak, that buffer becomes unsold stock carrying a holding cost. A forecast lets the firm trim that hedge—and perfect information removes the need for it almost entirely, since the firm makes only what it knows it can sell. Here, then, the value of information shows up less as "selling more" and more as "wasting less": it shrinks the defensive over-build rather than expanding output. That this is the dominant channel is a property of the payoff structure—when over-producing into a weak state (a convex holding cost) is the more painful mistake, information is most valuable as a brake. It is the same asymmetry that makes the efficiency map lean toward the true positive rate.

Where the Value Lives: Value Functions and State-wise VOI

The value of information is, by definition, the vertical gap between value functions. Computing $V(S)$ for all three regimes makes that gap explicit at every inventory level, and the differ-

ence $\Delta V(S) = V^{\text{regime}}(S) - V^{\text{no}}(S)$ *is* the value of information, state by state. The routine `solve_values_all_regimes()` returns all three value functions and both VOI curves; `plot_value_and_voi_by_stat` draws them as two stacked panels ($V(S)$ on top, $\Delta V(S)$ below).

Why Report at $S = 0$?

For a single headline number, evaluate the value functions at the empty-warehouse reference point $S = 0$ (printed by `print_voi_summary()`). It is a clean common baseline across regimes. But the state-wise difference $\Delta V(S)$ is more informative: it shows not just *how much* the forecast is worth, but *where*—at which inventory levels—it matters most.

Pricing the Forecast: VOI Rises with Accuracy

Sweeping the classifier's accuracy from a coin flip to perfection traces out the value of the forecast. At accuracy 0.5 the posterior collapses to the prior and VOI is exactly zero; at accuracy 1.0 efficiency reaches 100% by construction. The shape in between shows how the returns to forecast quality are distributed across the accuracy range. The main loop sweeps accuracy and records $V^{\text{signal}}(0) - V^{\text{no}}(0)$ at each point, against the perfect-information ceiling.

A Mediocre Model Is Worth Almost Nothing

A barely-better-than-random classifier captures almost none of the available value: its posterior hardly moves from the prior, so the firm hardly changes its behavior, so it hardly captures any gains. How sharply the value is back-loaded toward high accuracy is exactly what the accuracy sweep reveals for a given parameterization.

Not All Accuracy Is Equal

Letting the true positive rate (TPR) and true negative rate (TNR) vary separately reveals an asymmetry that a single "accuracy" number would hide. Sweeping both rates over a grid and recording efficiency at each (TPR, TNR) pair produces a two-dimensional map; efficiency typically responds more strongly to one rate than the other, depending on which kind of forecasting error is more expensive given the payoff structure.

The Economics, Not the Confusion Matrix, Picks the Error to Fix

When the cost of being caught short in a good state (under-producing, leaving profitable sales unmet) outweighs the cost of over-producing into a bad state (a bounded, convex holding cost), the firm values reliably calling the upside more than the downside. The practical lesson: if you can only improve one dimension of your model, let the payoff structure—not the classifier metrics—tell you which one.

A Note on Parameterization

How visible these effects are depends on the parameters. With a narrow demand spread or an expensive holding cost, the regimes' value functions sit almost on top of each other and the gaps are hard to see. Widening the demand spread (a larger gap between θ_H and θ_L) and lowering the holding-cost curvature h both enlarge the value of information and make the regime-by-regime plots more legible. The companion script exposes a `high_voi_params()` preset that does exactly this, used as the default in `main()`; the baseline `Params()` values are also printed for comparison.

Code–Theory Mapping

Theory	**Code**	**Section**
Posterior from confusion matrix, eq. (7)	`posterior()`	§4.4
Linear interpolant of V	`make_V_interp()`	§7.2
Sales value $W(A, \theta)$, eq. (9)	`sales_value()`	§5.2
Production step, $\max_q$ under belief	`best_belief()`	§5.2
Production step under perfect info	`best_perfect()`	§6
Bellman operator $T[V]$, eq. (11)	`bellman()`	§5
Fixed-point iteration	`solve_vfi()`	§7.4
All regimes + state-wise VOI	`solve_values_all_regimes()`	§8
Value of information at $S = 0$	`compute_voi()`	§8
Per-belief / per-θ optimal q	`q_opt_belief()`	§9.1
(and, knowing θ)	`q_opt_theta()`	§9.1
Production policies $q^\star(S)$ by regime	`extract_production_policies()`	§9.1
Policy plot	`plot_production_policies()`	§9.1
$V(S)$ and state-wise VOI plot	`plot_value_and_voi_by_state()`	§9.2
High-VOI parameter preset	`high_voi_params()`	§9.5
Summary printout	`print_voi_summary()`	§9.2
Figures and headline numbers	`main()`	§9

Note. Section numbers refer to this document. The companion script `inventory_voi.jl` is heavily commented and follows the same ordering.