Chapter 8 – Where AI Fits: Machine Learning as a Model, a Service, and a Risk

8.0 Before you start

Where we are. This chapter closes Part II. Chapter 4 named the data-driven family and deferred its risks here. Chapter 6 said the same about surrogates. Chapter 7 said the same about validating a model whose parameters mean nothing. Chapter 2 said, back in Sec. 2.6.5, that a trained network is not a twin – it is a *model*, one candidate for the model slot inside a twin – and that Chapter 8 would cover what changes when you put a learned model there.

All of that comes due now.

Something is different about this chapter, and it is worth saying out loud. This is the only Part II topic you may already know more about than the book does. Plenty of working software engineers have shipped a model, tuned a classifier, or at least argued about one at length. So this chapter does not explain what a neural network is, does not teach the train/test split (Chapter 7 already used it, under the name *hold-out*), and does not tell you that data quality matters.

What it does is the thing you probably have not done: **put a learned model inside a system that acts on a physical object, and account for what that changes.** Almost everything specific to twins in this chapter comes from that sentence – from the fact that the model's outputs move water, or a turbine blade, or a maintenance vessel, and that the physical thing then produces the data the next model is trained on.

The register, stated plainly. This chapter teaches where machine learning belongs in a twin, what it costs there, and how each of its failure modes shows up as an engineering consequence you can design against. **It teaches no machine learning.** No architectures, no training procedure, no loss functions, no framework names.

What you are assumed to know. Everything so far. Especially: Chapter 1's value metric; Chapter 2's Sec. 2.6.5 and what closing the loop changes; Chapter 3's model registry, store and actuation guard; Chapter 4's three model families, the white-to-black-box scale, and above all Sec. 4.4.3's argument about training a fault detector on data containing faults; Chapter 6's surrogates; and all of Chapter 7 – residual, bias, spread, hold-out, validity envelope, credibility argument, expiry.

The maths budget. Unchanged from Chapters 6 and 7: no derivatives, no integrals, no matrices, no probability notation, no named distributions, and no loss function written down. The chapter has one quantitative set piece – a **confusion matrix built from counts** (Sec. 8.5.1) – which is integers and two ratios. Chapter 7's *spread* and Chapter 6's `1/sqrt(N)` are reused, not reintroduced.

What this chapter deliberately does not cover. Any machine-learning method. How to build and run a Machine Learning (ML) pipeline – Chapter 14 owns the operational loop, and this chapter owes only what is *different* about it when the model is learned. Data engineering, labelling infrastructure and feature stores – Chapter 10. Service design and deployment topology – Chapters 11 and 12; edge inference and federated learning are named once each and handed on. Security in general – Chapter 3 Sec. 3.3.3 owns

it, and Sec. 8.5.5 covers only the attack surface that exists *because* a model is learned. Regulation and AI standards – Chapter 13. Generative AI as a product category – named for the two jobs it actually does in twins, not surveyed.

Learning objectives

By the end of this chapter, you will be able to:

1. **Decide** whether a proposed use of machine learning belongs in a twin's model slot, beside the model as a service, or nowhere at all, and **justify** the decision from the twin's value metric rather than from the technology.
2. **Name** the six jobs service-side AI does in deployed twins, and **state** for each what it needs and how it fails.
3. **Compute** a detector's miss rate and false-alarm rate from a table of counts, and **explain** why accuracy is the wrong number to report for a rare event.
4. **Identify** the four risks a twin adds on top of ordinary machine-learning practice, including the feedback loop a twin creates in its own training data.
5. **Extend** Chapter 7's credibility argument to cover a learned model, including a runtime input-coverage check and an expiry clause that a physics-based model does not need.

8.1 Two words that keep arriving together

"Digital twin" and "AI" are spoken in one breath so routinely that a newcomer could be forgiven for thinking one implies the other. It does not, in either direction, and Figure 8.1 is the untangling – four distinct relationships where meetings usually assume one.

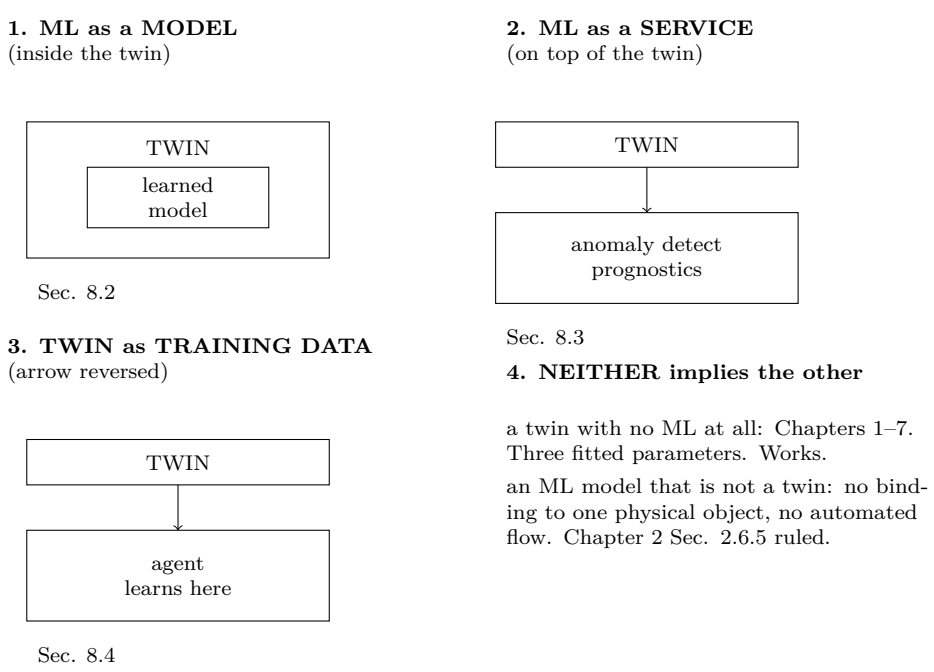

Figure 8.1 Four relationships, not one. Most confused meetings are two people holding different boxes.

Untangling them takes one paragraph and saves whole meetings.

A twin need not contain any machine learning. Chapters 1 to 7 built a complete, useful, decision-serving twin of a plant pot with three fitted parameters and no learning of any kind. It detects the fault Chapter 1 costed, it carries a credibility argument, and nothing in it would be improved by a neural network.

A machine-learning model is not a twin. Chapter 2 Sec. 2.6.5 already ruled on this: a trained network has no persistent binding to a specific physical object, no automated data flow in either direction, and no services. It is a *model* – a candidate for one slot in Chapter 3's architecture.

But they genuinely do reinforce each other, and pretending otherwise is as unhelpful as conflating them. The survey literature treats AI and machine learning as core enabling technologies for twins rather than optional extras [1], and the reason is structural: a twin is a standing supply of contextualised, timestamped, physically-grounded data about one object, which is exactly the input a learned model needs and exactly the thing most machine-learning projects lack.

So there are three separable claims, and it is worth knowing which one somebody is making:

Claim	What it means	Where this chapter treats it
"The twin uses AI"	A learned model sits in the model slot	Sec. 8.2
"The twin has AI features"	Learned models sit beside the model, doing jobs the model does not	Sec. 8.3
"We're using the twin for AI"	The twin generates training data or acts as a training environment for something else	Sec. 8.4

The three cost different amounts, fail differently, and need different evidence. A vendor saying "AI-powered digital twin" has made none of them yet, and the useful reply is "in the model, or beside it?"

The chapter's shape follows the title. Model (Sec. 8.2), service (Sec. 8.3 and Sec. 8.4), risk (Sec. 8.5 and Sec. 8.6). The three parts are deliberately unequal. The model part is mostly consolidation, because Chapters 4, 6 and 7 already did that work. The service part is the one that will surprise you, because **most of the machine learning in deployed twins is not in the model slot at all**. The risk part is where the chapter earns its place.

8.2 Machine learning as a model

8.2.1 What Part II already decided

Four chapters have circled this. Consolidated into one table, here is everything Part II established about putting a learned model in the model slot:

Established in	The finding
Ch4 Sec. 4.4.2	It captures effects nobody derived – a new column instead of a new derivation
Ch4 Sec. 4.4.2	It needs a different expert, often an easier one to hire
Ch4 Sec. 4.4.3	It needs data covering the conditions you care about, and will not tell you when it does not have them
Ch4 Sec. 4.4.3	Its parameters mean nothing, so no human can sanity-check them
Ch4 Sec. 4.4.3	It has no assumption ledger; the assumptions are in the choice of inputs and in what the data happened to contain
Ch4 Sec. 4.5	The hybrid families exist precisely to keep some of the physics-based advantages
Ch4 Sec. 4.6	Where a model sits on the white-to-black-box scale decides what kind of argument you can make about it
Ch6 Sec. 6.5.1	A surrogate is a *speed* device, fitted to a model rather than to reality, and can fail unexpectedly off-distribution
Ch7 Sec. 7.5.5	The validation *procedure* is unchanged; the *argument* loses three of its moves and must be rebuilt on data coverage

That is a lot of cost. It is also, in the right circumstances, worth paying, and the literature's framing is fair: data-driven models are reached for exactly when the underlying physics or principles are not fully available [2], and reviews of specific domains report that relying on data-driven models alone for precise prediction is difficult, so physics-based modelling is often needed to aid them [3].

8.2.2 The one question that decides it

Everything above collapses into a single question, and it is not a technical one:

> **Can you write down the mechanism?**

If somebody can derive the behaviour from a conservation statement, a material property or a datasheet, derive it. You get inspectable parameters, an assumption ledger, a validity envelope you can reason about, and a failure mode that drifts in a predictable direction. Chapter 4 got the pot's whole model out of "water is conserved" in about a page.

If nobody can – because the physics is unknown, or known but intractable, or spread across effects nobody has separated – then fitting is not a shortcut, it is the only route. Structural health monitoring reached this conclusion decades ago and it is the origin of the field's use of machine learning: represent normal conditions from measured data, then test for abnormality, because deriving the damage mechanism was not available [4].

Two corollaries that save arguments.

"We should use ML because we have a lot of data" is not a reason. Having data is a precondition, not a motivation. Chapter 1's question – what decision does this serve, and what does being wrong cost – is still the one that decides.

The answer is usually "some of both". Chapter 4 chose corrected physics for the pot: derived structure where the mechanism is known, a learned term for the part nobody derived. That is not a compromise, it is the design, and it is where a great deal of current practice sits [5]. Published work does exactly this shape – an analytical white-box friction model combined with a data-driven black-box term, because friction depends on load, temperature and time in ways that are hard to model analytically [6]. **Prefer the arrangement that keeps the largest inspectable part inspectable**, because Chapter 7 showed that the inspectable part is where the credibility argument comes from.

8.2.3 What a learned model does to the model registry

Chapter 3 Sec. 3.2.4 gave the twin a model registry: models with versions, resolvable by the runner, recorded against every prediction. A learned model strains it in three specific ways, and all three are yours to design.

A version is no longer just code. A physics model's version is its source plus its parameter set – Chapter 7 put `pot-3/v2` in the registry and that was sufficient. A learned model's behaviour is determined by the code, the trained weights, *and the data it was trained on.* Two of those three are large binary artifacts, and the third is a dataset that may no longer exist in the form it had. **Register all three, by content hash, or you cannot answer Chapter 7's question about which model produced a prediction.**

Retraining is a release. A refitted physics parameter set is a three-number diff a human can read. A retrained learned model is an opaque replacement whose behaviour differs everywhere by a little and somewhere by a lot. Chapter 7's golden trajectories (Sec. 7.3.3) become more important here, not less, because they are the only readable diff available: run version 3 and version 4 against the same recorded scenarios and report where they disagree.

The registry must carry the training range. Sec. 8.6.2 builds a runtime check out of it. The place for it to live is beside the model, versioned with it, because a model and the range it was trained on go stale together.

8.2.4 What it does to reproducibility

Chapter 5 Sec. 5.6 required that a run be reproducible from recorded inputs, and Chapter 7 Sec. 7.3.4 made reproducibility a precondition for having the credibility conversation at all. A learned model adds two threats.

Inference is usually deterministic; training almost never is. Given the same weights and the same input, most models produce the same output every time, and Chapter 5's requirement is met. But *retraining* on the same data generally does not reproduce the same weights, because of initialisation, data ordering, and hardware-level nondeterminism. So "we can rebuild the model from the data" is a weaker promise than it sounds. **Archive the weights, not just the recipe.**

Hardware and library versions change numeric results. A model that produces 0.61 on one accelerator may produce 0.62 on another. Usually harmless; occasionally

the difference between alerting and not, if the threshold sits between them. This is a golden-trajectory test, and it is exactly the "change nobody declared" case Chapter 7 Sec. 7.3.3 warned about.

8.2.5 What to ask

- Can somebody write down the mechanism? If yes, why are we fitting it?
- What part of this model is derived and what part is learned, and can the derived part run alone if the learned part refuses?
- What are the three artifacts of this version – code, weights, training data – and where is each?
- What happens to the credibility argument when this model is retrained?

8.3 Machine learning as a service

Here is the part that surprises people. **In most deployed twins, the machine learning is not in the model slot.** It sits beside the model, in Chapter 3's services layer, doing jobs the model does not do. Six jobs cover almost all of it.

8.3.1 Anomaly detection: learning what normal looks like

The job. Watch a stream of measurements, or a stream of residuals, and raise something when the pattern stops resembling ordinary operation.

Why it is the most common. It is the one job where a learned model has a decisive structural advantage: it needs no labelled faults. You train it on periods of normal operation and it flags departures. Diagnostic twins for offshore wind do exactly this – unsupervised learning fits a model of normal operation, flags departures from it and diagnoses them, on an operational floating turbine monitored through measurements [7]. The same pattern is the origin of machine learning in structural health monitoring: features representing normal conditions, then a test for novelty [4]. And twin reference implementations wire it in as a service in its own right, with a monitoring phase that watches the prediction error of the state estimator to detect anomalies [8].

Note carefully what that last one does. It runs the anomaly detector on the **residuals of a physics model**, not on the raw signal. That arrangement is much stronger than either half alone, and it is available to you the moment you have a model:

> The physics model says what should have happened. The residual is what did not. A learned detector on the residual stream is learning the shape of *model error*, which is a far smaller and better-behaved thing to learn than the shape of the world.

How it fails. It tells you *something changed*, not *what* or *whether it matters*. It will flag the day the greenhouse blinds were replaced with the same confidence as the day the pump failed. And it inherits Sec. 8.5.1's problem in full: if the "normal" training period contained faults, it has learned them as normal.

What it needs. A period of operation somebody is willing to certify as normal. That certification is a human act and it is the expensive part.

8.3.2 Prognostics: how long until it fails

The job. Estimate remaining useful life, or the probability of failure within a horizon, so maintenance can be scheduled rather than suffered.

This is the largest commercial application of twins that involves learning, and it has its own systematic literature [9]. Battery health is the most developed case, with hybrid approaches coupling a degradation model to a learned component [10], [11], and a review literature that is unusually candid about uncertainty [12].

How it fails, and it is the hardest failure in this section. Labels for "time until failure" require failures. You get one label per asset per lifetime, the label arrives years late, and the assets you most want to predict are the ones that have not failed. Techniques exist to stretch what data there is – transfer learning from a pretrained model adapted to a specific system is being applied to exactly this time-to-event problem in cyber-physical systems [13], and population-based approaches share information across a fleet of nominally similar structures [14]. Both are real, and neither manufactures a label that does not exist.

What to ask. How many failures are in the training data? If the answer is a small number, the honest claim is much narrower than the proposal will say.

8.3.3 Forecasting the inputs

The job. Predict what the twin's *inputs* will do, so the model can be run forward. Weather, demand, occupancy, price, inflow.

Why this one gets overlooked, and why it is the simplest to justify. It is often the highest value per unit of effort in the whole twin, because the physics model is already good and the thing limiting a day-ahead prediction is not the model – it is not knowing tomorrow's conditions. Chapter 7 Sec. 7.6.1 listed input and scenario uncertainty as the one source that *cannot* be reduced by better modelling. This is the service that reduces it.

How it fails. Gracefully, which is unusual and welcome. A forecast comes with a horizon over which it degrades, and everybody in the domain already knows this. The engineering requirement is to carry the forecast's own uncertainty into the twin's run rather than treating a forecast as a measurement – which is Chapter 6's Monte Carlo, fed by Chapter 7's ranges.

8.3.4 Perception: turning pixels and sound into state

The job. Get a state variable from a sensor that does not produce numbers directly. A camera that reads a gauge, counts items, or spots a misaligned part. A microphone that hears a bearing.

Why it belongs in a twin chapter. This is the service that can *remove an admitted limitation from the credibility argument.* Chapter 7's demonstrator argument conceded, under assumption A3, that the twin cannot distinguish a failed pump from a displaced dripper. No amount of moisture modelling fixes that – the two are identical in the data available. A camera makes them different. Sec. 8.7.3 does exactly this.

How it fails. Lighting, viewpoint, dirt on the lens, and the class of object it never saw. Perception models are also the ones most exposed to the input-manipulation attacks of

Sec. 8.5.5.

8.3.5 Optimisation and control policy

The job. Choose an action. Which setpoint, which schedule, which maintenance window.

The important distinction, because the two get called the same thing. *Optimisation* searches over candidate actions by running the model on each one – that is Chapter 6's Monte Carlo and Chapter 11's subject, and it involves no learning. *A learned control policy* replaces the search with a model that outputs the action directly, having been trained to.

The second is where reinforcement learning enters twin practice, and the attraction is speed: a policy answers in microseconds where a search answers in minutes. The cost is that you have replaced a procedure whose reasoning you can inspect – here are the candidates, here is the one that scored best, here is the score – with one you cannot.

> **The rule that follows, and it is the strongest recommendation in this chapter.** A learned control policy on a closed loop puts a model you cannot interrogate on the far side of Chapter 3's actuation guard. Chapter 2 Sec. 2.8 already made closing the loop the point where the obligations change. Do not make a learned policy the *first* thing you close the loop with. Ship the searching version, learn what actions it chooses and why, and replace it with a policy only when latency is genuinely the binding constraint – and then keep the search running as an oracle you can compare against.

8.3.6 Natural-language interfaces

The job. Let a human ask the twin a question in words, or have the twin summarise a situation in words.

This is the newest of the six and the one moving fastest. The framing in the literature is that generative models bring an ability to work with both natural language and physics-based simulation, and proposals exist to merge classical model-driven twins with generative AI into a more adaptable system [15].

Where it is genuinely good. Explaining the twin to a person who is not going to read a residual plot. Drafting the incident narrative. Turning "which pots were dry last week" into a query. These are interface problems, and the failure mode is a wrong sentence a human reads and evaluates.

Where it is dangerous. The moment its output reaches the command path without a human in between. A language model that can *call* the twin's actuation endpoint has been placed on the wrong side of the actuation guard, and Chapter 3 Sec. 3.2.7's rules apply with no exceptions: every command audited with the state that justified it, and a fail-safe defined per command.

> **A test worth applying to any proposal in this category.** Draw where the generated text goes. If it goes to a screen a human reads, it is an interface. If it goes to an endpoint that does something, it is a control system with an unusually poor specification.

8.3.7 Three placement notes, then on

Each of these is a real topic that this chapter names and hands on.

Where the inference runs. On the device, at the edge, or in the cloud. Work exists on giving constrained microcontrollers adaptive decision-making by pairing them with a twin that offloads heavy computation, validates inferences, and retrains models without disrupting real-time operation [16]. That is a deployment-topology decision – Chapters 9 and 12.

Training across sites without moving data. Federated approaches let multiple parties contribute to a model without pooling their raw data, and communication-efficient variants exist for twin settings [17]. Relevant when the data is commercially sensitive or too large to move. Chapter 12.

What it costs to run. A learned service has an inference bill per call and a training bill per version, and Chapter 5 Sec. 5.3.4's arithmetic – does it fit in the request? – applies to it unchanged.

8.3.8 What to ask about any service-side proposal

- Which of the six jobs is this, and what would we do instead without it?
- What labels does it need, and where do they come from?
- Does its output reach a human or an actuator?
- What happens when it is unavailable? (A service that has no defined behaviour when down is a dependency nobody costed.)

8.4 The arrow the other way: the twin as a training environment

So far the twin has been the thing and AI has been an ingredient. It runs the other way too, and often enough to have a name.

8.4.1 Two jobs

A systematic survey of what it calls *AI simulation* – using virtual training environments to develop AI agents on simulated, synthetic data – finds the practice splits into two, with digital twins as the high-fidelity environments that make it credible [18]:

A safe place to train. Roughly two thirds of the studied work uses the twin as a virtual training environment in which an agent can act, fail, and learn without breaking anything [18]. The examples are the obvious ones – a drone learning to fly, a robot learning motion responses to situations it has not met. The attraction is stark: the agent needs thousands of failures, and the physical asset can afford none.

A source of labelled data. The remaining third generates or labels datasets [18]. A simulation knows the ground truth by construction, which is precisely what the physical world will not tell you. Rendering a part under lighting conditions that have never occurred, with the correct label attached for free, solves the labelling problem of Sec. 8.3.4 in a way nothing else does. Generative approaches are being proposed for the same

purpose – synthesising data that resembles the real thing to overcome data limitations [15].

8.4.2 The sim-to-real gap, which is the whole difficulty

An agent trained in the twin has learned to succeed **in the twin**. It has learned the twin's assumptions, including the ones nobody wrote down, and including the numerical artifacts of Chapter 5's step size. The gap between that and working on the physical object is the **sim-to-real gap**, and it is not a detail.

Figure 8.3 shows why Chapter 7's limitations change character in this use.

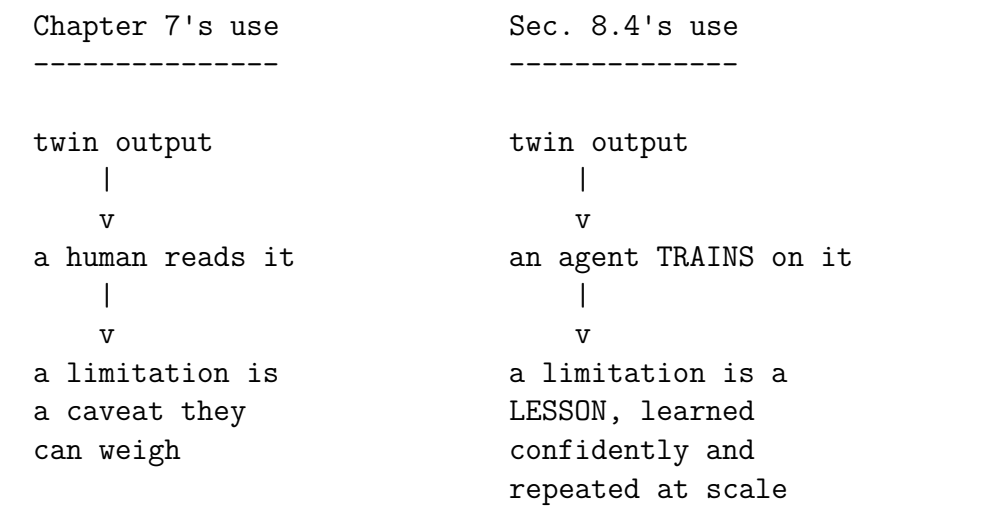

Figure 8.3 The sim-to-real gap restated as a Chapter 7 problem. The admitted limitations do not merely carry over -- they become the agent's confident mistakes.

Everything in Chapter 7 applies here with a twist. The credibility argument was about believing the twin's *outputs*. This is a use in which the twin's outputs become somebody's *training data*, so every limitation becomes a lesson the agent learns. Chapter 7's admitted limitations are the list of things your agent will be confidently wrong about.

Three consequences that are yours.

1. **The validity envelope is now a training-coverage specification.** If the twin is only valid between 350 and 900 reading units, an agent trained across its full range has been trained partly on fiction.
2. **Randomise what you are unsure of.** The standard mitigation is to vary the uncertain parameters across training runs so the agent cannot learn to depend on any single value. This is Chapter 6's Monte Carlo used for a different purpose, and Chapter 7's calibration spreads are where the ranges come from.

3. **Budget for the transfer, not just the training.** An agent that works in the twin is a milestone, not a deliverable.

   A sanity check before agreeing to this. Ask what the credibility argument for the twin says, and read its limitations section as a list of things the agent will get wrong. If nobody has written one, the twin is not ready to be a training environment, whatever its graphics look like.

8.5 Machine learning as a risk

Ordinary machine-learning risks – overfitting, leakage, imbalanced data, poor evaluation – apply here exactly as they do everywhere, and you know them. This section is about the four that a **twin** adds or sharpens, plus the attack surface.

8.5.1 The label problem, worked

Chapter 4 Sec. 4.4.3 made an argument and called it the strongest single argument in that chapter:

> **A model used to detect abnormality should not be taught what is normal by data that contains the abnormality.**

Chapter 4 asserted that this generalises. Here is the arithmetic that shows it, and it is the chapter's one set piece.

The setting. Chapter 1's baseline: watering faults occur and go undetected for about four days each. Suppose three months of ten-minute logging is available and nobody has labelled it. Somebody proposes training a detector on that history. Somebody else proposes Chapter 7's physics threshold. Both are evaluated on a test period containing **200 evening dose events, of which 6 were genuine watering faults** – 3 per cent, which is what a rare event looks like.

Detector A: the physics threshold from Chapter 7 Sec. 7.5.2. Expected step 56.4 reading units, alert when the observed step falls more than 9.0 units below it.

	The dose failed (6)	The dose was fine (194)
Alerted	6	2
Stayed silent	0	192

Detector B: the learned detector, trained on the unlabelled history – which, by Chapter 1's own baseline, contained watering faults nobody marked. It has therefore been taught that a dose producing no moisture step is a thing that sometimes happens.

	The dose failed (6)	The dose was fine (194)
Alerted	2	3
Stayed silent	4	191

Now compute accuracy, the number everyone reaches for first.

```
Detector A:  (6 + 192) / 200 = 198 / 200 = 99.0 per cent
Detector B:  (2 + 191) / 200 = 193 / 200 = 96.5 per cent
```

Both excellent. Ship either. Except:

```
Detector C, which is a function that returns "no alert" and nothing else:
            (0 + 194) / 200 = 194 / 200 = 97.0 per cent
```

The empty function beats the learned detector. That is not a rhetorical trick, it is what accuracy measures when one class is 3 per cent of the data, and it is why the number must not be reported for a rare event.

Report these two instead, each against its own denominator.

```
miss rate       = faults not alerted / faults          (the false negatives)
false-alarm rate = healthy doses alerted / healthy doses (the false positives)
```

Detector	Miss rate	False-alarm rate
A – physics threshold	0 / 6 = **0 per cent**	2 / 194 = **1.0 per cent**
B – learned on unlabelled history	4 / 6 = **67 per cent**	3 / 194 = **1.5 per cent**
C – the empty function	6 / 6 = **100 per cent**	0 / 194 = **0 per cent**

Now the detectors are distinguishable, and B is revealed as much closer to C than to A.

Convert to Chapter 1's units, which is the step that ends the argument. Each undetected fault costs about four days. Over this test period, A loses 0 days and B loses 4 x 4 = **16 experiment-days**. That is the sentence for the sponsor: not "96.5 per cent accurate" but "loses sixteen experiment-days per two hundred doses".

Three general lessons, and they outlive this example.

1. **Accuracy is meaningless for a rare event.** Ask for the miss rate and the false-alarm rate separately, each with its denominator. If a report gives you one number, it is hiding which of the two it traded away.
2. **The failure is in the labels, not the algorithm.** No better architecture rescues Detector B, because the training data says the fault is normal. This is the general form of Chapter 4's argument and it is why it generalises: **a learned detector's ceiling is set by what its training data called normal.**
3. **The fix is available and is not more machine learning.** Train on a period somebody has certified as fault-free, which turns the problem into the novelty detection of Sec. 8.3.1 – or, better here, put the learned detector on the *residuals of a physics model*, so that "expected" comes from a conservation law rather than from history.

8.5.2 Drift, and why it is worse for a learned model

Chapter 7 Sec. 7.4.7 established that a twin's model must be continually re-checked because the world changes under it, and Sec. 7.7.4 built the monitors: rolling residual bias, rolling residual spread, estimator gain. All of them work unchanged for a learned model. **Use them; nothing here replaces them.**

Three things are worse.

You cannot attribute the drift. When the pot's residuals developed a -4.8 bias in Chapter 7, the cause was in the assumption ledger: A1b, one drying rate for day and night. A named cause meant a targeted fix and an earned parameter. A learned model has no ledger. You can detect the drift and you can retrain, but you cannot say *what changed*, which means you also cannot say whether retraining will hold.

Retraining resets the credibility argument. Chapter 7 Sec. 7.7.2 listed "model version changed" as an expiry trigger. For a physics model a refit is three numbers and a re-run of the hold-out. For a learned model, retraining produces a different model, and every claim in the argument is about the old one. If the drift monitor fires monthly and the validation procedure takes three weeks, you have the backlog Chapter 7 Sec. 7.5.6 warned about – recognised in the literature as a scaling problem in its own right, since industrialising twins puts new demands specifically on the speed of validation and verification [19].

The drift may be in the inputs rather than the outputs. A learned model can go wrong without its residuals moving at all, if the inputs have shifted into a region where the residual happens to stay small by luck. Sec. 8.6.2's input-coverage check is the monitor for this, and it is a different monitor from the residual ones.

8.5.3 The feedback loop the twin creates in its own data

This one is specific to twins, it is not obvious, and it is the risk in this chapter most likely to be discovered late.

Be careful with the word. Chapter 2 Sec. 2.8's *control* loop is the twin acting on the physical twin. This is a **data** loop: the twin's actions change the data that trains its next model. Same word, different thing, and they arrive together.

The demonstrator, at Chapter 2's Stage 3. The twin now sets the doses, holding moisture near a target. It works. Six months later somebody retrains the drying model on the operational data and it is worse.

Here is why, and it is inevitable rather than unlucky:

1. A model of drying learns from variation in moisture. Dry spells and wet spells are the informative events.
2. The controller's entire job is to *remove* that variation. It succeeds.
3. So the operational data has almost no variation left to learn from.
4. The retrained model is worse, the control degrades, and the data gets slightly more informative again – which is the system limping toward an equilibrium of mediocrity rather than failing loudly.

Figure 8.2 draws the loop, because written as four numbered steps it reads like a sequence, and the damaging property is that it is a cycle with no alarm on it.

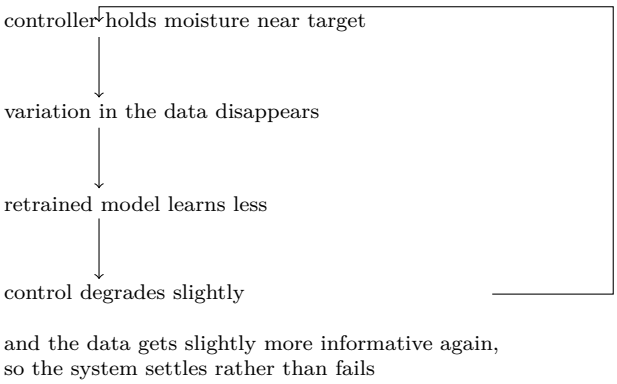

Figure 8.2 The data loop. Not Chapter 2's control loop: this one runs through the training set. It has no failure signal, which is why it is found late.

The absence of a failure signal is the actionable part. Nothing in the system reports this, so the only defence is to measure the *variation* in the training data over time and alert when it collapses – a monitor on the inputs rather than on the outputs.

This is Chapter 7's identifiability problem arriving by a different road. Sec. 7.4.4 showed that daily net-change data could not separate `k` from `g` because both effects acted together. Here the data cannot separate anything because the controller suppressed the effects. In both cases the information is not in the data, and no algorithm recovers it.

Three defences, in increasing order of cost.

Keep the calibration campaigns. Chapter 7's two campaigns were deliberate experiments, not operational logs, and this is a second reason to keep running them after deployment.

Log the controller's actions as first-class inputs. A model trained on moisture alone sees a flat line; a model that also sees the doses can at least learn the response. Chapter 3's audit record already requires this.

Deliberately perturb, occasionally. Let the moisture drift a little, on purpose, on a schedule, and treat the result as calibration data. This costs something real – and it is exactly the trade the identifiability discussion of Chapter 7 Sec. 7.4.4 said you would eventually face: an experiment that moves one effect without the other, on a physical twin you did not want to interrupt.

> **Ask this at design time, not after six months.** *What does this twin's own operation do to the data it will be retrained on?* If the answer is "removes the variation the model needs", you have found the problem in a meeting rather than in production.

8.5.4 Opacity, and the conversation you will actually have

Chapter 7 Sec. 7.7.3 listed the five questions an incident review asks, in order. Question 1 is what the twin predicted and measured; question 2 is which model version produced it. For a physics model there is a question 6 that gets asked informally and answered in one sentence: *why?* Because `k_day` is 4.28 and the window was 7.8 hours.

A learned model has no answer beyond "these inputs produced this output". That is a real cost, and it lands in three places: the incident review, the operator who has to decide whether to trust an alert at 3 a.m., and the regulator, if you have one.

The field's answer is explainability, and it is worth knowing the vocabulary without overestimating it. Explainable-AI surveys aimed at industrial settings frame model transparency as the key concept – a model revealing its internal operation on the inputs it processed – and are explicit that it matters most where a human is integrated into the process [20]. The same surveys are candid about the trade-off between accuracy and interpretability, and about the need for domain expertise in the explanation process itself [20].

What to do about it as an engineer, since you will not be building explainers.

- **Log the inputs, not just the output.** For a physics model the inputs can often be reconstructed. For a learned model they cannot, and without them question 1 has no answer. This is a storage cost you should agree to.
- **Prefer the arrangement that keeps something inspectable.** Corrected physics leaves a derived core that can be reasoned about even when the correction cannot. This is Chapter 4 Sec. 4.6's white-to-black-box scale cashing out as an operational property.
- **Prefer a model that reports its own uncertainty**, as Chapter 7 Sec. 7.5.5 recommended [12]. An output with a range on it lets a service degrade instead of guessing, which is worth more operationally than an explanation nobody reads.

8.5.5 The attack surface a learned model adds

Chapter 3 Sec. 3.3.3 owns security. This subsection covers only what is different **because** a model is learned, and hands the rest on.

Two things are different.

The training data becomes an attack surface. An adversary who can influence what the model learns can shape what it does, and the influence need not be an intrusion: in a twin that retrains on its own operational data, anyone who can affect the physical process can affect the next model. Note that this is Sec. 8.5.3's feedback loop with an adversary in it.

The inputs become an attack surface. Learned models can be induced to misclassify by inputs crafted for the purpose, and the twin security literature lists classifier manipulation among the attack classes to be studied [21]. Perception services (Sec. 8.3.4) are the most exposed, because their inputs are the least constrained.

The defences that belong in this chapter are the ones already built for other reasons: the input-coverage check of Sec. 8.6.2 rejects inputs unlike anything in training, whatever their provenance; Chapter 3's actuation guard stands between any model and any action; and Chapter 7's monitors notice a model whose behaviour has changed. The broader treatment – and the twin's own use as a security testbed, which is a genuine and separate topic – is Chapter 13's and the survey literature's [21].

8.5.6 What to ask

- What called the training data normal, and who certified it?
- Give me the miss rate and the false-alarm rate, with denominators.
- What does this twin's operation do to the data it will be retrained on?
- When this alerts at 3 a.m., what can the operator see besides the alert?
- What happens to the credibility argument the next time this is retrained?

8.6 What changes in the credibility argument

Chapter 7 Sec. 7.5.5 gave a four-row table of what a learned model takes away, and said that Chapter 8 would take it further. This is that, in three parts: the extended table, the check Chapter 7 named without building, and the expiry clause.

8.6.1 Chapter 7's table, extended into a checklist

Chapter 7's finding was that the validation *procedure* is unchanged and three of its *moves* are unavailable. Adding what to do instead:

Move	Physics-based	Learned	What replaces it
Inspect a parameter for plausibility	"Is 4.28 units/hour reasonable?"	Nothing means anything alone	Nothing replaces it. **Record the loss in the limitations section.**
Attribute a bias to a named assumption	A1b, immediately	Nothing to attribute to	A record of training-data composition, so drift can at least be correlated with what the data contained
State an envelope from the assumption ledger	Six rows, mostly free	No ledger exists	The **training-coverage table** of Sec. 8.6.2, derived from the data, and enforced at runtime
Predict the failure mode	Drifts as the assumptions predict	Confident output off-distribution, with no signal	The input-coverage check, plus a model that reports its own uncertainty where one is available
(new) Identify the model version	Source plus three numbers	Code, weights and data	Content hashes for all three in the registry (Sec. 8.2.3)

Move	Physics-based	Learned	What replaces it
(new) Re-issue after a refit	Re-run the hold-out; hours	A different model; the whole argument is about the old one	A validation procedure sized to the retraining cadence, or a slower cadence (Sec. 8.6.3)

And the harder question underneath, which Chapter 7 raised and could not settle. The standards practice that regulated domains rely on assumes a credibility assessment can be done once for a knowledge-driven model; for data-driven models that assumption is explicitly noted as unresolved [22]. That is not a gap in your project. It is the honest state of the field, and the correct engineering response is to make the *cadence* explicit rather than to pretend the one-off assessment transfers. Ongoing programme-level reviews say the same thing from the research side: verification, validation and uncertainty quantification for twins is an open agenda, and online approaches may be required [23].

8.6.2 Building the input-coverage check

Chapter 7 Sec. 7.5.5 recommended checking that inputs resemble the training range and did not build one. It is about twenty lines, and here it is for the demonstrator's corrected-physics model, whose learned term takes air temperature, humidity and light.

Step 1 – record the range, per input, at training time. Not at review time, when nobody can reconstruct it.

Input	Training minimum	Training maximum	Outside the range
Air temperature	14 C	27 C	Refuse; fall back to physics-only
Relative humidity	35 per cent	78 per cent	Refuse; fall back to physics-only
Light level	0	820	Refuse; fall back to physics-only
Moisture reading	372	902	Already refused by Chapter 7's envelope (350 to 900)

Step 2 – store it with the model, versioned together, in Chapter 3's registry. A model and the range it was trained on go stale at the same moment.

Step 3 – check before every call, in the service, not in the model. The model cannot refuse; the code that calls it can.

Step 4 – decide what "refuse" means, per input, in advance. This is the part that gets skipped and the part that matters. Here the answer is available and cheap: **fall back to the physics-only model.** The prediction gets worse in a known direction and the twin keeps working.

> **Notice where that fallback came from.** It exists because Chapter 4 chose corrected physics over a pure learned model for this pot. A twin whose model slot is entirely learned has no fallback – refusing means having no prediction at all. **The graceful-degradation path is bought at model-selection time, four chapters earlier, and cannot be added later.**

Step 5 – log every refusal, and alert on the rate. A rising refusal rate is the earliest warning of distribution shift you will get, and it arrives before the residuals move.

Now the honest limitation, which must go in the credibility argument. Checking each input against its own range is **necessary and not sufficient**. Suppose training contained hot dry days and mild humid days, but never a hot humid day. An input of 26 C at 76 per cent humidity passes every individual check – both values are inside their ranges – and is a combination the model has never seen. Per-dimension coverage cannot detect that, and a proper treatment of joint coverage is beyond this chapter and beyond most projects.

So state it as a limitation rather than solving it: "input coverage is checked per dimension; unusual *combinations* of individually ordinary inputs are not detected." A reviewer who reads that sentence trusts the rest of the document more, not less.

8.6.3 The expiry clause a learned model needs

Chapter 7 Sec. 7.7.2 required every credibility argument to state its expiry conditions. A learned model adds three rows to that table:

Expiry trigger	Detected by	Why it is new
Model retrained	Release pipeline	Retraining produces a different model; nothing in the old argument transfers
Input-coverage refusal rate exceeds its threshold	The check of Sec. 8.6.2	Distribution shift, detectable before the residuals move
Training data's certification as "normal" is called into question	A human, after an incident	Sec. 8.5.1: if the training period contained the fault, every claim about detection is wrong retroactively

The third is the uncomfortable one, because it invalidates the argument *backwards*. When somebody discovers that the pump was intermittently failing during the month you trained on, every alert and every silence since deployment is in question. Physics models have no equivalent.

8.7 Worked example: where AI goes in the demonstrator

Four proposals arrive. Following Chapter 6 Sec. 6.9 and Chapter 7 Sec. 7.8's pattern, each is decided on the value metric, and the answer is sometimes no.

8.7.1 Proposal 1: replace the fault detector's model with a learned one

The pitch. "We have three months of data. Train a model to predict the next reading; alert on the gap. No modelling expert needed."

Declined, and Sec. 8.5.1 has the arithmetic. The training history contains unlabelled watering faults, so the model learns a failed dose as ordinary. Miss rate 67 per cent against the physics threshold's 0 per cent; 16 experiment-days lost per 200 doses. The accuracy figure of 96.5 per cent that would have appeared in the proposal is beaten by a function that returns "no alert".

And the cheaper objection, which comes first. Chapter 8 Sec. 8.2.2's question: can somebody write down the mechanism? Yes – Chapter 4 did it in a page from "water is conserved". There is nothing to learn here.

8.7.2 Proposal 2: a learned correction on the drying rate

The pitch. "The model ignores temperature, humidity and light, and Chapter 7's week 3 residuals still have a spread of 2.0. Add a learned term."

Accepted, with three conditions. This is Chapter 4 Sec. 4.5.1's corrected physics – the arrangement Chapter 4 already chose for this pot – and it is the right shape because g, the expected step per 100 ml, stays derived from conservation. Sec. 8.5.1's objection does not apply, because the fault detector's expectation is not being learned.

The conditions:

1. **The input-coverage check of Sec. 8.6.2**, with the physics-only fallback and refusal logging.
2. **The physics-only model stays deployable.** It is the fallback, so it is not decommissioned and its golden trajectories keep running.
3. **A hold-out that shows the correction earns its place.** Chapter 7 Sec. 7.4.6's rule, unchanged: it must reduce the residual spread on a week neither the fit nor the fix has seen. If it does not beat 2.0 on fresh data, it is three artifacts of maintenance for nothing.

Note what this is worth if condition 3 fails, which is a real possibility. Then the answer is no, and finding that out costs one week of held-out data.

8.7.3 Proposal 3: a camera on the pot

The pitch. "A cheap camera and a small vision model can tell whether the dripper is over the soil."

Accepted, and it is the best of the four, because of what it does to the credibility argument rather than what it does to the model.

Chapter 4's assumption A3 said that all delivered water reaches the sensed soil, and Chapter 4 was blunt that this was design-limiting: a dose that arrives but misses the sensor is *indistinguishable* from a pump failure. Chapter 7's credibility argument had to carry that as an admitted limitation: "cannot distinguish a failed pump from a displaced dripper."

No amount of moisture modelling removes that limitation, because the two cases are identical in the available data. A camera makes them different. This is Sec. 8.3.4's point in its strongest form: **a perception service can delete a row from the limitations section, which is something no improvement to the model can do.**

What it costs, stated honestly: a camera, a labelled dataset of displaced and correctly-placed drippers (small, and you can create the positive examples deliberately, which is a rare luxury), a service, and its own credibility claim with its own miss rate and false-alarm rate. It also inherits Sec. 8.3.4's failure modes – lighting, a dirty lens – so its envelope needs stating and its unavailability needs a defined behaviour.

Whether to build it is Chapter 1's question, not this chapter's. Does distinguishing the two faults change what anybody does? If both faults send the same researcher to the same rig, the distinction is worth little and the camera is a nice-to-have. If a displaced dripper can be fixed remotely and a failed pump cannot, it is worth a great deal. Ask before building.

8.7.4 Proposal 4: a learned watering policy

The pitch. "Rather than the fixed schedule, learn a policy that decides doses from the current state."

Declined for now, with a route to yes, following Sec. 8.3.5's rule. This would put a model nobody can interrogate directly behind the actuation guard, on the first closed loop this system has ever had. The latency argument does not apply – Chapter 4 Sec. 4.5.3 recorded that the pot's physics model evaluates in nanoseconds, and Chapter 5 Sec. 5.3.4 costed a whole run at 32 microseconds – so a search over candidate schedules is affordable, and Chapter 6 Sec. 6.7 already sized it.

The route to yes: ship the searching version, run it for a season, and see what it chooses. If a learned policy is still wanted afterwards, it can be trained against the search's decisions and evaluated against the search as an oracle, which is a far better position than training it against nothing.

And Sec. 8.5.3's warning applies to whichever version ships. A controller holding moisture at target removes the variation the drying model learns from. Whatever closes this loop, the calibration campaigns of Chapter 7 keep running, the doses are logged as inputs, and somebody schedules the occasional deliberate drift.

8.7.5 The scorecard

Proposal	Decision	Deciding reason
Learned fault-detection model	**No**	Mechanism is derivable; training data contains the faults (Sec. 8.5.1)
Learned drying correction	**Yes**, with coverage check, fallback and a hold-out	Absorbs effects nobody derived, without learning the fault expectation

Proposal	Decision	Deciding reason
Camera for dripper placement	**Yes**, if the distinction changes a decision	Removes an admitted limitation, which modelling cannot
Learned watering policy	**Not yet**	Learned policy behind the actuation guard on a first closed loop

Two of four accepted, one conditionally, one deferred – and the accepted ones are accepted for reasons that have nothing to do with the technology being good. That is the chapter's thesis in miniature, and it is the same shape as Chapter 6 Sec. 6.9's and Chapter 7 Sec. 7.8's: **the question is never whether the method works, it is whether this decision needs it.**

8.8 Faded example: the offshore turbine, with AI

Chapters 4 through 7 each took the offshore turbine one step further: model families, run sizing, simulator choice, and a credibility argument with two outputs at two levels of rigour. Now place the machine learning. Two parts are worked; four are yours.

The system, recapped. A monitoring twin for a floating offshore wind turbine: a reduced-order structural model, a Kalman filter estimating loads from a handful of sensors, and a fatigue service reporting remaining life. Two outputs – maintenance scheduling (advisory) and life extension (certification, and a much harder evidence bar).

(a) The anomaly detector – worked. This is Sec. 8.3.1's job and the strongest candidate in the system. Diagnostic twins for floating offshore wind do exactly this: unsupervised learning builds a model of normal operation, detects anomalies and supports diagnosis on an operational turbine [7]. Three things make it a good fit here, and each is a general lesson.

It needs no failure labels, which is decisive, because Sec. 8.3.2's label problem is acute for a fleet of assets that mostly have not failed.

It runs on residuals, not raw signals. The Kalman filter already produces a prediction and a measurement every step; their difference is the natural input, and it is a far smaller thing to learn than the sea state.

Its output goes to a human, so Sec. 8.3.6's actuation test is passed trivially and Chapter 3's guard is not involved.

(b) Where the reduced-order model sits – worked, because it is the classification people get wrong. The reduced-order model is Chapter 6's surrogate: fitted to a higher-fidelity model, solving a speed problem. It is tempting to file it under "the twin uses AI". Resist. Whether it was built by projection or by fitting a network, it is *in the model slot* and it is validated against the model it was reduced from – which validates the reduction, not the physics. Chapter 7 Sec. 7.9(b) already made this point and it is worth repeating because the classification decides what evidence is owed.

Now yours.

(c) The life-extension output feeds a certification decision. Using Sec. 8.6.1's extended table, say which rows make a *learned* component hardest to defend in that setting, and whether you would allow one in the fatigue path at all. *Hint: which row invalidates an argument backwards?*

(d) Sec. 8.5.3's feedback loop was posed for a controller. Does it apply here? The twin is advisory, so it does not act – but it does schedule maintenance. Work out whether maintenance driven by the twin changes the data the twin's next model is trained on, and in which direction. *Hint: what happens to the population of observed near-failures when you get good at preventing failures?*

(e) Build the input-coverage table for the anomaly detector, following Sec. 8.6.2. Name three input dimensions, and say what "refuse" should mean for an offshore asset where the fallback cannot be "fall back to the physics model" in the same cheap way it was for the pot. *Hint: what is the safe behaviour of a monitoring service that does not know whether to trust itself?*

(f) A vendor proposes using the twin as a training environment for a control agent, per Sec. 8.4. Write the three questions from Sec. 8.4.2's sanity check as they apply here, and say what in Chapter 7's turbine credibility argument would have to exist first.

8.9 Posed problem: reviewing the AI twin proposal

No solution is given. This is the deliverable, and it is deliberately the document you are most likely to be handed in real life.

The situation. You are the engineering reviewer for a proposed twin of a hospital's chilled-water plant: four chillers, cooling towers, pumps, and a building load that varies with weather and occupancy. The vendor's proposal promises an "AI-native digital twin" that will:

1. predict plant energy use an hour ahead;
2. detect chiller faults early from an "AI anomaly engine trained on two years of building management system data";
3. recommend chiller sequencing to minimise energy cost, with an option to apply the recommendations automatically in a later phase;
4. answer operator questions in natural language.

The proposal says the anomaly engine "achieves 98.7 per cent accuracy" and that "the AI model continuously learns from live data".

Produce a review of no more than four pages containing:

1. **A classification of each of the four features** against Sec. 8.1's three claims and, where relevant, Sec. 8.3's six jobs. Say for each whether the learning is in the model slot or beside it.
2. **Your objection to "98.7 per cent accuracy"**, in the form of the two numbers you require instead and the denominators you require with them. Estimate, from a

stated assumption about how often a chiller fault occurs, what accuracy an empty function would achieve.

3. **The question that decides feature 2**, from Sec. 8.5.1: what certified the two years of data as normal? Say what you will do if the answer is "nothing did", and note that a hospital plant with two years of unlabelled history has almost certainly had faults in it.
4. **An analysis of "continuously learns from live data"** using Sec. 8.5.2 and Sec. 8.5.3. Two things to establish: what happens to the credibility argument on each retrain, and what feature 3 does to the data feature 2 trains on once sequencing is automated.
5. **A position on feature 3's automatic phase**, using Sec. 8.3.5's rule and Chapter 2 Sec. 2.8. State what must exist before you would agree, and who owns the safety argument for a hospital's cooling.
6. **A boundary drawing for feature 4**, following Sec. 8.3.6's test: where does the generated text go, and what is your requirement if the answer is ever "to an endpoint"?
7. **The input-coverage table you will require**, per Sec. 8.6.2, with at least four input dimensions, the fallback behaviour for each, and the per-dimension limitation stated honestly.
8. **The three expiry conditions** from Sec. 8.6.3, with owners, plus one sentence on which of them can invalidate the argument retroactively.

What a good review looks like. It does not object to AI. It classifies each feature, prices each one separately, and accepts the ones that are sound – feature 1 is very likely the best value in the proposal and the review should say so. It notices that features 2 and 3 interact through the data even though the proposal presents them as independent. It asks who certified the training data before it asks anything about the model. And it says plainly that the automatic phase of feature 3 needs a safety argument that this review is not, and that no amount of accuracy substitutes for.

8.10 Summary

Seven things, tied to the five objectives.

1. **"AI" and "digital twin" are three separable claims** (Sec. 8.1): the learning is in the model slot, beside it as a service, or the twin is being used to train something else. They cost differently, fail differently and owe different evidence. "In the model, or beside it?" is the question that starts the real conversation. *(Objective 1.)*
2. **One question decides the model slot** (Sec. 8.2.2): can somebody write down the mechanism? If yes, derive it – you get inspectable parameters, an assumption ledger and a predictable failure mode. If no, fitting is the only route. The answer is usually "some of both", and the right arrangement keeps the largest inspectable part inspectable. *(Objective 1.)*
3. **Most machine learning in deployed twins is not in the model slot** (Sec. 8.3). Six jobs: anomaly detection, prognostics, forecasting the inputs, perception, optimisation and control policy, and language interfaces. Forecasting is often the best value per unit of effort; perception is the only one that can delete a row from the limitations section; a learned control policy behind the actuation guard is the

one to be slowest about. *(Objective 2.)*

4. **The arrow runs the other way too** (Sec. 8.4). A twin is a safe place to train an agent and a source of labelled data it is otherwise impossible to get. The sim-to-real gap is the whole difficulty, and Chapter 7's limitations section is the list of things the agent will be confidently wrong about. *(Objective 2.)*
5. **Accuracy is the wrong number for a rare event** (Sec. 8.5.1). Of 200 doses with 6 faults, the physics threshold missed 0 and the learned detector missed 4 – yet scored 99.0 and 96.5 per cent, and an empty function scored 97.0. Report the miss rate and the false-alarm rate with their denominators, then convert to the value metric: 16 experiment-days lost. **A learned detector's ceiling is set by what its training data called normal**, which is Chapter 4's argument, generalised. *(Objective 3.)*
6. **Four risks a twin adds** (Sec. 8.5): drift you cannot attribute because there is no assumption ledger; a data feedback loop in which the twin's own control removes the variation its next model needs; opacity that lands in the incident review and the 3 a.m. alert; and an attack surface that now includes the training data. *(Objective 4.)*
7. **The credibility argument gains three expiry rows and a runtime check** (Sec. 8.6). The input-coverage check is twenty lines, is stored with the model, refuses per dimension, falls back, and logs – and its graceful-degradation path was bought back in Chapter 4 by choosing corrected physics over a pure learned model. Per-dimension coverage is necessary and not sufficient, and saying so in the document is what makes the rest of it credible. *(Objective 5.)*

And Part II's closing note, since this is where it ends. Five chapters ago you could not have said what a model was. You still cannot build one, and that was never the goal. What you can do now is sit in a room with the people who can, follow what they are arguing about, ask the question that changes the answer, and know which of their promises will need evidence later and what that evidence looks like. Part II's whole claim was that this is a learnable skill distinct from becoming a modeller, and that a software engineer who has it is worth far more to a twin project than one who has neither. Part III now builds the system all of this runs in.

8.11 Exercises

Solutions or hints follow. Each exercise names the objective it exercises.

8.11.1 *(Objectives 1 and 2.)* Classify each proposal as model-slot AI, service-side AI, twin-as-training-environment, or none of the three – and, where it is service-side, say which of Sec. 8.3's six jobs it is: (a) a network that predicts a pump's outlet pressure from its inlet pressure and speed, called by the simulation runner; (b) a model that reads a dial from a webcam; (c) generating ten thousand rendered images of a valve in different lighting to train a defect classifier; (d) a dashboard that highlights the five sensors with the largest residuals; (e) a language model that drafts the weekly plant report.

8.11.2 *(Objective 1.)* For each of these, decide whether to derive or to fit, and say why in one sentence: (a) the volume of water in a tank of known dimensions at a known level; (b) how long a specific brand of filter lasts before clogging in this particular water; (c) the

current drawn by a motor at a given load, where the datasheet exists; (d) how long an operator will delay responding to an alert at night.

8.11.3 *(Objective 3.)* A vibration-based bearing-fault detector is tested on 500 machine-hours containing 8 hours of genuine fault. It alerts during 5 of the fault hours and during 20 of the healthy hours. Build the confusion matrix, compute accuracy, the miss rate and the false-alarm rate, and compute what accuracy a detector that never alerts would achieve. Then say which single number you would put in the credibility argument's claim sentence.

8.11.4 *(Objective 3.)* Continuing 8.11.3: the team proposes lowering the threshold, which raises the alerts to 7 of 8 fault hours and 60 of 492 healthy hours. Recompute both rates. State the condition under which this is an improvement, in terms of the relative cost of a missed fault and a false alarm, and give one operational reason a high false-alarm rate can make a detector worse than useless even when its miss rate improves.

8.11.5 *(Objective 4.)* A traffic-management twin learns to predict junction queue lengths and uses the predictions to retime the lights. It is retrained monthly on operational data. Describe the feedback loop in four steps, following Sec. 8.5.3, and propose one defence that does not require degrading the traffic control.

8.11.6 *(Objective 4.)* Your anomaly detector was trained on "January through March, a normal quarter". In August somebody discovers that a sensor was reading 4 per cent low for all of February. State three things that are now in question, and say which of them can be resolved without retraining.

8.11.7 *(Objective 5.)* Write the input-coverage table for a learned model that predicts a building's next-hour electricity demand from outdoor temperature, hour of day, day of week, and yesterday's peak demand. Give a range for each, a refusal behaviour for each, and identify which dimension makes the per-dimension limitation of Sec. 8.6.2 most dangerous here.

8.11.8 *(Objective 5.)* Chapter 7's credibility argument for the demonstrator had five parts. Rewrite its **limits** paragraph on the assumption that Proposal 2 of Sec. 8.7.2 has shipped – the drying correction is learned. Add what must be added and remove nothing that Chapter 7 admitted.

8.11.9 *(Objectives 1 and 4.)* A colleague argues: "the physics model is just a model with hand-fitted parameters, so it is machine learning with fewer steps – the distinction in this chapter is a false one." Write the strongest version of their argument in two sentences, then the reply. *This is a genuinely good objection and the reply is not that they are wrong about everything.*

8.11.10 *(Objectives 1-5, and the one that leaves the book.)* Take the real plant-controller rig and the credibility argument you wrote for exercise 7.12.10. Pick exactly one of Sec. 8.3's six jobs, justify the choice from your own value metric, and write a one-page proposal for it containing: the job, what it needs, its failure mode, the two rates you will hold it to, its input-coverage table, and the row it adds to your expiry conditions. Then decide, honestly, whether you would fund it.

Solutions and hints

8.11.1. (a) Model slot – it is called by the runner and produces state. (b) Service-side, Sec. 8.3.4's perception job. (c) Twin as training environment, specifically Sec. 8.4.1's dataset-generation half. (d) **None of the three.** Sorting residuals by size is arithmetic; a dashboard that does it is not AI, and the exercise is here because such a feature is routinely sold as one. (e) Service-side, Sec. 8.3.6, with the boundary question: the report goes to a human, so it is an interface.

8.11.2. (a) Derive – geometry, and the mechanism is a formula. (b) Fit – clogging depends on this water's particulates in ways no datasheet covers; this is Sec. 8.2.2's "nobody can write it down". (c) Derive from the datasheet, then check the residual; if the residual has structure, that is the case for a learned correction, not for replacing the datasheet. (d) Neither, and this is the trap: it is a human-behaviour question that a twin should not be quietly answering with a model. Measure it, report it as a distribution of observed delays, and let it inform the *decision latency* Chapter 1 Sec. 1.2 defined – but do not build a predictor of operator behaviour without saying out loud that you are doing so.

8.11.3. Fault hours 8, healthy hours 492.

	Fault (8)	Healthy (492)
Alerted	5	20
Silent	3	472

Accuracy = (5 + 472) / 500 = 477 / 500 = **95.4 per cent**. Miss rate = 3 / 8 = **37.5 per cent**. False-alarm rate = 20 / 492 = **4.1 per cent**. A detector that never alerts scores 492 / 500 = **98.4 per cent** accuracy – better than the real one, again. For the claim sentence, neither rate alone is right: put both, with denominators, and if forced to one, the miss rate, because it is the one the value metric is about.

8.11.4. New matrix: alerted 7 of 8 faults, 60 of 492 healthy. Miss rate = 1 / 8 = **12.5 per cent**. False-alarm rate = 60 / 492 = **12.2 per cent**. It is an improvement when the cost of one missed fault exceeds roughly the cost of the 40 extra false alarms it buys – 2 extra faults caught for 40 extra alarms, so about 20 false alarms per fault caught. For a bearing whose failure destroys a machine, worth it many times over; for one that merely needs replacing sooner, not. **The operational reason:** alarm fatigue. A detector firing 60 times over 500 hours will be muted, and a muted detector has a miss rate of 100 per cent regardless of what the test said. That effect is invisible in every number above.

8.11.5. (i) The model learns to predict queues from the variation in observed queues. (ii) The retimed lights exist to reduce that variation and do. (iii) The next month's training data contains less of the queue-building behaviour the model needs. (iv) Predictions degrade, timing degrades, queues return, and the system oscillates around mediocre. **A defence that costs no control quality:** log the signal timings as first-class model inputs, so the model learns the *response* to timing rather than the bare queue history. (Deliberate perturbation also works and is the other standard answer, but it degrades control, which the question excluded.)

8.11.6. In question: (i) every anomaly-detection result since deployment, because February's readings were part of what defined normal; (ii) the detector's stated miss

and false-alarm rates, which were measured against a baseline that was itself wrong; (iii) any downstream decision taken on a February-influenced alert or silence. **Resolvable without retraining:** (iii), by re-reviewing the affected decisions from the audit record – if Chapter 3's record was built. (i) and (ii) require retraining on corrected data and re-running the evaluation. Note that this is Sec. 8.6.3's third expiry trigger, the one that invalidates backwards.

8.11.7. *Hint, not a solution.* Ranges should come from the data, not from what seems sensible. Outdoor temperature and yesterday's peak get numeric min and max; hour of day and day of week are categorical, and "outside the range" for them means a value never seen, which is rare and worth thinking about anyway. **The dangerous dimension is temperature in combination with day of week:** the training data almost certainly contains hot weekdays and cool weekends, and a hot public holiday passes every per-dimension check while being a combination the model has never seen. That is Sec. 8.6.2's limitation with a specific victim.

8.11.8. *Hint.* Chapter 7's limits paragraph admitted the sensor range, the 12-hour horizon, assumption A3, the 16 per cent detection floor, and assumption A5. All of those stand. Add: the learned correction's training ranges for temperature, humidity and light, with the statement that outside them the twin falls back to physics-only and the prediction degrades in a known direction; the per-dimension limitation on coverage checking; and the fact that the correction's contribution has been validated only against the weather that occurred during the training and hold-out periods. A good answer also notes that the *claim* sentence may now need two versions – one with the correction active and one for the fallback.

8.11.9. Their argument, stated strongly: "Both models have parameters whose values come from data. Chapter 7 fitted `k` and `g` by minimising residuals, which is what training is; the only difference is the number of parameters, and treating three as a different kind of thing from three million is arbitrary." **The reply, and it concedes the first half.** They are right that calibration and training are the same operation, and this book has said so – Chapter 7 Sec. 7.4.3 called the objective a choice and pointed at the same optimisation literature that trains networks. The difference is not the fitting, it is **what the structure came from**. The pot model's structure came from a conservation statement, which is why it has an assumption ledger, an inspectable `k`, a failure mode that drifts in a predictable direction, and a validity envelope derivable without any data. A learned model's structure came from a family chosen for flexibility, so none of those four exist and Sec. 8.6.1's table has to replace them. The distinction that survives is not fitted-versus-not; it is **derived structure versus assumed structure**, and everything this chapter says about risk follows from that, not from the parameter count.

8.11.10. No solution. One prediction: the honest answer to the last question will more often be no than you expect, and writing down what a service needs is what makes that answer visible before the money is spent rather than after.

8.12 Where to go next

In this book. Part II ends here and Part III builds the system. Chapter 9 is the connector that supplies every measurement this chapter's services consume. Chapter 10 is

the store and the data engineering – and it is where the training-data provenance that Sec. 8.2.3 and Sec. 8.6.3 depend on is actually built; a reader who found this chapter's registry requirements demanding should read Chapter 10 as the answer to them. Chapter 11 builds the services of Sec. 8.3 properly, including the optimisation that Sec. 8.3.5 distinguished from a learned policy. Chapter 12 is where the deployment questions of Sec. 8.3.7 – edge, cloud, federated – get decided, and where platforms that bundle ML tooling with twin tooling are assessed; open-source frameworks that integrate simulation models and learned models side by side already exist and are surveyed [24], [25]. Chapter 13 covers standards and regulation, including the AI-specific compliance landscape this chapter named once [26]. **Chapter 14 is the one this chapter leans on hardest**: every retraining trigger in Sec. 8.6.3 is a Chapter 14 workflow, and the validation-cadence problem of Sec. 8.5.2 is Chapter 14's to solve operationally. Chapter 15 asks what happens when twins compose, which is where the fleet-scale learning of Sec. 8.3.2 becomes an architecture question.

In the literature, if you want more. Grouped by what you would read them for.

- *AI and twins, the overall picture:* [1] for enabling technologies, [5] for the modelling-perspective view of why hybrid arrangements dominate, and [27] for AI-in-cyber-physical-systems including the safety and data-quality standards work now under development.
- *The twin as a training environment:* [18] is the systematic survey and the source of Sec. 8.4's two-way split; [15] for the generative angle on synthetic data.
- *Learning as a service, by job:* [7] for anomaly detection on a real offshore asset, [4] for the novelty-detection tradition it comes from, [9] for the prognostics literature, [12] and [10] for the battery case done with uncertainty taken seriously, and [13] and [14] for the two standard responses to label scarcity.
- *Risk and trust:* [20] for explainability in industrial settings, [21] for the twin security landscape including attacks on learned models, and [22] for the clearest statement that one-off credibility assessment does not transfer to data-driven models.
- *Consulted, not drawn on above:* [16] on twins supporting on-device learning for constrained hardware, [17] on federated training in twin settings, [28] on stating twin fidelity as knowledge equivalence, [29] on dependability across the data spectrum, and [19] on why validation speed is the bottleneck when twins are produced industrially rather than one at a time.

In the demonstrator. Exercise 8.11.10 is the assignment, and it is deliberately a proposal rather than an implementation, because the skill this chapter exists to teach is deciding. If you want to build something anyway, build the input-coverage check of Sec. 8.6.2 against the model you already have – it is twenty lines, it needs no machine learning at all, and the first time it refuses a reading you will understand why this chapter put it in the credibility argument rather than in the model.

8.13 References

[1] S. Mihai *et al.*, "Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2255–2291, 2022, doi: 10.1109/COMST.2022.3208773.

[2] Y. Jiang, S. Yin, K. Li, H. Luo, and O. Kaynak, "Industrial applications of digital twins," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2207, p. 20200360, Aug. 2021, doi: 10.1098/rsta.2020.0360.

[3] Z. Chen *et al.*, "Service oriented digital twin for additive manufacturing process," *Journal of Manufacturing Systems*, vol. 74, pp. 762–776, Jun. 2024, doi: 10.1016/j.jmsy.2024.04.015.

[4] K. Worden, G. Tsialiamanis, E. J. Cross, and T. J. Rogers, "Artificial Neural Networks," in *Machine Learning in Modeling and Simulation: Methods and Applications*, T. Rabczuk and K.-J. Bathe, Eds., Cham: Springer International Publishing, 2023, pp. 85–119. doi: 10.1007/978-3-031-36644-4_2.

[5] A. Rasheed, O. San, and T. Kvamsdal, "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective," *IEEE Access*, vol. 8, pp. 21980–22012, 2020, doi: 10.1109/ACCESS.2020.2970143.

[6] M. Heithoff, M. Trinh, J. Michael, B. Rumpe, and C. Brecher, "A Digital Shadow for Accurate Robot Motion Control: Integrating Data with Friction Models," Grand Rapids, USA, Jul. 2025. doi: 10.5283/epub.77667.

[7] F. Stadtmann and A. Rasheed, "Diagnostic Digital Twin for Anomaly Detection in Floating Offshore Wind Energy." arXiv, Jun. 2024. doi: 10.48550/arXiv.2406.02775.

[8] C. Gomes *et al.*, "Digital Twin Tutorial: The Incubator Case Study," in *Engineering Trustworthy Software Systems: 6th International School, SETSS 2024, Chongqing, China, April 14–21, 2024, Tutorial Lectures*, J. P. Bowen, C. Gomes, and Z. Liu, Eds., Singapore: Springer Nature, 2025, pp. 68–101. doi: 10.1007/978-981-96-4656-2_3.

[9] R. van Dinter, B. Tekinerdogan, and C. Catal, "Predictive maintenance using digital twins: A systematic literature review," *Information and Software Technology*, vol. 151, p. 107008, Nov. 2022, doi: 10.1016/j.infsof.2022.107008.

[10] A. Thelen, M. Li, C. Hu, E. Bekyarova, S. Kalinin, and M. Sanghadasa, "Augmented model-based framework for battery remaining useful life prediction," *Applied Energy*, vol. 324, p. 119624, Oct. 2022, doi: 10.1016/j.apenergy.2022.119624.

[11] T. Li *et al.*, "Coupling a capacity fade model with machine learning for early prediction of the battery capacity trajectory," *Applied Energy*, vol. 389, p. 125703, Jul. 2025, doi: 10.1016/j.apenergy.2025.125703.

[12] A. Thelen, X. Huan, N. Paulson, S. Onori, Z. Hu, and C. Hu, "Probabilistic machine learning for battery health diagnostics and prognostics—review and perspectives," *npj Materials Sustainability*, vol. 2, no. 1, pp. 1–33, Jun. 2024, doi: 10.1038/s44296-024-00011-1.

[13] Q. Xu, T. Yue, S. Ali, and M. Arratibel, "Pretrain, Prompt, and Transfer: Evolving Digital Twins for Time-to-Event Analysis in Cyber-physical Systems." arXiv, Apr. 2024. doi: 10.48550/arXiv.2310.00032.

[14] P. Gardner, L. A. Bull, J. Gosliga, N. Dervilis, and K. Worden, "Foundations of population-based SHM, Part III: Heterogeneous populations – Mapping and transfer," *Mechanical Systems and Signal Processing*, vol. 149, p. 107142, Feb. 2021, doi: 10.1016/j.ymssp.2020.107142.

[15] C. Savaglio, V. Barbuto, F. Mangione, and G. Fortino, "Generative Digital Twins: A Novel Approach in the IoT Edge-Cloud Continuum," *IEEE Internet of Things Magazine*, vol. 8, no. 1, pp. 42–48, Jan. 2025, doi: 10.1109/IOTM.001.2400035.

[16] A. Barbone, N. Bicocchi, M. Martinelli, R. Morandi, and M. Picone, "On-device AI and digital twins: A synergistic approach to intelligent cyber-physical systems," *Future Generation Computer Systems*, vol. 175, p. 108068, Feb. 2026, doi: 10.1016/j.future.2025.108068.

[17] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-Efficient Federated Learning and Permissioned Blockchain for Digital Twin Edge Networks," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2276–2288, Feb. 2021, doi: 10.1109/JIOT.2020.3015772.

[18] X. Liu and I. David, "AI simulation by digital twins: Systematic survey, reference framework, and mapping to a standardized architecture," *Software and Systems Modeling*, Aug. 2025, doi: 10.1007/s10270-025-01306-0.

[19] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, vol. 1, no. 5, pp. 313–320, May 2021, doi: 10.1038/s43588-021-00072-5.

[20] C. Trivedi *et al.*, "Explainable AI for Industry 5.0: Vision, Architecture, and Potential Directions," *IEEE Open Journal of Industry Applications*, vol. 5, pp. 177–208, 2024, doi: 10.1109/OJIA.2024.3399057.

[21] A. Jaber, I. Koufos, and M. Christopoulou, "A Comprehensive State-of-the-Art Review for Digital Twin: Cybersecurity Perspectives and Open Challenges," in *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, L. Barolli, Ed., Cham: Springer Nature Switzerland, 2025, pp. 78–98. doi: 10.1007/978-3-031-76462-2_8.

[22] M. Viceconti, M. De Vos, S. Mellone, and L. Geris, "Position Paper From the Digital Twins in Healthcare to the Virtual Human Twin: A Moon-Shot Project for Digital Health Research," *IEEE Journal of Biomedical and Health Informatics*, vol. 28, no. 1, pp. 491–501, Jan. 2024, doi: 10.1109/JBHI.2023.3323688.

[23] *Foundational Research Gaps and Future Directions for Digital Twins*. Washington, D.C.: National Academies Press, 2024. doi: 10.17226/26894.

[24] S. Gil, P. H. Mikkelsen, C. Gomes, and P. G. Larsen, "Survey on open-source digital twin frameworks–A case study approach," *Software: Practice and Experience*, vol. 54, no. 6, pp. 929–960, Jun. 2024, doi: 10.1002/spe.3305.

[25] S. Infante *et al.*, "Integrating FMI and ML/AI models on the open-source digital twin framework OpenTwins," *Software Practice and Experience*, Mar. 2024, doi: 10.1002/spe.3322.

[26] P. Singh, N. Meratnia, M. Beliatis, and M. Presser, "Navigating the International Data Space To Build Edge-Driven Cross-Domain Dataspace Ecosystem: English," Aug. 2024.

[27] J. Chae, S. Lee, J. Jang, S. Hong, and K.-J. Park, "A Survey and Perspective on Industrial Cyber-Physical Systems (ICPS): From ICPS to AI-Augmented ICPS," *IEEE Transactions on Industrial Cyber-Physical Systems*, vol. 1, pp. 257–272, 2023, doi: 10.1109/TICPS.2023.3323600.

[28] N. Zhang, R. Bahsoon, N. Tziritas, and G. Theodoropoulos, "Knowledge Equivalence in Digital Twins of Intelligent Systems," *ACM Transactions on Modeling and Computer Simulation*, vol. 34, no. 1, pp. 1–37, Jan. 2024, doi: 10.1145/3635306.

[29] Q. Xu, "Traversing the Data Spectrum: Path to Dependable Cyber-Physical Systems through Digital Twins," Doctoral thesis, 2023. Accessed: Apr. 05, 2024. [Online]. Available: https://www.duo.uio.no/handle/10852/106058