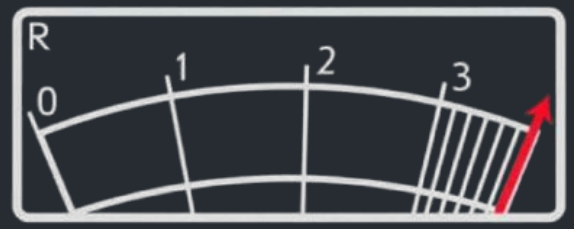
When the Meter Maxes Out

Chernobyl Disaster Lessons for ML Systems in Production

PyData Global 2025

Idan Richman Goshen
Data Science
Team Lead @ Lusha

Images: РИА Новости, HBO



3.6 ROENTGEN
NOT GREAT, NOT TERRIBLE

The truth was thousands of times higher!

People trusted the instrument because it gave a number. any number.

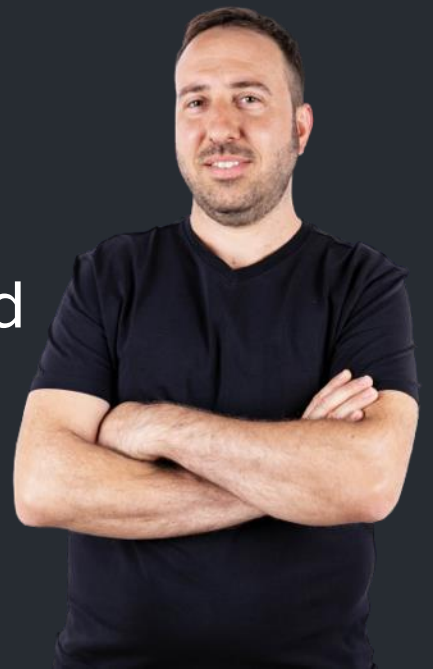
Instrumentation is not ground truth it is a model of the truth.

Image: THEHUTT

About Me

Idan Richman Goshen

- Economist turned Data Scientist (~8 years ago)
- Coding since childhood (drawing polygons in assembly 🤷‍♀️)
- Board member, Eretz-Ir NGO, making cities community-oriented
- Experience across both startups and large enterprises



Timeline of the Disaster

1

1970

Construction of the Chernobyl Nuclear Power Plant and the city of Pripyat begins

2

1977–1983

Reactors 1-4 go online, powering the region

3

April 25, 1986

Preparations for a safety test on Reactor 4 begin

4

April 26, 01:23 AM

Reactor core explosions occur during the test

5

April 27-28

Pripyat's 49,000 residents evacuated (36 hours post-explosion). Soviet Union acknowledged the accident only after Sweden detects radiation

6

Recovery & Containment

1986: "Sarcophagus" completed

2000: Reactor 3 shuts down - final plant closure

2016: New massive steel arch installed to secure the reactor for ~100 years

Before the Explosion: Critical Failures

April 25, 1986

Operational Pressure

Safety Override

Feedback Loop

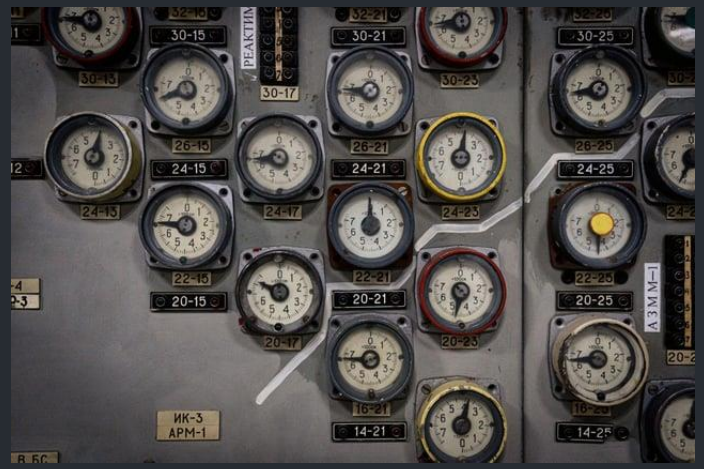
Operational Pressure

The test was delayed because the electrical grid demanded full power during the day, so they pushed ahead at night with a less experienced and tired crew.

A recommender system built in a hurry.

What went wrong?

A bug in the user's table caused all user features to become null, and your model failed silently.



Core lessons

- **Speak up**
- **If you end up compromising, make sure all stakeholders are aware of the risks**
- **Expect the worst - prepare fallbacks**
- **"Don't deploy on Friday" ☺**

Safety Override

The operators disabled the Emergency Core Cooling System, automatic shutdown logic, and several alarms “just for this test”.

Forecasting electricity demand during a severe storm. Demand and supply patterns break, lines start tripping - all those alerts fire continuously.

What went wrong?

The very alarms designed to catch these events were muted because everything looked abnormal already.

Even in expected special cases - like Black Friday or a storm - we can predict **only part** of the abnormal behavior.

Core lesson

Special operating modes need *their own* safety envelope:

- What “expected abnormal” looks like
- What signals remain non-negotiable
- Hard limit conditions for stop/revert

If you must bypass something, **document it, announce it, and create compensating safeguards.**

Feedback Loop

The RBMK’s positive void coefficient made it unstable at low power - the very regime of the turbine test.

A NER Transformer model deployed in "Shadow Mode" with stable P99 latency at 900ms.

What went wrong?

That 900ms latency frustrates users. They hit refresh -> one request becomes two, causing more timeouts and more refreshes. The system enters a positive feedback loop of retry storms

Core lessons

- **Your model isn’t an isolated component**
- **Ponder second-order interactions**
- **Use Canary deployments for a more realistic “Closed Loop” system (expose 1% of real users to the new model).**

The Explosion: The Point of No Return

April 26, 01:23 AM

Failed Fallbacks

False Metric

The Unexpected truth

Image: Igor Kostin / Sygma via Getty

Failed Fallbacks

AZ-5 Button: Operators pressed the emergency shutdown button to stop the instability. But under the test conditions, it acted as a detonator.

A new model is deployed, with the legacy service set as fallback.

What went wrong?

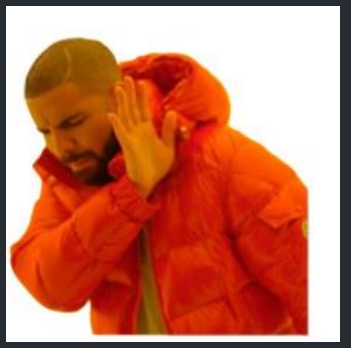
An upstream missing feature kills the new model, dropping 100% of traffic to the fallback. The legacy service - now running on a single micro-instance - is hit with a tsunami of requests, and crashes instantly.

Core lessons

- **Don’t plan only for the happy path**
- **Account for all dependencies your system rely on**
- **Be in sync with all cross-team dependencies**
- **Consider simpler and maintainable fallback solutions**

A Quick Test

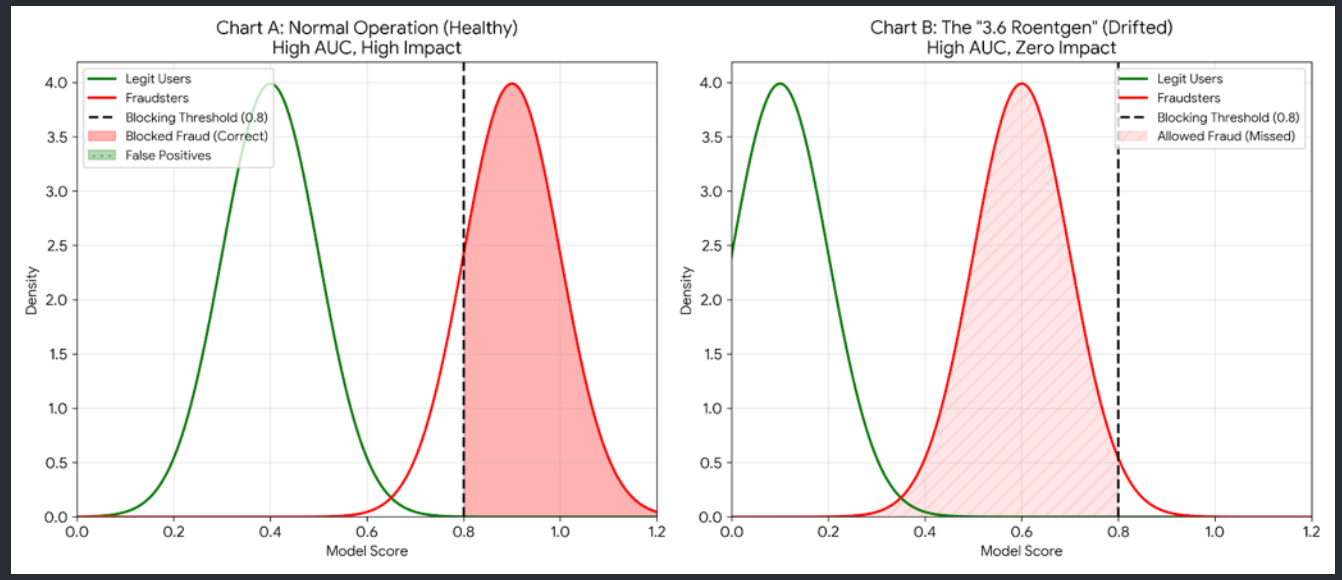
When you see the following, how do you react?

		
A timestamp feature with value of 1970-01-01		✗
An ID field showing -1		✗
A probability predicted as exactly 1.0 or 0.0	✗	

False Metric

The control room dosimeters maxed out at 3.6 R/h, while in reality it was around 15,000 R/h.

Fraud detection model dashboard shows a **stellar AUC around 0.95**. Meanwhile, in reality, you are losing $100k an hour.



What went wrong?

Downstream process used model scores with a threshold of 0.8. Over time model scores drifted down. Fraudsters still got higher scores than legit users, but below the threshold.

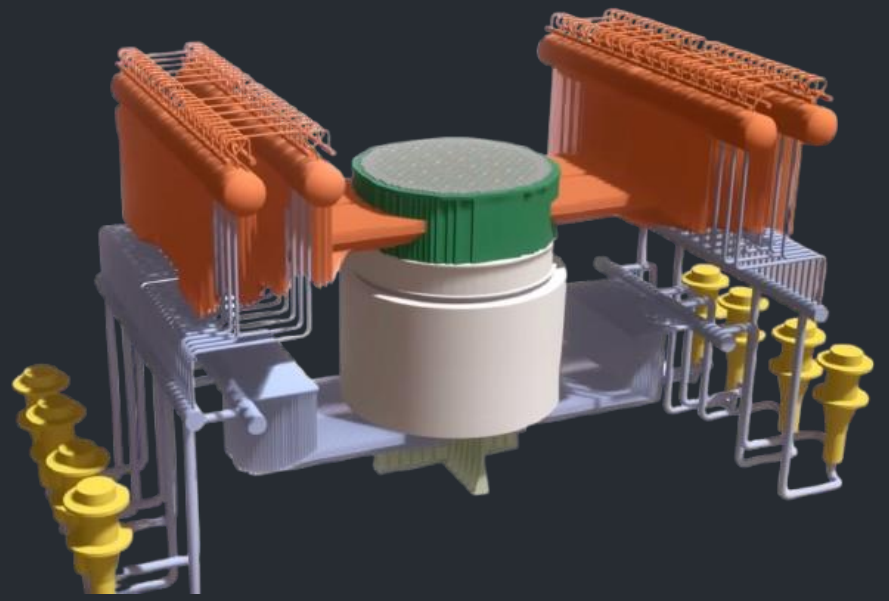
Core lessons

- **A metric is a single-dimensional model of the truth**
- **Track multiple metrics (e.g.: Avg. model score)**
- **Collect and track downstream/business metrics, even if lagged**

The Unexpected truth

RMBK core explosion considered “impossible”… until it wasn’t.
It is an inherent design flaw that wasn’t communicated.

- “The model is stable.” → You won’t test extreme inputs
- “The system is robust.” → You won’t simulate drift
- “It can’t go wrong.” → You won’t build fallbacks



Core lessons

- **Your system can fail in more ways than you expect**
- **Favor business metrics and common sense over theoretical explanations**
- **Act professionally - Communicate your model’s limitations**

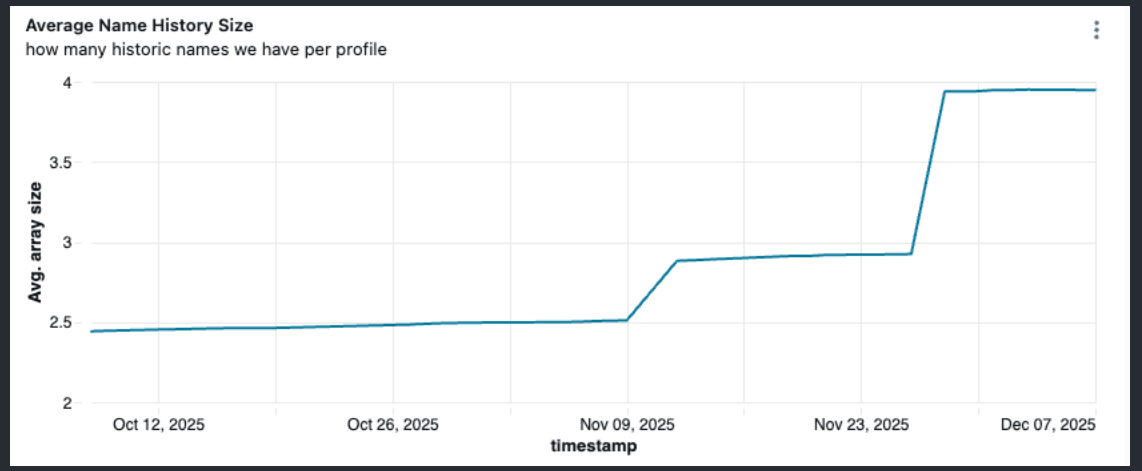
Pripyat’s 49,000 residents' evacuation

36 hours after the explosion

Radiation levels in Pripyat were abnormal almost immediately after the explosion. Symptoms appeared and external cities reported spikes. The authorities waited for conclusive proof before acting.

Big failures rarely begin with dashboards turning red.

By the time certainty arrives - customers complaining, KPIs collapsing - you’ve already expanded the blast radius



Model input distribution slowly shifts

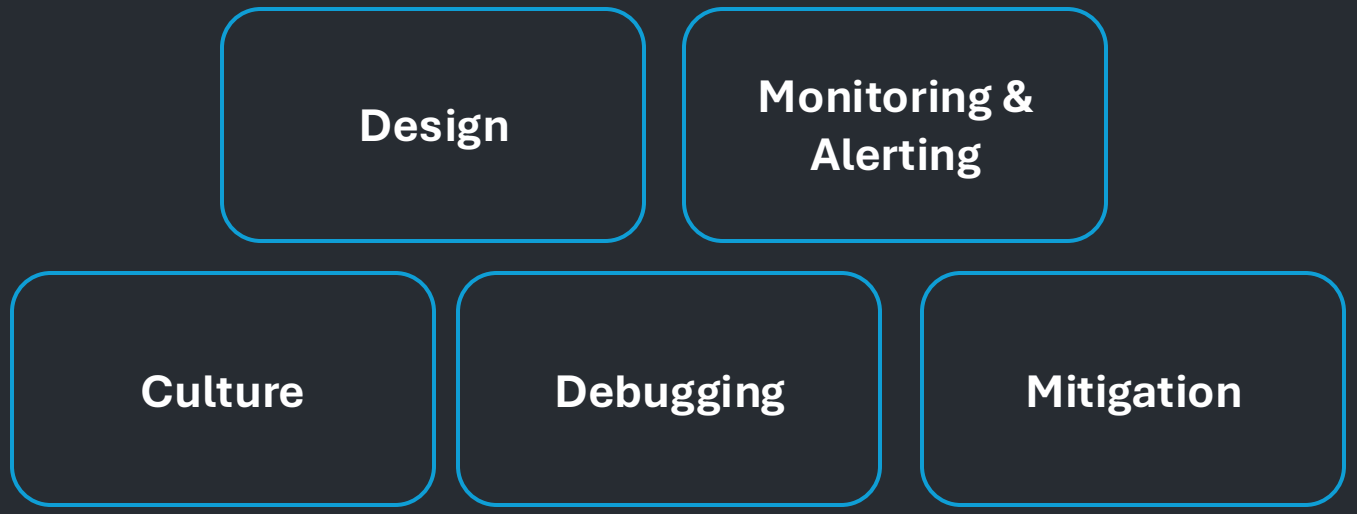
Core lesson

Professionals don’t escalate when they’re certain; they escalate when something looks off

To Summarize

Smart people, doing reasonable things, in a familiar environment - **until** the wrong conditions align and normal rules stop applying.

Often, it’s not that single small mistake, it’s a collection of practices that turns a mistake into a disaster.



Thank You !

How was the talk?