[DRAFT] Practitioner's Guide to Real-time Machine Learning: Principles, Patterns and Lessons Learned

(Work in progress)

rtmlbook.queirozf.com

Felipe Q. B. Almeida

April 2026

`df31c0dd6b40ee92a39763e522d1fc39d92d0f71`

Contents

Hello!

Welcome to the Practitioner's Guide to Real-Time Machine Learning. This book provides practical guidance for building and maintaining real-time machine learning systems in production.

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

About This Book

This book covers the entire lifecycle of real-time ML systems, from initial project planning to steady-state operations and scaling.

Principles: High-level guidelines that apply to the whole end-to-end system

Patterns: things that are self-contained and common enough to merit a short "name". Examples: shadow-mode deployment, pre-mortem. No need for a lot of context in the name

Lessons Learned: are very short sentences with "statements of fact", "orders", "sugestions" or "warnings" for specific, practical cases, in the form of:

- (statement)
  - "SE principles are also important in RT-ML"
- (order)
  - "Do X"
  - "Don't do Y"
  - "Choose features acording to value and effort"
- (suggestion)
  - "Prefer X over Y"
- (warning)
  - "X doesn't usually end well"
  - "Strategy X is not robust and frequently fails"

What this book is

Collection of practical advice observed by RTML practitioners in the industry

What this book is not

Is there a PDF version of the book?

Yes. Every new release triggers a new PDF version. Download the latest PDF here.

1 Real-time Machine Learning: What and Why

TODO bullets

this book contains practical advice on how to use ML models together with real-time systems. in simple terms, this means connecting a previously trained ML model in regular software and performing inference in real-time. see two examples below

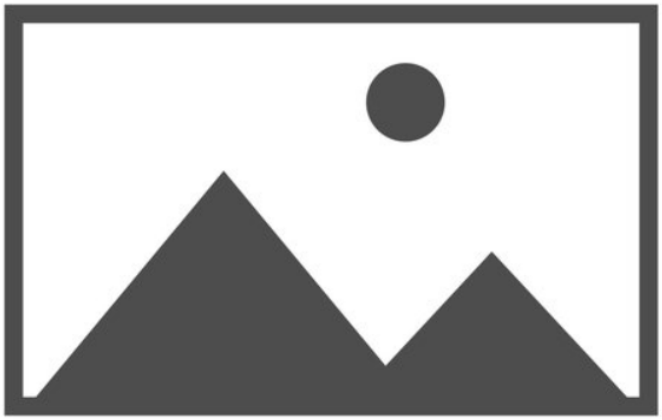

Figure 1.1: example-real-time-credit-underwriting

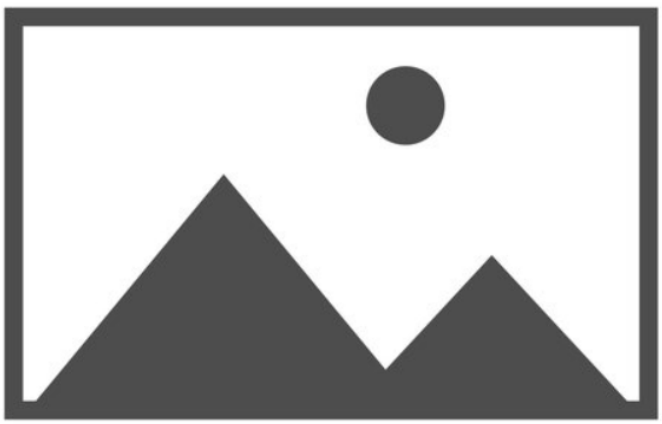

Figure 1.2: example-chatgpt

> **Real-time vs Online Machine Learning**
>
> the corect term is real-time this is why we'll use it. online means something else...

it could well be that in the future, most systems will be RTML systems, but there will always be use-cases that cannot be driven by ML, because they don't support probabilistic decisioning. This is in fact a major reason why RTML projects fail and we'll explain it in more detail in the next chapter

link text in this chapter, we will.....

1.1 Example 1: Fraud detection in Credit Card Transactions

1.2 Example 2: E-commerce "what to show next"

1.3 Traditional vs ML-enabled systems

IMG: a diagram with two parts. on the left, traditional systems (systems with clear deterministic logic). on the right, ml-enabled systems (systems that have ml components in them)

well, that looks just like any other system to me. Isn't this just software engineering? What's so different about ML-enabled RT software?

- ML-enabled systems have different failure modes and they can fail silently
  - a model outputting garbage outputs is arguably **worse** than software that's out of service, because it will keep making bad business decisions.
- ML-enabled systems need a very different monitoring setup from regular software
- The performance of ML-enabled systems decays over time
- ML-enabled systems are **non-deterministic** and data-dependent by construction, so the usual testing strategies don't work
- The skills involved in training and maintaining a RT model are different from those in traditional sofware teams (math, statistics, etc)

TODO figure out how to add a link to deterministic heuristics here and how they are sometimes the signal

understood, but where does RT enter the picture? Aren't all applications of ML like this?. No. see next

1.4 Real-time vs Batch ML

IMG: a continuation/zoom in of the previous diagram, but now the ml-enabled systems are split into two: systems with real-time ML models and systems with batch ML models only

explain the usual RTML flow

Well, isn't *all* production ML like this? How else is it done?

- "ad-hoc" ML operation: input datasets are built and scored when needed and then the output is manually sent to whichever team will use it to make decisions.
- batch production ml: input datasets are automatically built and scored every day/week or month.

Well, why doesn't everyone just use real-time ML then? It looks much better

- it's more expensive and riskier and sometimes the use case doesn't justify the cost (e.g. too few instances being scored…)
- sometimes the use-case is not that well defined (who will use the scores, what the decision layer policy will look like, etc) so a batch model is a good MVP until the use-case is solid enough to justify a RT ML model.
- sometimes it just doesn't make sense because the customer doesn't need a real-time answer. E.g. a decision of whether or not to lend 1 billion dollars to a big company is something that takes weeks and maybe months to decide. It makes little sense to have this be a RT ML flow.
- modeling teams are sometimes far away from "engineering" teams (sometimes in different business units) so it's easier to have a flow where datasets get scored and then "passed on" to the engineering teams from time to time.
  - differences in mindsets, jargon, incentives, etc.
  - modeling teams don't always want to cede "power" to engineering teams.

1.4.1 Summary