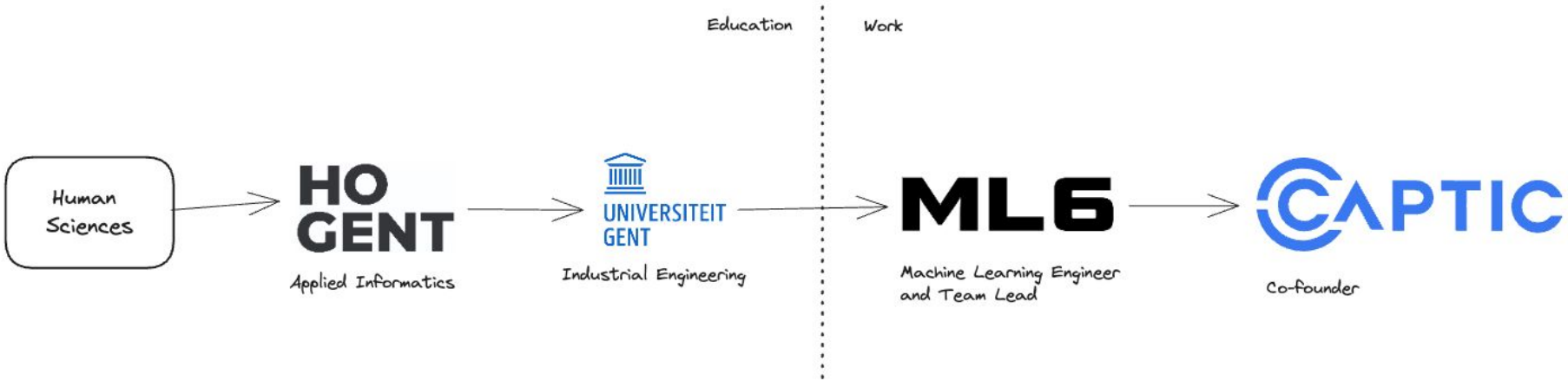
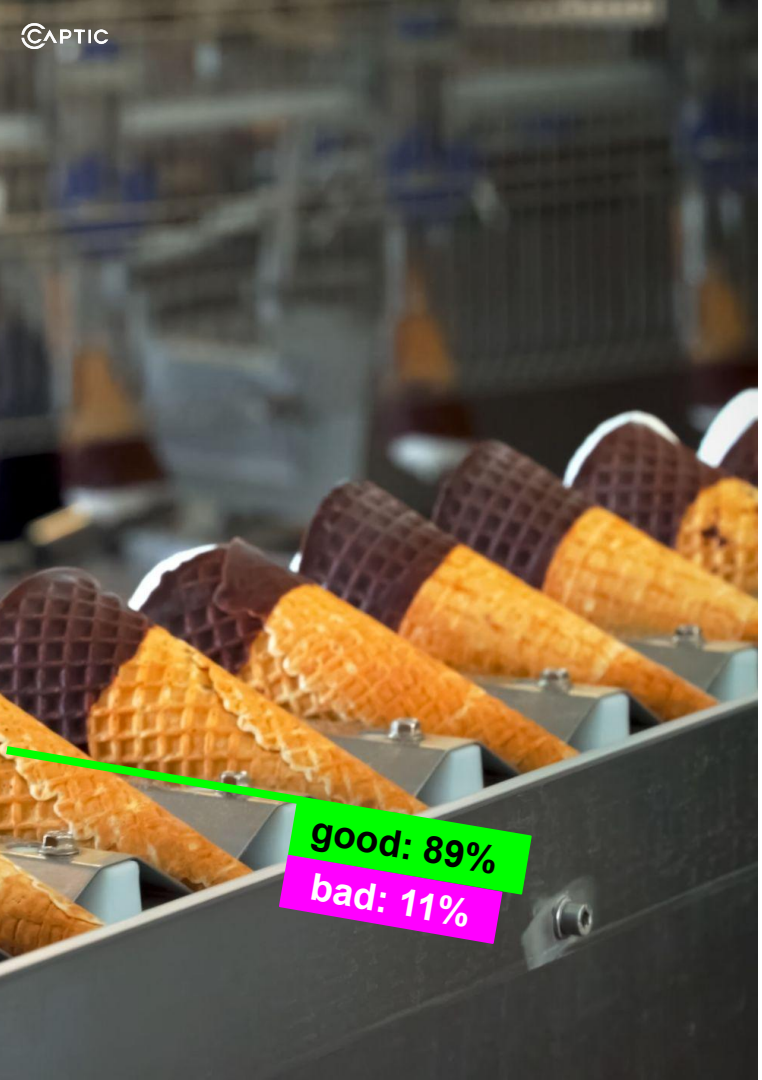
The ML workflow in practice.

Our learnings at Captic distilled for the new generation of ML Engineers

Who's this guy?





About Captic.

We create done-for-you AI-vision systems to digitize and automate food manufacturing

Meet Captic.

Founded in 2022

By Jonathan Kesteloot & Tim De Smet

ML6 Spin-off

ML6 = One of Europe's biggest AI consultancy firms

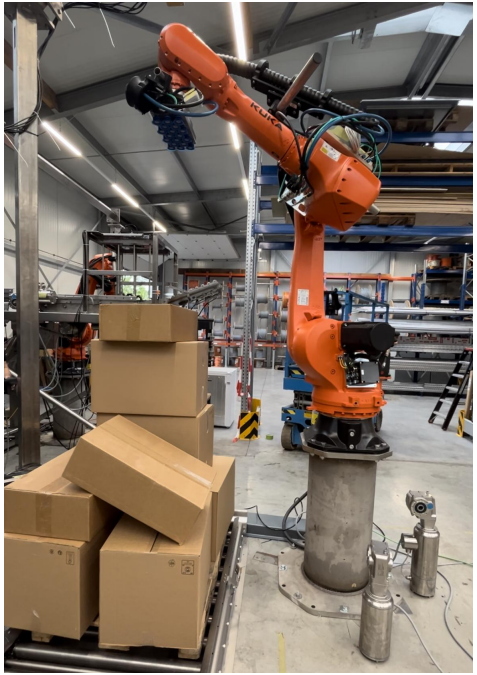
Food focus

We focus on food processing companies, so we speak your language

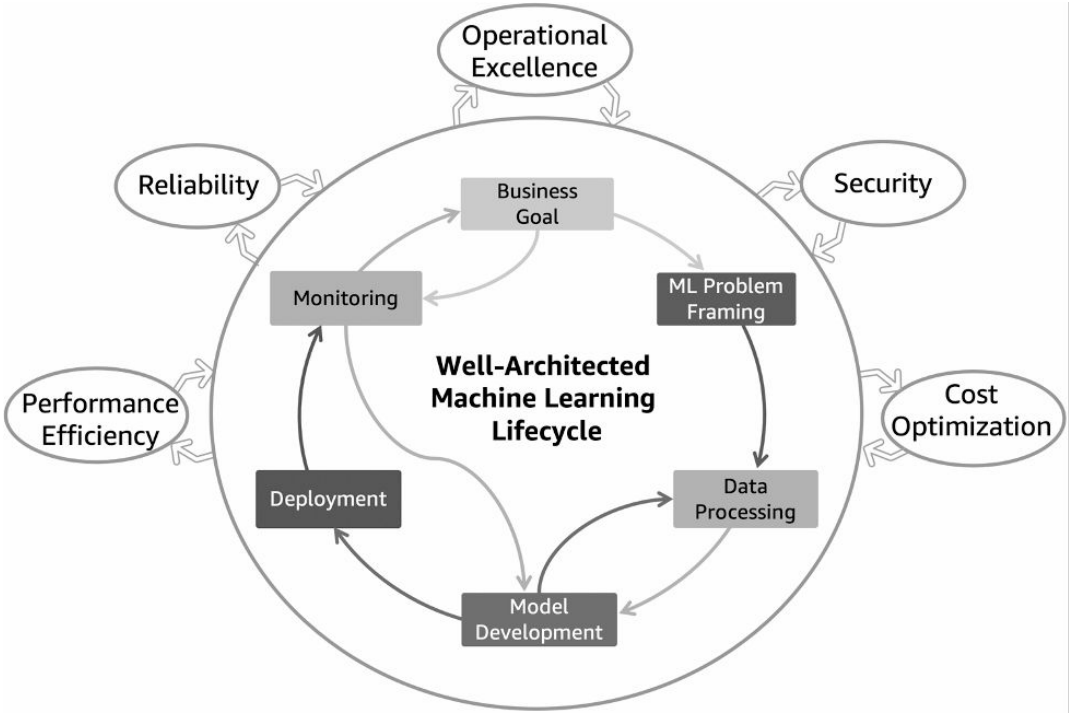
AI-powered

We create done-for-you AI-vision systems to digitize and automate

Some examples:



"Done for you"


Operational Excellence
Reliability
Security
Business Goal
Monitoring
ML Problem Framing
Well-Architected Machine Learning Lifecycle
Performance Efficiency
Cost Optimization
Deployment
Data Processing
Model Development

Our partners:

Microsoft

ADVANTECH

Mondelēz International

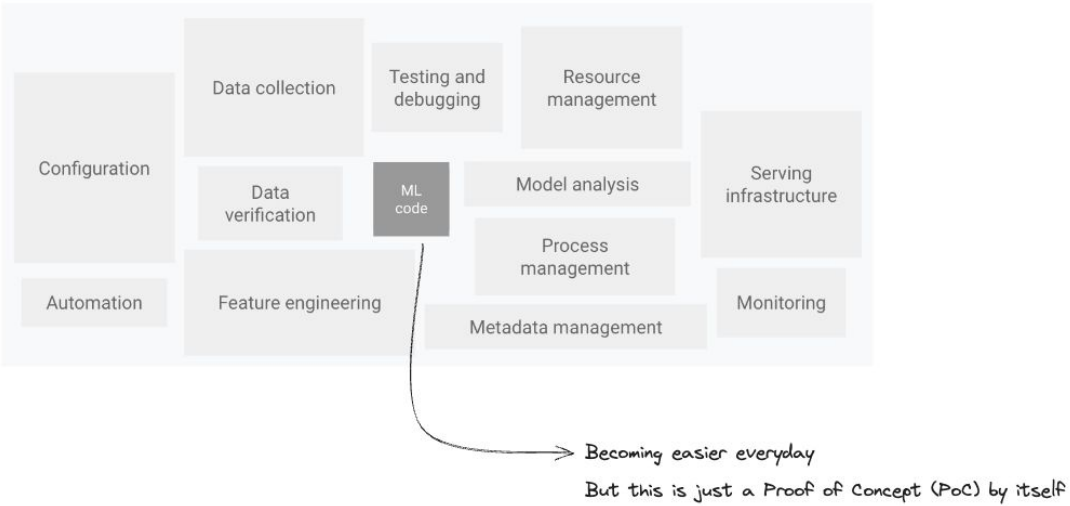
ML6

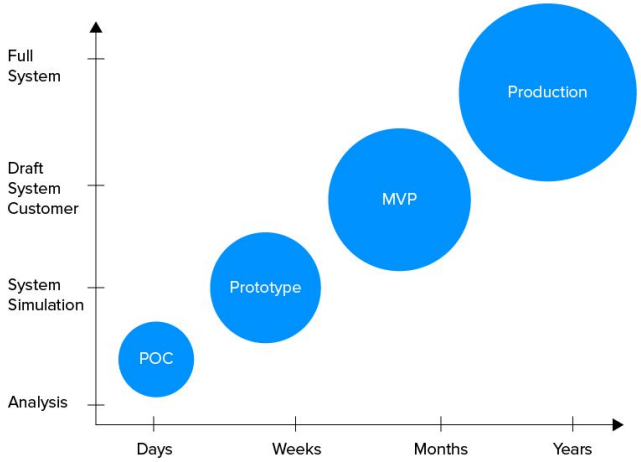
Some of our customers:

Why MLOps?

Why MLOps is a crucial topic for any ML Engineer.

ML by itself isn't enough.





Why they teach you MLOps.

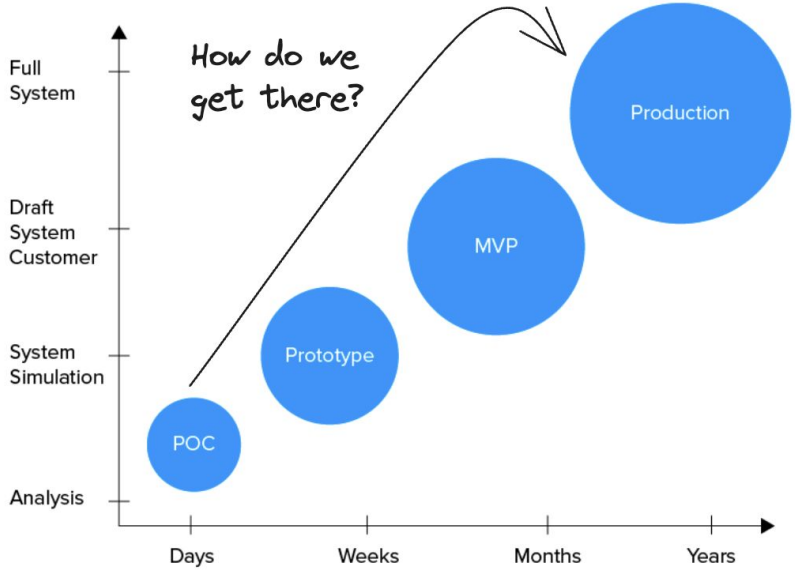
Following a YouTube tutorial and creating a notebook is not hard. But it doesn't solve the business goal.

ML becomes valuable in production, not before. That's when it's no longer a gimmick.

How do we get there?

Standardization and streamlining of ML lifecycle management (= **MLOps**)

→ allows you to **get to** - **and stay in** - **production**.



ML Engineering basics.

Python

Almost everything is done in python.

Git

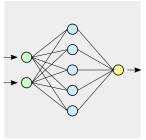
Must have skill in order to work within a team.

Docker

A key tool for any software engineer.

Cloud

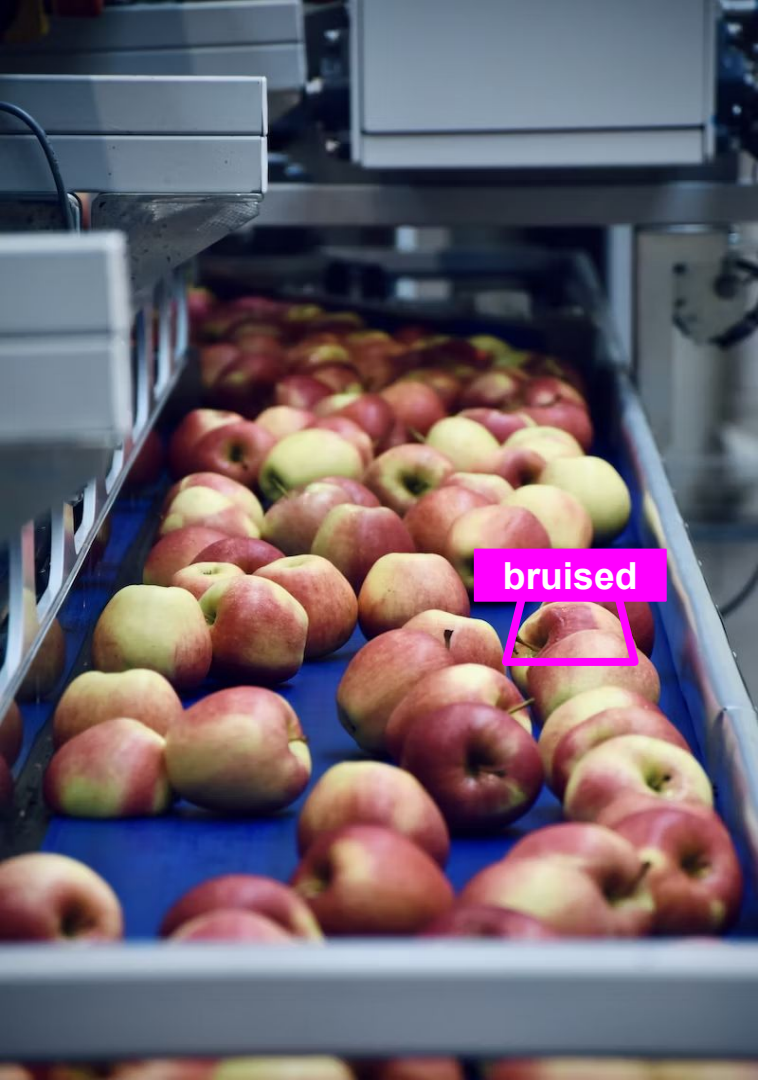
The Cloud gives you huge storage, powerful compute and much more.



ML Knowledge

Understanding of the types of ML tasks, models and frameworks, allow you to problem-solve.

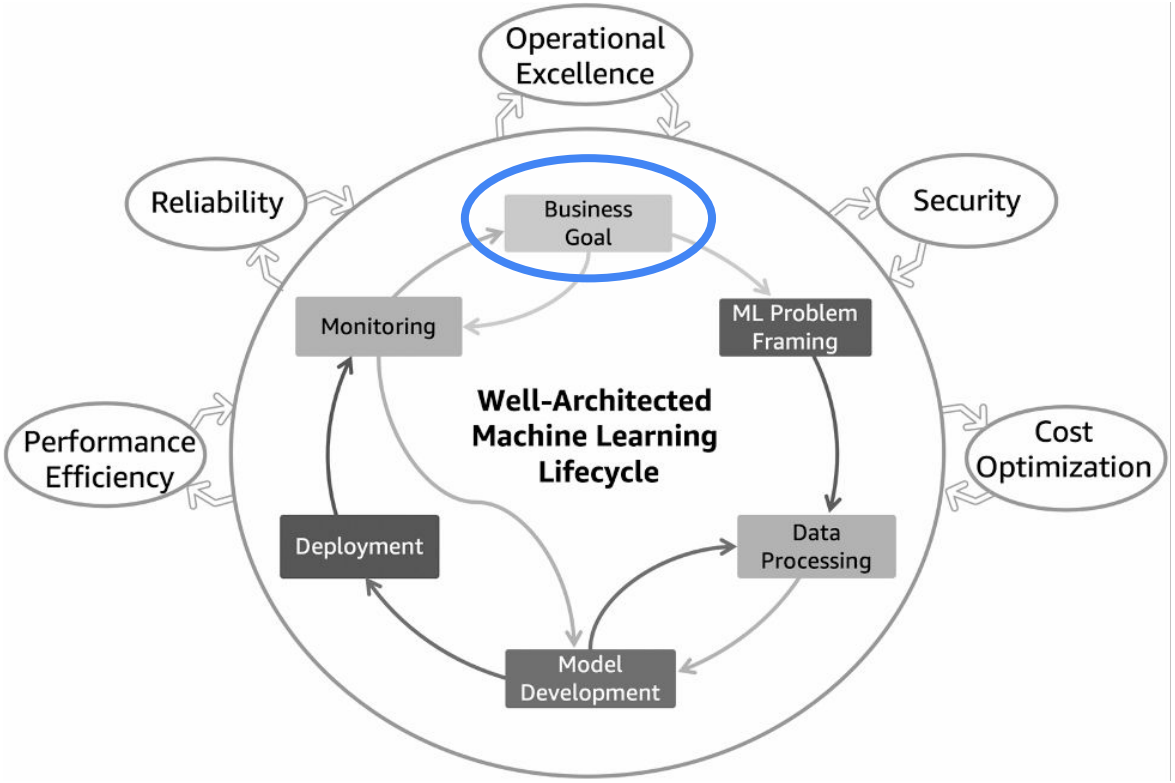
(The more experienced you are, the better your chances)



Real-world ML Workflow.

Let’s solve a (Captic flavored) example use case together.

The ML Lifecycle.



Business Goal.

Tip: Don’t engineer for the sake of engineering

Questions you need to ask yourself:

1. What real problem are we trying to solve? (What is the goal?)
   - Staffing issues?
   - Safety issues?
   - Quality issues?
   - Throughput issues?
   - Waste issues?
   - …

2. Is ML the best way to solve the problem?
   - No?
   - Yes?

Business Goal: A pitfall.

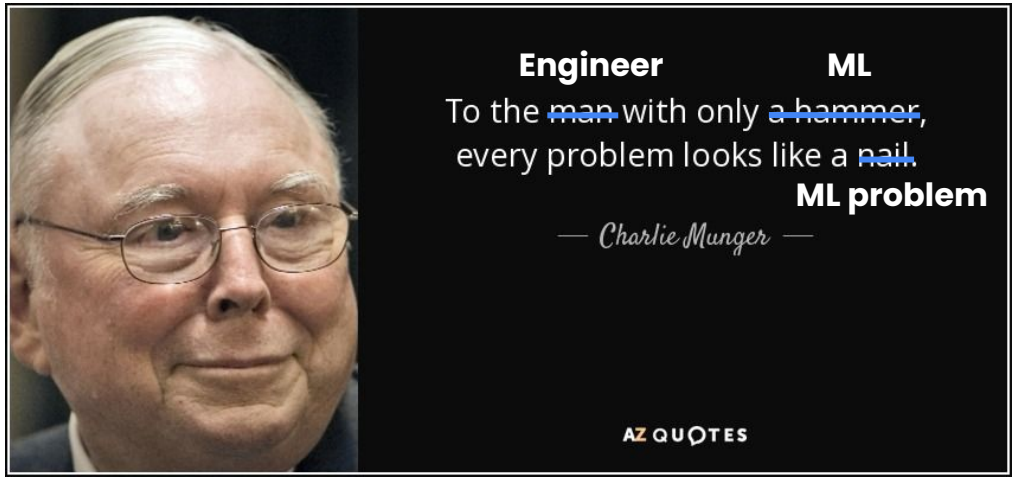
Business is always looking for the:

- Biggest ROI
- Most likely ROI

ML solutions require a lot of:

- Knowledge
- Skill
- Ongoing effort

→ Driving up the cost and risk of failure



Tip: If the problem can be solved without ML, then don't use it just because it seems cool

Business Goal: Example (1).

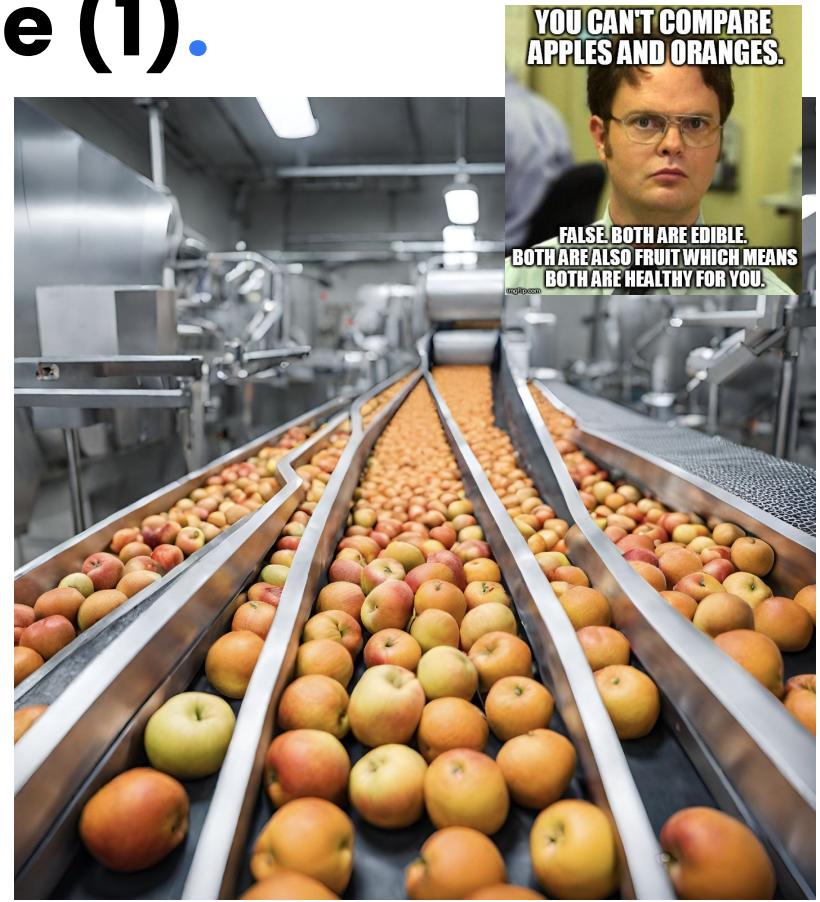
Company that compares apples to oranges

Problem:
- Labor cost keeps increasing
- Finding talent is hard

Result:
- Shrinking margins

Opportunity:
→ Automation is needed

✅ Question 1 (goal) answered!



Business Goal: Example (2).

Question 2 (Use ML)?

Some kind of sensor is needed.

→ Weigher? Too similar
→ Camera? Makes sense

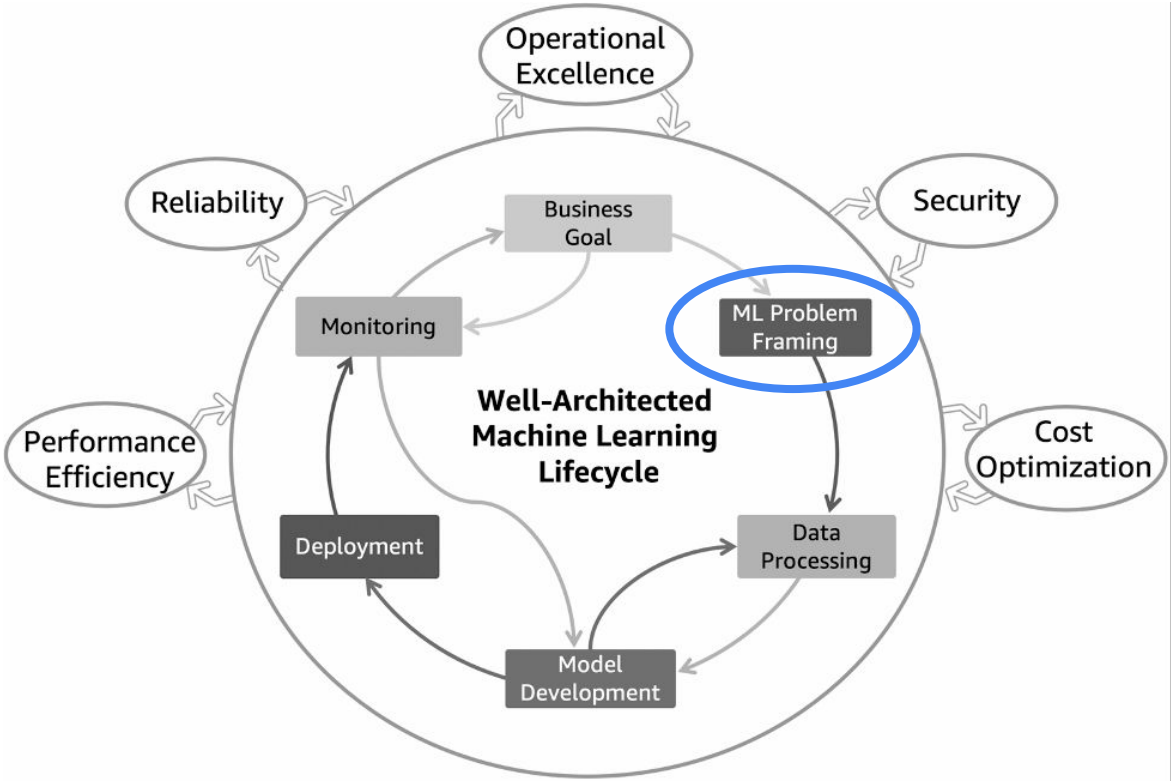
Can we use traditional methods?

- Color? Could work but won't handle variety well
- Shape? Too similar

We've exhausted all other options
→ We'll use ML to solve this problem

= Automation through AI-Vision

The ML Lifecycle.



ML Problem Framing.

Our case: Automation through AI-Vision

Questions to ask yourself:

1. What do we want to predict?
2. Do we have performance expectations?

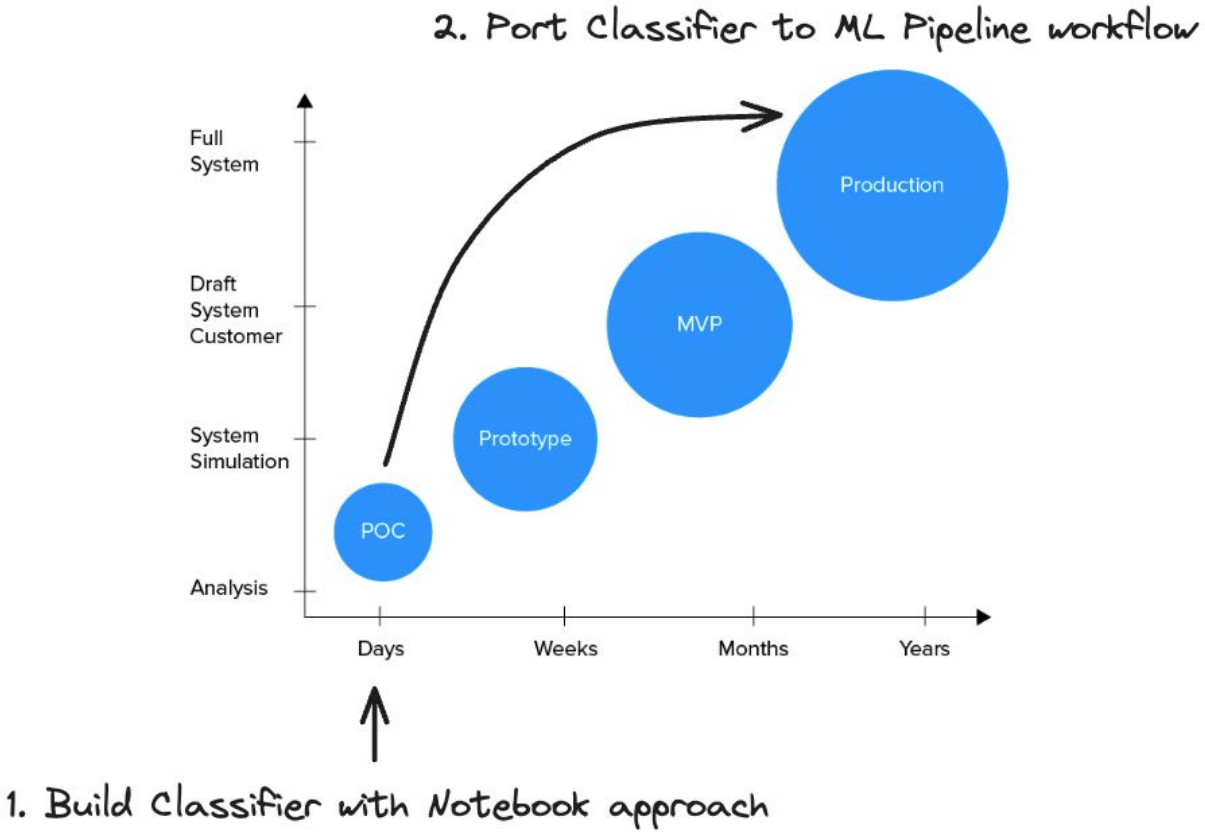
Answers:

1. We need to be able to distinguish apples from oranges
   → Image classification

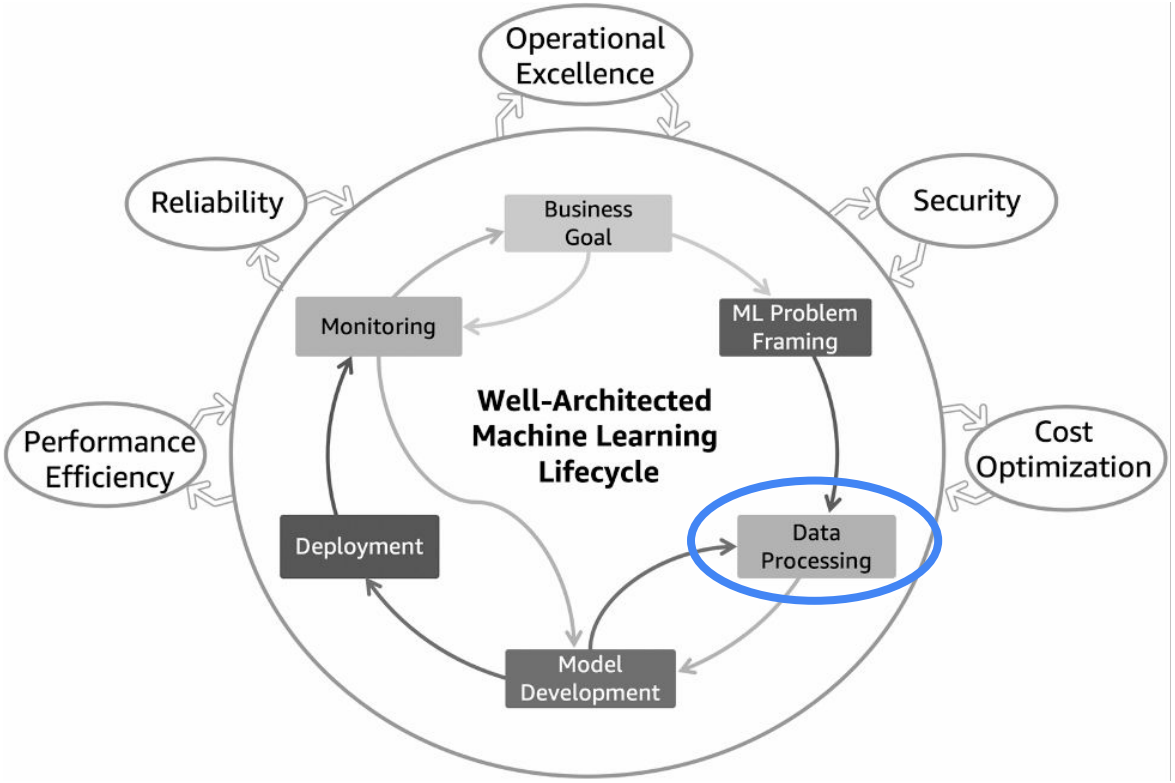
2. Requirements
   a. We will need the system to work real-time → Deployed on the edge
   b. Both are equally important (mistaking an apple for an orange is as bad as vice versa)

The Lab.

During the lab, you will:



The ML Lifecycle.

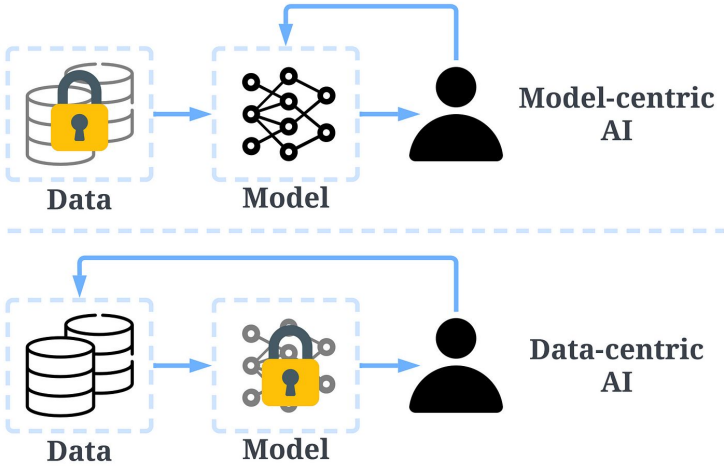


Data processing.

Questions to ask yourself:

- What data do we need?
- How much data do we need?
- Do we need particular examples that are hard to come by?
- How will we collect the data?
- Where do we store the data?
- How will we label it?
- In what format do we store it?
- Do we need to modify the data?
- How long will we keep the data?
- ...

Data is hard work:



Data processing: Example.

What data do we need?
→ Images of apples and oranges in their real setting under different circumstances

How much data do we need?
→ Hard to know for sure, our classifier will probably only need a few hundred

Do we need particular examples that are hard to come by?
→ We need to take into account edge cases (e.g. rot → we can stage those if needed)

How will we collect the data?
→ Bring a camera to the processing plant

Where do we store the data?
→ We'll go with an SD card for now, but the Cloud is the obvious choice (blob storage)

How will we label it?
→ There are many tools out there. We use Azure's one, but Label Studio is a great open-source one

The Lab.

During the lab, we work with a toy dataset:

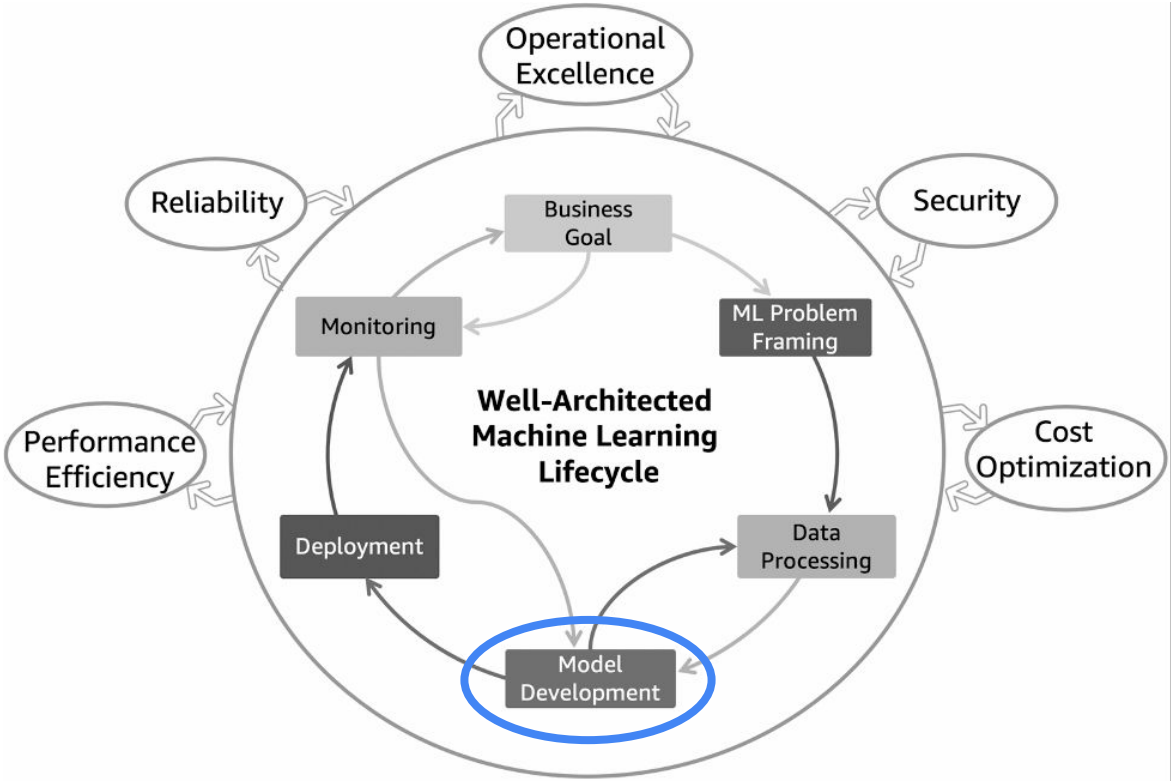
→ We download 10 images of each class in a script

Note:

- This is not nearly enough (but it allows us to demonstrate the notebook and pipeline approach)
  - Our model will drastically overfit on the data
- This is not how you'd want to do it (fast and dirty workaround to avoid complex infrastructure)

Challenge: Can you get your hands on more data?

The ML Lifecycle.



Model selection.

There are many different approaches you can take:

1. Build from scratch → Not done often

2. Give AutoML a go
   - Tends to be worse than SOTA-models, but perhaps good enough
   - Always good to have a baseline
   - Cloud costs

3. Use existing and proven architecture
   - Find them on Papers With Code
   - Use them
     - GitHub
     - HuggingFace
     - Keras
     - ...

Model training.

Tip: Training on specialized hardware makes it a lot faster. Try to avoid using your own laptop. Google Colab gives you free GPU usage.

It's crucial that training is reproducible, this requires you track everything

- What dataset
- What augmentations
- What model
- What parameters
- …

You can use a tool like MLflow for this.

Additional "Hacks":

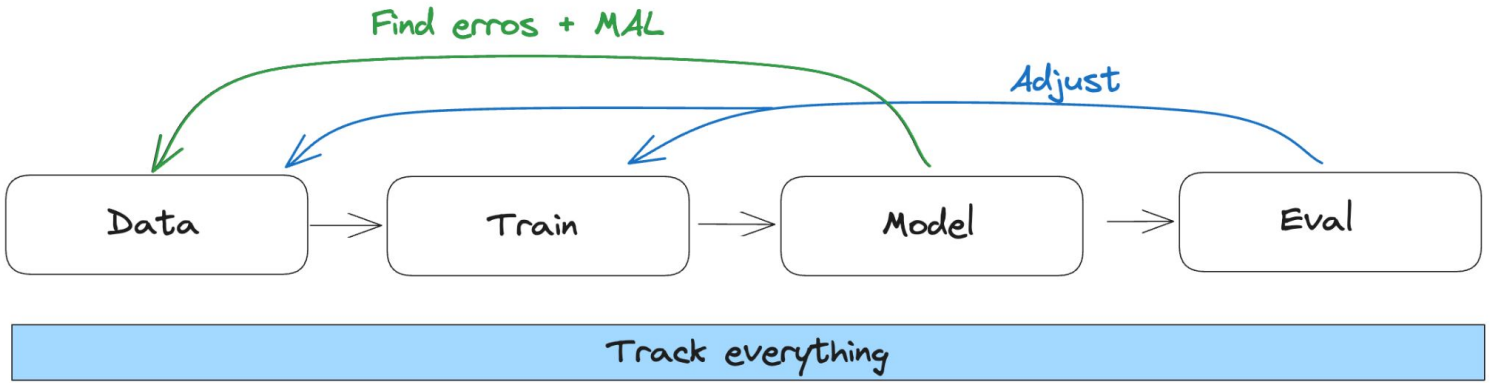
- **Transfer learning =** start from model weights for similar task → Faster (and better)
- **Hyperparameter tuning** = try different parameters for different runs to see which are best
- **Data augmentation =** generate more data by changing the data itself

Model evaluation.

Evaluating the model properly is key. Make sure your test dataset is high-quality and balanced.

This step should also be tracked for reproducibility.

Model development is never done → Iterative process



ML Pipelines.

= Automated sequence of steps to build, train, evaluate and deploy machine learning models. Used to streamline the end-to-end process.

Why?

- Help organize and automate
- Keeps track of everything → reproducibility
- Can scale as needed in the Cloud
- Splitting into steps allows for collaboration
- Enforces systematic approach

Many flavors:

- Kubeflow
- TFX
- Azure
- …

The Lab.

Model selection

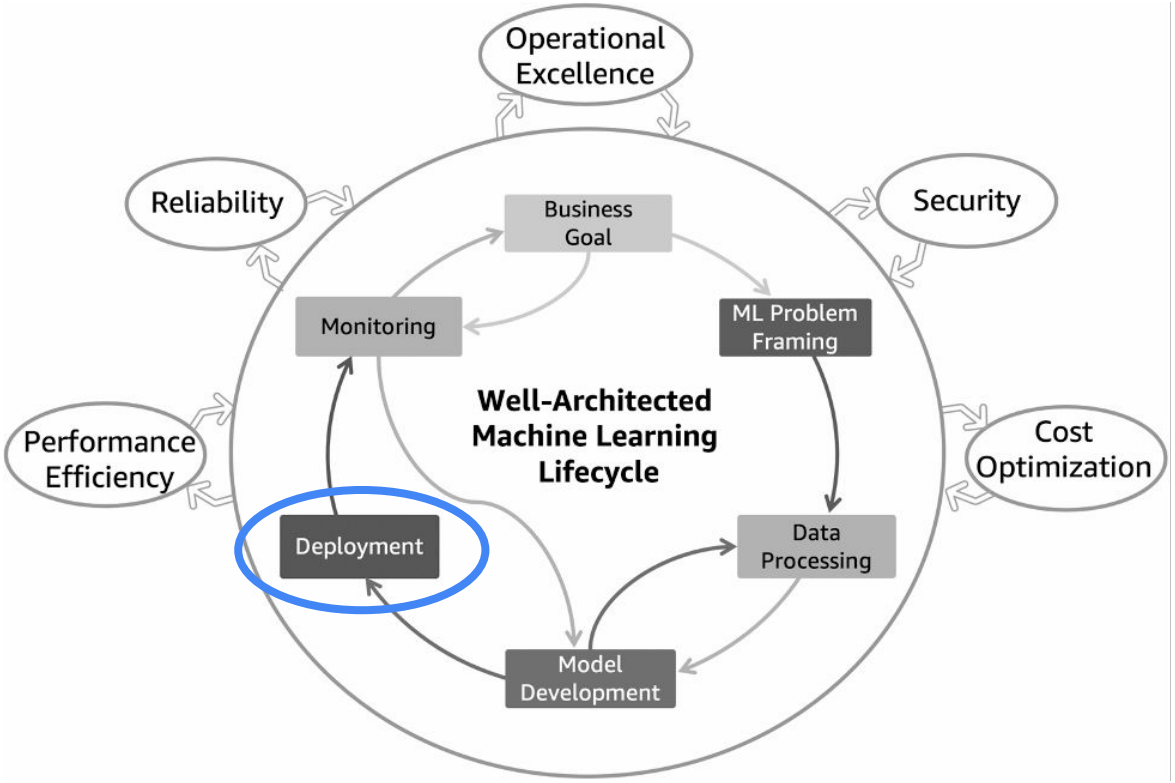
→ Build from scratch using Keras

Model training

→ Use optimized hardware
→ Reproducible (and automated) ML Pipeline with tracking in Azure

Model evaluation
→ Reproducible (and automated) ML Pipeline with tracking in Azure

The ML Lifecycle.



Deployment.

There are many ways to deploy an ML model:

- Real-time serving
- Serverless
- Batch processing
- Edge deployments

Questions to ask yourself:

- How do I want to work with my model?
- How fast should I get an answer?
- What's my budget?
- How do we automate the release process? (CI/CD)
- What and how do we monitor?

Deployment: Example.

We need our Classifier to work real-time. Real-time serving?
→ No, we can't let our production rely on whether we have internet connection.

We need an **Edge Deployment**

There are many options for the hardware:
- Coral
- Raspberry Pi
- Nvidia Jetson
- ...

What do they have in common?
They're tiny, so our model will have to be as well

Model implications.

Since we want our model to run on Edge, we need to make it tiny and fast

→ Best to start with a model that is already pretty small

Many techniques for model compression:
- Pruning = removing unimportant weights
- Quantization = reducing precision of weights
- Knowledge distillation = training a smaller student model that learns from the bigger teacher

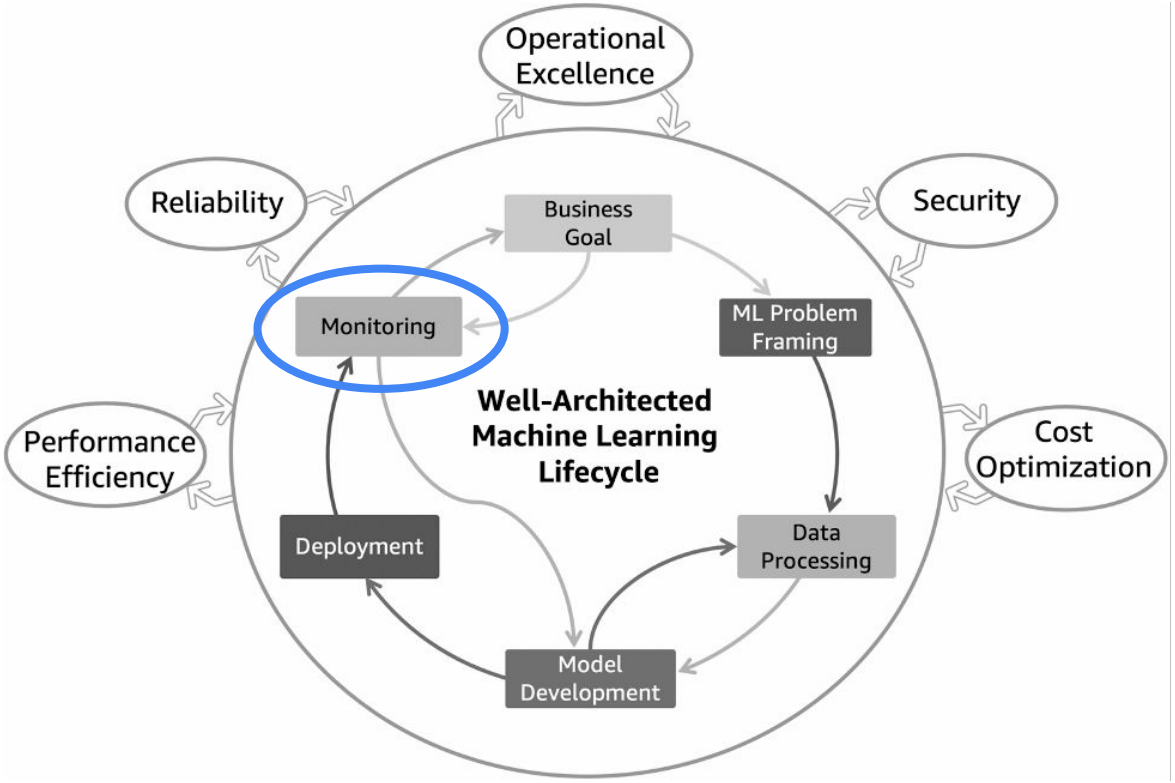
There are tools that do this for you:
- In TensorFlow
- TensorFlow Lite
- In ONNX
- …

The Lab.

Since we don’t have hardware at our disposal, we won't cover edge deployments.

Instead, the model will be deployed in the cloud using Azure facilities.

The ML Lifecycle.

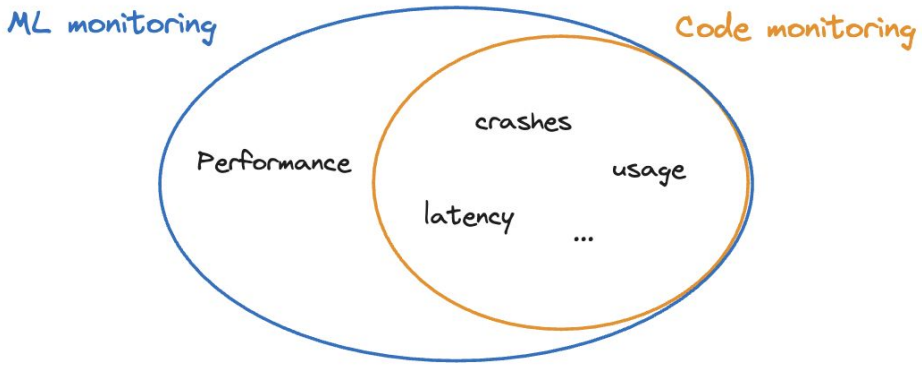


Monitoring.

Why?

- We want to know how our model is doing
- Every model will get bad over time. How fast depends on the use case

ML monitoring is different from normal code monitoring
→ Bad doesn't mean crash (latency, up, …)



Model Decay.

No model lives forever, but the speed of decay varies.

Usual culprits:
- Data drift = changing of input data
- Concept drift = relationship between input and output has changed

Example of Data drift:
Our lens is dirty so our images look different

Example of Concept drift:
The Quality manager's interpretation of a certain class changes over time

→ Monitoring allows us to spot model decay and to retrain (back to model development)

Monitoring.

What can be monitored:

- User feedback collection
  - Instances themselves
  - How many over time
- Output
  - Confidence scores
  - Distributions as expected?
    - Dataset
    - Previous models
- Annotated data

You'll also want to monitor whether you're actually solving the business problem.

(You'll learn more about monitoring in upcoming lessons)

Want to learn more?

- Follow us on LinkedIn
- Apply for an internship