

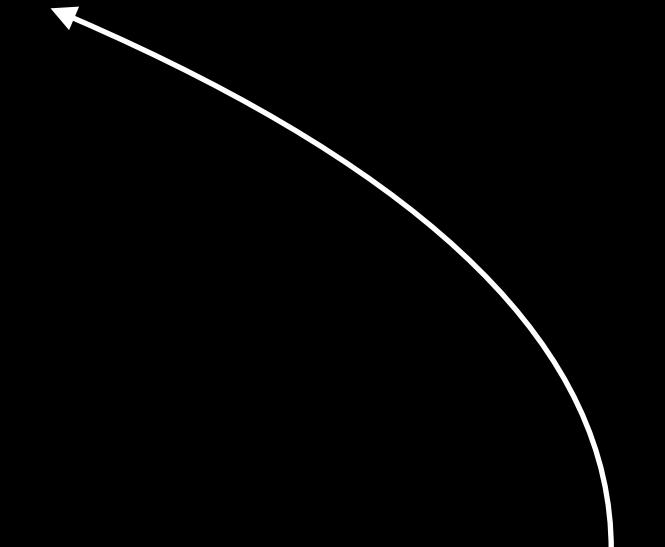
# Machine Learning Roadmap 2020

(a machine-learning-flavoured-visual-iterative-  
living-mind-map/compass)

# “What is machine learning?”

**The curious internet-dwelling user (maybe you)**

Machine learning is turning things (data) into numbers and **finding patterns** in those numbers.



The computer does this part.  
How?  
Math.  
(we'll cover a little on this later)

# Traditional programming (software 1.0)

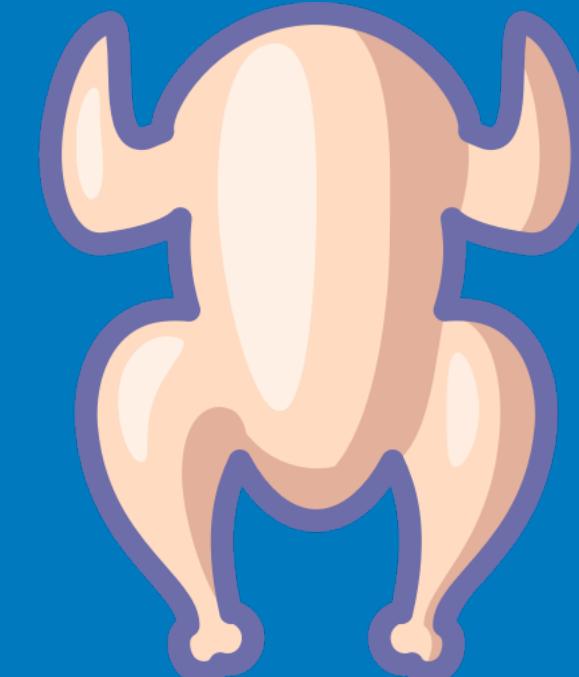
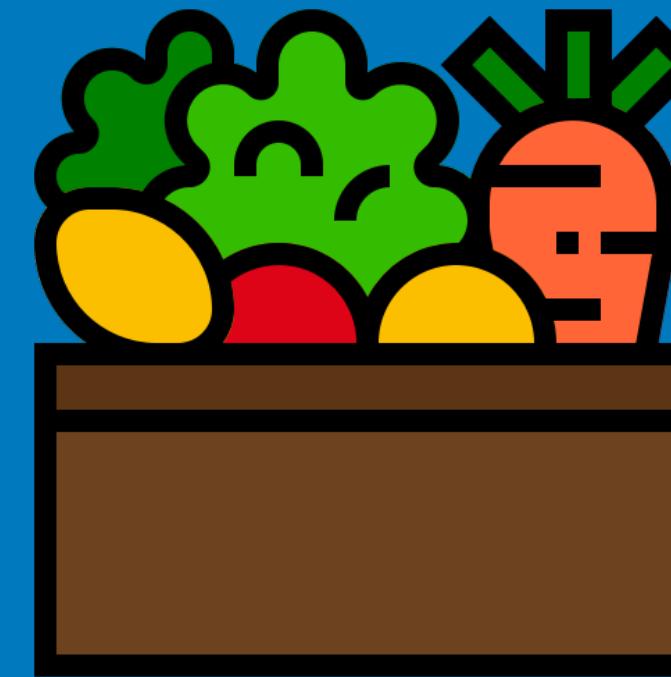
(does not require 2.0)

vs.

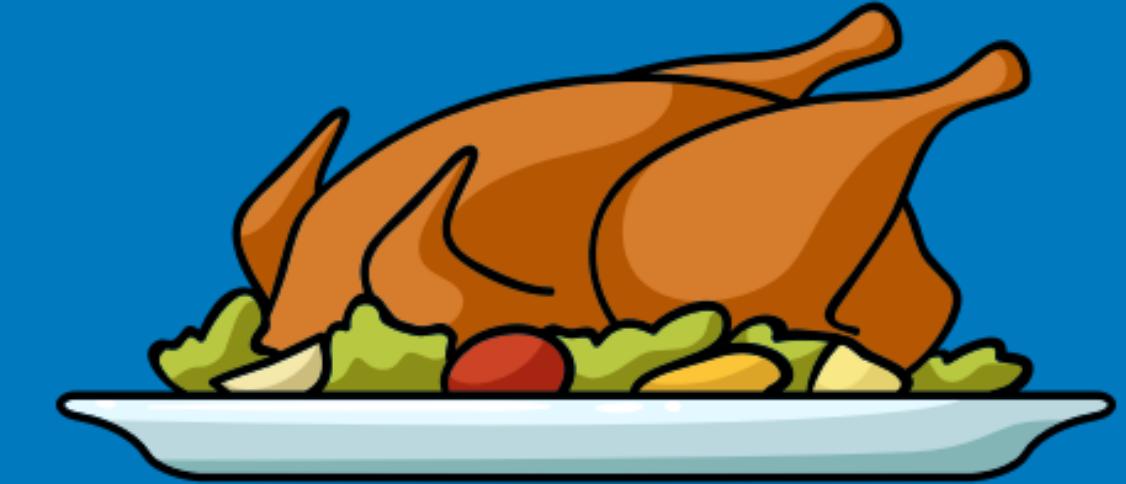
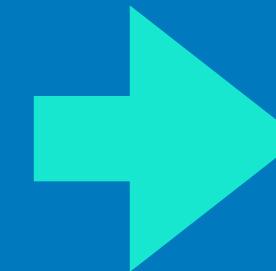
(requires 1.0)

# Machine learning (software 2.0)

## Traditional programming



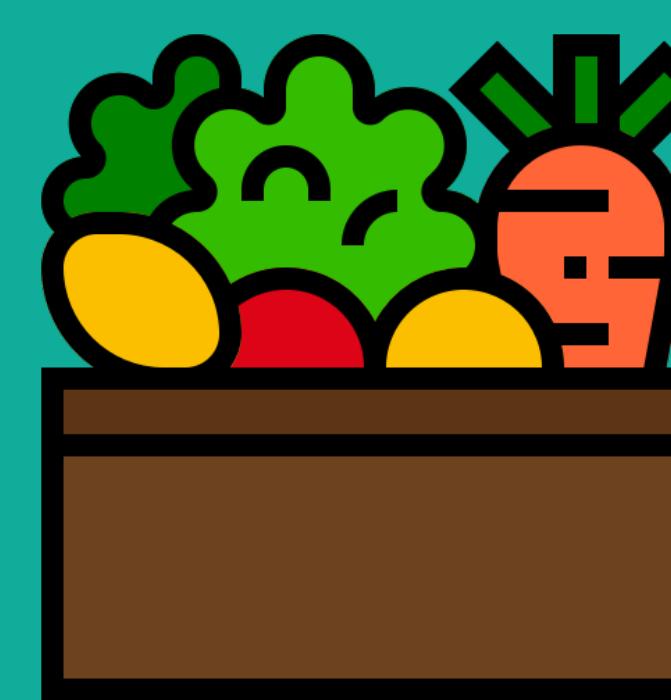
1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables



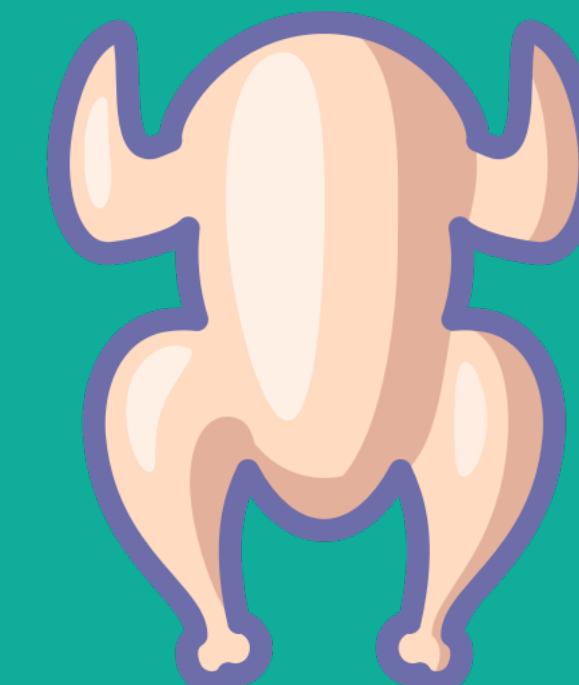
Starts with

Makes

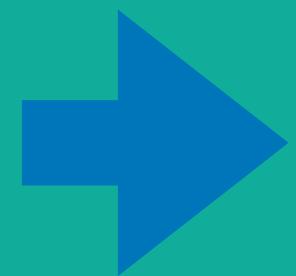
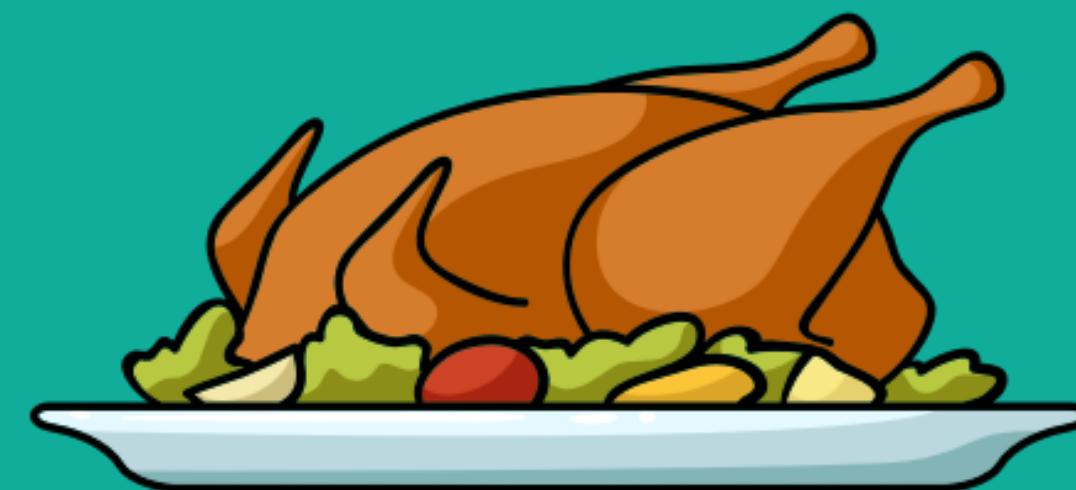
## Machine learning algorithm



Inputs



Output



1. Cut vegetables
2. Season chicken
3. Preheat oven
4. Cook chicken for 30-minutes
5. Add vegetables

Starts with

Figures out

# **“Why use machine learning?”**

**Another curious (perhaps even more curious than before) internet dweller**

Good reason: ~~Why not?~~

Better reason: Can you think of  
all the rules?

(probably not)

(maybe not very simple...)

“If you can build a **simple rule-based** system that doesn’t require machine learning, do that.”

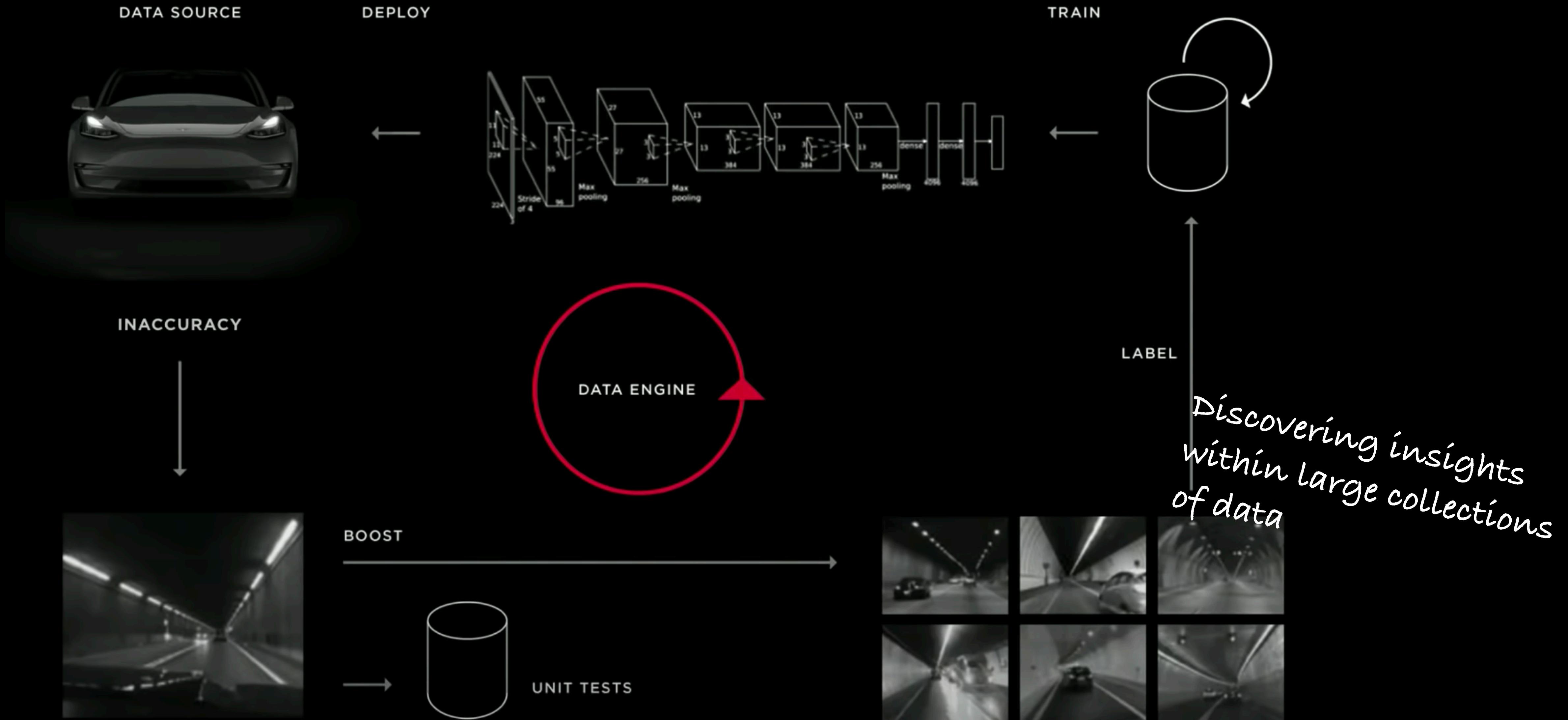
A wise software engineer... (actually rule 1 of Google’s Machine Learning Handbook)

# What machine learning is good for



- **Problems with long lists of rules**—when the traditional approach fails, machine learning may help.
- **Continually changing environments**—machine learning can adapt ('learn') to new scenarios.
- **Discovering insights within large collections of data**—can you imagine trying to go through every transaction your (large) company has ever had by hand?

Problems with long lists of rules



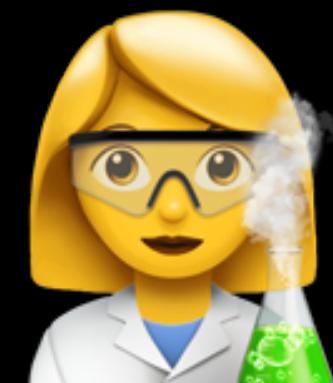
Continually changing environments

Source: [Tesla Autonomy Day video](#)

# What we're going to cover (broadly)

1. 🤔 **Machine Learning Problems**—what does a machine learning problem look like?
2. 🌱 **Machine Learning Process**—once you've found a problem, what steps might you take to solve it?
3. 🔧 **Machine Learning Tools**—what should you use to build your solution?
4. 📏 **Machine Learning Mathematics**—what exactly is happening under the hood?
5. 📚 **Machines Learning Resources**—okay, this is cool, how can I learn all of this?

How:



# How to approach this roadmap



(actually a compass)

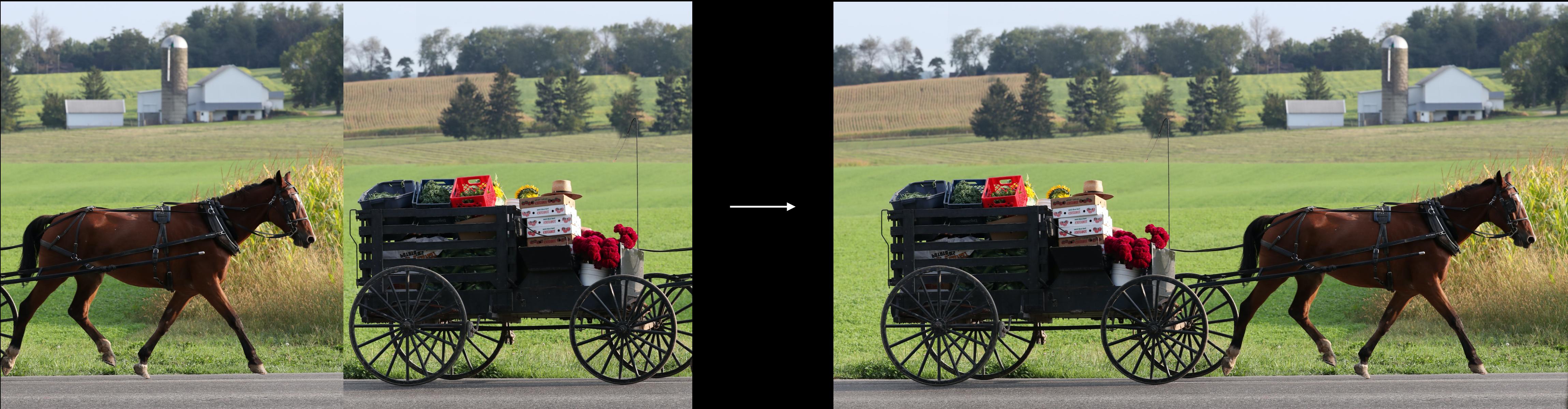
- Explore
- Comment  
*(what's missing?)*
- Give feedback, give advice
- Share
- Follow your curiosity *(there's a lot, so don't expect to get it all)*
- Explore  
*(twice on purpose)*

# Ready?

Okay, let's go.

# 1. 🤔 Machine Learning Problems

# 1. 🤔 Machine Learning Problems



# 1. 🤔 Machine Learning Problems

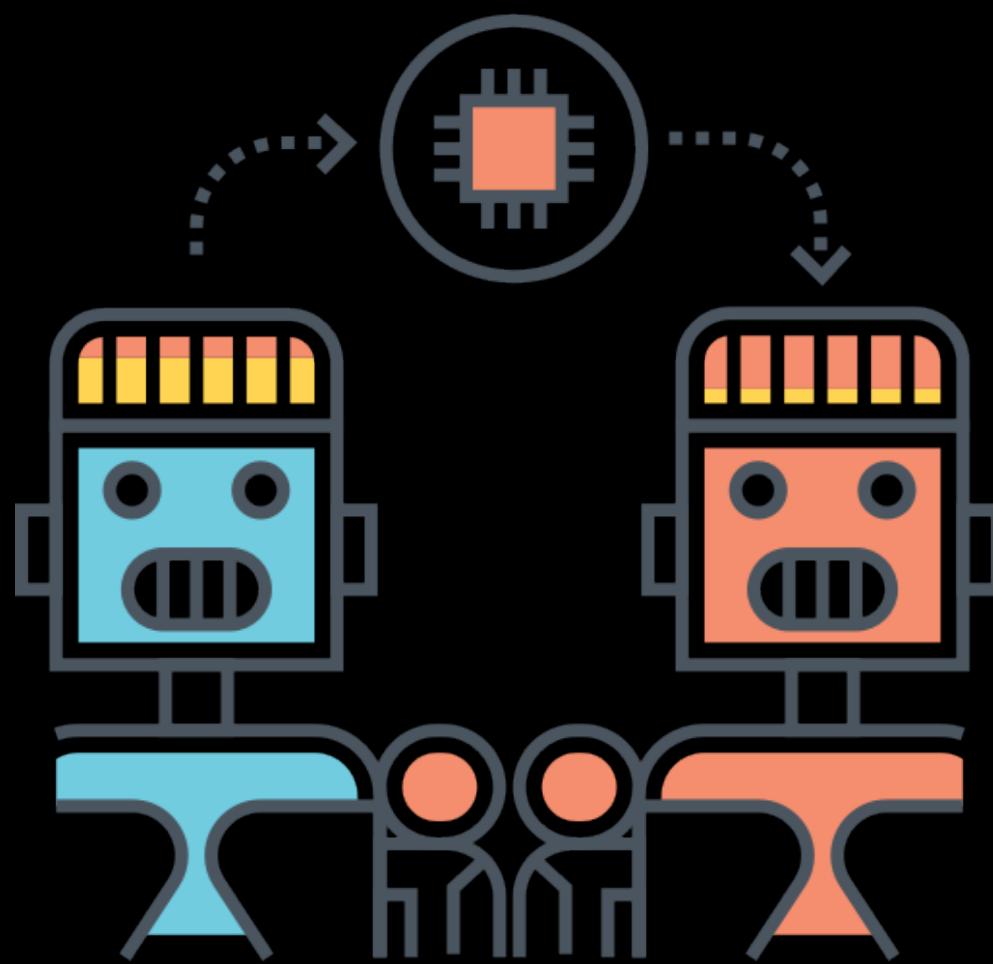
(categories of learning)



Supervised  
Learning



Unsupervised  
Learning



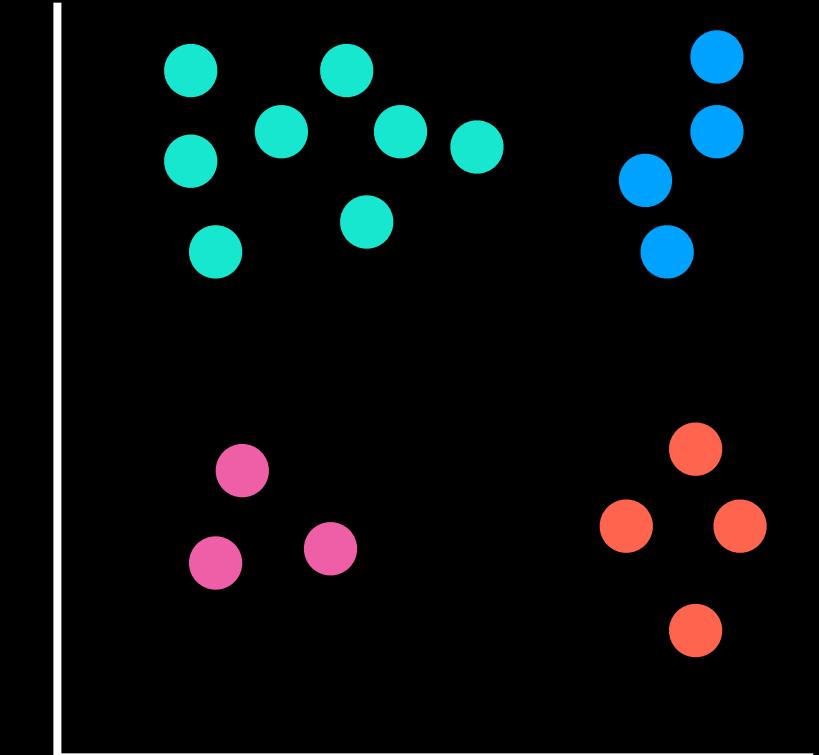
Transfer  
Learning



Reinforcement  
Learning

# 1. 🤔 Machine Learning Problems

(problem domains)



Classification

Regression

Clustering

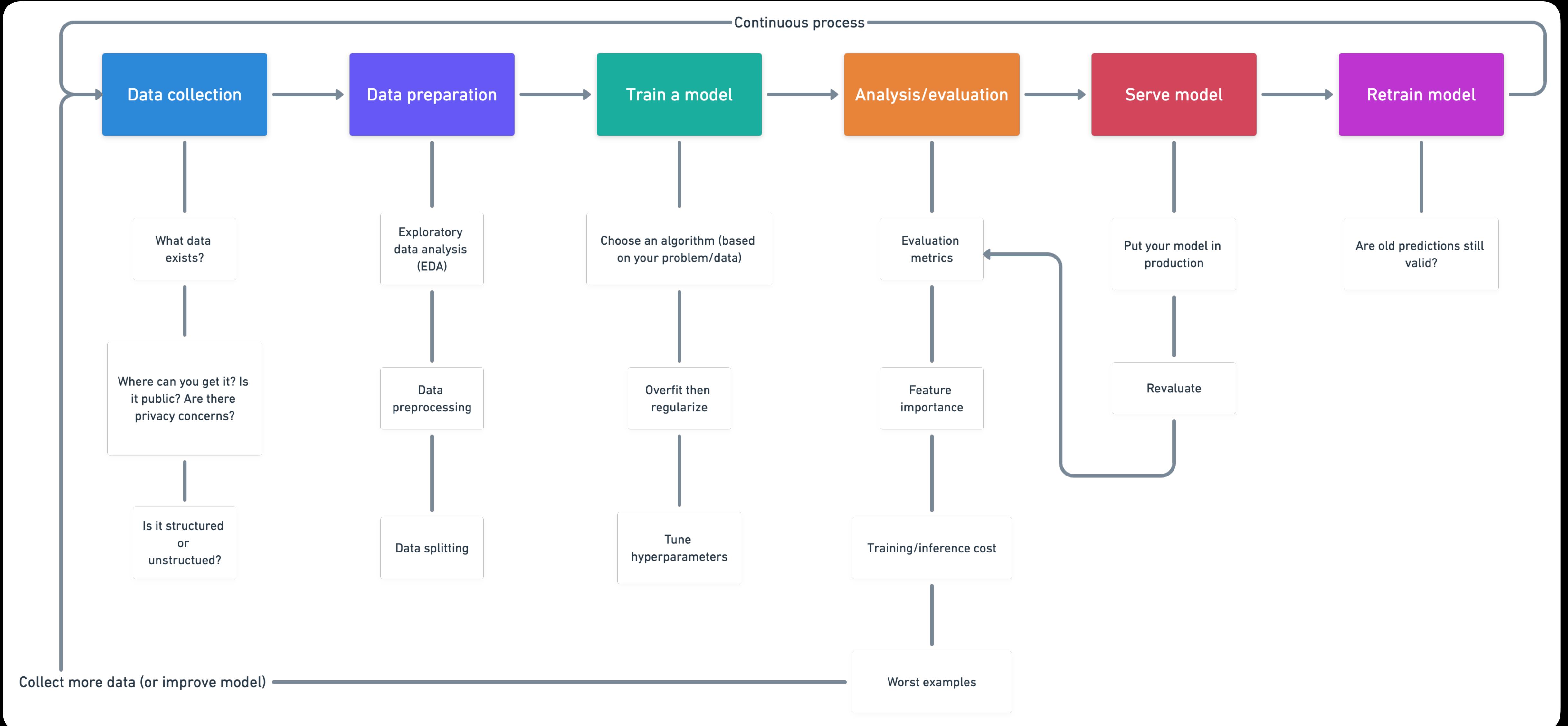
Dimensionality reduction

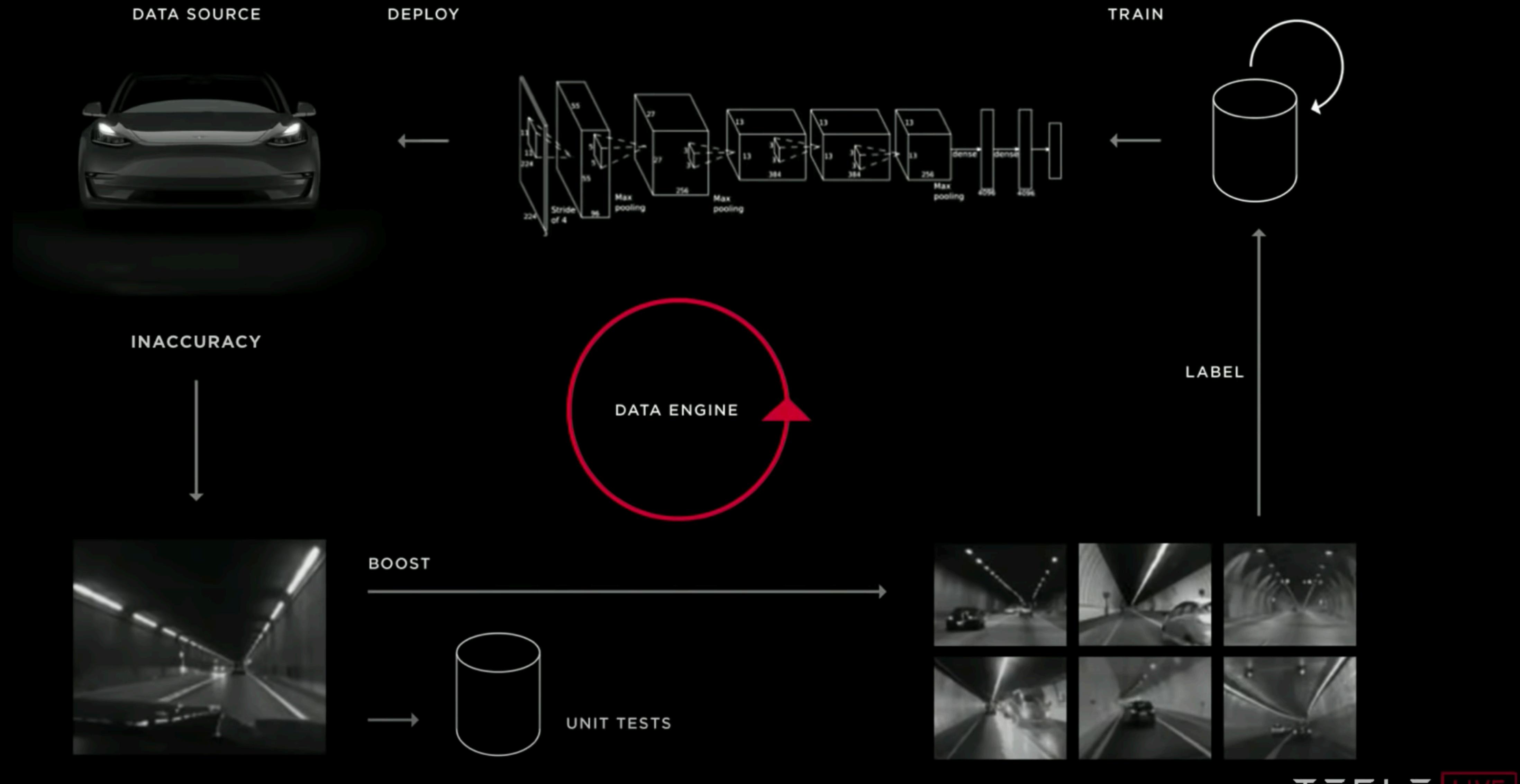
(dropped) (most important)

ID	Weight	Heartrate	Age	Heart Disease?
0	76	54	55	0
1	81	42	34	0
2	90	70	47	0
3	67	100	79	1

## 2. Machine Learning Process

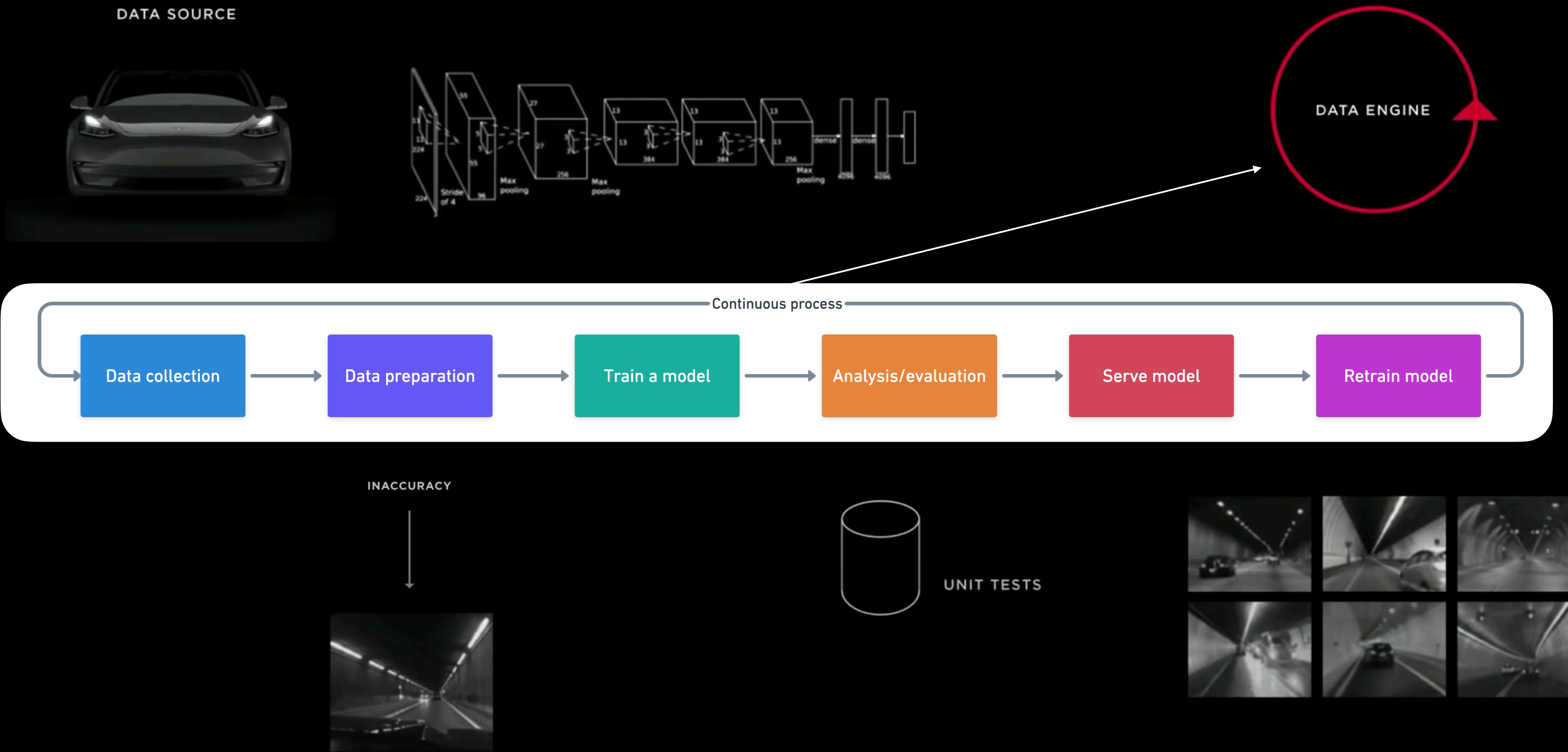
## 2. 🍔 Machine Learning Process



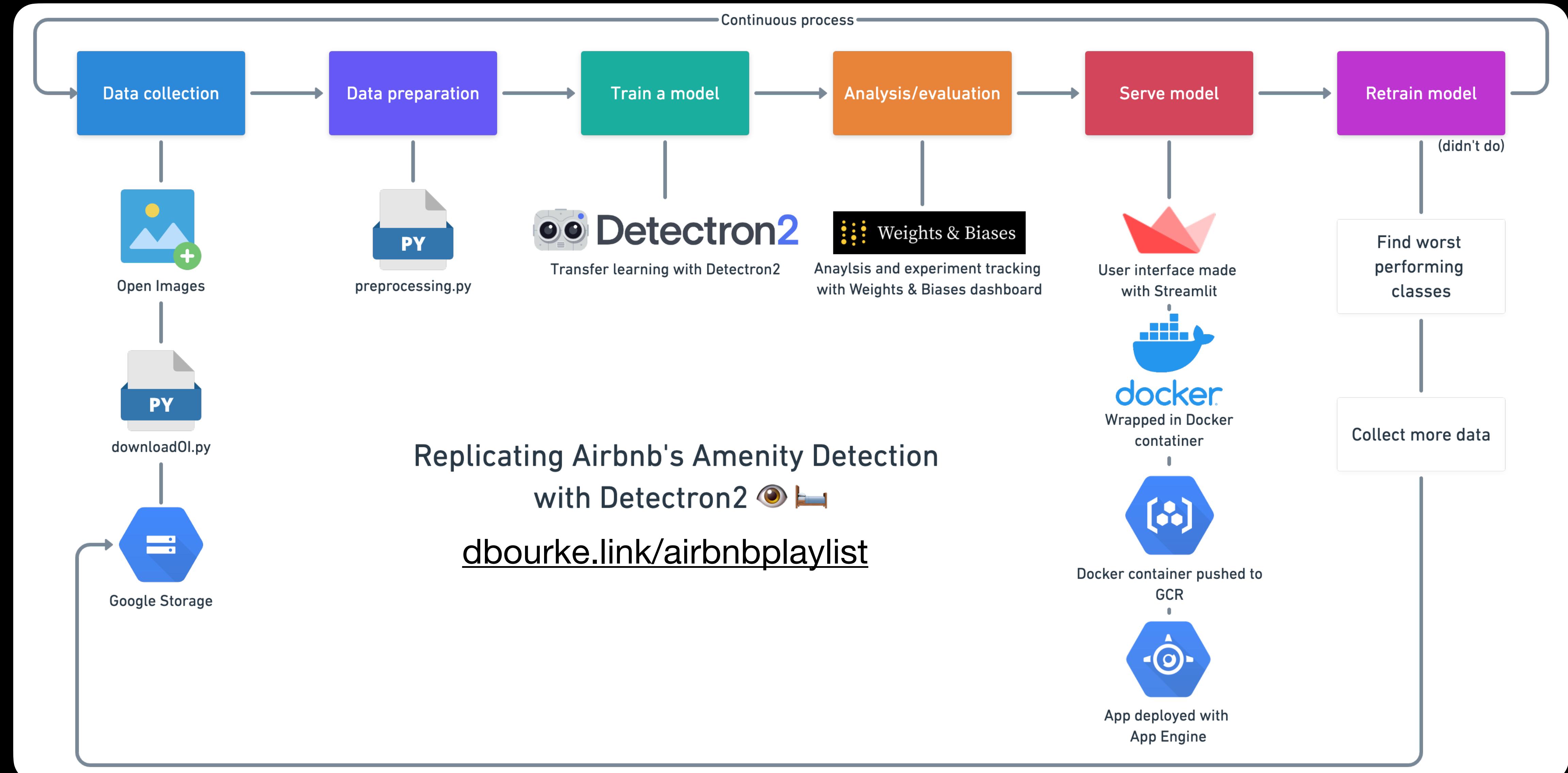


TESLA LIVE

**Source: Tesla Autonomy Day video**



## 2. 🍗 Machine Learning Process



### 3. Machine Learning Tools

# 3. Machine Learning Tools

Libraries/code space



TensorFlow.js  
TensorFlow Lite



Experiment Tracking



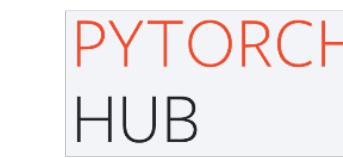
by Weights & Biases



Pre-trained models



TensorFlow Hub



HuggingFace Transformers

Data and model Tracking



by Weights & Biases

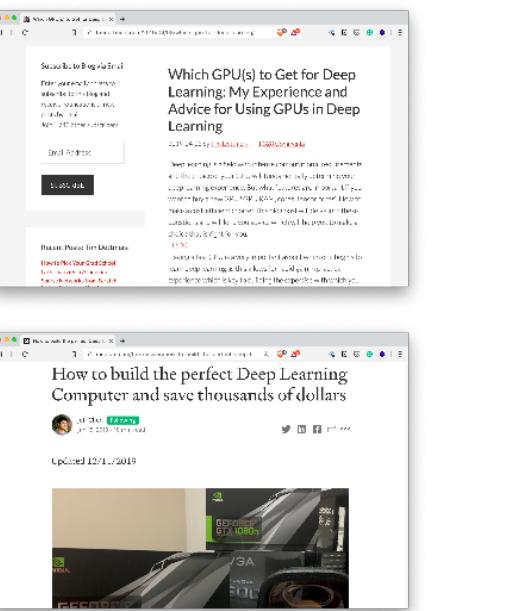


Data version control

Cloud Compute Services



Hardware  
(building your own deep learning PC)



AutoML & hyperparameter tuning



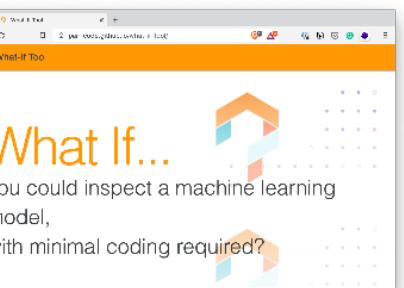
by Weights & Biases



Google Cloud AutoML



Explainability



ML Lifecycle



SELDON



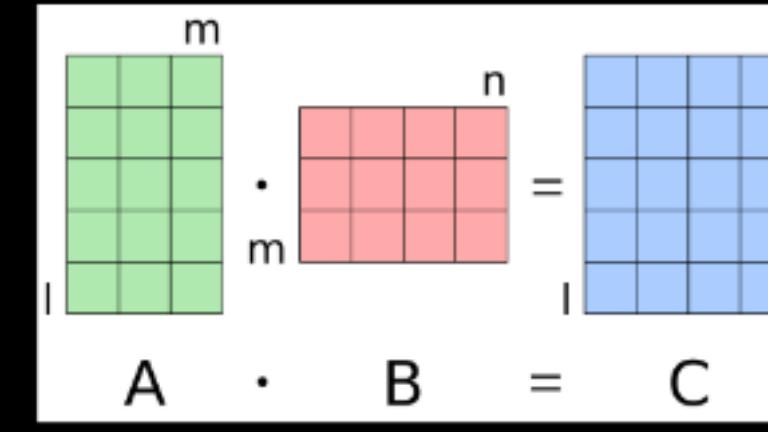
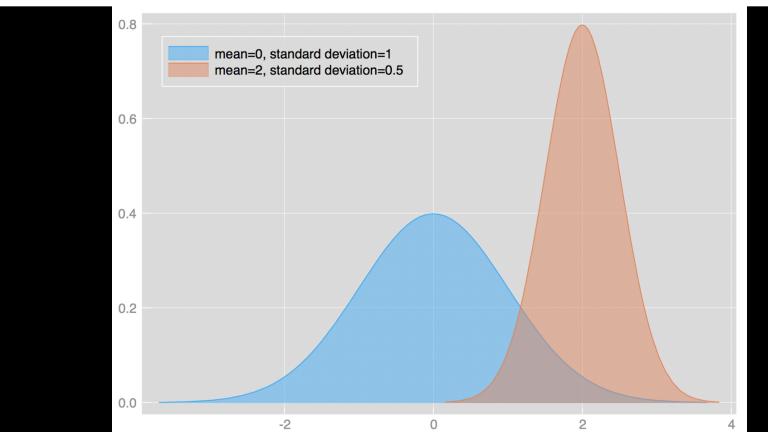
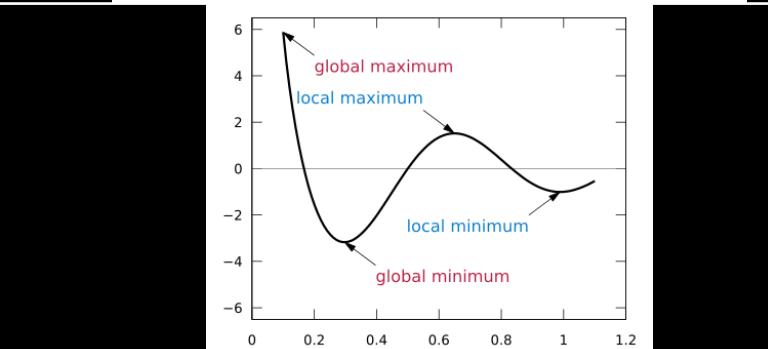
User Interface Design



# 4. Machine Learning Mathematics

(some of the main ones)

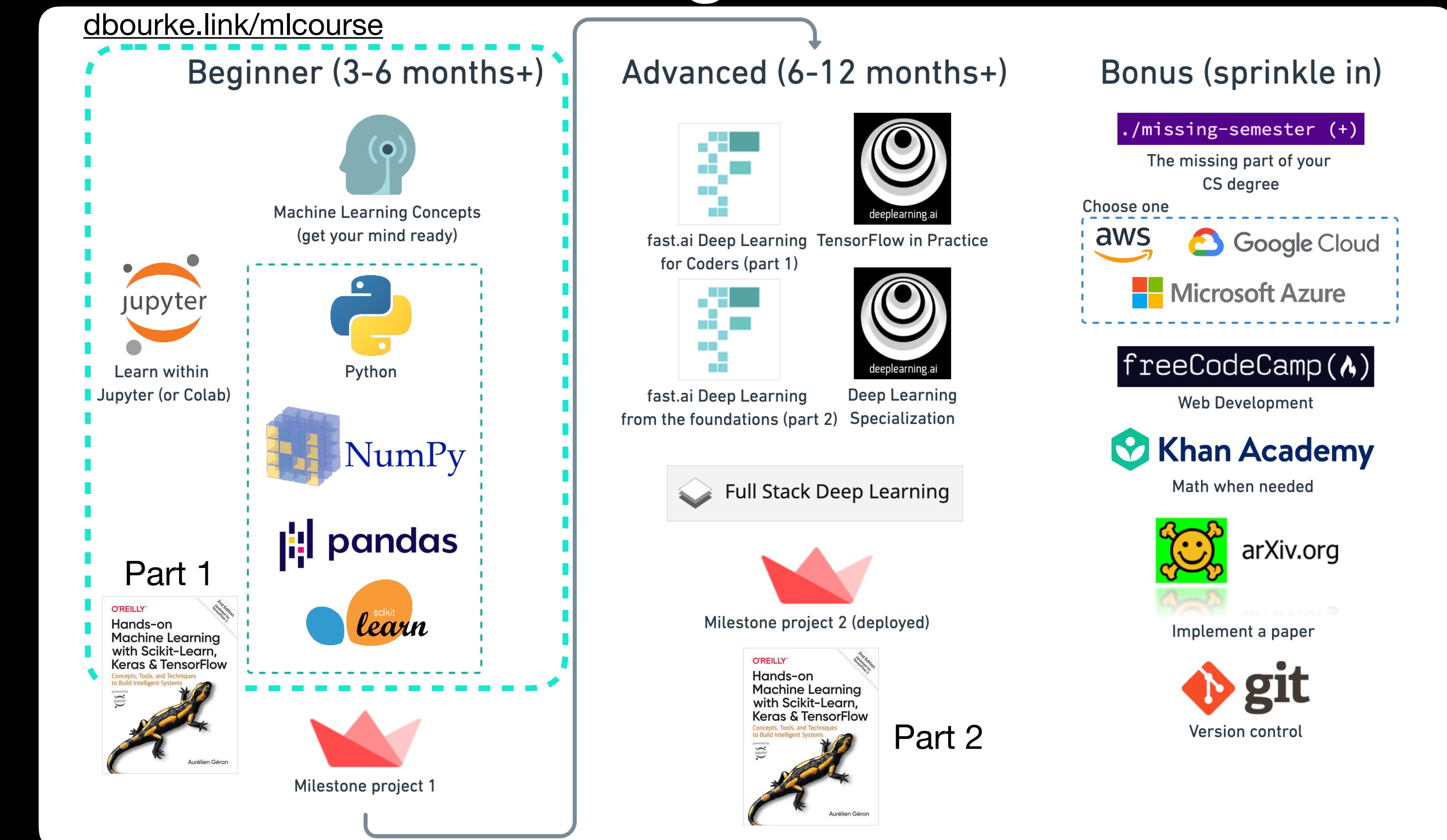
# 4. Machine Learning Mathematics

<b>Linear Algebra</b>	$\begin{array}{l} 2x + y - z = 8 \\ -3x - y + 2z = -11 \\ -2x + y + 2z = -3 \end{array}$	
<b>Matrix Manipulation</b>		
<b>Multivariate Calculus</b>	$f(x, y) = \frac{x^2 y}{x^4 + y^2}$	
<b>The Chain Rule</b>	$\frac{d}{dx} [f(g(x))] = f'(g(x))g'(x)$	
<b>Probability + Distributions</b>		
<b>Optimization</b>		

## 5. 📚 Machine Learning Resources

(where to start learning)

# 5. Machine Learning Resources



# 5. 📚 Machine Learning Resources

(example curriculums)

Daniel Bourke's Self-Created AI Masters Degree + Other Learnings

Resources

- Siraj's YouTube Channel
- Deep Learning Books
- Probability Summary
- Calculus and Statistics Cheat Sheets from Siraj
- AI: a practical approach textbook
- Free Data Analytics Pathway on Springboard
- A-Z Data Science Course - Udemy
- Fast.ai - Deep Learning for Coders Course

Projects

- PROJECT: Build an Adversarial Search Agent
- PROJECT: Create a Domain-Independent Planner
- PROJECT: Use Hidden Markov Models to Recognise American Sign Language

To-do

- Udacity Intro to Data Analysis Course
- Khan Academy Courses
- Read this on linear regression
- Update Udacity Professional Profile
- AI Key Terms List
- Optional Reading - Artificial Intelligence - A Modern Approach (AIMA): Chapter 5.1-5.2
- Reading - AIMA: Chapter 5.3-5.4
- Reading - Multi-player-Alpha-beta-

Doing

- Read Grokking Deep Learning
- AIND TERM 1: Foundations of AI
- Start Udacity AIND Repository on GitHub

Done - September 2017

- Solve Sudoku Puzzle by Hand
- Read Peter Norvig's Blog Post on Solving any Sudoku with AI
- PROJECT: Solving Sudoku with AI

[dbourke.link/aimastersdegree](http://dbourke.link/aimastersdegree)

JB How I learned web development

Jason Benn

Machine learning  
Communal living  
Future of work  
Living well  
About me

How I learned web development, software engineering, & ML

Ever since Cal Newport wrote about how I learned how to code in [Deep Work](#) by reading books and going to a bootcamp (which thousands of people do every year - I'm not special), I've been getting a few emails a week asking for more details. Most often, people ask for the specific books I read as I was preparing, or general advice on breaking into the field. Below is a representative sample. If I sent you this link to you, I hope it's helpful! If you have any further questions, please [email me](#), and I'll use them to improve this guide.

"We greatly overestimate what we can do in one year. But we greatly underestimate what is possible for us in five years." -Peter Drucker. This guide probably would've scared the shit out of me back in 2013 when I was starting. But it's worth it! Software engineering is creative, intellectually stimulating, high-impact, flexible (you can work on almost any problem you want), more social than people realize (pair programming is great!), you can work outside or abroad, it's well compensated, and it has an unlimited skill ceiling. The last part is the *best*, because if you're more systematic about studying than your peers, then it's just a matter of time before you'll be so in-demand that you'll be able to dictate the terms of your employment such that your lifestyle is however you want it to be. I recommend it to most everyone that thinks they might learn to like it.

tl;dr: two steps to software mastery

"Hi Jason! I've read from Cal Newport's book 'DeepWork' that you already finished reading as many as 18 books in a few weeks/months by the time you attended the notorious Dev boot camp to become a full stack programmer. Could you tell me which are those software books you were reading. I'm looking for a career change into programming, so any inputs you give would give me would surely help me here."

<https://jasonbenn.com/post/how-i-learned-to-code>

## 5. 📚 Machine Learning Resources

(some useful places to visit)



sotabench



Papers with Code



Made with ML

# End

Summary, plus next steps

# What we're going to covered (broadly)

1. 🤔 **Machine Learning Problems**—what does a machine learning problem look like?
2. 🔄 **Machine Learning Process**—once you've found a problem, what steps might you take to solve it?
3. 🔧 **Machine Learning Tools**—what should you use to build your solution?
4. 📏 **Machine Learning Mathematics**—what exactly is happening under the hood?
5. 📚 **Machines Learning Resources**—okay, this is cool, how can I learn all of this?

How:



# What's missing?



- Instance vs. model-based learning
- Machine learning pipelines (still emerging on best practices)
- More specific topics (such as computer vision, NLP, RL...)
- Probably much more  
*(leave a comment)*

# Sources (thank you's)



- Daniel Formoso's machine learning mind maps
- Hands-on Machine Learning Book (2nd edition)
- 100-page Machine Learning Book
- Countless blog posts (check the mind map for links)

Keep learning  
Keep creating

*(machine)*