Applications of Machine Learning

Applicazioni Informatiche del Machine Learning

LECTURE 1 | GÉRON CH. 1–2

What machine learning is, and how we will work

BSc Mathematics of Artificial Intelligence · Third year

23 lectures

Textbook: Géron, *Hands-On Machine Learning with Scikit-Learn and PyTorch*

06:30 Your phone waited to finish charging



It sat at most of a charge all night and topped up just before your alarm, because it has learned when you usually get up.

06:58 Your watch knows you slept badly



Before the alarm goes off, your wrist has already decided which parts of the night were deep sleep, which were dreaming, and which were you scrolling in the dark.

07:01 Your phone recognises you before coffee



Bed hair, no glasses, half asleep, in the dark — and the phone still unlocks for you, and not for your sibling.

07:05 Your inbox has a bouncer



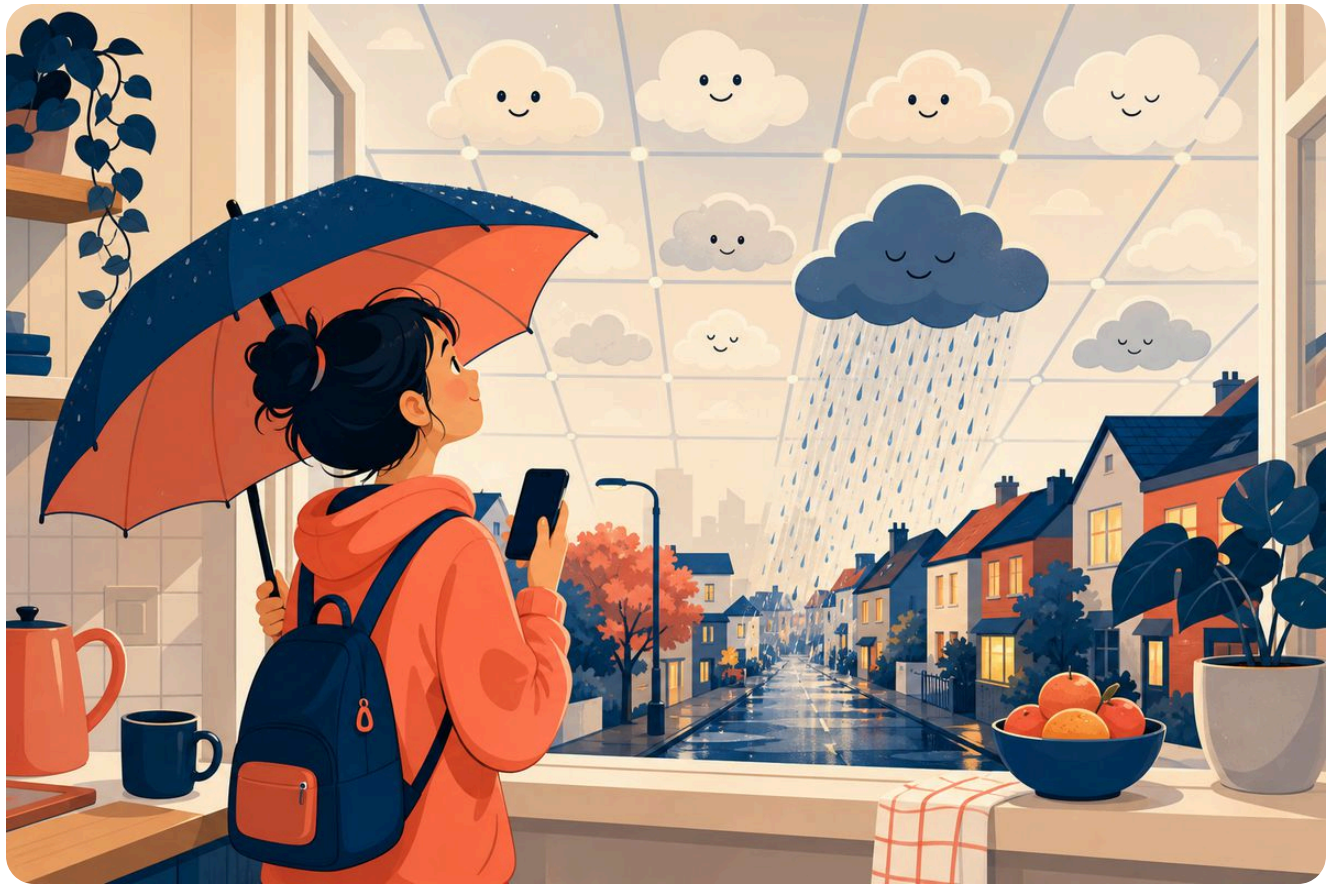
The prince who wants to share his fortune with you never reaches your inbox. Billions of messages a day are turned away at the door.

07:12 “Sounds good, thanks!”



The three little replies under an email or a chat were not written by you, and somehow they are usually exactly what you were going to say.

07:15 Will it rain? Ask a neural network



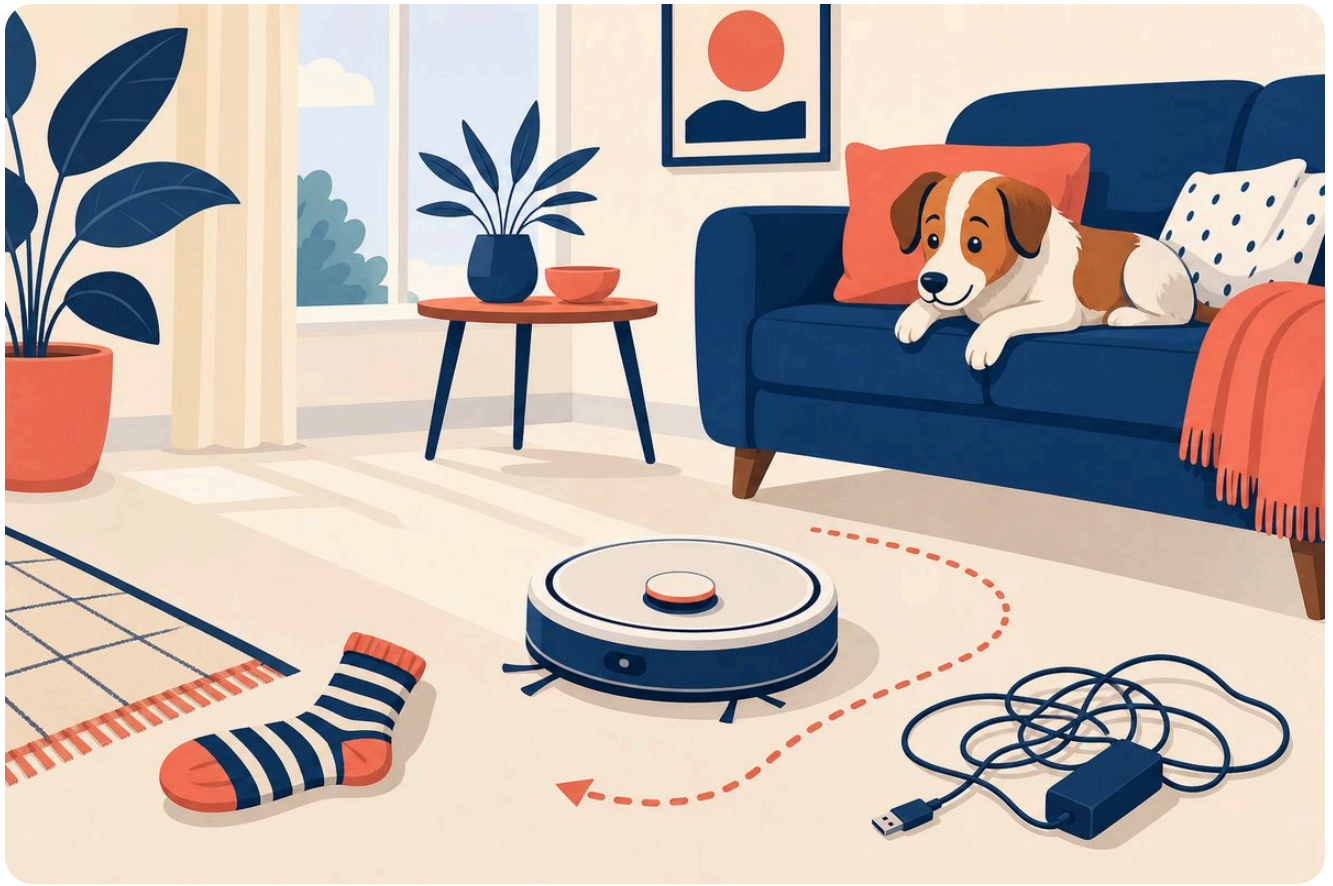
Some of the world’s big forecasting centres now run machine-learned forecasts alongside the physics, and they hold their own.

07:20 Your feed is not in chronological order



Out of everything your friends posted overnight, a few hundred things were chosen, and ordered, for you alone. Your flatmate sees a different feed.

07:45 The robot vacuum steers round the sock



It maps the flat, recognises the charging cable, the sock and the thing the dog left behind, and goes around all three.

08:00 Your car braked before you did



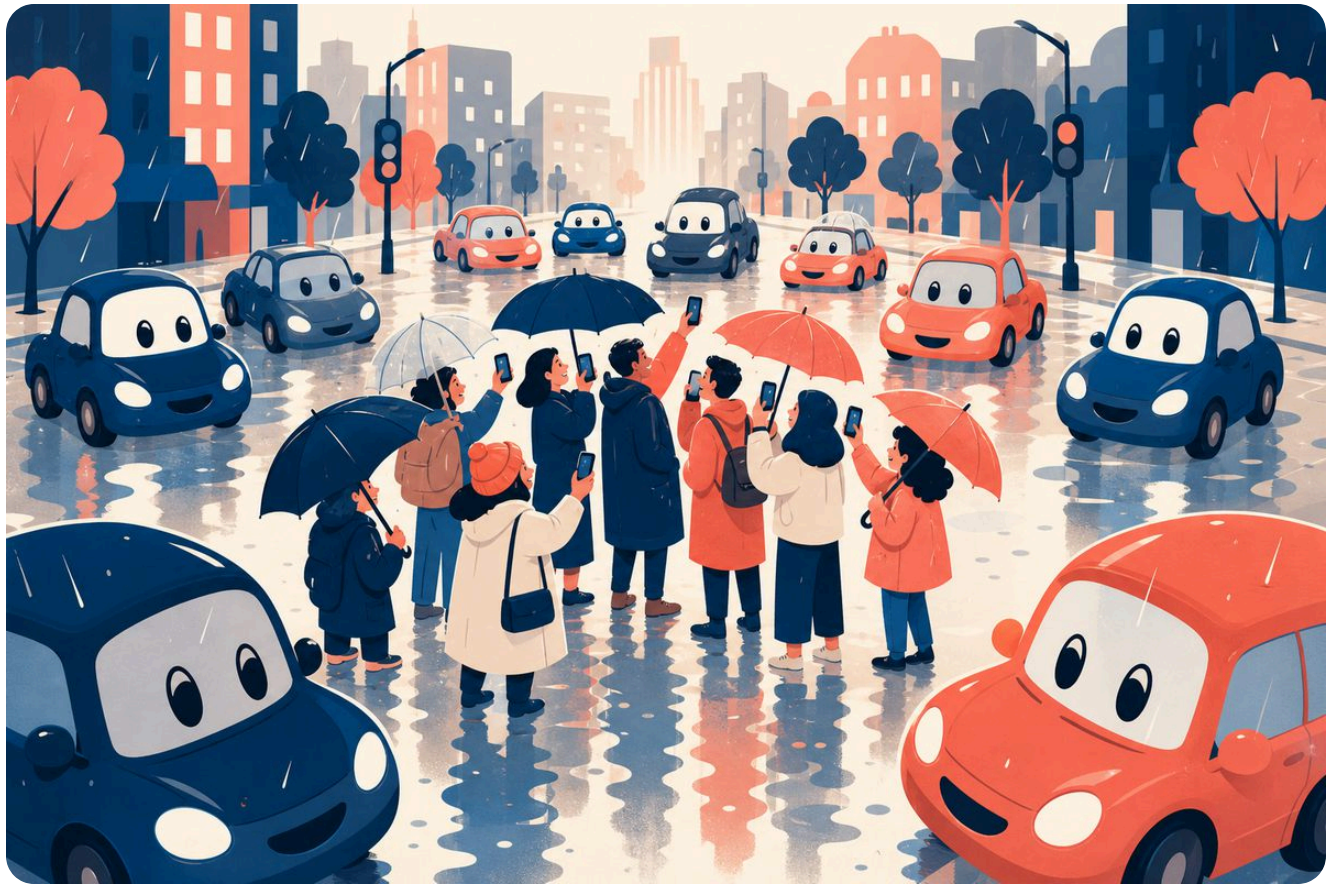
A cyclist appears from behind a parked van and the car stops on its own. Automatic emergency braking is required on every new car sold in the EU.

08:05 “You will arrive at 8:41”



An arrival time is not a measurement. It is a prediction about traffic that has not happened yet, and it is usually right to the minute.

08:10 Why the ride costs more in the rain



When you open a ride-hailing app, it has already guessed how many people are about to want a car, where, and how much they will pay.

08:12 Nobody in the driver’s seat



In some cities you can already call a taxi that drives itself, all day, every day, through ordinary traffic.

08:15 A mixtape from a stranger who knows you



Once a week a playlist appears of songs you have never heard, by artists you have never heard of, and you like most of them.

08:25 The gate that read your number plate



Drive into a restricted-traffic zone without a permit and a camera reads your plate, at speed, in the rain — and the fine finds you.

08:30 Name that tune in three seconds



Through café noise and a coffee machine, a few seconds of a song are enough for your phone to tell you exactly which recording it is.

09:00 Reading a menu you cannot read



Point your camera at a sign in a language you do not speak, and the words change on the screen into yours, in the same font, on the same wall.

09:20 The owl knows what you will forget



Language apps decide which word to ask you today based on when you are most likely to forget it — and they send the guilt-trip notification at the right moment too.

09:45 “Was this you?”



A sign-in to your account from a new country, on a new device, at an odd hour. The service asks you before it lets the stranger in.

10:00 Ten links, out of the whole web



You type four words, misspell one of them, and of all the pages that exist, the one you wanted is in the first three.

10:15 It finishes your sentences



The three words above your keyboard are a guess about what you are going to type next — and the guess is trained on how *you* write.

10:16 Autocorrect: hero and villain



It fixes the typo you did not notice, and then, with complete confidence, turns your message to your professor into something else entirely.

10:30 Copying text out of a photo



You photograph the whiteboard at the end of the lecture, and later you select the text in the photo and paste it into your notes.

11:00 “Hey, set a timer for ten minutes”



Your voice, with your accent, over a running tap, becomes text, then an intention, then an action. In under a second.

11:30 Your pair programmer never sleeps



You type the name of a function and the rest of it appears in grey. Most of the time it is right. This course is about the rest of the time.

12:00 You asked a machine to explain your homework



A chat assistant that drafts your emails, explains a proof, and plans a trip — and sometimes says something false with exactly the same confidence.

12:30 That picture never happened



Type a sentence and get a picture. Every illustration in this lecture was made that way, from a one-paragraph description.

13:00 Subtitles nobody typed



Live captions on a video, a lecture, or a call are written as people speak — including, if you are unlucky, a creative interpretation of your surname.

13:30 Describing a photo to someone who cannot see it



A screen reader comes to a photo nobody captioned and says what is in it: two people smiling on a beach, at sunset.

14:00 The dog barked, and nobody heard it



On a video call, the barking dog, the drill next door and your messy bedroom all vanish, and only your voice and your face are left.

14:30 Select all images with traffic lights



Proving you are not a robot has, at times, also meant labelling pictures for one — the early puzzles helped digitise old books.

15:00 People who bought this also bought…



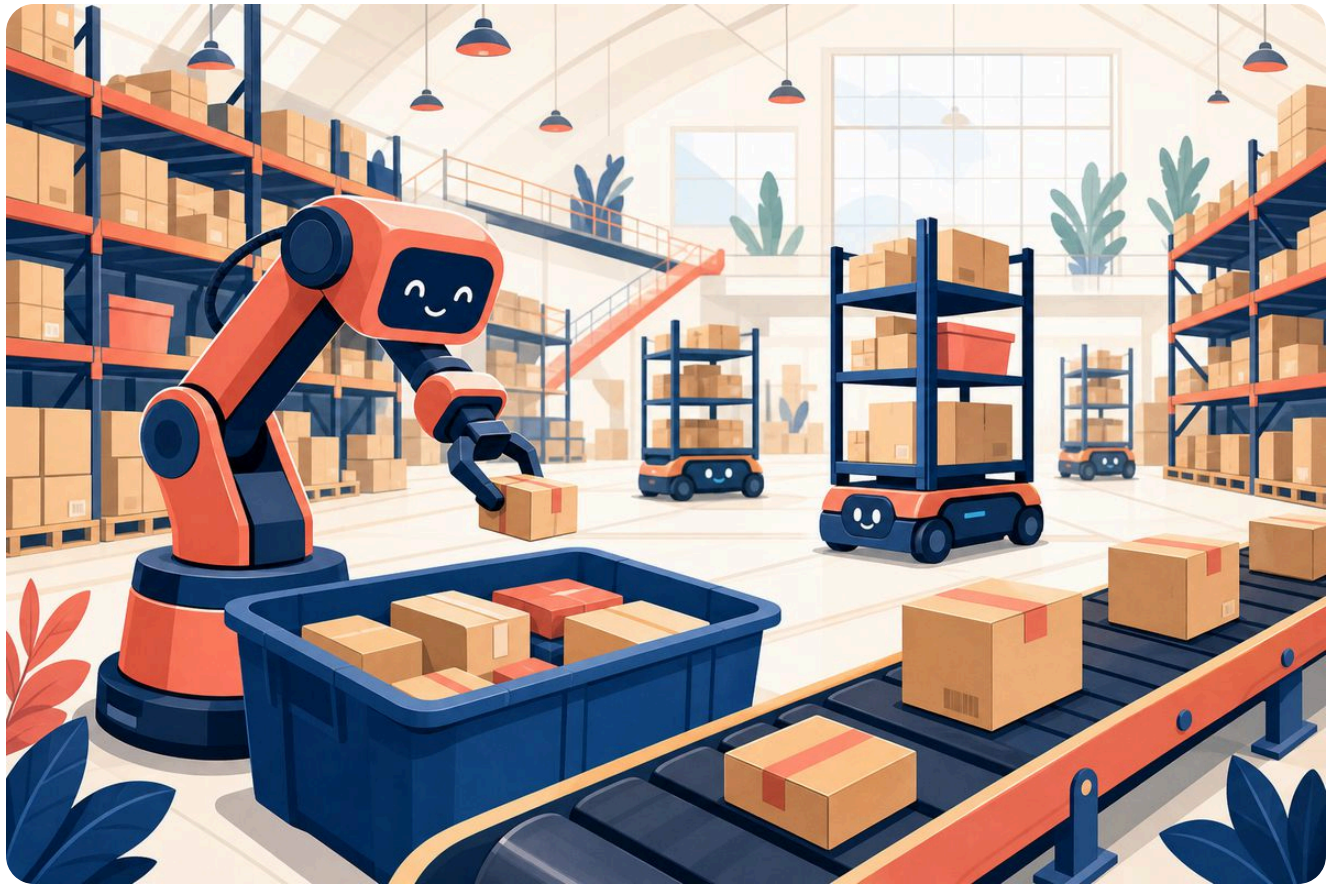
You came for a phone charger. You leave with a charger, a cable, a stand, and a lamp you did not know you needed.

15:05 Why your bank texted you at 3 a.m.



The card was used in another country, for an odd amount, at an odd hour, and someone — something — noticed within a second.

15:10 Your parcel was picked by a robot



Somewhere in a warehouse, a machine that learned to grasp things it had never seen picked up your charger and put it in a box.

15:30 The ad that seems to read your mind



You mention running shoes once, and for a week the internet is full of them. Every ad you see was chosen in the instant the page loaded.

15:45 A machine read your CV first



For many jobs, before any person sees your application, software has ranked it. The EU’s AI Act counts that as high-risk.

16:00 “What plant is this?”



Point your phone at a leaf, a dog, a pair of trainers or a maths problem, and it tells you what it is — or where to buy it.

16:30 Should the bank lend you money?



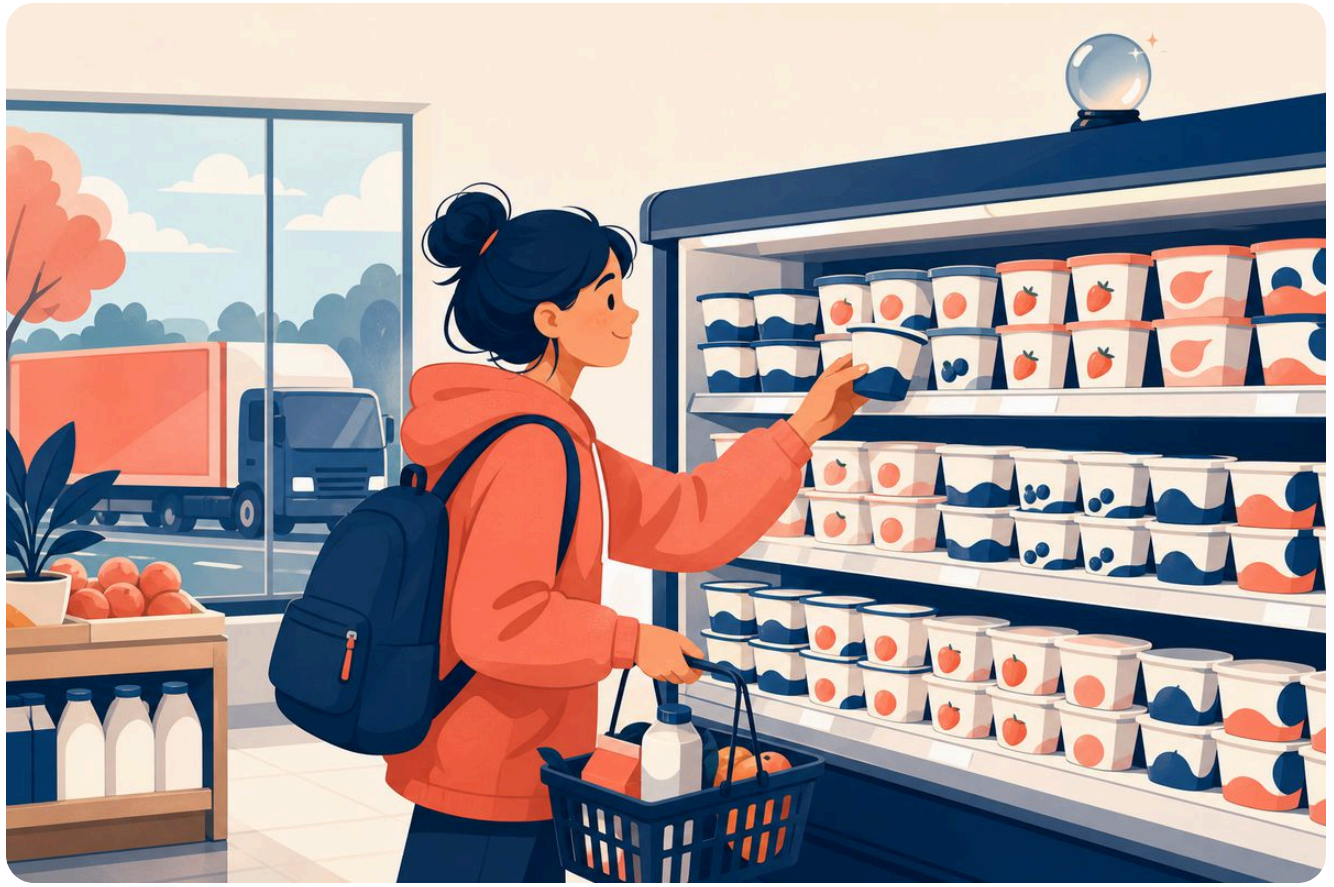
Whether you get the loan, and at what rate, is shaped by a score a model computes from your history. The AI Act calls this high-risk too.

17:00 “Show me dogs at the beach”



Thousands of photos, none of them labelled, and your gallery finds every dog, every beach, every birthday cake, and every photo of your grandmother.

17:30 Why your yoghurt was on the shelf



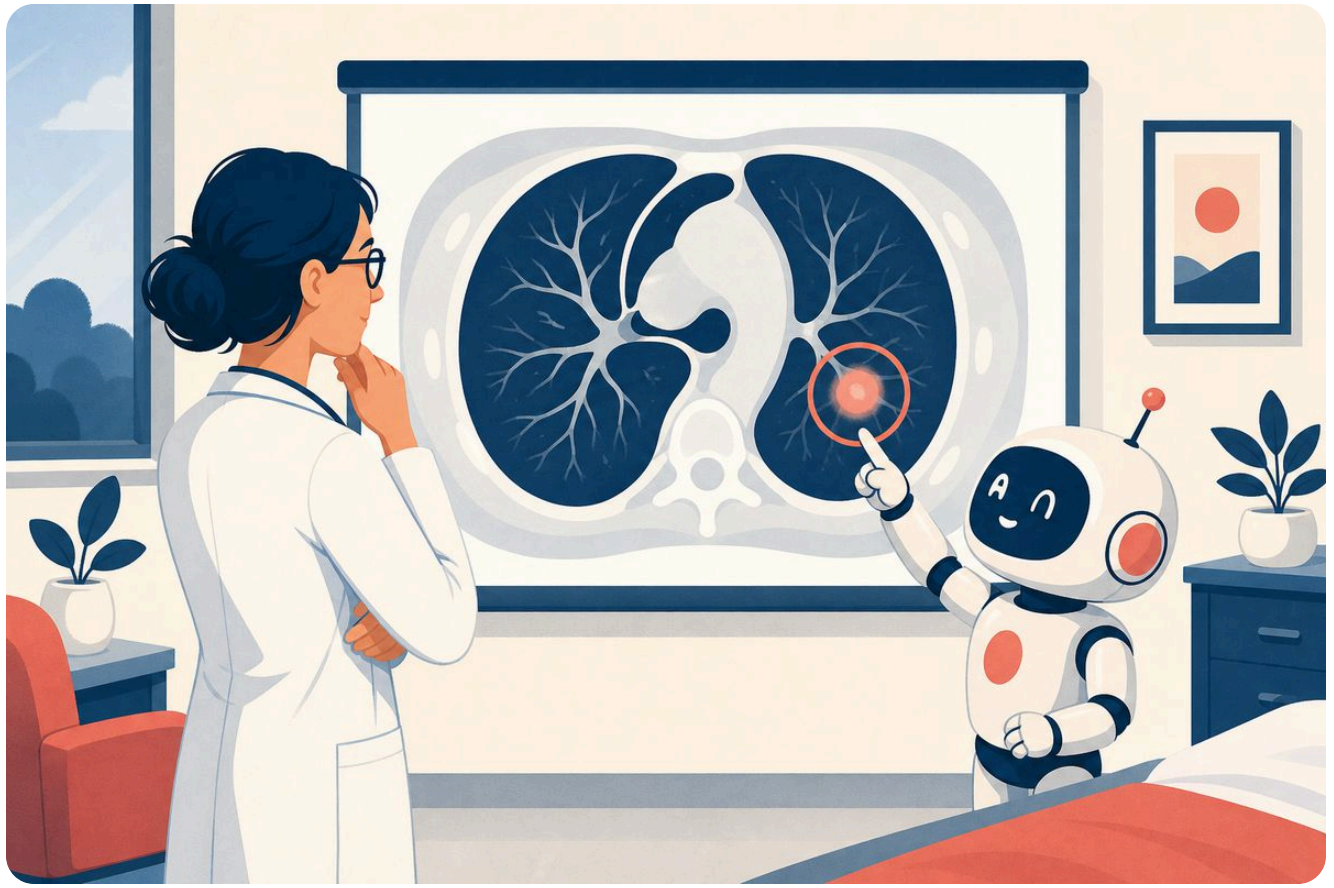
The supermarket ordered it days ago, store by store, because it predicted roughly how many people would buy it on a sunny afternoon.

18:00 Your watch noticed before you did



On an evening run, a smartwatch can flag an irregular heart rhythm, or call for help if you fall and do not get up.

18:30 A second reader for your scan



In some hospitals a model now reads screening images alongside the radiologist, and points at the spot worth a second look.

18:45 A protein, folded on a computer



Predicting the shape of a protein from its sequence was a fifty-year-old problem. A neural network largely solved it, and its creators shared the 2024 Nobel Prize in Chemistry.

19:30 Photographing the dark



A phone camera with a lens the size of a lentil takes a sharp, bright picture of a candlelit dinner. Most of that picture was computed, not captured.

20:00 Dog ears, in real time



The filter finds your eyes, nose and mouth, follows them as you move, and puts puppy ears exactly where they belong, thirty times a second.

20:15 The lights stay on at dinner time



Everyone cooks at once. The grid operator predicted the evening peak hour by hour, so the power plants were ready for it.

20:30 Who keeps the internet (mostly) clean



Every minute, hours of video are uploaded to the big platforms. Most of what should never be seen is removed before anyone sees it.

20:45 Offside, decided by a skeleton



In the big competitions, cameras track a stick-figure skeleton of every player, and the offside line is drawn in seconds.

21:30 Pixels your graphics card never drew



In many modern games, part of every frame on your screen is not rendered at all. It is invented, and you cannot tell which part.

22:00 Who appears on your dating app



The profiles you are shown, and the order you see them in, are a recommendation — just one where the items can also say no.

23:30 Just one more episode



Even the picture on each show was chosen for you. Someone who loves romance and someone who loves action see the same film with different artwork.

23:55 One last scroll: what is this flat worth?



Property sites show an estimated price for almost any home, one you are not even selling. That estimate is the model we start building right now.

PART ONE

The brief

You have just been hired

You are a data scientist at a real-estate investment company.

> **THE TASK**
>
> Use California census data to build a model of housing prices. The data gives population, median income and median housing price for each *block group* — the smallest unit the US Census publishes. We will call them **districts**.

Your model predicts a district's **median housing price** from the other metrics.

The first question is never technical

What exactly is the business objective?

- Building a model is not the goal
- How will the company use the output?
- The answer determines the framing, the algorithm, the metric, and how much effort is worth spending

What the output feeds

Your model's prediction is fed to **another machine learning system**, together with other signals, to decide whether a district is worth investing in.

> *TERMINOLOGY*
>
> *A piece of information fed to an ML system is called a signal — after Shannon's information theory. You want a high signal-to-noise ratio.*

✗ Note the consequence: nobody downstream ever looks at your prediction. A component that quietly degrades inside a chain of components is not noticed by anyone — which is why the evidence has to come from you.

What is the current solution?

District prices are currently estimated **manually by experts**: a team gathers information and applies complex rules.

> **THE NUMBER THAT MATTERS**
>
> When the team can check their estimates against the truth, they are typically off by more than **30%**.

Remember this figure — and read it carefully. It is a **relative** criterion: 30% *of the district's price*. The errors we will measure are in dollars, which is not the same thing, and we come back to this when we measure them.

It is also the stakeholder's claim about their own team, not a measurement we made. We will treat it as an assumption and label it as one wherever it appears.

Frame the problem

Before writing any code, answer four questions:

1. What kind of **training supervision** does this need?
2. Is it **classification, regression**, or something else?
3. Should we use **batch or online** learning?
4. What **assumptions** are we making?

Pause here and answer them yourself before the next slide.

Questions 1–3, answered

Question	Answer	Because
Supervision	✓ **supervised**	every district carries its expected output
Task	✓ **multiple, univariate regression**	predict one value, from several features
Learning mode	✓ **batch, offline**	no data stream, and 20,640 rows fit in memory

> ***THE ALTERNATIVES, FOR REFERENCE***
>
> *Unsupervised (no labels) · semi-supervised (some labels) · self-supervised (labels from the data itself) · reinforcement (rewards). Predict several values per district and it becomes multivariate regression; add a data stream and it becomes online learning.*

Three answers, one minute. The fourth question is the one that can sink the project.

4 · Check the assumptions

List and verify what everyone is assuming. This catches serious problems early.

> **THE ONE THAT WOULD SINK US**
>
> What if the downstream system converts our prices into *categories* — “cheap”, “medium”, “expensive” — and discards the numbers? Then this is a **classification** problem and we have built the wrong thing.

We asked. They need the actual prices. We may proceed.

PART TWO

The data

one look at everything, before anything is split

Automate the download

```
from pathlib import Path
import pandas as pd
import tarfile
import urllib.request

def load_housing_data():
    tarball_path = Path("datasets/housing.tgz")
    if not tarball_path.is_file():
        Path("datasets").mkdir(parents=True, exist_ok=True)
        url = "https://github.com/ageron/data/raw/main/housing.tgz"
        urllib.request.urlretrieve(url, tarball_path)
        with tarfile.open(tarball_path) as housing_tarball:
            housing_tarball.extractall(path="datasets", filter="data")
    return pd.read_csv(Path("datasets/housing/housing.csv"))

housing_full = load_housing_data()
```

Write a function, not a manual download: the data will change, and you will need this on another machine.

The structure

One row per district, ten attributes. Start with `head()` and `info()`, always, before anything else.

```
>>> housing_full.info()
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   longitude           20640 non-null   float64
 1   latitude            20640 non-null   float64
 2   housing_median_age  20640 non-null   float64
 3   total_rooms         20640 non-null   float64
 4   total_bedrooms      20433 non-null   float64
 5   population          20640 non-null   float64
 6   households          20640 non-null   float64
 7   median_income       20640 non-null   float64
 8   median_house_value  20640 non-null   float64
 9   ocean_proximity     20640 non-null   object
```

Observation: 207 districts are incomplete

`total_bedrooms` has **20,433** non-null values, not 20,640.

✗ 207 districts are missing this feature.

> **THREE OPTIONS, ALL LEGITIMATE**
>
> Drop those districts · drop the whole attribute · fill the gaps with a chosen value. We will return to this in the build.

One attribute is not numerical

```
>>> housing_full["ocean_proximity"].value_counts()
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
```

Repeated values from a small set: this is a **categorical** attribute, not free text.

✗ Note `ISLAND`: five districts. Rare categories will cause trouble later.

Summary statistics

```
housing_full.describe()
```

- `count` ignores nulls — that is why `total_bedrooms` reads 20,433
- `std` is the standard deviation: how dispersed the values are
- `25%`, `50%`, `75%` are the quartiles and the median

For a bell-shaped distribution the 68–95–99.7 rule applies: about 68% of values fall within 1σ of the mean, 95% within 2σ, 99.7% within 3σ.

Plot everything

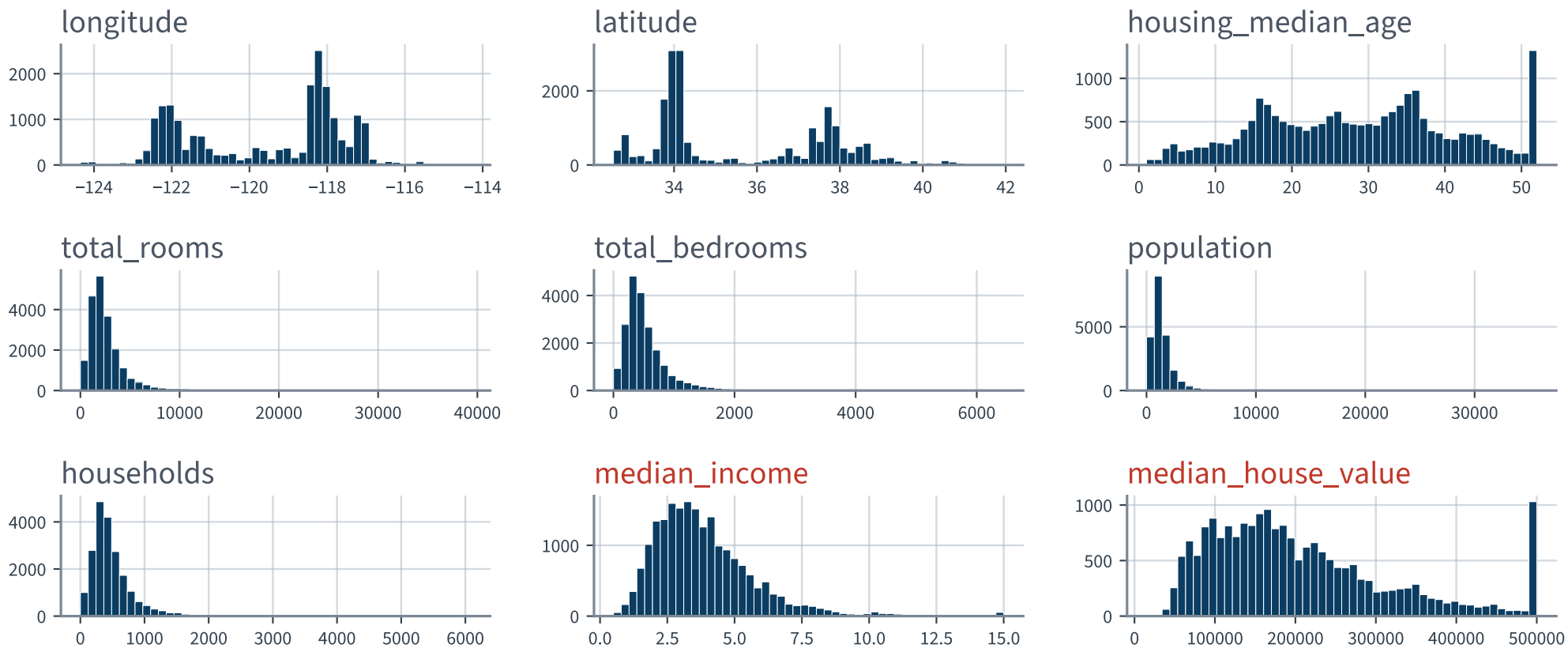
```
import matplotlib.pyplot as plt

housing_full.hist(bins=50, figsize=(12, 8))
plt.show()
```

A histogram shows how many instances fall in each value range. Four things should jump out.

Nine attributes, fifty bins

housing_full.hist(bins=50)



Take thirty seconds. What is wrong with the bottom two?

Observation 1 · The income is not in dollars

`median_income` ranges roughly 0.5 to 15. That is not a salary.

> The data was scaled and capped: at 15 (actually 15.0001) at the top and 0.5 (0.4999) at the bottom. The numbers are roughly tens of thousands of dollars — 3 means about $30,000.

Preprocessed attributes are normal. **Understanding how they were computed is not optional.**

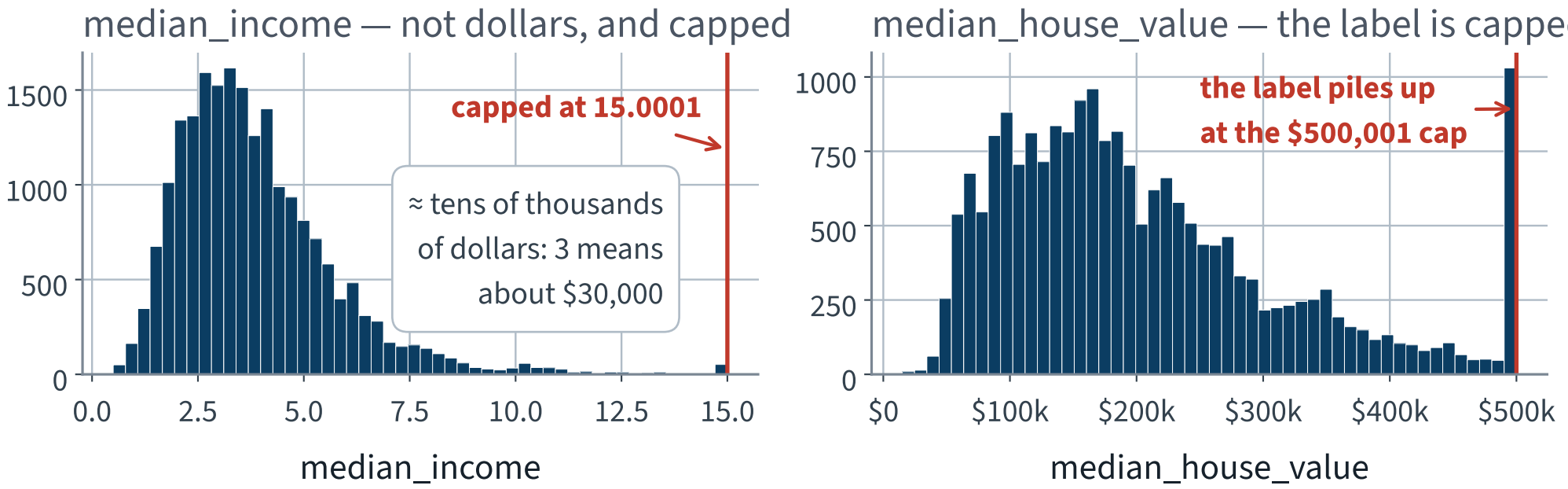
Observation 2 · The target is capped

`median_house_value` is capped — and it is our **label**.

> **WHY THIS IS SERIOUS**
>
> Your model will learn that prices never exceed the cap. If the client needs accurate predictions beyond $500,000, the model is unusable there.

Two honest fixes: collect proper labels for the capped districts, or remove them from *both* the training and the test set.

Both spikes, enlarged



✗ 965 districts — 4.7% of the dataset — sit at the cap, $500,001: a label that is not their price. It is a ceiling.

Observation 3 · The scales differ wildly

`total_rooms` runs from 2 to 39,320. `median_income` runs from 0.4999 to 15.0001.

> **WHICH MATTERS FOR SOME METHODS AND NOT OTHERS**
>
> **Scaling is required for:** gradient descent, logistic regression, SVMs, k-nearest neighbours, k-means, PCA, neural networks. Anything that measures a distance, penalises a coefficient, or takes a step.
>
> **Scaling does nothing for:** decision trees, random forests, and ordinary least squares. Trees split on one feature at a time, so any monotone rescaling gives the same tree; least squares is *equivariant* under an invertible affine map of the features — the fit is identical and the coefficients simply absorb the change.

✗ Linear regression, trees and forests — the models this dataset will be fitted with — are all in the second list. It is also exactly why scaling before the split, a classic leak, will cost nothing here: an invertible affine map, applied to an estimator that does not care. We will measure that rather than assert it.

Observation 4 · Many attributes are skewed right

They extend much further to the right of the median than to the left.

This makes patterns harder for some algorithms to detect. Later we will transform them toward something more symmetric — typically by taking a square root or a logarithm.

A feature following a power law is a good candidate for $\log$: districts of 10,000 people are ten times rarer than districts of 1,000, not exponentially rarer.

Stop. Before we look any closer

Four observations from four histograms, and we have not yet plotted one variable against another. That is exactly the moment to stop.

The only way to know how a model generalises is to try it on cases it has never seen.

- Split the data: a **training set** and a **test set**
- The error rate on new cases is the **generalisation error**
- Low training error with high generalisation error means **overfitting**

Typically 80/20 — but with 10 million instances, holding out 1% is plenty.

The notebook for this lecture

Open Lecture 1 in Colab

- **Runs on free CPU**
- Everything on today's data slides, in order: fetch, `info()`, the missing values and the categories, the histograms, the range of every column, and the cap — counted, not squinted at
- Every code cell carries the specification that would produce it — read the box, predict the output, then run

> **WHAT TO DO WITH IT BEFORE THE NEXT LECTURE**
>
> Run it top to bottom once. Then change one thing — `bins=50` to the default `bins=10` in the histogram cell — and look for the two spikes. One argument, and it decides whether you ever see the cap.

Design exercise · in the manner of an ML system-design interview

The task: a property marketplace’s instant home valuation

> **THE BRIEF**
>
> A property marketplace — the app where homes are listed and searched — wants an **instant valuation** on every listing and on an owner’s own home page. Buyers read it before offering, owners before selling, and the agents who pay for leads get it with each lead. The business measures leads sold and owners who list; a wrong number costs a listing, a buyer, or a visit from the advertising regulator. The question: a model at all?

- **Given:** the public record of past sales — address, date, price — joined to what is known of each home, and current asking prices
- **Scale:** tens of millions of homes; tens of thousands of sales a month; a few a year in a small town, thousands in a city
- **Consumers of the output:** the visitor who reads it, the search ranking, and the agents who buy the leads

✓ Nine steps follow, each with the pitfall that catches a first attempt and what to do about it. The model at the centre is today’s: none yet — the framing, the baseline, the label, one look at the data and a held-out set

Step 1 — Frame it: what is the business objective?

- Building a model is not the goal. The marketplace wants owners to list and buyers to enquire; the valuation is a means, and its use decides the framing, the metric and the effort worth spending
- Four questions before any code: every sale carries its price, so **supervised**; one number out, so **regression**; sales arrive monthly and fit in memory, so **batch**; and what is everyone assuming?
- Ask each consumer's team how the number is used: as a price, or as a band — above or below asking?

> **PITFALL**
>
> The search team uses only whether the valuation is above or below the asking price, to badge a listing "good value". For that consumer the system is a classifier, and every regression metric chosen from here on measures the wrong thing. We asked: the visitor needs the price, the badge needs the band, and a price can be turned into a band but not the reverse — so the output is the price, the band is derived downstream, and the answer is written down before any code.

Step 2 — What is the current solution, and what must be beaten?

- Today the page shows the home's last recorded sale price, moved by the region's price index since that date; an agent's valuation where one exists
- That is the **baseline**: a lookup, no model, computed in a line. Any model is judged by how much it beats it, on sales neither has seen
- The product manager says the current number is "usually within a tenth of the sale price": a claim, a relative one, and not yet a measurement

> **PITFALL**
>
> Without the baseline's error on the held-out sales, any error the model reports looks like progress, the release is approved on it, and "within a tenth" becomes marketing copy. Measure the lookup first, on exactly the held-out sales the model will be measured on, and label the figure an assumption wherever it appears.

Step 3 — The label is a sale, not a wish

- One row per **sale**: the price paid, the date, and what was known about the home on that date
- Asking prices are plentiful — every listing has one — and they are not the label: a home listed at one price and sold at another has one price, the second
- Homes that never sold have no label. The estimate for them is a prediction about a sale that has not happened, judged only when it does

> **PITFALL**
>
> Training on asking prices, because there are ten listings for every sale. The model then learns what sellers hope for, which runs above what buyers pay by a margin that differs by town and by season; owners list too high on its advice, and the error against real sales is never seen, because the test set is wishes too. Label with recorded sales only; keep the asking price as a feature at most, never as the answer.

Step 4 — Look at the whole table once, before anything else

- `head()`, `info()`, `describe()`, a histogram of every column: which columns are incomplete — floor area for old sales, year built for many — and which are categorical, with a rare level such as *houseboat*
- The scales — area in square metres, price in hundreds of thousands — and the skew: a few palaces far to the right
- The spikes: a price piled up at one value, an area recorded as zero where it was unknown

> **PITFALL — A CAPPED LABEL**
>
> The public record truncates prices above a reporting threshold, so the most expensive sales all carry the same value. A model learns that prices never exceed it, and the error on exactly the homes with the richest leads is invisible, because the held-out sales are capped too. Count the spike; then obtain the true prices, or remove the capped sales from *both* the training and the test set, and say so.

Step 5 — Set the test set aside before looking any closer

- The only way to know how a valuation generalises is to try it on sales it has never seen: split into a training set and a test set now, before one variable is plotted against another
- The share held out follows the size: a fifth of a few thousand sales, far less of tens of millions
- Low error on the training sales and high error on the held-out ones is **overfitting**, and the training error alone can never show it

> **PITFALL**
>
> A home sold twice is in the record twice, and a random split puts one sale on each side. The second is not a case the model has never seen: same address, same area, nearly the same price, and the error the release rests on reads better than it is. Hold out *homes*, not sales, so that every address is on one side only, and check that it is.

Step 6 — How the error will be measured, and against what

- The error is one number computed on the held-out sales: valuation against price paid, over every home there
- Measured in money, a mistake of twenty thousand on a flat and on a mansion count the same; the owner of the flat does not agree
- The product team's criterion is **relative** — a share of the price — and an error in money is not the same thing

> **PITFALL**
>
> The model is tuned on an error in money, dominated by the expensive homes, and reported to a team whose criterion is a share of each price. The two can move in opposite directions: better on the mansions, worse on the flats that are most of the traffic. Write the metric beside the criterion before anything is fitted, report the error in both forms, and report it by price band, so that nobody approves a release on the average.

Step 7 — When a rule or a lookup table is the better answer

Situation	The better answer
The home sold recently, or carries a fresh valuation	show the record, moved by the index: a **rule**
A small town with a handful of sales a year	the area's median price per square metre, by type and size band, refreshed monthly: a **table**
A city with thousands of sales and prices that change street by street	a **model** — if it beats the table on held-out sales by a margin the visitor can feel

PITFALL

A model is built because the roadmap said so. In the towns with a handful of sales it is fitted on a dozen rows, overfits them, and loses to the table it replaced — and nobody sees it, because the error is averaged over the country, where the cities dominate; the owners there notice. Decide per area: the rule where a record exists, the table where sales are few, the model where it is measured to win.

Step 8 — After launch: a prediction about a market that moves

- The held-out error was measured on last year's sales. Prices drift, and the number shown this spring is judged by sales that close this summer
- Batch learning, on a schedule: refit on the new sales each month, re-measure on the sales closed since, and only then go live; the data team is on call for the error by region
- The number changes the market it predicts: owners set asking prices from it, and buyers anchor on it

PITFALL

An error measured once, before launch, is quoted for years, while the site's own number steers the asking prices that become the next sales. One marketplace bought homes on the strength of its own valuations, overpaid for thousands of them, and closed the business. Keep measuring on the sales that close after each release, record what the site showed on the day of each sale so that its pull can be seen, and report the error month by month, never once.

Step 9 — Ship it: the pipeline, and the pitfalls in one table

Objective agreed with each consumer → the lookup measured as the baseline → recorded sales as the label → one look at the whole table → homes held out before exploring → the metric written beside the criterion → rule, table or model per area → measured on the sales after launch → refit monthly.

Pitfall	Address it by
The number consumed as a band, not a price	asking each consumer first; the price, then the band
No baseline; a claim taken as a measurement	the lookup measured on the same held-out sales
Asking prices used as labels	recorded sales only; the asking price a feature at most
A label truncated at a reporting threshold	counting the spike; true prices, or out of both sets
The same home on both sides of the split	holding out homes, not sales, before exploring
An error in money, a relative criterion	the metric written first; both forms, by price band
A model where a record or a table answers	rule, table or model per area; the model where it wins
One error quoted for years, in a market it steers	re-measuring after each release; the shown number kept

What we did

1. Spent a day with machine learning: fifty-one applications you already use, from the phone that unlocks to the episode that plays next
2. Framed the problem before touching the data — supervised, multiple univariate regression, batch
3. Checked the assumptions with the downstream team, and found the output is consumed as a *price*, not as a category
4. Looked at the whole dataset once: 207 incomplete districts, a categorical attribute with a five-district level, mismatched scales, skew
5. Found two caps: an income that is not in dollars, and a target where 965 districts — 4.7% — sit at the cap, $500,001, a ceiling instead of a price
6. Stopped before looking any closer: generalisation is measured on cases the model has never seen, so the test set comes before the exploration
7. Asked whether a property marketplace's instant home valuation should be machine learning at all, in nine steps — the objective agreed with each consumer, a lookup measured as the baseline, recorded sales as the label, one look at the table, homes held out before exploring, the metric written beside the criterion, a rule or a table where they win, and the error re-measured after launch — each step named for the pitfall it avoids

NOT PRESENTED — FOR AFTERWARDS

Exercises Lecture 1

five, in the style of the written paper

Exercises — Lecture 1

1 The brief says the output feeds a downstream system that expects a price. Name the one thing this fact settles about the problem framing, and one thing it leaves open. [4]

2 `info()` reports that 207 of the 20,640 districts have no value for `total_bedrooms`. Give the three legitimate ways of handling the incomplete districts, and say which of them still leaves a decision to make. [4]

3 `total_rooms` runs from single digits to tens of thousands, `median_income` from about 0.5 to 15. Name two methods for which this difference in scale matters and two for which it does not, and give the reason for one of the latter. [5]

Solutions on Lecture 2's deck.

Exercises — Lecture 1 (continued)

4 The median income column is capped at 15. Give one consequence for the *model* and one for the *evaluation*, and say which of the two you would raise with the client first. [5]

5 A model's error is low on its training set and high on new districts. Name the condition, and explain why the training error alone could never have revealed it. [4]

Solutions on Lecture 2's deck.

LECTURE 2 · GÉRON CH. 2

The end-to-end project

Data snooping, and a stratified split — measured, not assumed
Exploring the training set: geography, a colour scale, correlations, ratios
Least squares: what fitting a linear model minimises, and why its stationary point is the minimum

Run the Lecture 1 notebook first