



Breaking Into AI

A roadmap for developers who already write code. 60 lessons, one a day, about three months. You will not finish everything, and that is fine — stop at Track 4 and you are already employable.

7 TRACKS • 60 LESSONS • ONE BUILD EVERY WEEK



Before you start

The first language was the hard one

You spent months on JavaScript learning how to think like a programmer. Python takes two weeks, because only the syntax is new. That stays true for every language after this one.

The rule

Build something small every week. Code you wrote badly teaches more than ten lessons you only read.



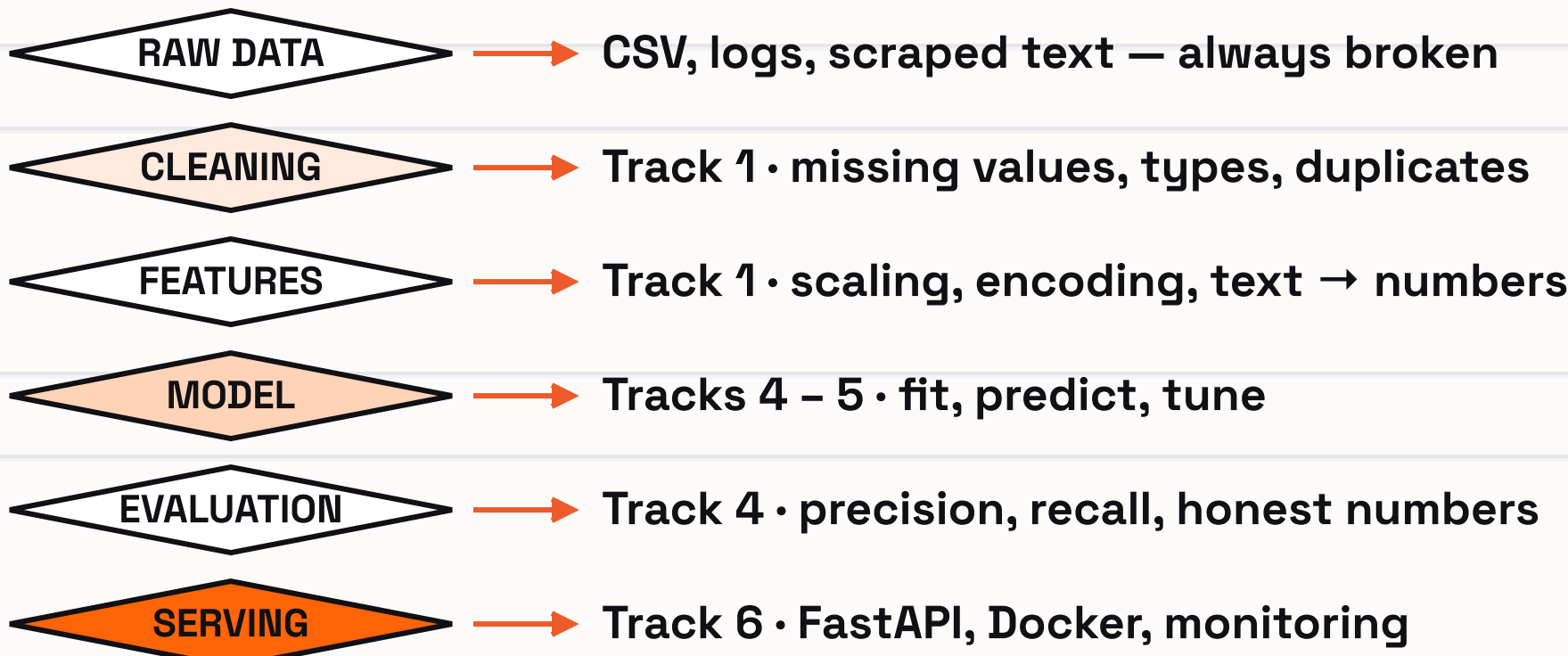
The seven tracks

FOLLOW IN ORDER

TRACK	LESSONS	PACE	WHAT YOU CAN DO AT THE END
0 • JavaScript → Python	1–8	2 weeks	Write any script you can already write in JS
1 • Handling data	9–18	2 weeks	Load, clean and explain any messy dataset — 70% of the job
2 • The maths	19–24	20 min a day, in parallel	Read the equations in a paper without stalling
3 • Statistics	25–29	1 week	Know whether a result means anything
4 • Classical ML	30–42	3 weeks	Get hired. Train, evaluate and explain real models
5 • Deep learning	43–60	4 weeks	Build and train networks; work with LLMs
6 • Shipping it	61–65	1 week	Put a model behind an API your app can call



What the job actually looks like



THE JOB IS THE WHOLE STACK — NOT ONLY THE MODEL LAYER



The tools you will end up using

 Python

 Jupyter

 NumPy

 pandas

 scikit-learn

 PyTorch

 Hugging Face

 Colab

 Kaggle

 FastAPI

 Docker

 Git

XGBoost

matplotlib

seaborn

Start tomorrow. One lesson. Then the next one.

Build for problems near you. There are datasets, languages and use cases in your own community nobody has worked on yet. A model trained on data only you can collect is worth more than a fifth copy of a famous tutorial project.



Python & Data

You are translating, not starting over — then you learn the part of the job everyone skips.

0 From JavaScript to Python

LESSONS 1–8 · 2 WEEKS

Two weeks. Only the syntax is new.

Python

Anaconda

Jupyter

JavaScript

- 1

Setup and why Python
Every AI library, paper and tutorial lives here. Install Python, learn **venv** and **pip** — Python’s npm.
- 2

The syntax map
Indentation instead of braces, **None** not null, no semicolons. Put a JS file next to a Python one and compare.
- 3

Lists, dicts, tuples, sets
Arrays and objects you already know. Tuples (fixed) and sets (unique) are the new ones.
- 4

Loops and comprehensions
`[x*2 for x in nums]` — the feature that makes JS developers like Python.
- 5

Functions and modules
Default and keyword arguments, imports, and how a Python project is laid out.
- 6

Files, JSON and errors
Reading data, **try/except**, and reading a traceback from the bottom up.
- 7

Jupyter notebooks
Run a cell, look, run the next. Notebooks explore; **.py** files ship.
- 8

Project: rewrite an old script
Take something you built in JS and rewrite it in Python.
★ **This turns fear into confidence.**

1 Handling data

LESSONS 9–18 · 2 WEEKS

This is 70% of the real job. Do not skip it to reach neural networks faster.

NumPy

pandas

Plotly

matplotlib

seaborn

- 9

NumPy arrays
The array everything in AI is built on. Shape, axis, dtype.
★ **Most AI bugs are shape bugs.**
- 10

Indexing, slicing, masks
Selecting rows, columns and blocks: `arr[arr > 5]`.
- 11

Broadcasting and vectorization
Operating on whole arrays instead of looping. This is how all AI code is written.
- 12

pandas DataFrames
A spreadsheet in code. Load a CSV, then `.head()` `.info()` `.describe()`.
- 13

Filtering, grouping, merging
`.loc`, `groupby`, and joining two datasets on a key. The daily work.
- 14

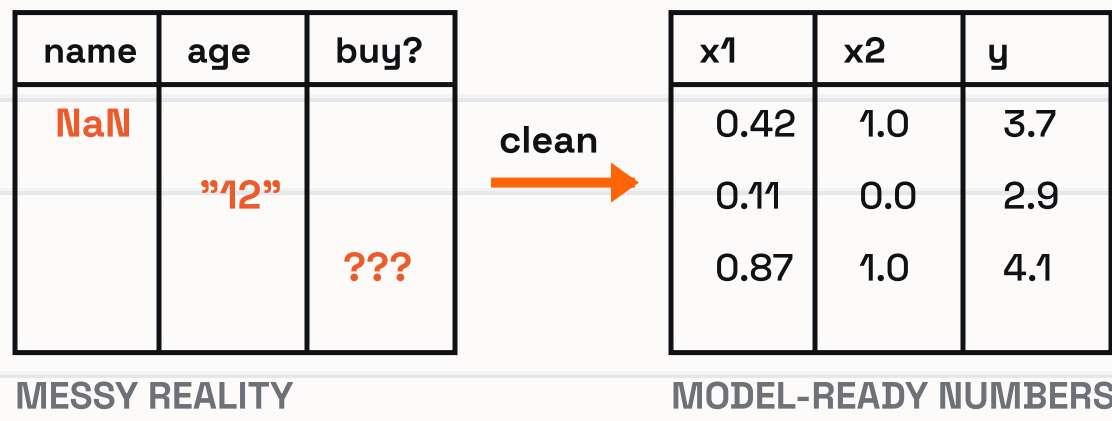
Cleaning messy data
Missing values, duplicates, wrong types, inconsistent spellings.
★ **Real data is always broken.**
- 15

Plotting: matplotlib & seaborn
Histograms, scatter plots, correlation heatmaps. See your data instead of guessing.
- 16

Exploratory data analysis
The routine you run on every new dataset before modelling anything.
- 17

Train/test split & leakage
Why you hide data from yourself, and how future information fakes a great score.
- 18

Scaling, encoding, text → numbers
Words become IDs, categories become one-hot columns, features share one scale.



LESSON 14 · EVERY MODEL STARTS AS A TABLE YOU HAD TO FIX



Maths & Statistics

Run this in parallel, 20 minutes a day. Keep coding while you do it.

2 The maths you need

LESSONS 19–24 • IN PARALLEL

Six ideas. Every one of them shows up in code later.

- 19

Rearranging formulas & composition
f(g(x)) repeated is literally a neural network.
- 20

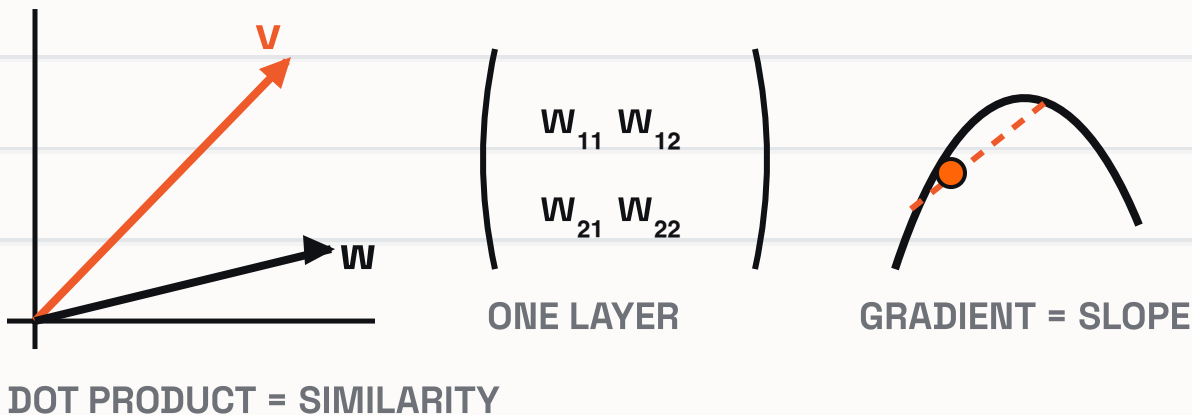
Logarithms and sigma notation
log sits inside almost every loss function; Σ is a sum over your data.
- 21

Vectors and the dot product
A list of numbers, and the one operation that measures similarity.
- 22

Matrices and multiplication
The rule, and why dimensions must line up. One layer of a network.
- 23

Matrices as transformations
A matrix moves every point in space.
★ Linear algebra stops being arithmetic here.
- 24

Derivatives, chain rule, gradient
Slope, nested functions, a vector pointing uphill.
★ The chain rule IS backpropagation.



LESSONS 21–24 • THE THREE PICTURES WORTH MEMORISING

3 Statistics

LESSONS 25–29 • 1 WEEK

How to know whether your result means anything.

- 25

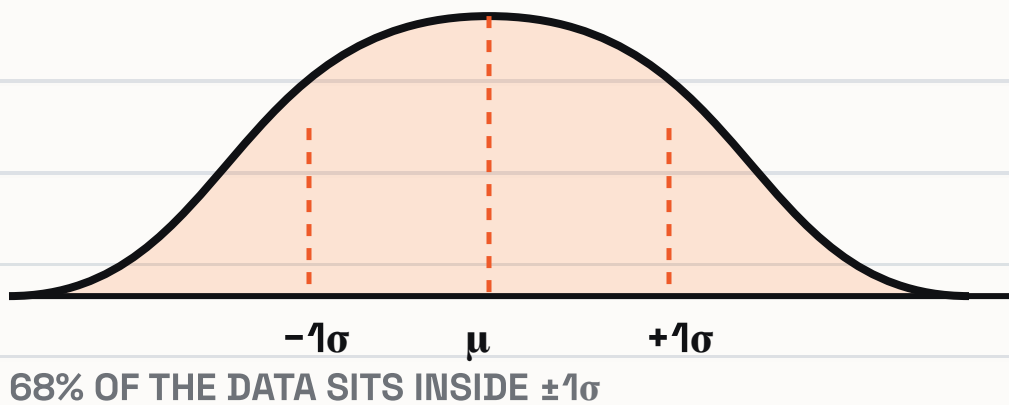
Mean, variance, distributions
Typical value and spread. An average with no spread is half a story.
- 26

The normal distribution
Why the bell curve is everywhere, and what “two standard deviations” means.
- 27

Probability and conditionals
 $P(A \text{ given } B)$ — the foundation of nearly all prediction.
- 28

Bayes’ theorem
Updating a belief when new evidence arrives.
- 29

Correlation $\neq$ causation
Two variables moving together proves nothing — and your sample quietly represents the wrong people.



LESSON 26 • THE SHAPE BEHIND ALMOST EVERYTHING

Why this track sits in the middle

Track 4 is full of numbers that look impressive and mean nothing: an accuracy of 99% on a dataset where 99% of rows are one class, a correlation that is really a coincidence, a great score that came from leaked data. Statistics is what stops you from shipping those.



Classical Machine Learning

Where you become useful. Do not rush this to reach deep learning — most paid work lives right here.

4 Classical machine learning

LESSONS 30-42 · 3 WEEKS

Stop here and you are already employable.

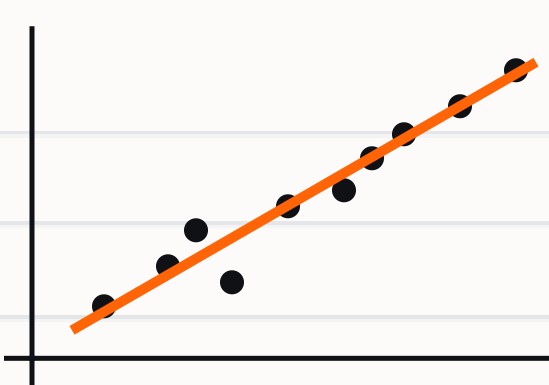
scikit-learn

pandas

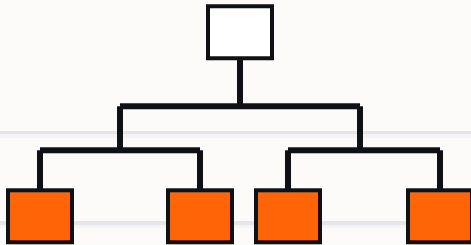
NumPy

XGBoost

LightGBM



REGRESSION · FIT A LINE



TREES · SPLIT, SPLIT, DECIDE

TP	FP
FN	TN

CONFUSION MATRIX · HOW IT FAILS

LESSONS 32, 39, 36 · FIT A LINE, SPLIT THE DATA, THEN LOOK AT HOW IT FAILS

- 30

What ML actually is
Instead of writing rules, you show examples and let the machine find the rule.
- 31

The four organising words
Supervised vs unsupervised, regression vs classification.
- 32

Linear regression & cost functions
Fit a line, then measure how wrong it is. Everything else is a variation.
- 33

Gradient descent in practice
Watch the loss drop, and see what a bad learning rate looks like.
- 34

Logistic regression
Bending a line into a classifier with the sigmoid function.
- 35

Your first scikit-learn model
fit, predict, score.
★ Four lines that will carry you for years.
- 36

Precision, recall, F1, confusion
Accuracy alone is dangerous. These show how your model fails.
- 37

Overfitting and underfitting
Memorising versus learning.
★ The central problem of the entire field.
- 38

Regularization & cross-validation
Penalise complexity; test on several splits so one lucky split cannot fool you.
- 39

Trees, forests, boosting
XGBoost still beats neural networks on most table data.
★ Know this and save months.
- 40

Clustering and PCA
Finding groups with no labels, and squeezing many columns into a few.
- 41

Feature engineering & imbalance
Better inputs beat fancier models. When 99% of rows are one class, accuracy lies.
- 42

Project: end-to-end ML
Real data cleaned by you. Three models, compared honestly, explained in one page.
★ For a non-technical reader.

The habit that gets you hired

Anyone can call `.fit()`. Being able to say why the model is wrong, on which rows, and what it would cost the business — that is the job.

Project 42 is your portfolio

Real data you cleaned yourself, three models compared honestly, and one page a non-technical reader understands. One of these beats ten tutorials.



Deep Learning

Slow down. This is where the modern field lives — and where every LLM you use comes from.

5 Deep learning and neural networks

LESSONS 43–60 • 4 WEEKS

Free GPUs are enough. You do not need expensive hardware.

PyTorch

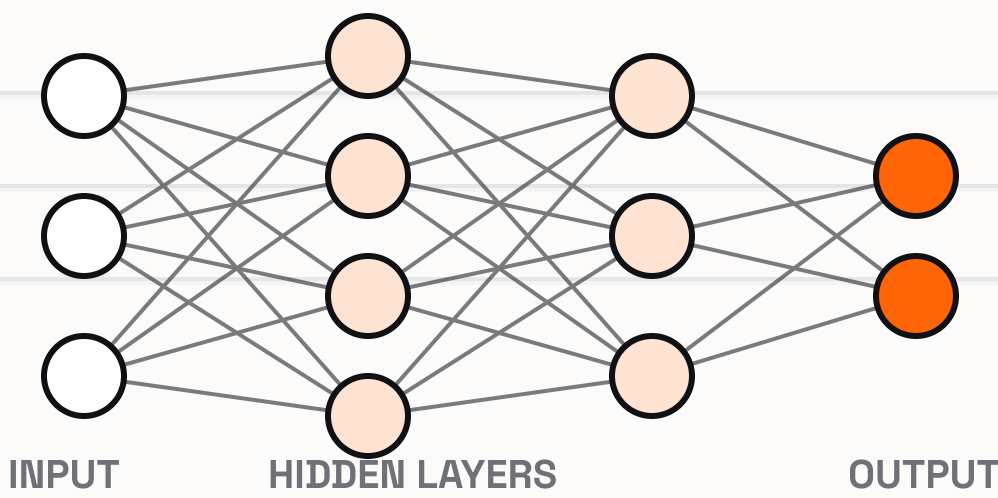
Hugging Face

Colab

Kaggle

NVIDIA GPU

TensorFlow



$$z = w \cdot x + b$$
$$a = \text{ReLU}(z)$$
$$\text{loss} \rightarrow \nabla \rightarrow \text{step}$$

THE LOOP THAT NEVER CHANGES

← BACKPROP: THE ERROR TRAVELS BACK

LESSONS 44–51 • FORWARD, LOSS, ZERO GRADS, BACKWARD, STEP

- 43

When deep learning wins & loses
Wins on text, images, audio, huge data.
Loses on small spreadsheets.
- 44

The neuron and the layer
Weights, bias, sum, activation. Stack them and you have a network.
- 45

Activation functions
ReLU, sigmoid, tanh. Without them the network collapses into linear regression.
- 46

The forward pass
Matrix multiply, add bias, activate, repeat. That is a prediction.
- 47

Loss functions and softmax
Mean squared error for numbers, cross-entropy for categories.
- 48

Backpropagation
Sending the error backwards to see what each weight contributed.
★ The chain rule at work.
- 49

Optimizers and learning rate
SGD, momentum, Adam.
★ The most important setting you will ever tune.
- 50

PyTorch tensors and autograd
NumPy arrays that live on a GPU and remember their own history.
- 51

The training loop, by hand
Forward, loss, zero grads, backward, step.
★ Write it from memory five times.
- 52

Datasets, DataLoaders, GPUs
Batching and shuffling on free Colab or Kaggle GPUs.
★ You do not need expensive hardware.
- 53

Dropout, early stopping, augmentation
The tools that stop a big network from memorising your data.
- 54

Debugging a model that won't learn
Loss flat, exploding, or suspiciously perfect. A checklist you reuse for years.
- 55

CNNs and transfer learning
Adapt someone else's trained model with a few hundred images.
★ This is how small teams compete.
- 56

Embeddings
Turning words, users or products into vectors where similar things sit close.
- 57

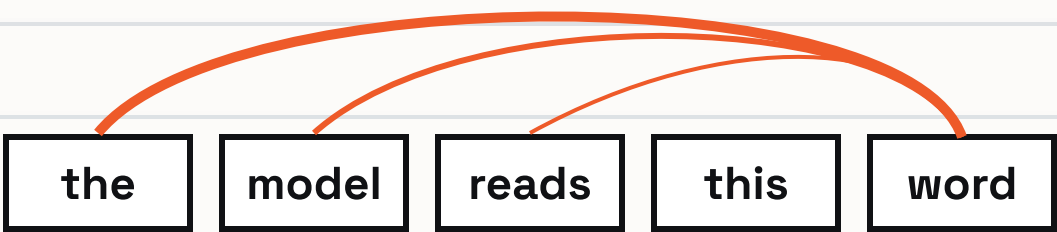
Attention and the transformer
Letting a model choose which parts of the input matter.
★ The engine behind every LLM.
- 58

Tokenization and fine-tuning
How text is chopped up, and adapting a pretrained model.
★ Low-resource languages suffer most — open research.
- 59

Prompting, RAG, fine-tuning
Three levels of working with LLMs. Pick the cheapest one that solves the problem.
- 60

Project: train from scratch
Small PyTorch model, your own cleaned data, no shortcuts. Write down every mistake.

ATTENTION: the last word looks back



THICKER ARC = MORE WEIGHT. THIS IS THE TRANSFORMER.

Lessons 57–59 • where the LLMs come from

Attention lets a model decide which words matter to which. Tokenization decides what it sees in the first place — and it is worst for low-resource languages, Somali included. That is not a limitation to complain about; it is open research with very few people in it.



Shipping It & How To Study

Short, and the part your full-stack background makes easy.

6 Shipping it

LESSONS 61–65 • 1 WEEK

A model nobody can call is a notebook, not a product.

FastAPI

Docker

React

GitHub

Git



PART 6 IS THE PART YOUR FULL-STACK BACKGROUND ALREADY COVERS

- 61

Notebook to script
Turn exploration code into a file that runs end to end without you.
- 62

Serve it with FastAPI
Wrap the model in an HTTP endpoint, then call it from React.
★ This part you already know.
- 63

Deploy it
Docker basics, cheap hosting, and self-hosted model vs calling an API.
- 64

Latency, cost, monitoring
What tutorials ignore and workplaces argue about. Models get worse quietly.
- 65

Responsible deployment
Test for bias, be honest about limits, tell users when they are talking to a machine.

How to study

Daily • 60–90 minutes

- 1 20 min — read the lesson
- 2 30 min — write your own code, without copying
- 3 10 min — break it, then fix it
- 4 5 min — one sentence explaining what you learned

If you cannot explain it in a sentence, you have not learned it.

Weekly & when stuck

-
- Stuck? Go [here](#), never forward. Almost every wall is a gap two lessons earlier.
- Ship the small thing anyway. A finished ugly project beats a perfect unfinished one.

Three warnings

- Watching a course **feels** like progress and is not. For every hour you watch, write an hour of code that was not in the video.
- You will feel stupid regularly. So does everyone in this field, including the confident ones online.
- Ignore most new model announcements. The fundamentals in Tracks 1, 4 and 5 have not changed in a decade.

Resources

Start here

- **Kaggle Learn** — short practical micro-courses for Python, pandas and intro ML
- **Andrew Ng**, Machine Learning Specialization — clearest foundation for Track 4

Then

- **fast.ai**, Practical Deep Learning for Coders — built for programmers, code first
- **3Blue1Brown**, Neural Networks — 3 hours, watch before Track 5
- **Hugging Face NLP Course** — fastest route into transformers and LLMs

Books

- **Hands-On Machine Learning**, Géron — the one book to buy
- **Mathematics for Machine Learning** — free PDF, for Track 2