AI Research Labs

Section 1 of 6 · 5 lessons

Machine Learning Foundations

What machine learning does, how it differs from traditional programming, how models learn from data, and why learning from a finite dataset has limits.

1.1 What is Machine Learning?

1.2 Types of Machine Learning

1.3 The ML Modelling Process

1.4 Fundamental Limits of Machine Learning

1.5 Empirical Risk and Population Risk



Free course with quizzes, progress tracking and Scoopy, an AI tutor:
airesearchlabs.si/learn/ml

Machine Learning Foundations · Edition November 2026 · © 2026 AI Research Labs. Free for personal study.

Where this section fits

Every lesson is a step in building a model. These are the steps this section teaches; the Big picture PDF and the online map show how they connect to the rest.

1

SUPERVISED LEARNING · STEP 1

Frame the problem

What exactly are we predicting, and what decision will it drive?

Lessons here: 1.1, 1.2, 1.3 · feeds into Get, split and prepare the data

◆

UNDERNEATH BOTH MAPS · FOUNDATION

No free lunch

Lessons here: 1.4 · feeds into Choose the model, Choose the model, Evaluate honestly

◆

UNDERNEATH BOTH MAPS · FOUNDATION

Empirical vs population risk

Lessons here: 1.5 · feeds into Evaluate honestly

What is Machine Learning?

Machine learning is learning a prediction rule from data instead of writing it by hand. What that means, how it differs from traditional programming, and why the decision matters more than the prediction.

The one-sentence version

Machine learning (ML) is about making predictions and decisions from data. Instead of a programmer writing the rule, an algorithm *learns* the rule from examples. The term was coined by Arthur Samuel in 1959, whose checkers program learned to play better than he did by adjusting its own scoring rule from experience.

Traditional programming vs machine learning

In traditional programming you write the rules and the computer applies them. In machine learning you supply examples of inputs and the right outputs, and the algorithm finds a rule that maps one to the other.

	TRADITIONAL PROGRAMMING	MACHINE LEARNING
You provide	Rules + data	Data + answers (examples)
Computer produces	Answers	Rules (a fitted model)
Spam filter example	"If the subject contains FREE!!! then spam"	Learn which word patterns separate 100,000 labelled spam and non-spam emails
Breaks when	The world changes faster than you can update the rules	New data stops looking like the training data

ML shines when the rule is too complicated to write down (recognising a face), changes over time (fraud patterns), or must be personalised (recommendations).

A model is a function

Whatever the method, the output of learning is a function f that takes inputs X (also called **features** or **predictors**) and returns a prediction $\hat{y} = f(X)$ of the **target** Y (also called the response, outcome or label). The hat on $\hat{y}$ always means "estimated" or "predicted".

$$\underbrace{X}_{\text{inputs}} \longrightarrow \boxed{f} \longrightarrow \underbrace{\hat{y}}_{\text{prediction}}$$

ML is not one technique. It is a toolbox of models (linear regression, logistic regression, SVMs, K-means, trees, neural networks...) and of methods for fitting and checking them.

Predictions exist to support decisions

A prediction is rarely the end goal. A hospital predicts readmission risk so it can decide who gets a follow-up call; a bank predicts default so it can decide whether to lend. That has a big consequence that runs through this whole course: **the costs of different mistakes matter**. Missing a sick patient is not the same as calling a healthy one, so "most accurate" and "most useful" are not always the same model.

> BIG IDEA
>
> Machine learning is not about fitting the data you have. It is about predicting data you haven't seen yet, well enough to make good decisions.

Where ML sits

ML is part of the broader field of artificial intelligence, and it grew out of statistics and pattern recognition. Statistics and ML share most of their goals but historically differ in emphasis: statistics focuses more on inference (is this effect real?) and ML more on prediction at scale. The two are converging, and this course uses both perspectives: you will compute p-values *and* worry about test-set performance.

> HOW IT'S ASKED
>
> "Define machine learning and explain how it differs from traditional programming." A full answer mentions: learning a rule from data rather than coding it, the model as a function from inputs to outputs, and generalisation to new data as the goal.

Cheatsheet

Machine learning

Learning a prediction rule f from data so that $\hat{y} = f(X)$ predicts Y well on **new** data.

Features vs target

Features / predictors X: the inputs. **Target / response / label** Y: what we predict.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/foundations/what-is-machine-learning

Types of Machine Learning

Regression, classification, clustering and forecasting; supervised vs unsupervised learning; parametric vs non-parametric models. How to tell which kind of problem you are looking at.

Four common tasks

The first question for any ML problem is: **what kind of thing are we predicting?**

TASK	WHAT IS PREDICTED	EXAMPLE	IN THIS COURSE
Regression	A number	House price, CO concentration	Section 3
Classification	A category	Spam or not; diabetic or not	Sections 4 and 6
Clustering	Groups, with no labels given	Customer segments	Section 6
Time-series forecasting	Future values in time order	Electricity demand next month	Time leakage, 2.8

A quick test: if the answer is a quantity you could average (price, temperature), it is regression. If it is a name from a fixed list (yes/no, cat/dog), it is classification.

Features and labels

Every row of a dataset is one observation. The columns you use as inputs are **features** (X). The column you want to predict is the **label** or target (y), the "answer" for that row.

PATIENT	GENE G1	GENE G2	GENE G3	CANCER (LABEL Y)
P1	1	0	1	**1**
P2	0	1	0	**0**

G1 to G3 are features; the cancer column is the label. The model's job is to learn how features relate to the label so it can predict the label for a new patient.

Supervised vs unsupervised learning

- **Supervised learning:** every training example has inputs X *and* a known output Y. We learn a mapping from X to Y. Regression and classification are supervised (linear and logistic regression, SVMs).
- **Unsupervised learning:** we only have X, no labels. We look for structure: groups (clustering), or a compressed representation. K-means is unsupervised.

> RULE OF THUMB
>
> If the dataset has a column you want to predict, it's supervised. If you're asking "what natural groups are in here?", it's unsupervised.

Parametric vs non-parametric models

A **parametric** model has a fixed shape described by a finite list of numbers (parameters). Linear regression is a straight line (or plane) defined by its coefficients $\beta_0, \beta_1, \ldots$. A **non-parametric** model assumes much less about the shape and lets the data decide; K-nearest neighbours predicts by averaging the closest training points.

	PARAMETRIC	NON-PARAMETRIC
Examples	Linear and logistic regression	K-nearest neighbours, decision trees
Data needed	Modest	Lots
Interpretability	High: read the coefficients	Often lower
Main risk	Wrong assumed shape (bias)	Instability, overfitting (variance)

Generative vs discriminative (in brief)

Some approaches model how the data are generated, including the inputs themselves; others focus only on the boundary or relationship needed to predict Y from X. Logistic regression and SVMs are discriminative: they go straight for the decision.

Common doubts

GOING DEEPER **What exactly does "label" mean?**

The label is the **actual answer** for a training row: the value of the target y that the model is trying to predict. Features X are the input information; the label y is the known outcome. In the gene table above, G1, G2 and G3 are features and the cancer column (1 or 0) is the label.

This distinction matters for data leakage later: "validation labels influenced feature selection" means the *answers* of validation rows were used to make a decision, which is not allowed.

FIRST PASS **Does non-parametric mean the model has no parameters?**

No. It means there is no **fixed, finite** set of parameters that defines the model. Linear regression is parametric: whatever the amount of data, it is defined by its coefficients. K-nearest neighbours is non-parametric: it keeps every training point and predicts from the closest ones, so what it must store grows with the data. It still has settings, such as K and the distance measure, but those are hyperparameters you choose.

Cheatsheet

Supervised vs unsupervised

Supervised: have X and Y; learn $X \to Y$ (regression, classification). **Unsupervised:** only X; find structure (clustering).

Parametric vs non-parametric

Parametric (linear/logistic regression): fixed shape, few parameters, less data, interpretable, risk of bias. **Non-parametric** (K-NN): flexible, needs more data, risk of variance.

Regression vs classification

Regression predicts a **number**. Classification predicts a **category**.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/foundations/types-of-machine-learning

The ML Modelling Process

The practical workflow from problem definition to deployment, why it is a loop and not a pipeline, and the difference between an algorithm and its complexity.

The workflow

1. **Define the problem** and the decision the prediction will support. Decide the task type.
2. **Obtain and explore the data.** Identify the target and candidate predictors. Look for missing values, odd units, duplicates.
3. **Split the data** into training, validation and test sets *before* learning anything from it.
4. **Choose candidate models** and fit them on the training data.
5. **Compare them** on validation data, or by cross-validation.
6. **Inspect errors and assumptions.** Go back and revise features, data or model.
7. **Evaluate the final model once** on the test set, then deploy and monitor it.

> A LOOP, NOT A PIPELINE
>
> Step 6 sends you back to earlier steps. Evaluation often reveals that the task definition, the features or the model choice needs to change. What must *not* loop is the test set: it is used once, at the very end.

A revision analogy

Think of studying with past papers. **Training data** are the questions you practise on. **Validation data** are a mock exam you use to decide how to revise. **Test data** are the real exam, which you must not see in advance. If you memorise the practice answers you score 100% on them and may still fail the real exam. That is overfitting.

Algorithm vs complexity

Two different choices hide inside "choosing a model":

	MODEL ALGORITHM	MODEL COMPLEXITY
What it is	The learning technique	How flexible that technique is allowed to be
Examples	Linear regression, logistic regression, decision tree, SVM	Polynomial degree, tree depth, number of features, K in K-NN, penalty λ
Tree example	"Use a decision tree"	"Depth 2" (simple) vs "depth 20" (very flexible)

Settings that control complexity and are chosen by you rather than fitted from the data are called **hyperparameters**. Coefficients are fitted on the training set; hyperparameters are chosen on validation data.

Feature construction

The process includes building new features from domain knowledge: a ratio of debt to income, the day of the week from a timestamp, or X^2 to capture curvature. Task definition, features, model choice and evaluation are connected decisions, not isolated steps.

Common doubts

GOING DEEPER **What is the difference between model algorithm and model complexity?**

The **algorithm** is the learning technique (logistic regression, decision tree, random forest…). The **complexity** is how sophisticated or flexible that particular model is allowed to be.

Example: a decision tree of depth 2 and one of depth 20 use the same algorithm but have very different complexity. The depth-20 tree can carve out tiny regions around individual training points, so it is far more likely to overfit.

Cheatsheet

ML workflow

Define problem → explore data → **split** → fit candidates on train → compare on validation/CV → inspect & revise → test **once** → deploy & monitor.

Hyperparameter

A setting chosen by you, not fitted from data (degree, depth, K, λ, C). Choose it on validation data or by CV.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/foundations/the-ml-modelling-process

Fundamental Limits of Machine Learning

The No Free Lunch theorem, why no algorithm is best for every problem, and the conditions that make learning from data possible at all.

No Free Lunch

The **No Free Lunch theorem** (Wolpert) says that, averaged over *all possible* problems, no learning algorithm beats any other. An algorithm that does brilliantly on one family of problems must do correspondingly worse on another.

WHAT IT MEANS IN PRACTICE

There is no universally best model. A method works when its assumptions suit the problem. So you compare candidate methods on data relevant to your task, and you make your assumptions explicit.

COMMON MISREADING

No Free Lunch does **not** say all algorithms perform equally on your dataset. It is an average over every conceivable problem, most of which look like noise. Real problems have structure, and an algorithm whose assumptions match that structure will do better.

Why learning is possible at all

Training examples alone can never pin down what happens at inputs you haven't seen. Learning works only because of two things:

- **Structure:** the real relationship is reasonably smooth or simple, so nearby inputs have similar outputs.
- **Representative data:** enough observations, drawn from the same population you will predict for.

Every model brings assumptions (its *inductive bias*): linear regression assumes a straight-line relationship, K-NN assumes nearby points behave alike. Those assumptions are what let it fill in the gaps between training points.

The promise behind every model

Most methods assume training and future data are independent draws from the **same** distribution. If you train on patients from one hospital and deploy in another country, that promise is broken, and the model can fail however clever the algorithm.

Two cultures (background)

Leo Breiman's essay *Statistical Modeling: The Two Cultures* contrasts the **data-modelling culture** (assume a stochastic model, check goodness of fit, interpret parameters) with the **algorithmic culture** (treat the mechanism as unknown, judge models by predictive accuracy). Two ideas from it are useful later:

- **Rashomon effect:** many quite different models can fit equally well, so individual coefficients can be unstable (see multicollinearity, 3.9).
- **Occam's razor:** a tension between simplicity and accuracy, which regularisation (Section 5) manages explicitly.

Cheatsheet

No Free Lunch

Averaged over all problems no algorithm wins. In practice: no universally best model, so compare candidates on relevant data and state your assumptions.

NFL misreading

Wrong: "all algorithms do equally well on my data". Right: it is an average over every possible problem; real problems have structure.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/foundations/fundamental-limits-of-machine-learning

Empirical Risk and Population Risk

Population risk is the error we truly care about; empirical risk is the error we can compute. The gap between them is the central problem of machine learning.

The setting

Assume each observation (X_i, Y_i) is an independent draw from some unknown **data-generating distribution** $P(X, Y)$. We never see P itself, only a finite sample of n draws from it. We want a function f whose predictions $f(X)$ are close to Y for *new* draws from the same P.

Loss: scoring a single prediction

A **loss function** $L(Y, f(X))$ measures how bad one prediction is. For regression the usual choice is squared loss $(y - \hat{y})^2$. Choosing the loss is a real modelling decision (lesson 2.1).

Two risks

POPULATION RISK: WHAT WE CARE ABOUT

$$R(f) = \mathbb{E}\big[L(Y, f(X))\big]$$

The average loss over the whole, unknown population.

EMPIRICAL RISK: WHAT WE CAN COMPUTE

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} L\big(Y_i, f(X_i)\big)$$

The average loss over the n observations in our sample.

Because P is unknown, $R(f)$ can't be computed. $\hat{R}_n(f)$ is our estimate of it. On the training data it is the **training error**.

WORKED EXAMPLE: COMPUTING EMPIRICAL RISK

A model predicts $\hat{y} = (8, 5, 10, 3)$ for true values $y = (11, 5, 7, 4)$.

Residuals $y - \hat{y} = (3, 0, -3, 1)$.

Squared-loss empirical risk: $\frac{1}{4}(9 + 0 + 9 + 1) = \frac{19}{4} = 4.75$.

Absolute-loss empirical risk: $\frac{1}{4}(3 + 0 + 3 + 1) = 1.75$.

Same predictions, two different risks, because the two losses punish errors differently.

Why the gap matters

Fitting a model means making $\hat{R}_n(f)$ small. But a flexible enough model can make training error tiny by memorising the sample, noise and all, while its population risk stays large. The difference

$$\text{generalisation gap} = R(f) - \hat{R}_n(f)$$

is the heart of Section 2. We can't compute it exactly, so we estimate $R(f)$ using data the model was not trained on.

HOW IT'S ASKED

"Explain the difference between population risk and empirical risk. Which one can be computed?" Say: population risk is expected loss over the unknown distribution and cannot be computed; empirical risk is average loss on the sample and can. Minimising training loss does not guarantee minimising population risk.

Cheatsheet

Population risk

$R(f) = \mathbb{E}[L(Y, f(X))]$: expected loss over the unknown population. Cannot be computed.

Empirical risk

$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f(X_i))$: average loss on the sample.

Generalisation gap

$R(f) - \hat{R}_n(f)$. Estimated with held-out data.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/foundations/empirical-risk-and-population-risk