# Machine Learning

What Machine Learning actually is -- the difference between training and inference, the three types of learning (supervised, unsupervised, reinforcement), and how overfitting and underfitting can make training go wrong.

**Beginner** 18 min

Last updated: August 22, 2026
Read online: https://www.learnforgex.com/en/topics/machine-learning

---

# Machine Learning

The previous lesson, "What Is Artificial Intelligence?", ended with a map: AI contains Machine Learning, which contains Deep Learning, which is the technique behind most of today's Generative AI. This lesson zooms into the second circle on that map. If AI is the broad idea of "software that performs tasks requiring apparent intelligence," Machine Learning (ML) is the specific *mechanism* most modern AI systems actually use to get there: **learning behavior from data, instead of having it hand-written by a programmer.** By the end of this lesson you'll know what "training" actually means, the three main ways a system can learn, and the two failure modes -- overfitting and underfitting -- that explain why a trained system sometimes gets things wrong.

## What Is Machine Learning?

Machine Learning is the subfield of AI in which a system improves its performance on a task by learning patterns from data, rather than following rules that a human explicitly programmed. Concretely: instead of writing `if (email contains "free money") then spam = true`, you show the system thousands of emails already labeled "spam" or "not spam," and a *learning algorithm* automatically works out which patterns (which words, senders, formatting quirks) tend to separate the two categories. The output of that process is a **model**: a fixed structure (a function/architecture that produces predictions) combined with the learned **parameters** (numbers) inside it. The thing that actually runs the training is called the **learning algorithm** -- it's not the model itself, it's the process that *produces* the model. Once training is done, you're left with a model that no longer needs the learning algorithm to run: it can look at a brand-new, never-before-seen email and predict whether it's spam.

This is the exact idea "What Is Artificial Intelligence?" introduced as "AI is software whose behavior is learned from data or examples." Machine Learning is what that sentence is describing in practice.

## Why Does It Exist?

The previous lesson used the example of "is this photo a picture of a cat?" to show why hand-written rules break down on messy, real-world problems. Machine Learning exists because it turns hand-coding a solution into a different, more tractable problem: **instead of describing the pattern yourself, you collect examples of it and let an algorithm find the pattern for you.** This shift matters for a very practical reason -- for a huge class of problems (recognizing images, understanding language, predicting prices, detecting fraud), nobody has ever managed to write a reliable, explicit rule set by hand, but computers have repeatedly managed to *learn* one from enough examples.

> A useful gut-check: if you could plausibly write the logic yourself as a reasonably sized set of `if` / `else` rules, you probably don't need machine learning -- plain code will be simpler, faster, and easier to debug. Save ML for problems that aren't easily solved with clear, stable rules, where learning patterns from data actually provides an advantage.

## Training vs. Inference

Every machine learning system's life has two distinct phases, and confusing them is one of the most common sources of misunderstanding for newcomers:

- **Training** is the (often slow, computationally expensive) process of showing a model many examples and adjusting its internal parameters so its predictions get better and better. This happens *once* (or periodically, when you want to update the model), usually offline, sometimes taking minutes, hours, or -- for the large models covered later in this course -- weeks on specialized hardware.
- **Inference** is using an *already-trained* model to make a prediction on new input. This is what happens every time a spam filter checks an incoming email, or a phone unlocks with face recognition. Inference is typically fast (milliseconds) and is what end users actually experience.

Think of training as "studying for an exam" and inference as "taking the exam using what you studied." A model that performs its training phase well but is never actually used for inference is useless in practice -- and a model that's fast at inference but was trained on bad data will confidently produce bad predictions.

## Three Types of Learning: Supervised, Unsupervised, Reinforcement

Not all machine learning works the same way. There are three broad categories, distinguished by what kind of data the system learns from and what "feedback" (if any) it gets:

- **Supervised Learning:** the system learns from data that already has the "correct answer" attached -- for example, photos already labeled "cat" or "not cat," or houses with their known sale price. The system's job is to learn the mapping from input to correct output, so it can predict the answer for new, unlabeled inputs. This is by far the most common and most mature category, and it's what the spam filter example above describes.
- **Unsupervised Learning:** the system is given data with **no** correct answers attached, and has to find structure in it on its own -- for example, grouping customers into segments based on purchasing behavior, without anyone telling it in advance what the segments should be (a technique called *clustering*).
- **Reinforcement Learning:** the system (called an *agent*) learns by taking actions in an environment and receiving a *reward* or *penalty* as feedback, gradually learning which actions lead to better outcomes over time -- the way a game-playing AI learns strategy by playing millions of games against itself. This category is conceptually the closest to the "AI Agents" category later in this course, though the agents you'll build there use a different, more modern set of techniques built on large language models rather than classic reinforcement learning.

## Features and Labels: What a Model Actually Learns From

Two terms come up constantly once you look inside any supervised learning system:

- A **feature** is a measurable piece of input information the model uses to make its prediction -- for an email spam filter, features might include the number of exclamation marks, whether the sender is a known contact, or the presence of certain words. Choosing and preparing good features (a process called *feature engineering*) used to be one of the most time-consuming and skill-dependent parts of building an ML system, especially for the classical ML techniques covered in this lesson, where features largely had to be designed by hand. You'll see in the later "Deep Learning" lesson that this isn't equally true of every ML technique.
- A **label** is the correct answer attached to a training example -- "spam" or "not spam," a house's actual sale price, a photo's correct category. Labels are what make supervised learning "supervised": a human (or some automated process) had to supply them in advance.

A **dataset**, then, is simply a large collection of examples, each with its features (and, for supervised learning, its label). The quality and size of this dataset is often *more* important to a model's final performance than which specific learning algorithm is used -- a theme this course will return to.

## Overfitting and Underfitting

Once a model is trained, how do you know if the training actually worked *well*? Two named failure modes describe the most common ways it can go wrong:

- **Overfitting** happens when a model learns the training data *too* precisely -- including its noise and quirks -- rather than the general pattern underneath it. An overfit model gets excellent scores on the exact examples it was trained on, but performs poorly on new, unseen data, because it effectively "memorized the answers" instead of learning the underlying rule.
- **Underfitting** is the opposite problem: the model is too simple (or wasn't trained enough) to capture even the real pattern in the data, so it performs poorly on both the training data and new data.

The practical fix for overfitting almost always involves evaluating a model on data it has never seen during training (see the next section) and using techniques -- more training data, simpler models, or deliberately limiting how tightly the model fits -- to keep it generalizing well rather than memorizing.

## How Do We Know a Model Is Good?

The standard practice is to split the available dataset into three parts before training even begins:

- **Training set:** the examples the model actually learns from.
- **Validation set:** examples held back from training, used while making model or hyperparameter choices (for example, deciding which architecture or which settings perform better).
- **Test set:** examples that are never used in training or in those choices -- held back completely, used exactly once to honestly measure the model's *final* performance after every decision has been made.

The reason for three-way split, not two, is simple: if you use the validation set to pick a model, your choice becomes indirectly tuned to that data too -- so you still need a genuinely untouched test set. The point of this lesson isn't to go deep into how validation is done, just to build the right mental model: the training set learns, the validation set picks, the test set gives the final, honest score. A model that scores well on the training set but poorly on the test set is a textbook symptom of overfitting.

> A model's training-set score is *not* a trustworthy measure of real-world quality -- it's almost always higher than the model's true performance, precisely because the model has seen those exact examples before. Whenever you read a claim about how "accurate" an AI system is, the important question is always: accurate on *what data*, that the model never saw during training?

This course's later category on AI Evaluation goes much deeper into how modern systems (especially large language models) are actually measured -- this lesson only needs you to understand the basic training/test split that underlies all of it.

## From Machine Learning to Deep Learning: What's Next

Everything in this lesson applies to machine learning broadly, regardless of which specific algorithm is doing the learning -- and historically, there have been many: decision trees, linear regression, support vector machines, and others, most of which predate the recent AI boom by decades. The next lesson, "Deep Learning," zooms into one specific family of learning algorithms -- **neural networks**, and specifically *multi-layered* ones -- that turned out to scale dramatically better than older techniques once enough data and computing power became available (recall the "What Is Artificial Intelligence?" lesson's mention of the 2012 turning point). Deep learning didn't replace the concepts in this lesson -- training, inference, supervised learning, overfitting -- it's a specific, extremely successful *technique for doing them*, and everything you've just learned still applies to it directly.

## Best Practices

- Before reaching for machine learning, check whether the problem can be solved with plain, explicit code (see the "Why Does It Exist?" tip) -- ML adds real cost (data collection, training infrastructure, ongoing monitoring) that isn't justified for problems with clear, stable rules.
- Always evaluate a model on data it did not train on. A number without a clear answer to "measured on what test data?" should not be trusted.
- Treat your dataset as seriously as your code. A well-tuned learning algorithm trained on bad, biased, or too-small data will reliably produce a bad, biased, or unreliable model -- garbage in, garbage out applies here more than almost anywhere else in software.

## Common Mistakes

- **Judging a model only by its training performance.** As covered in "Overfitting and Underfitting," a model can score close to perfect on training data while performing badly on real, new inputs. Always ask about test-set performance specifically.

- **Assuming "machine learning" means "neural networks."** As the next lesson will make clear, neural networks (deep learning) are one family of ML techniques among several -- widely used today, but not synonymous with the field.
- **Treating reinforcement learning and supervised learning as interchangeable.** They solve very different kinds of problems (labeled prediction vs. learning through trial-and-error feedback) -- see "Three Types of Learning: Supervised, Unsupervised, Reinforcement" -- and mixing up which one a system actually uses leads to wrong expectations about how it can be trained or improved.

## Summary, Cheat Sheet, and Glossary

### Summary

- Machine Learning is the subfield of AI where systems learn behavior from data rather than from hand-written rules -- it's the specific mechanism behind the general idea introduced in "What Is Artificial Intelligence?"
- Every ML system has two phases: **training** (learning from examples, usually slow) and **inference** (using what was learned, usually fast).
- There are three broad types of learning: **supervised** (learns from labeled examples), **unsupervised** (finds structure with no labels), and **reinforcement** (learns from reward /penalty feedback).
- **Overfitting** (memorizing training data too precisely) and **underfitting** (failing to capture the real pattern) are the two main ways training can go wrong -- caught by evaluating on a held-out test set.
- Deep Learning (next lesson) is a specific, highly successful family of ML techniques based on neural networks -- not a replacement for the concepts in this lesson, but a technique for applying them at scale.

### Cheat Sheet

- Model = architecture (the function that produces predictions) + learned parameters. Learning algorithm = the training process that *produces* the model, not the model itself.
- Training = learning from examples. Inference = using what was learned.
- Feature = an input the model uses. Label = the correct answer (supervised learning only).
- Training set = the model learns. Validation set = model/hyperparameter choices get made. Test set = final performance gets honestly measured.
- Overfit = too closely matches training data, fails on new data. Underfit = too simple, fails on everything.
- Always evaluate on a test set the model never trained on.

### Glossary

- **Machine Learning (ML):** the subfield of AI where systems learn behavior from data instead of explicit rules.
- **Model:** the trained output of a learning process -- parameters plus an algorithm for using them to make predictions.
- **Training / Inference:** the two phases of an ML system's life -- learning from data, and then using what was learned on new input.
- **Supervised / Unsupervised / Reinforcement Learning:** the three broad categories of machine learning, based on what kind of data and feedback the system learns from.
- **Feature / Label:** an input the model uses to predict / the correct answer attached to a training example.
- **Overfitting / Underfitting:** learning the training data too precisely to generalize / failing to learn the real pattern at all.
- **Training set / Validation set / Test set:** the data a model learns from / the held-back data used when making model or hyperparameter choices / the untouched data used exactly once to honestly measure how well the model actually performs.