# Decision Trees

A decision tree is a flowchart of yes/no questions that ends in a prediction. How to

## Introduction

Some models are black boxes: they give an answer, and you cannot easily say why. A **decision tree** is the opposite. It is a flowchart of yes/no questions, and you can follow it with your finger from the top to the answer. That is why trees are the favourite first model for teaching — and for any job where someone will ask "why did it decide that?"

Picture a teacher deciding whether a student will pass. First question: did the student study 5 hours or more? If yes, follow one branch; if no, follow the other. On each branch sits another question — did they attend class regularly? — and so on, until you reach a leaf: a final prediction, Pass or Fail. The tree learned those questions from the training data: at each step it picked the question that split the students most cleanly into pass and fail groups.

Trees have one weakness, and it is worth meeting early: a tree that keeps asking questions until every single training student sits alone in their own leaf has memorised the class, not learned the pattern. That tree is deep, brittle, and fails on new students. The fix — limiting how deep the tree may grow — is the doorway to lesson ML.7.

How does the tree choose its questions? At each node it tries candidate splits — "hours 5 or more?", "attended regularly?" — and measures how clean the two sides are. A split is clean when each side holds mostly one class: mostly passes on one side, mostly fails on the other. The tree picks the cleanest split it can find, then repeats the same game inside each side. The plain name for "cleanliness" is purity, and the usual measure is called Gini or entropy — you do not need their maths now, only the idea: the tree prefers splits that separate the classes most sharply.

This also explains a tree's second superpower: it can use features in combination. "Studied 5+ hours" alone may not separate the class, and "attended regularly" alone may not either, but the pair — first one question, then the other on each branch — can. That is why trees often beat a single rule, and why reading the path from root to leaf tells you the exact combination of reasons behind one prediction.

*(Roman Urdu: Decision tree aik flowchart hai — har mor par aik haan/naa sawal, aur aakhir mein jawab. Is ko ungli rakh kar parha ja sakta hai, is liye is ki wazahat asaan hai.)*

## Key points

- **A decision tree is a flowchart of questions.** Each inner node asks a yes/no question about one feature; each branch is an answer; each leaf is a final prediction.
- **The tree learns its questions from data.** At each node it picks the split that separates the classes (or values) most cleanly.

- **Reading a tree needs no maths.** Follow the questions from the root to a leaf, and you can explain any prediction in plain sentences.
- **Trees handle both jobs.** Leaves can hold a category (classification) or a number (regression) — the shape of the tree is the same.
- **Trees are easy to explain but easy to overgrow.** A very deep tree fits every quirk of the training data and fails on new data. This is overfitting, covered in lesson ML.7.
- **Depth is the main control.** Limiting the tree's depth (how many questions deep it may go) keeps it simple and general.
- **One tree can be unstable.** Change a few training rows and the tree's questions can change completely. Later lessons in your journey will show how forests of trees fix this.
- **At each node the tree picks the cleanest split.** Clean means each side holds mostly one class; the tree tries candidate questions and keeps the one that separates the classes most sharply.
- **A leaf's prediction is the majority class in that leaf.** If a leaf holds 8 passes and 2 fails, it predicts Pass — and that 8-to-2 split also tells you how sure it is.
- **Feature importance can be read from the tree.** Features asked near the root, splitting many examples cleanly, matter more than features asked deep down on tiny branches.

## Examples

### Example 1 — Read a tiny tree by hand

A tree trained on students, predicting Pass or Fail:

```
Studied 5+ hours?
├── YES → Attended class regularly?
│         ├── YES → Pass
│         └── NO  → Fail
└── NO  → Fail
```

Follow it for three students. A student who studied 7 hours and attended regularly: YES, YES → Pass. A student who studied 7 hours but skipped class: YES, NO → Fail. A student who studied 2 hours: NO → Fail, and the tree never even asks about attendance. Every prediction comes with its own plain-words explanation — that is the tree's superpower.

### Example 2 — The same tree in code (scikit-learn)

This example needs scikit-learn installed first: `pip install scikit-learn`. It uses only the standard `DecisionTreeClassifier` on a tiny hand-made table:

```
from sklearn.tree import DecisionTreeClassifier

# features: [hours studied, attended regularly (1 = yes, 0 = no)]
X = [[7, 1], [7, 0], [2, 0], [3, 1], [8, 1], [1, 0], [6, 1], [4, 0]]
y = [1, 0, 0, 0, 1, 0, 1, 0]  # 1 = Pass, 0 = Fail

model = DecisionTreeClassifier(max_depth=2, random_state=42)
model.fit(X, y)
print("Prediction for 7 hours, attended:", model.predict([[7, 1]]))
print("Prediction for 2 hours, absent:", model.predict([[2, 0]]))
```

What the result means: the model learns a small tree (at most 2 questions deep, thanks to `max_depth=2`) and predicts Pass for the regular attendee and Fail for the low-hours absentee. Limiting depth is the simplest defence against the overgrown, memorising tree.

### Example 3 — A tree that grew too deep

Give the same tree no depth limit and only 8 students, and it can keep splitting until each leaf holds exactly one student — perfect training score, and a tree that has learned quirks like "the student who studied exactly 4 hours and skipped class failed, so all such students fail". On a new class, that quirk fails. A shallow tree that gets 7 out of 8 right has learned a pattern; a deep tree that gets 8 out of 8 has memorised a list.

### Example 4 — Worked example: a fruit tree, traced by hand

A tree sorts fruit into Apple or Orange using two features: colour and weight:

```
Red?
├── YES → Heavy (over 150 g)?
│         ├── YES → Apple
│         └── NO  → Cherry (not in our two labels — the tree must still choose, so it says
Apple, the majority)
└── NO  → Orange
```

Trace three fruits. A red apple weighing 180 g: YES, YES → Apple, correct. A red cherry weighing 20 g: YES, NO → the tree says Apple, wrong — this leaf mixes small red fruit, and the tree has no better question left for it. An orange weighing 200 g: NO → Orange at once, weight never asked. The lesson is practical: a tree is only as good as its features. Give it a feature that separates cherries from apples — for example, "has a stone?" — and the mistake disappears. When a tree makes a silly error, first ask whether it was ever given the feature that would have prevented it.

### Example 5 — Worked example: reading a tree back in code (scikit-learn)

This example needs scikit-learn installed first: `pip install scikit-learn`. It trains the tree from Example 2 and prints it as plain text, so you can read the exact questions it learned:

```
from sklearn.tree import DecisionTreeClassifier, export_text

X = [[7, 1], [7, 0], [2, 0], [3, 1], [8, 1], [1, 0], [6, 1], [4, 0]]
y = [1, 0, 0, 0, 1, 0, 1, 0]  # 1 = Pass, 0 = Fail

model = DecisionTreeClassifier(max_depth=2, random_state=42)
model.fit(X, y)
print(export_text(model, feature_names=["hours", "attended"]))
```

Run it and compare the printed tree with the hand-drawn tree in Example 1. The questions may not match word for word — the tree picks the cleanest split it finds, and with only 8 rows, small differences change the split — but the shape should be similar: hours first, then attendance on one branch. If the printed tree surprises you, that is useful information: it means your mental model of the data and the data itself disagree, and the data is worth a second look.

### Real-world case — Why hospitals and banks still love trees

When a hospital model flags a patient as high risk, a doctor will ask "why?" — and a decision tree can answer in plain sentences: "because the patient is over 65 and has a fever above 39 degrees". A black-box model cannot answer that question nearly as cleanly. The same applies in lending, where a refused applicant has a right to an explanation. That explainability is why trees, and forests of trees, remain in daily use in medicine and finance even though newer models sometimes score a little higher. In jobs where the decision affects a person's health or money, being able to explain the decision is part of the model's job, not a bonus.

## Quick recap

- A decision tree is a flowchart: questions at nodes, predictions at leaves.
- You can read and explain any prediction by following the branches.
- The tree picks, at each node, the split that separates the data most cleanly.
- Deep trees memorise training data and fail on new data — limit the depth.
- Trees work for both classification (a label) and regression (a number).

## Common mistakes

1. **Letting the tree grow without a depth limit** — perfect training score, poor real results. Fix: set `max_depth` to a small number (2 or 3 to start) and compare test scores.
2. **Trusting a tree's explanation too far** — the tree explains its own logic, but that logic may be built on quirks. Fix: check the tree against the test set, not just its own story.
3. **Forgetting that small data changes can reshape the tree** — one new row can change the top question. Fix: treat a single tree as a starting model, not a final truth.

1. **Reading a leaf as a certainty** — a leaf with 6 passes and 4 fails is not sure, it is just leaning. Fix: check how many examples sit in the leaf, and treat mixed leaves as uncertain.

2. **Using a tree on features it never saw** — a tree trained on hours and attendance cannot use a new feature like "sleep hours" until it is retrained. Fix: retrain when the features change, and compare the new tree with the old one.

## Practice

1. Draw the tree from Example 1 on paper and follow it for a student who studied 9 hours and did not attend class. What is the prediction?
2. Change one label in the Example 2 data (make the [3, 1] student a Pass) and think: how might the tree's questions change?
3. Run the code in Example 2 with `max_depth=3` instead of 2, then with no `max_depth` at all. Compare the training scores and write one line about what you notice.

1. Draw the fruit tree from Example 4, then add a third question that fixes the cherry mistake. Show the new tree and trace the cherry through it.
2. Run the code in Example 5 with `max_depth=3`. Does the printed tree gain a new question? Write one line about whether that question looks like a real pattern or a quirk of 8 rows.
3. Think of a decision you make by flowchart in daily life (for example, "should I take an umbrella?"). Draw it as a decision tree with at least three questions and two leaves that predict the same answer by different paths.

## Practice quiz

**1. How does a decision tree decide?**

- By guessing
- By asking yes/no questions in order, following a branch to an answer
- By voting
- By drawing a chart

**Answer:** By asking yes/no questions in order, following a branch to an answer

**2. Why are decision trees easy to explain?**

- They are small
- You can follow the exact path of questions to the answer
- They use no maths
- They never make mistakes

**Answer:** You can follow the exact path of questions to the answer

**3. What is the risk of a very deep tree?**

- It runs slowly
- It memorises the training data — overfitting

- It forgets the data
- It cannot ask questions

**Answer:** It memorises the training data — overfitting

---

**Watch the video lesson:** https://youtu.be/uU16-qr1fvo

**Read online:** https://scopetreasure.com/notes/ml-decision-trees · More free tools and notes at https://scopetreasure.com