Classification

Classification predicts a category, not a number — pass or fail, spam or not spam. A

Introduction

Some predictions are choices between categories: is this email spam or not, will this student pass or fail, is this photo a cat or a dog? These are **classification** jobs. The answer is a label from a fixed list, not a number on a scale. Lesson ML.4 predicted numbers; this lesson predicts categories, and the change matters more than it looks.

The most common beginner classification model is called **logistic regression**. The name is confusing — despite "regression" in its name, it is a classification model, not a number predictor. You only need to know its name and its job for now: it learns a boundary between categories and predicts which side a new example falls on. It does not draw a straight line through prices; it draws a dividing line between groups.

Then comes the trap. The usual score for classification is **accuracy**: the share of predictions that were correct. Accuracy sounds like the whole story, and often it is not. If 95 out of 100 emails are not spam, a lazy model that says "not spam" every single time scores 95% accuracy — and catches zero spam. A single number can hide a useless model. This lesson teaches you to look past it.

Two ideas make classification much clearer once you see them: probability and threshold. Most classifiers do not just output "Pass" or "Fail" — they first output a chance, such as "70% chance of Pass". The final label comes from a **threshold**: by default, 50% or more means Pass, below 50% means Fail. That threshold is a choice, not a law of nature, and changing it changes the mistakes you make. Lower the threshold to 30% and the model passes more students — catching more real passes, but also passing some who will fail. Raise it to 70% and the opposite happens.

This is why "which mistakes matter more?" is the real question in classification. For a spam filter, wrongly binning a real email hurts more than letting one spam through, so you might raise the bar for calling something spam. For an illness screening test, missing a sick patient hurts most, so you lower the bar for flagging. The model gives you the chances; you, knowing the real costs, set the bar. A classifier without that conversation is only half a solution.

(Roman Urdu: Classification mein jawab aik category hota hai — pass ya fail, spam ya nahi. Accuracy kabhi kabhi dhoka deti hai, is liye sirf aik number par bharosa mat karo.)

Key points

- **Classification predicts a category.** The answer comes from a fixed set of labels: spam/not spam, pass/fail, cat/dog.
- **Binary classification has two labels; multi-class has more.** Spam filtering is binary; sorting photos into cat, dog and bird is multi-class.

- **Logistic regression is a classification model, despite its name.** It learns a boundary between categories. You will meet its full maths much later; the name and the job are enough now.
- **Accuracy = correct predictions ÷ all predictions.** Simple, useful, and sometimes dangerously misleading.
- **Accuracy misleads when the classes are unbalanced.** If one class dominates the data, always guessing that class gives a high score while learning nothing.
- **Always ask "which mistakes does it make?"** Missing spam is one kind of error; marking a real email as spam is another. The two mistakes do not hurt equally.
- **Report accuracy together with the mistakes.** Lesson ML.7 shows the confusion matrix, the plain-words table that makes those mistakes visible.
- **Most classifiers output a probability first, then a label.** The label comes from comparing that probability with a threshold, usually 50%.
- **Moving the threshold trades one mistake for another.** A lower threshold catches more of the positive class but raises false alarms; a higher threshold does the reverse.
- **Multi-class classification just adds more labels.** Sorting photos into cat, dog and bird uses the same idea: the model picks the label with the highest chance.

Examples

Example 1 — Tiny pass/fail example, by hand

Eight students, hours studied and the true result:

Hours	True result
1	Fail
2	Fail
3	Fail
4	Fail
6	Pass
7	Pass
8	Pass
9	Pass

A simple rule — "5 hours or more means Pass" — gets all 8 right: accuracy 8 ÷ 8 = 100%. On this clean toy data the boundary is obvious. Real data is messier: a student who studied 6 hours can still fail, and the model must weigh the odds, not find a perfect wall.

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

This example needs scikit-learn installed first: `pip install scikit-learn`. It uses only the standard `LogisticRegression` model on the tiny table above:

```
from sklearn.linear_model import LogisticRegression

hours = [[1], [2], [3], [4], [6], [7], [8], [9]]
result = [0, 0, 0, 0, 1, 1, 1, 1]  # 0 = Fail, 1 = Pass

model = LogisticRegression()
model.fit(hours, result)
print("Prediction for 5 hours:", model.predict([[5]]))
print("Training score:", model.score(hours, result))
```

What the result means: the model learns the boundary between Fail and Pass, predicts for a 5-hour student (the borderline case), and reports its training score. Note carefully: this score is on the training data, so it is not proof the model works on new students — for that you need a held-out test set, as in lesson ML.3.

Example 3 — Accuracy misleading you

A clinic tests 100 people, of whom 5 actually have a rare illness and 95 do not. A model that predicts "healthy" for everyone scores 95% accuracy — impressive-looking, and completely useless, because it finds zero sick patients. The model's real job was finding the 5, and accuracy hid that failure. Whenever one class is rare, accuracy alone is the wrong report card.

Example 4 — Worked example: probabilities and the threshold, by hand

A model gives these pass chances for five students:

| Student | Hours | Chance of Pass | Label at 50% threshold | |---|---|---|---| | A | 2 | 10% | Fail | | B | 4 | 40% | Fail | | C | 5 | 55% | Pass | | D | 6 | 75% | Pass | | E | 8 | 95% | Pass |

Now suppose the true results come in: A fails, B passes, C passes, D passes, E passes. At the 50% threshold the model gets 4 out of 5 right — it only missed B, the borderline student at 40%. Lower the threshold to 35% and B becomes a Pass too, giving 5 out of 5 — but that same lower threshold would also pass a future student at 36% who actually fails. There is no free win: every threshold choice buys one kind of correctness with the other kind of mistake. The table, not the single accuracy number, is what lets you see that trade.

Example 5 — Worked example: probabilities in code (scikit-learn)

This example needs scikit-learn installed first: `pip install scikit-learn`. It trains the same logistic regression as Example 2, then asks for chances instead of labels:

```
from sklearn.linear_model import LogisticRegression

hours = [[1], [2], [3], [4], [6], [7], [8], [9]]
result = [0, 0, 0, 0, 1, 1, 1, 1]  # 0 = Fail, 1 = Pass

model = LogisticRegression()
model.fit(hours, result)
print("Chances for 4, 5 and 6 hours:")
print(model.predict_proba([[4], [5], [6]]))
```

`predict_proba` returns, for each student, the chance of Fail and then the chance of Pass. Run it and read the middle row carefully: the 5-hour student sits near the boundary, so the chances are close to even — that is the model honestly saying "I am unsure here". A good habit is to treat near-50% predictions as "unsure, check by hand" rather than as confident answers. Try predicting for 3 hours and 8 hours too, and notice how the chances move away from 50% as you move away from the boundary.

Real-world case — How a bank decides a loan application

A bank's loan model is a classifier: approve or refuse. Its features include income, past loan history and current debts; its label is whether past borrowers repaid or defaulted. The model outputs a chance of

repayment, and the bank sets the threshold by business judgement: a low threshold approves more people but accepts more defaults; a high threshold is safer but turns away good borrowers. This is also why banks must explain refusals — a bare "computer says no" is not acceptable when someone's livelihood is at stake, which is one reason explainable models like decision trees (lesson ML.6) stay popular in lending even when fancier models score slightly higher.

Quick recap

- Classification predicts a category from a fixed list of labels.
- Logistic regression is a classification model — the name is a historical accident.
- Accuracy = correct ÷ total. Easy to compute, easy to misuse.
- With unbalanced classes, a lazy model can score high accuracy and learn nothing.
- Always look at which mistakes a model makes, not just how many.

Common mistakes

1. **Reporting accuracy alone on unbalanced data** — 95% can mean "found nobody". Fix: also report the mistakes per class (the confusion matrix in lesson ML.7).
2. **Using a regression model for a category job** — predicting "pass = 1.4" makes no sense. Fix: use a classifier like logistic regression for labels.
3. **Judging the model on training data only** — a perfect training score is not a real result. Fix: split train/test first, and judge on the test set.

1. **Treating a 51% prediction as confident** — a prediction just over the threshold is nearly a coin flip. Fix: report the probability, and flag near-threshold cases for a human check.
2. **Forgetting that the threshold is a choice** — accepting the default 50% without asking what each mistake costs. Fix: decide the threshold from the real costs of false alarms versus misses, not from habit.

Practice

1. In Example 1, change the rule to "4 hours or more means Pass". Recompute accuracy by hand. What changed?
2. Make up your own 10-email spam example with 9 not-spam and 1 spam. What accuracy does an "always not spam" model get? Write one sentence about why that score lies.
3. Run the code in Example 2 and predict for 4 hours and 6 hours. Do the answers match the rule from Example 1?

1. Using the table in Example 4, what happens to accuracy if the threshold is raised to 80%? Work it out row by row and write one sentence about the trade you just made.
2. Make up a classification problem where a false alarm is worse than a miss (for example, flagging a genuine customer as a fraudster). Write down which threshold direction you would choose, and why.

3. Run the code in Example 5 and write down the Pass chance for 5 hours. Is the model confident or unsure there? Answer in one sentence.

Practice quiz

1. What does classification predict?

- A number
- A category, like pass or fail
- A group of similar rows
- A chart

Answer: A category, like pass or fail

2. What is accuracy?

- How fast the model runs
- The share of predictions the model got right
- The number of features used
- How much data was used

Answer: The share of predictions the model got right

3. Why can accuracy mislead?

- It is hard to calculate
- With imbalanced classes, a lazy model that always guesses the common class still scores high
- Accuracy ignores the test set
- It only works for numbers

Answer: With imbalanced classes, a lazy model that always guesses the common class still scores high

---

Watch the video lesson: https://youtu.be/cdRwAxypS5k

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