AI Research Labs

Section 4 of 6 · 11 lessons

Logistic Regression and Classification

How classifiers produce probabilities, how probabilities become decisions, and how to evaluate classification beyond accuracy.

4.1 Regression versus Classification

4.2 Odds, Log-odds and the Sigmoid Function

4.3 The Logistic Regression Model

4.4 Maximum Likelihood Estimation

4.5 Cross-entropy / Log-loss

4.6 The Confusion Matrix

4.7 Accuracy, Precision, Recall and Specificity

4.8 Sensitivity and Specificity Trade-offs

4.9 Threshold Selection and Classification Costs

4.10 ROC Curve and AUC

4.11 Class Imbalance



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.

?

SUPERVISED LEARNING · THE FORK

A number or a category?

Is the target a quantity, or one of a few classes?

Lessons here: 4.1 · feeds into Choose the model, Choose the model

C1

SUPERVISED LEARNING · CLASSIFICATION LANE · STEP C1

Choose the model

How do I turn inputs into a class, or a probability of one?

Lessons here: 4.2, 4.3 · feeds into Turn predictions into decisions

C2

SUPERVISED LEARNING · CLASSIFICATION LANE · STEP C2

Fit it

Which coefficients make the observed labels most probable?

Lessons here: 4.4, 4.5 · feeds into Control complexity

C3

SUPERVISED LEARNING · CLASSIFICATION LANE · STEP C3

Score it

How good is it at catching the class that matters?

Lessons here: 4.6, 4.7, 4.10, 4.11 · feeds into Turn predictions into decisions, Evaluate honestly

6

SUPERVISED LEARNING · STEP 6

Turn predictions into decisions

Where do I draw the line, given what each mistake costs?

Lessons here: 4.8, 4.9

Regression versus Classification

When the outcome is a category, we model the probability of each class. Why ordinary linear regression is a poor fit for yes/no outcomes, and why logistic "regression" is a classifier.

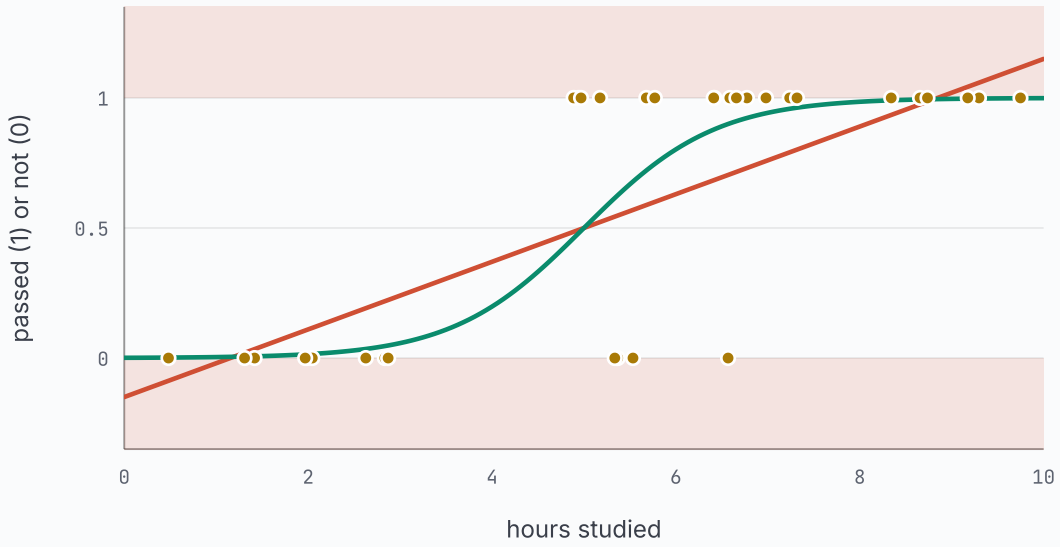
Categories, not quantities

Classification predicts a category: spam or not, diabetic or not, which digit. For two classes we code $Y \in \{0, 1\}$, where 1 is the class of interest ("positive"). The natural target is the **probability** of class 1:

$$P(X) = \Pr(Y = 1 \mid X)$$

A class prediction then comes from comparing that probability with a threshold (lesson 4.9).

Why not just use linear regression?



Fitting a straight line to 0/1 outcomes produces "probabilities" outside [0, 1] (shaded). The S-shaped logistic curve doesn't.

- **Impossible predictions:** a line goes below 0 and above 1, which can't be probabilities.
- **Broken assumptions:** when Y is only ever 0 or 1, the noise can't be "constant variance, roughly normal".
- **Fake ordering:** with more than two unordered classes (red, green, blue), coding them 1, 2, 3 imposes a meaningless order and spacing.

Why is it called logistic *regression*?

Because it **is** a regression, of the **log-odds** on the predictors: $\ln \frac{P}{1-P} = \beta_0 + \beta_1 X_1 + \dots$ (4.3). It outputs a continuous probability. It becomes a **classifier** once you add a decision rule like "predict 1 if $P > 0.5$". The name describes the model; the use is classification.

	LINEAR REGRESSION	LOGISTIC REGRESSION
Outcome	A number	A class (via a probability)
Models	$\mathbb{E}[Y \mid X]$ directly	Log-odds of $Y = 1$
Output range	$(-\infty, \infty)$	$(0, 1)$
Fitted by	Least squares (closed form)	Maximum likelihood (numerical)
Evaluated by	RMSE, MAE, R^2	Confusion matrix, sensitivity, specificity, AUC, log loss

Common doubts

FIRST PASS **Why is logistic regression used for classification when it's called regression?**

The model itself is a regression: it fits a linear function of the predictors to the **log-odds** of the outcome and outputs a probability between 0 and 1. Classification happens in an extra step, when you turn that probability into a class with a threshold (e.g. "predict 1 if $p > 0.5$"). So: regression for the probability, classification for the decision.

Cheatsheet

Linear vs logistic regression

Linear: number, models $\mathbb{E}[Y|X]$, OLS, RMSE/R^2.
Logistic: class via $P(Y = 1|X) \in (0, 1)$, linear in log-odds, MLE, confusion matrix/AUC.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/regression-versus-classification

Odds, Log-odds and the Sigmoid Function

Probability, odds and log-odds are three views of the same thing. How to convert between them by hand, and how the sigmoid function squashes any score into a probability.

Odds: betting language

$$\text{odds} = \frac{p}{1-p}$$

$p = 0.75$ gives odds of $0.75/0.25 = 3$: "3 to 1 in favour". $p = 0.5$ is odds 1 (even money). $p = 0.2$ is odds 0.25 ("1 to 4"). Odds run from 0 to ∞.

Log-odds (the logit)

$$\text{logit}(p) = \ln \frac{p}{1-p} \in (-\infty, \infty)$$

Taking the log stretches odds over the whole real line, which is perfect for a linear model: any value of $\beta_0 + \beta_1 X$ is a valid log-odds.

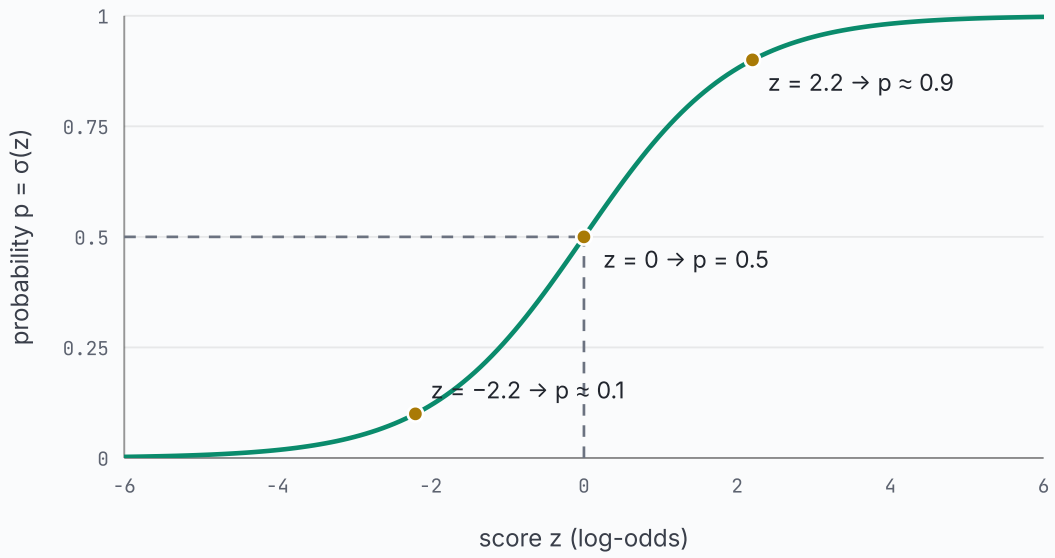
PROBABILITY P	ODDS $P/(1-P)$	LOG-ODDS
0.10	0.111	−2.20
0.20	0.25	−1.39
0.50	1	0
0.75	3	1.10
0.80	4	1.39
0.90	9	2.20

Notice the symmetry: p and $1-p$ have log-odds of equal size and opposite sign.

The sigmoid: back from score to probability

The inverse of the logit is the **logistic (sigmoid) function**:

$$p = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$$



Any real score maps to a probability strictly between 0 and 1. The curve is steepest at z = 0 (p = 0.5) and flattens near 0 and 1.

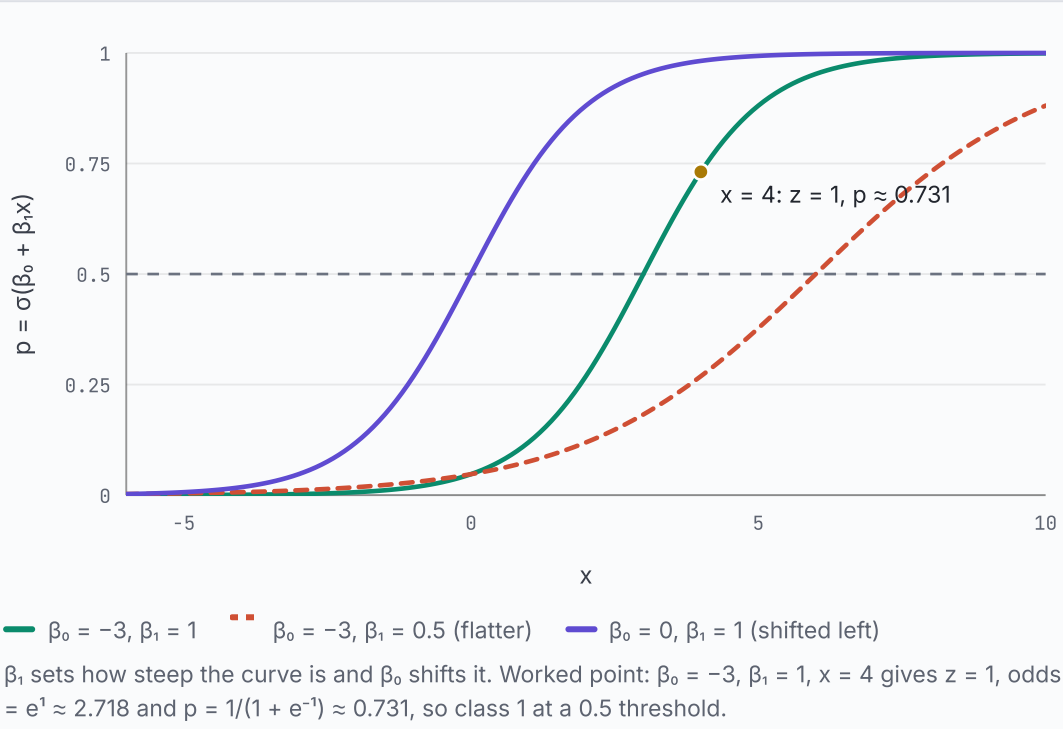
CONVERSIONS YOU MUST BE ABLE TO DO

Probability → odds → log-odds: $p = 0.8 \to$ odds $0.8/0.2 = 4 \to \ln 4 = 1.386$.

Log-odds → probability: $z = -0.6 \to$ odds $e^{-0.6} = 0.549 \to p = 0.549/1.549 = 0.354$. Equivalently $p = 1/(1 + e^{0.6})$.

Handy values: $e^{-1} \approx 0.368$, $e^{0.5} \approx 1.649$, $\ln 2 \approx 0.693$, $\ln 3 \approx 1.099$, $\ln 10 \approx 2.303$.

FROM SCORE TO PROBABILITY



β_1 sets how steep the curve is and β_0 shifts it. Worked point: $\beta_0 = -3$, $\beta_1 = 1$, $x = 4$ gives $z = 1$, odds $= e^1 \approx 2.718$ and $p = 1/(1 + e^{-1}) \approx 0.731$, so class 1 at a 0.5 threshold.

Cheatsheet

Odds & log-odds

odds $= \frac{p}{1-p}$; logit $= \ln \frac{p}{1-p}$. $p = \frac{\text{odds}}{1+\text{odds}}$.

Sigmoid

$\sigma(z) = \frac{1}{1+e^{-z}}$. $z = 0 \to 0.5$; $z = \pm 2.2 \to 0.9/0.1$.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/odds-log-odds-and-sigmoid

The Logistic Regression Model

The model as a linear score passed through a sigmoid, or as linear regression on the log-odds. Interpreting coefficients as odds ratios, computing predictions by hand, and the linear decision boundary.

Two equivalent views

THE MODEL

$$z = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p, \qquad P(X) = \Pr(Y = 1 \mid X) = \frac{1}{1 + e^{-z}}$$

$$\iff \qquad \ln \frac{P(X)}{1 - P(X)} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

View 1 squashes a linear score into a probability. View 2 is a linear model of the log-odds.

Interpreting coefficients

INTERPRETATION

β_j is the change in **log-odds** of $Y = 1$ for a one-unit increase in X_j, holding the other predictors fixed. Equivalently, e^{β_j} is the factor by which the **odds** are multiplied: the **odds ratio**.

- $\beta_j > 0 \Rightarrow e^{\beta_j} > 1$: higher X_j raises the probability of $Y = 1$.
- $\beta_j < 0 \Rightarrow e^{\beta_j} < 1$: higher X_j lowers it. $\beta_j = 0$: odds ratio 1, no effect.
- The effect on the **probability** is not constant: it depends on where you start (biggest near $p = 0.5$, tiny near 0 or 1).

WORKED EXAMPLE: CUSTOMER CHURN

log-odds $= -2 + 0.8x_1 - 0.5x_2$, where x_1 = number of complaints and x_2 = years as a customer.

Odds ratios: $e^{0.8} = 2.23$ and $e^{-0.5} = 0.61$. Each extra complaint multiplies the odds of churning by about 2.2, holding tenure fixed. Each extra year multiplies them by 0.61, a 39% reduction.

Prediction for 3 complaints, 2 years: $z = -2 + 2.4 - 1.0 = -0.6$, so $p = 1/(1 + e^{0.6}) = 0.354$. At threshold 0.5 we predict "no churn".

Probability isn't additive: going to 4 complaints moves z from -0.6 to 0.2 and p from 0.354 to 0.550 (+0.196). The odds rose by the same factor, 2.23, but the probability change depends on the starting point.

What does a predicted 0.85 mean?

A predicted $P(X) = 0.85$ means: among cases like this one, the model estimates that **85% belong to class 1**. It's a probability of class membership, not an 85% "amount" of anything. A patient with $p = 0.85$ for diabetes isn't "85% diabetic"; the model gives an 85% chance that they are diabetic.

The decision boundary is linear

With the default rule "predict 1 if $\hat{P} > 0.5$", and since $P > 0.5 \iff z > 0$, the boundary is where the linear score is zero:

$$\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0$$

That's a straight line in 2-D, so logistic regression is a **linear classifier** even though its probability curve is S-shaped. For the churn model the boundary is $0.8x_1 - 0.5x_2 = 2$. Changing the threshold to t shifts the line parallel to itself (to $z = \ln \frac{t}{1-t}$); it doesn't change the model.

THREE COMMON ERRORS

1. "A one-unit increase raises the probability by β." No: it adds β to the log-odds.
2. Treating an odds ratio of 2 as "twice as likely". It means twice the odds; the probability ratio is smaller unless p is small.
3. Reading coefficients causally from observational data, or comparing magnitudes across predictors with different units.

Common doubts

FIRST PASS **What does a predicted probability of 0.85 actually mean? Is it 85% of something?**

It's the model's estimated probability that this case **belongs to class 1**. Among many cases with the same predictor values, it expects about 85% to be positive. It isn't a measurement like "85% progressed" or "85% severity". The output is about class membership; the class itself is still yes or no.

Cheatsheet

Logistic coefficient

β_j: change in log-odds per unit X_j. e^{β_j}: odds ratio (multiplies the odds). Not a constant change in probability.

Decision boundary

Predict 1 if $\hat{p} > t \iff \hat{\beta}_0 + \hat{\beta}^\top x > \ln \frac{t}{1-t}$ (0 when $t = 0.5$). Linear.

Odds ratio 2 ≠ "twice as likely"

It doubles the **odds**. The probability ratio is smaller unless p is small.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/logistic-regression-model

Maximum Likelihood Estimation

How logistic regression is fitted: choose coefficients that make the observed labels as probable as possible. The likelihood, the log-likelihood, a worked example, and why there is no closed form.

The idea

Unlike OLS there's no formula for logistic regression's coefficients. Instead we use **maximum likelihood estimation (MLE)**: choose β to make the labels we actually observed as probable as possible under the model.

Likelihood for one observation

With predicted probability $P_i = P(x_i)$:

$$\Pr(Y_i = y_i) = P_i^{y_i}(1 - P_i)^{1-y_i}$$

This is just a compact way to write "P_i if $y_i = 1$, and $1 - P_i$ if $y_i = 0$".

All observations, and the log

For independent observations, the likelihood multiplies them. Products of many small numbers are awkward, so we take logs. The log is increasing, so the maximiser doesn't change:

LOG-LIKELIHOOD

$$\ell(\beta) = \sum_{i=1}^{n} \Big[y_i \ln P_i + (1 - y_i) \ln(1 - P_i) \Big]$$

Each term is the log of the probability the model gave to what actually happened. ℓ is always ≤ 0, and closer to 0 is better.

WORKED EXAMPLE

Labels $y = (1, 0, 1)$; predicted probabilities $p = (0.9, 0.2, 0.6)$.

Probabilities of what happened: 0.9, $1 - 0.2 = 0.8$, 0.6.

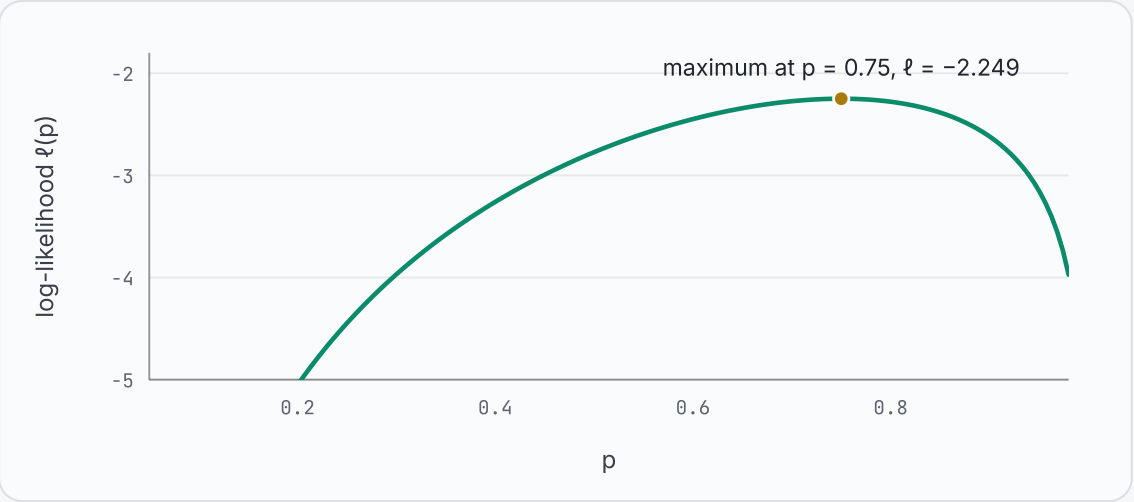
$\ell = \ln 0.9 + \ln 0.8 + \ln 0.6 = -0.105 - 0.223 - 0.511 = -0.839$.

Mean cross-entropy $= 0.839/3 = 0.280$ (lesson 4.5).

THE SIMPLEST MLE: NO PREDICTORS

Labels $y = (1, 1, 0, 1)$ and a single probability p for everyone. $\ell(p) = 3 \ln p + \ln(1 - p)$.

Derivative: $3/p - 1/(1 - p) = 0 \Rightarrow p = 3/4$. The MLE is the proportion of ones, which matches intuition.



Fitting in practice

- Maximise $\ell(\beta)$, which is the same as minimising $-\ell(\beta)$, which is the same as minimising the cross-entropy loss. So logistic regression is ERM with cross-entropy.
- There's **no closed form**: setting the derivative to zero gives equations in which β sits inside the sigmoid. They're solved numerically (gradient descent, Newton's method).
- The log-likelihood is **concave**, so there are no bad local optima: the numerical solver finds the global best.

Testing coefficients (Wald z-test)

As with OLS, test $H_0 : \beta_j = 0$ with $z_j = \hat{\beta}_j / \mathrm{SE}(\hat{\beta}_j)$, which is approximately standard normal for large samples. Reject at 5% if $|z_j| > 1.96$. A 95% CI for the odds ratio is $\exp\left(\hat{\beta}_j \pm 1.96\,\mathrm{SE}\right)$.

WORKED EXAMPLE

$\hat{\beta}_1 = 0.8$, SE $= 0.3$: $z = 2.67$, $p \approx 0.008$, significant. CI for β_1: $0.8 \pm 0.59 = (0.21, 1.39)$; for the odds ratio: $(e^{0.21}, e^{1.39}) = (1.23, 4.01)$. It excludes 1, consistent with the test.

PERFECT SEPARATION

If a predictor perfectly splits the classes, the likelihood keeps increasing as the coefficient grows, so the MLE diverges (huge $\hat{\beta}$ and SEs). Regularisation (5.x) fixes this.

Common doubts

GOING DEEPER **Why does logistic regression need gradient descent when OLS has a formula?**

Both set the **gradient** (the vector of slopes of the objective, one per coefficient) to zero. For OLS the objective is quadratic, so the gradient is linear in β and the equations can be solved directly: $\hat{\beta} = (X^\top X)^{-1} X^\top y$.

For logistic regression, β sits inside the sigmoid, so the equations can't be rearranged into a formula. Instead you start somewhere and repeatedly step **against** the gradient of the cross-entropy, $\beta \leftarrow \beta - \eta \nabla J(\beta)$, until it stops improving. The objective is convex, so this finds the single best β. The one exception is perfectly separable data, where the coefficients keep growing forever.

Cheatsheet

Logistic log-likelihood

$\ell(\beta) = \sum_i [y_i \ln P_i + (1 - y_i) \ln(1 - P_i)]$. Maximise ⇔ minimise cross-entropy.

Wald test

$z = \hat{\beta} / \mathrm{SE}$, reject if $|z| > 1.96$. OR CI: $e^{\hat{\beta} \pm 1.96 \mathrm{SE}}$.

OLS vs MLE fitting

OLS: minimise RSS, closed form. Logistic MLE: maximise log-likelihood, numerical (concave, no bad local optima).

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/maximum-likelihood-estimation

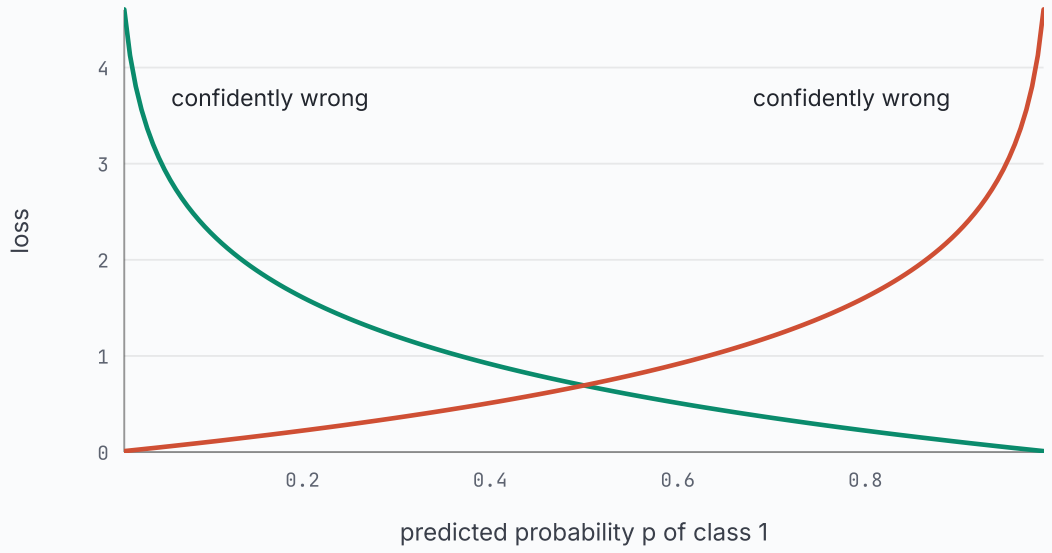
Cross-entropy / Log-loss

Cross-entropy is the negative average log-likelihood. Why it punishes confident wrong predictions so hard, how to compute it, and how it differs from accuracy as a measure.

BINARY CROSS-ENTROPY (LOG-LOSS)

$$\mathrm{CE} = -\frac{1}{n}\sum_{i=1}^{n}\Big[y_i \ln p_i + (1-y_i)\ln(1-p_i)\Big] = -\frac{\ell(\beta)}{n}$$

For each observation, only one term is active: $-\ln p_i$ if $y_i = 1$, and $-\ln(1-p_i)$ if $y_i = 0$. So the loss is "minus the log of the probability you gave to the truth".



Loss is near zero when the model is confident and right, and shoots towards infinity when it is confident and wrong.

P GIVEN TO THE TRUTH	0.99	0.9	0.5	0.1	0.01
$-\ln p$	0.010	0.105	0.693	2.303	4.605

Why use it to fit classifiers?

- It is the maximum-likelihood loss for a yes/no (Bernoulli) outcome.
- It rewards good **probabilities**, not just correct classes. Predicting 0.51 and 0.99 for a true positive both count as "correct" for accuracy, but cross-entropy prefers 0.99.
- It is smooth and, for logistic regression, convex: easy to optimise.

Cross-entropy vs accuracy

	ACCURACY	CROSS-ENTROPY
Uses	Hard class predictions	Predicted probabilities
Depends on threshold	Yes	No
Rewards confidence	No	Yes (and punishes overconfidence)
Good for fitting	No (not smooth)	Yes

SIGN CONVENTIONS

The log-likelihood ℓ has no minus sign and is **maximised**. Cross-entropy is $-\ell/n$ and is **minimised**. Both are correct; don't mix them in one sentence.

Cheatsheet

Log-loss values

$-\ln p$: 0.99→0.01, 0.9→0.105, 0.5→0.693, 0.1→2.303, 0.01→4.6.

Sign of the log-likelihood

ℓ is maximised (≤ 0). Cross-entropy $= -\ell/n$ is minimised (≥ 0).

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/cross-entropy-log-loss

The Confusion Matrix

Once you pick a threshold, every prediction is a true positive, false positive, true negative or false negative. How to build the table, read it, and which errors are "misses" and which are "false alarms".

Four outcomes

Fix a threshold and classify every observation. Choose one class as the **positive** class: the important one (disease present, fraud, alert needed). Each prediction then falls into one of four cells:

	PREDICTED POSITIVE	PREDICTED NEGATIVE
Actual positive	**TP** true positive (a hit)	**FN** false negative (a **miss**)
Actual negative	**FP** false positive (a **false alarm**)	**TN** true negative

Statisticians also call FP a **Type I error** and FN a **Type II error**. Row totals are the actual class sizes ($P = TP + FN$, $N = FP + TN$); column totals are how many the model flagged.

CHECK THE LAYOUT

Some books and software put predictions in rows and actuals in columns. Always read the labels before computing anything.

Building one

WORKED EXAMPLE

Six patients, threshold 0.5:

PATIENT	p	PREDICTED	ACTUAL	CELL
A	0.92	+	+	TP
B	0.71	+	−	FP
C	0.55	+	+	TP
D	0.40	−	+	FN
E	0.15	−	−	TN
F	0.08	−	−	TN

So TP = 2, FP = 1, FN = 1, TN = 2.

Which error is worse?

It depends on the decision, never on the maths. In cancer screening, a miss (FN) can be fatal while a false alarm (FP) means an extra test: FNs are worse. In spam filtering, a false alarm hides a real email from you: FPs are worse. This judgement drives the choice of metric (4.7) and threshold (4.9).

Cheatsheet

Confusion matrix

Rows actual, columns predicted. TP hit · FN **miss** · FP **false alarm** · TN. FP = Type I, FN = Type II.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/confusion-matrix

Accuracy, Precision, Recall and Specificity

Every metric you can compute from a confusion matrix, what question each one answers, a full worked example, and why accuracy can be dangerously misleading.

THE METRICS

$$\text{Accuracy} = \frac{TP + TN}{n}, \qquad \text{Error rate} = \frac{FP + FN}{n} = 1 - \text{Accuracy}$$

$$\text{Sensitivity (recall, TPR)} = \frac{TP}{TP + FN} \qquad \text{Specificity (TNR)} = \frac{TN}{TN + FP}$$

$$\text{FPR} = \frac{FP}{FP + TN} = 1 - \text{specificity} \qquad \text{Precision (PPV)} = \frac{TP}{TP + FP}$$

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

METRIC	QUESTION IT ANSWERS	DENOMINATOR
Sensitivity / recall	Of the actual positives, what fraction did we catch?	Actual positives
Specificity	Of the actual negatives, what fraction did we correctly clear?	Actual negatives
Precision	Of the cases we flagged, what fraction were truly positive?	Predicted positives
Accuracy	What fraction of all predictions were right?	Everything

SMOKE ALARM

Sensitivity: when there's a fire, how often does the alarm ring? **Specificity:** when there's no fire, how often does it stay quiet? **Precision:** when the alarm rings, how often is there really a fire? A hair-trigger alarm has high sensitivity and low specificity; a sleepy one the reverse.

WORKED EXAMPLE: ALL METRICS FROM ONE TABLE

$n = 1000$: TP = 40, FN = 10, FP = 30, TN = 920.

- Accuracy $= 960/1000 = 0.96$
- Sensitivity $= 40/50 = 0.80$
- Specificity $= 920/950 = 0.968$; FPR $= 30/950 = 0.032$
- Precision $= 40/70 = 0.571$
- $F_1 = \frac{2(0.571)(0.80)}{0.571+0.80} = 0.667$

Baseline: predicting "negative" for everyone gives accuracy $950/1000 = 0.95$ with sensitivity 0. Our 0.96 barely beats it. Precision (0.57) is much lower than sensitivity or specificity here, because positives are rare.

Prevalence matters for precision

Sensitivity and specificity condition on the **actual** class, so they don't depend on how common positives are. Precision conditions on the **predicted** class, so it does.

THE RARE-DISEASE SURPRISE

Prevalence 1%, sensitivity 90%, specificity 90%, 1,000 people. Sick: 10, of whom 9 test positive. Healthy: 990, of whom 99 test positive (10%). Total positives 108, of which only 9 are sick: **precision = 9/108 = 8.3%**. A "90% accurate" test is wrong more than 9 times out of 10 when it says positive.

ACCURACY HIDES RARE CLASSES

A fraud model with 99.5% accuracy on data with 0.5% fraud may simply predict "not fraud" for everything. Always compare with the majority-class baseline and report class-specific metrics.

Cheatsheet

Sensitivity / recall / TPR

$\frac{TP}{TP+FN}$: share of actual positives caught.

Specificity / TNR

$\frac{TN}{TN+FP}$: share of actual negatives cleared. FPR $= 1-$ spec.

Precision / PPV

$\frac{TP}{TP+FP}$: share of flagged cases truly positive. Depends on prevalence.

Accuracy & F1

Acc $= \frac{TP+TN}{n}$. $F_1 = \frac{2PR}{P+R}$.

High accuracy, rare class

Compare with the majority-class baseline. Report sensitivity, specificity, precision.

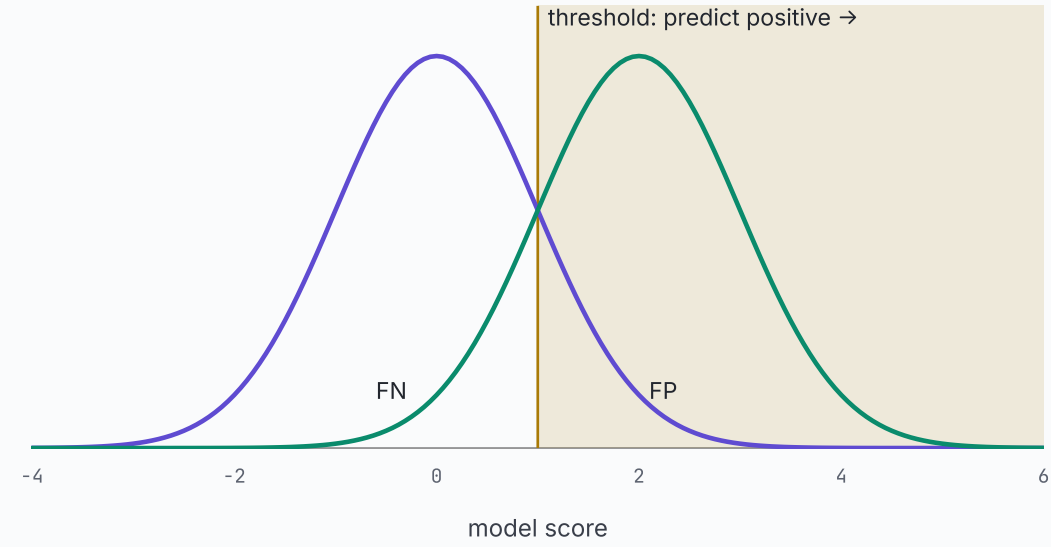
Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/accuracy-precision-recall-and-specificity

Sensitivity and Specificity Trade-offs

Moving the threshold trades misses for false alarms. Why you can't raise sensitivity and specificity together with the same model, and how to reason about which to favour.

Two overlapping distributions

A classifier gives each case a score (for logistic regression, $\hat{p}$). Positives tend to score higher than negatives, but the two distributions overlap. Wherever you put the threshold, some negatives land above it (false positives) and some positives below it (false negatives).



Negatives right of the threshold are false positives; positives left of it are false negatives. Sliding the line trades one for the other.

Which way does each move?

THRESHOLD	FLAG	SENSITIVITY	SPECIFICITY	FN	FP
Lower (e.g. 0.5 → 0.2)	More cases	↑	↓	↓	↑
Raise (e.g. 0.5 → 0.8)	Fewer cases	↓	↑	↑	↓

At the extremes: threshold 0 flags everyone (sensitivity 1, specificity 0); threshold 1 flags no one (sensitivity 0, specificity 1). Neither is useful.

ONLY A BETTER MODEL MOVES BOTH

With a fixed model, the threshold just slides along a trade-off. To improve sensitivity *and* specificity together you need a model whose score distributions overlap less: better features, more data, a better method. That's what the ROC curve and AUC measure (4.10).

Precision and recall trade off too

Lowering the threshold usually raises recall (sensitivity) and lowers precision, since you flag more marginal cases. Precision–recall is the more informative pair when positives are rare and true negatives are plentiful and uninteresting.

HOW IT'S ASKED

"A team lowers the threshold until sensitivity is 100%, then boasts about it. What's the catch?" Flagging nearly everyone achieves perfect sensitivity at the cost of a huge number of false positives (specificity and precision collapse). Report both sides of the trade-off.

Cheatsheet

Threshold effects

Lower t → more flagged → sens ↑, spec ↓, FN ↓, FP ↑, precision usually ↓. Raise t → the reverse.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/sensitivity-and-specificity-trade-offs

Threshold Selection and Classification Costs

When a miss and a false alarm cost different amounts, choose the threshold that minimises expected cost. The theory, a worked table, and how to choose it without leaking the test set.

0.5 is a choice, not a law

The default "predict 1 if $\hat{p} > 0.5$" minimises the number of mistakes when both kinds cost the same. When they don't, pick the threshold that minimises **expected cost**:

$$\text{Cost}(t) = c_{FP} \cdot FP(t) + c_{FN} \cdot FN(t)$$

The theoretical threshold

For a single case with (well-calibrated) probability p: alerting costs $(1-p)\,c_{FP}$ in expectation; not alerting costs $p\,c_{FN}$. Alert when the second is bigger:

COST-OPTIMAL THRESHOLD

$$\text{alert if} \quad p > \frac{c_{FP}}{c_{FP} + c_{FN}} \qquad (\text{equal costs} \Rightarrow 0.5)$$

If a miss is costlier than a false alarm, the threshold moves **down**, so you flag more cases.

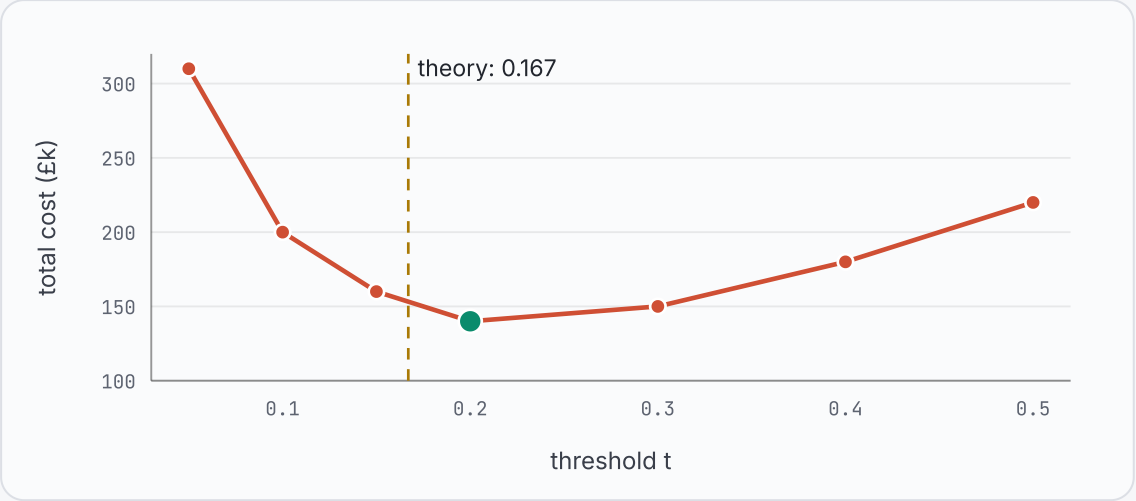
WORKED EXAMPLE: POLLUTION ALERTS

A false alarm costs £2,000; a missed dangerous reading costs £10,000. Theory: alert if $p > 2000/12000 = 0.167$.

Empirically, on 1,000 validation cases:

THRESHOLD	FP	FN	COST $= 2000\,FP + 10000\,FN$
0.5	10	20	£220,000
0.2	40	6	**£140,000** (best)
0.1	90	2	£200,000

The lowest cost is at 0.2, close to the theoretical 0.167. Going lower produces too many false alarms.



The same logic applies when the score is a regression prediction: e.g. alert when predicted CO exceeds t, where the real event is CO above a legal limit of 5. Because misses are costlier, the best t ends up a little *below* 5: alert slightly early.

MOVING THE THRESHOLD (40 POSITIVES, 160 NEGATIVES)

T	TP	FN	FP	TN	ACCURACY	SENSITIVITY	SPECIFICITY	PRECISION	COST
0.1	40	0	122	38	0.390	1.000	0.237	0.247	£244,000
0.2	38	2	93	67	0.525	0.950	0.419	0.290	£206,000
0.3	37	3	61	99	0.680	0.925	0.619	0.378	£152,000
0.5	28	12	28	132	0.800	0.700	0.825	0.500	£176,000
0.7	16	24	9	151	0.835	0.400	0.944	0.640	£258,000
0.9	5	35	1	159	0.820	0.125	0.994	0.833	£352,000

Cost uses £2,000 per false positive and £10,000 per false negative. The cost-minimising threshold for calibrated probabilities is t* = 2000 / (2000 + 10000) ≈ 0.167, far below 0.5. Try it interactively in the online lesson.

CHOOSE THE THRESHOLD ON VALIDATION DATA

The threshold is a hyperparameter. Choosing the cost-minimising threshold on the test set and reporting the test cost leaks the test set: the reported cost is optimistic. Pick it by cross-validation or on a validation set; test once. Choosing it on training predictions is also optimistic, because the model is overconfident on data it was fitted to.

Cheatsheet

Cost-optimal threshold

Alert if $p > \frac{c_{FP}}{c_{FP}+c_{FN}}$. Costlier misses → lower threshold.

Expected cost

$\text{Cost}(t) = c_{FP}\,FP(t) + c_{FN}\,FN(t)$; choose t on validation data.

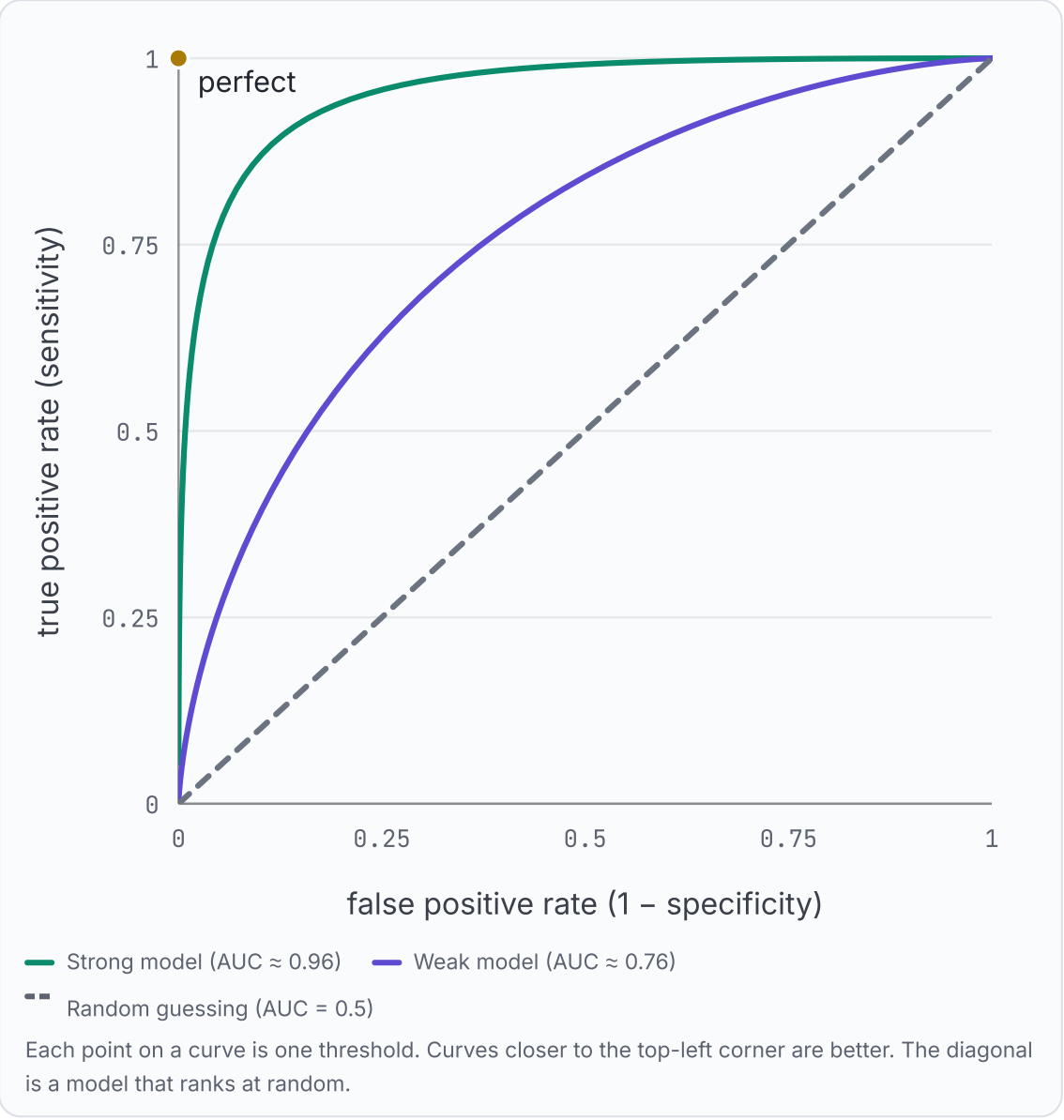
Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/threshold-selection-and-classification-costs

ROC Curve and AUC

The ROC curve shows every sensitivity–specificity trade-off a model can offer. AUC summarises it as the probability a random positive outranks a random negative. How to compute both by hand.

The curve

For every threshold t from 1 down to 0, compute the true positive rate (sensitivity) and the false positive rate (1 – specificity). Plot TPR against FPR. That's the **ROC curve** (receiver operating characteristic).



Each point on a curve is one threshold. Curves closer to the top-left corner are better. The diagonal is a model that ranks at random.

- **Top-left corner** (FPR 0, TPR 1) is perfect.
- **The diagonal** is random guessing: flagging a random fraction of cases catches the same fraction of positives and negatives.
- Moving along a curve = moving the threshold. The curve itself doesn't change with the threshold.

AUC

MEANING

AUC (area under the ROC curve) is the probability that a randomly chosen positive gets a **higher score** than a randomly chosen negative. 0.5 = random ranking; 1.0 = perfect ranking.

AUC BY COUNTING PAIRS

Positives scored $(0.9, 0.6)$; negatives $(0.7, 0.3, 0.2)$. There are $2 \times 3 = 6$ positive–negative pairs. A pair is "correct" if the positive scores higher:

- 0.9 beats all three negatives: 3
- 0.6 beats 0.3 and 0.2 but not 0.7: 2

AUC $= 5/6 = 0.833$. (Ties count as half.)

TRACING THE CURVE FROM THE SAME SCORES

THRESHOLD	FLAGGED	TP	FP	TPR	FPR
0.8	0.9	1	0	0.5	0
0.65	0.9, 0.7	1	1	0.5	1/3
0.5	0.9, 0.7, 0.6	2	1	1	1/3
0.1	all five	2	3	1	1

Raising the threshold lowers both TPR and FPR.

What AUC does and doesn't tell you

AUC TELLS YOU	AUC DOES NOT TELL YOU
How well the model **ranks** positives above negatives	Which threshold to use
Performance across all thresholds (threshold-independent)	Accuracy, precision or cost at your operating point
A fair comparison of rankers, unaffected by class balance	Whether the probabilities are well calibrated

AUC ≠ ACCURACY

An AUC of 0.92 doesn't mean 92% accuracy. And a model with AUC 0.55 isn't necessarily useless at every threshold, though it ranks only slightly better than chance.

Cheatsheet

ROC & AUC

ROC: TPR vs FPR over all thresholds. AUC = P(random positive scored above random negative). 0.5 random, 1 perfect.

AUC by counting

For every (positive, negative) pair, score 1 if the positive ranks higher, 0.5 if tied. AUC = total / (#pos × #neg).

AUC misreadings

AUC 0.9 ≠ 90% accuracy. It doesn't choose a threshold or account for costs.

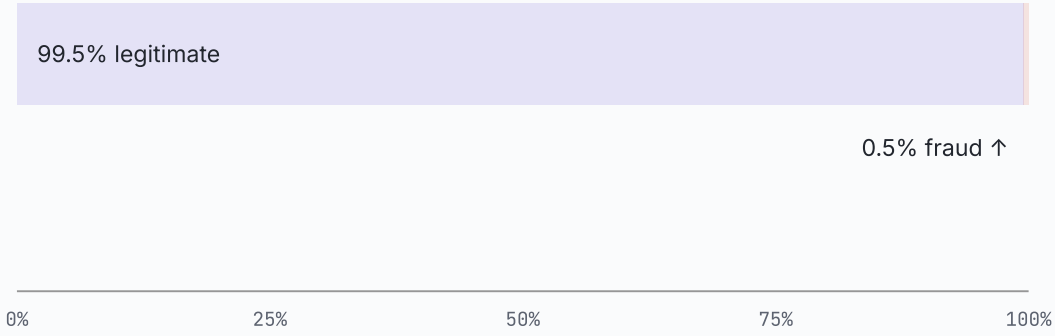
Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/roc-curve-and-auc

Class Imbalance

When the interesting class is rare, a model can score high accuracy while catching nothing. Baselines, the right metrics, threshold moves, oversampling and stratified sampling, and the oversampling traps.

The accuracy trap

When the interesting class is rare (fraud, a rare disease, equipment failure), a classifier minimising overall error can learn to ignore it. With 0.5% fraud, predicting "not fraud" for everyone scores **99.5% accuracy** and catches zero fraud.



Share of transactions by class. A model can be right 99.5% of the time by never predicting the tiny class that matters.

Remedies

1. **Use the right metrics:** sensitivity, specificity, precision, ROC/AUC, a confusion matrix; and compare with the **majority-class baseline**.
2. **Move the threshold** using the costs of each error (4.9).
3. **Oversample the rare class** (or undersample the common one) in the *training* data, so the model pays attention to it.
4. **Use class weights**: count each rare-class error as more expensive in the loss.

A stratified oversampling procedure

1. Split the data into set A (the class of interest) and set B (everything else).
2. **Training set:** randomly take 50% of A, plus an equal number of cases from B. This makes training balanced.
3. **Validation set:** the remaining 50% of A, plus enough of B to **restore the original class ratio**.

Result: a somewhat worse overall error rate, but typically a much better error rate on the cases that matter. Validation stays representative of the real world.

OVERSAMPLING TRAPS

- **Never oversample before splitting.** Copies of the same record land in both training and validation, inflating scores (leakage).
- **Evaluate on data with the original class ratio**, otherwise precision and accuracy are meaningless for deployment.
- After oversampling, predicted probabilities are no longer calibrated to the real prevalence; adjust before using them for cost-based thresholds.

HOW IT'S ASKED

"We balanced the data by duplicating the rare class, then cross-validated. Comment." Duplicates of the same case end up in training and validation folds, so the model is tested on near-copies of its training data. Oversample only within each training fold.

Cheatsheet

Oversampling rule

Split first. Oversample only the training part (inside each CV fold). Evaluate on the original class ratio.

Imbalance toolkit

Right metrics + majority baseline · cost-based threshold · oversample/undersample training data · class weights.

Online version with the interactive quiz: airesearchlabs.si/learn/ml/logistic-regression/class-imbalance