Quantitative Research Methods

Chapter 4: Classification

Prof. Dr. Christoph Weisser

HSBI

Summer Semester 2026

The course at a glance

	Lecture	Topic
	1	Ch. 1 + 2 — Introduction; what is statistical learning
	2	Ch. 2 — Accuracy, bias–variance, Bayes classifier, KNN
	3–4	Ch. 3 — Linear regression (estimation, inference, diagnostics)
▶	**5–6**	**Ch. 4 — Classification (logistic regression, confusion matrix, ROC/AUC)**
	7	Ch. 5 — Resampling (validation, CV, bootstrap)
	8	Ch. 6 — Model selection and regularisation (ridge, lasso)
	9	Ch. 7 — Moving beyond linearity (splines, GAMs)
	10	Ch. 8 — Tree-based methods (trees, forests, boosting)
	11	Ch. 10 — Deep learning (MLPs, CNNs, training)
	12	Ch. 13 — Multiple testing (FWER, FDR)

Twelve sessions of 180 minutes. ▶ marks today's chapter. Short exercises every ∼20 min; extended exercises every ∼45 min; each chapter has a companion lab notebook.

Contents

Optional and advanced material is collected in the **appendix** at the end of the deck: the **generative models** (Bayes refresher, LDA, QDA, naive Bayes, Exercises 4.5–4.7), the classifier comparison (Extended Exercise 4.4), the derivations (Extended Exercises 4.2–4.3), and GLMs / Poisson regression.

Notation in this chapter

Symbol	Meaning
$p(X) = \Pr(Y = 1 \mid X)$	probability of class 1 given predictors X
$p(X)/(1 - p(X))$	odds of class 1 (ranges over $(0, \infty)$)
$\log\big[p(X)/(1 - p(X))\big]$	log-odds (logit); linear in X in logistic regression
$\beta_0, \beta_1, \ldots$; e^{β_j}	coefficients; multiplicative effect of X_j on the odds
π_k	prior probability of class k *(appendix)*
$f_k(x)$	class-conditional density of X within class k *(appendix)*
μ_k, Σ	class-k mean vector; (shared) covariance matrix *(appendix)*
$\delta_k(x)$	discriminant score of class k; predict the largest *(appendix)*
TP, FP, TN, FN	true/false positives and negatives (confusion matrix)
sensitivity; specificity	$\mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$; $\mathrm{TN}/(\mathrm{TN} + \mathrm{FP})$
AUC	area under the ROC curve (threshold-free ranking quality)

Symbols marked *(appendix)* belong to the generative models (LDA, QDA, naive Bayes), which are optional material at the end of the deck.

Sources and references

Primary textbook

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An Introduction to Statistical Learning, with Applications in Python.* Springer.

About these slides

Each method is introduced with motivation, intuition, formal definition, worked example, and interpretation. Slide titles are descriptive; formulas are read symbol-by-symbol.

Learning objectives for this chapter

By the end of this chapter you will be able to:

- **Explain** why ordinary linear regression cannot model a categorical response.
- **Fit** logistic regression by maximum likelihood and **interpret** its coefficients on the odds scale.
- **Extend** the model to several predictors and **recognise** confounding in the fitted coefficients.
- **Read** confusion matrices, sensitivity, specificity, precision, ROC (receiver operating characteristic) curves, and AUC; **tune** the threshold to your loss function.
- **Run** the whole workflow in Python (`statsmodels` and `scikit-learn`).

Optional (appendix). The *generative* classifiers — the Bayes refresher, LDA, QDA and naive Bayes, their comparison with logistic regression and KNN, and the GLM / Poisson extension — are covered in the appendix and are not part of the taught thread.

Roadmap of this chapter

The taught thread — one model done properly

1. **Why** linear regression fails on a categorical response.
2. **Logistic regression** — discriminative, linear in the log-odds: the model, maximum likelihood, coefficients on the odds scale, several predictors and confounding.
3. **Evaluation** — confusion matrix, sensitivity / specificity / precision, ROC and AUC, choosing the threshold.
4. **Python lab** — the same workflow end-to-end.

In the appendix (optional)

The *generative* family — LDA, QDA and naive Bayes, built from Bayes' theorem — together with the head-to-head comparison against logistic regression and KNN, and the GLM framework that ties linear, logistic and Poisson regression together.

Outline

Why a separate chapter on classification?

When the response Y is a category — not a number — everything we know from Chapter 3 has to be rebuilt.

Three reasons

1. Most real-world decisions are *discrete*: pay vs. don't pay, fraud vs. legitimate, malignant vs. benign.
2. Linear regression on a 0/1 response gives meaningless predictions outside $[0, 1]$.
3. For $K > 2$ classes, no single linear model can encode the qualitative ordering — we need a different framework.

Concrete client problem: the Default data

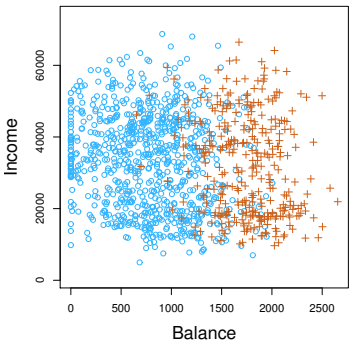
The brief

A bank wants to decide whether a credit card application should be approved. They give us:

Variables

- Response: `default` (Yes / No)
- Predictors: `balance`, `income`, `student` (Yes / No)
- $n = 10{,}000$ historical customers

The bank's question: *"what is the probability of default for this new customer?"*



Source: Figure 4.1 — ISLP, James et al. (2023).

Why ordinary linear regression fails on 0/1 data

Tempting answer: code $Y = 1$ for default, $Y = 0$ otherwise, and run OLS. Three reasons this fails.

Three problems

1. Predictions can fall *outside* $[0, 1]$ — nonsensical as probabilities.
2. For $K > 2$ classes, the implicit numerical coding $(1, 2, 3)$ imposes an arbitrary ordering and arbitrary distances.
3. Errors are heteroscedastic by construction: $\mathrm{Var}(Y \mid X) = p(1 - p)$ depends on X.

We need a model that produces well-calibrated probabilities and handles multiple classes naturally.

Exercise 4.1 — Why not linear regression? [Concept]

Exercise

A colleague wants to predict a patient's emergency diagnosis, coding the three unordered categories as $Y = 1$ (stroke), $Y = 2$ (overdose), $Y = 3$ (seizure), and fitting ordinary least squares (OLS).

1. What implicit assumption does the coding $1, 2, 3$ impose on the three diagnoses, and why is it inappropriate here?
2. Suppose instead there are only two classes coded $0/1$ and we fit OLS. Name two problems with the fitted values as probabilities.
3. Would relabelling the classes change the OLS fit? What does your answer imply about using linear regression here?

Solution — Exercise 4.1 (1/2)

Solution

Method. Ask what the numerical coding *implicitly assumes* about the response, then check each assumption against what a probability model must deliver.

(1) The coding imposes order and equal spacing. Using $1, 2, 3$ tells the model that stroke $<$ overdose $<$ seizure and that the "distance" from stroke to overdose equals that from overdose to seizure. The diagnoses are *unordered categories*; any such ordering and spacing is arbitrary and has no medical meaning. Concretely, a fitted value of 1.5 would assert a patient is "halfway between stroke and overdose" — clinically meaningless.

(2) Two problems with OLS on a $0/1$ response.

- Fitted values are unbounded, so predictions can fall *outside* $[0, 1]$ and cannot be read as probabilities (e.g. a *negative* "default probability").
- The errors are heteroscedastic by construction: $\mathrm{Var}(Y \mid X) = p(1 - p)$ depends on X, violating the constant-variance assumption of OLS — so its standard errors and p-values are invalid as well.

Solution — Exercise 4.1 (2/2)

Solution

(3) Yes, relabelling changes the fit. A different assignment of the numbers $1, 2, 3$ to the categories produces a different response vector and hence a different least-squares model and different predictions — swapping the codes of stroke and seizure reverses the fitted ordering although the data are unchanged. Because the answer depends on an arbitrary choice, linear regression is *not appropriate* for an unordered multiclass response. (For two classes, 0/1 OLS happens to give the same classification as linear discriminant analysis — see the appendix — but its fitted values are still not valid probabilities.)

Interpretation. What we actually need is a model that outputs calibrated probabilities in $(0, 1)$ and treats the classes symmetrically — exactly the gap that logistic regression (two classes) and its multinomial/softmax extension ($K > 2$) fill in this chapter.

Common mistake

hoping a larger sample fixes the problem. No amount of data repairs an arbitrary coding — the flaw sits in the model specification, not in sampling noise.

Where this chapter is used in industry

Sector	Concrete application	Which decision it drives
Retail banking	Probability of default (PD) on card, loan and mortgage applications	Accept/decline cut-off, risk-based pricing, IFRS 9 loss provisions
Payments	Card-fraud and anti-money-laundering transaction screening	Block at checkout, or send to a human review queue
Telco, SaaS, banking	Monthly churn (attrition) propensity scoring	Who receives a retention offer this month
Insurance	Underwriting referral; claims-fraud and claims-severity triage	Which applications and claims a human examines
Healthcare, pharma	Diagnostic and screening tests; biomarker and spectral panels (classic LDA territory: appendix)	Who is referred for a confirmatory test
Tech, operations	Spam, content and support-ticket classification (multinomial)	Auto-route to a queue, or escalate to an agent

In industry — The common thread

All six deliver a *probability*, not a verdict. The verdict comes from a **threshold chosen on business costs** — and where the decision has to be explained to a customer or a regulator, the explainable model usually wins.

Industry case in depth: credit scoring (probability of default)

In industry — The model behind an accept/decline decision

Response: 90+ days past due within 12 months of scoring. **Predictors:** utilisation, months since last delinquency, recent bureau enquiries, income and tenure, loan-to-value.

The fitted output (illustrative)

Utilisation gets a positive coefficient, a delinquency-free year a negative one; scorecards rescale the log-odds into points calibrated so that a fixed number of points *doubles the odds*.

Which decision — and who signs off

The PD drives the cut-off, the rate offered, IFRS 9 expected credit loss ($\text{PD} \times \text{LGD} \times \text{EAD}$) and Basel capital — and an independent validation function, not the analyst, signs it off (SR 11-7, ECB TRIM). Day to day it is the credit-committee member and the branch lead who live with that cut-off.

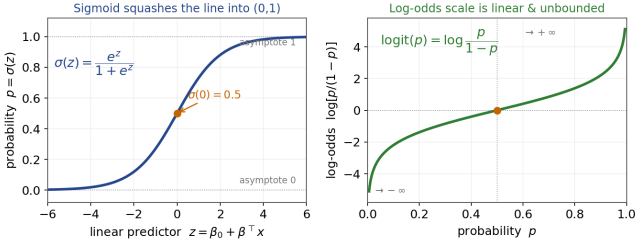
Why logistic regression — and one caveat

A declined applicant must be told *why*: adverse-action reason codes under US ECOA/Reg B come straight from the coefficients. Caveat: the data contain only *accepted* applicants, so the sample is selected.

Outline

Why we need a logistic curve

We need to map any linear predictor $z = \beta_0 + \beta^\top x$ to a probability in $(0, 1)$.



The logistic function

$\sigma(z) = e^z/(1 + e^z)$ is smooth, monotone $0 \to 1$, and symmetric about $\sigma(0) = 0.5$ (left). Its inverse, the *logit*, stretches $(0, 1)$ onto the whole real line (right) — so a *linear* model belongs on the logit scale.

The logistic regression model

Formal model

$$p(X) := \Pr(Y = 1 \mid X) = \frac{e^{\beta_0 + \beta^\top X}}{1 + e^{\beta_0 + \beta^\top X}}.$$

Equivalently, the *log-odds* (logit) are linear:

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

Why two forms?

They are the *same* model written two ways. The first guarantees $p(X) \in (0, 1)$; the second shows the model is linear on the log-odds scale — so all the intuition from Chapter 3 carries over, just on a transformed response.

Reading the model symbol-by-symbol

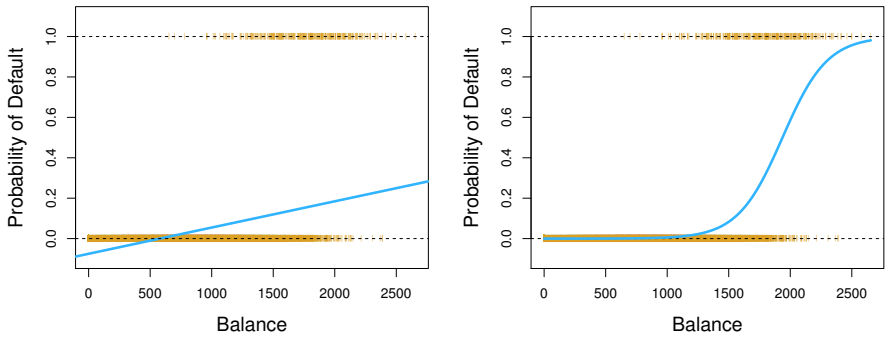
How to read this

Symbol	Meaning
Y	Binary response, 0 or 1
$p(X)$	$\Pr(Y = 1 \mid X)$, the conditional probability
$p(X)/(1 - p(X))$	The *odds* of $Y = 1$
$\log[p/(1 - p)]$	The *log-odds* or logit, in $(-\infty, +\infty)$
β_j	Effect of X_j on the *log-odds*
e^{β_j}	Multiplicative effect of X_j on the *odds*

Why the logit?

The log-odds are unbounded, so a linear model on this scale is natural. The sigmoid then squashes back into $(0, 1)$.

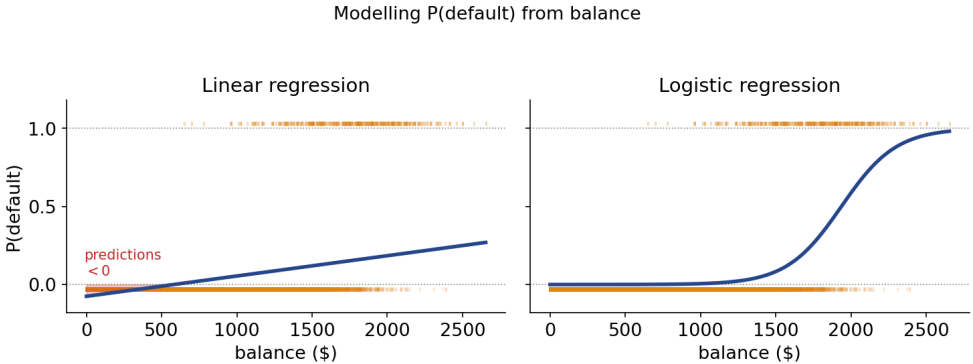
The logistic curve in pictures



Linear regression (left) versus logistic regression (right). Logistic regression keeps probabilities in $[0, 1]$ everywhere.

Source: Figure 4.2 — ISLP, James et al. (2023).

Recreating Figure 4.2 on the Default data



Takeaway

Fitting a straight line to the 0/1 default indicator (left) drives predicted probabilities *below zero* at small balances. The logistic curve (right) stays inside $[0, 1]$ by construction.

Exercise 4.2 — Computing $p(X)$ from coefficients [Math]

Exercise

A simple logistic regression of `default` on `balance` gives log-odds $\eta = \beta_0 + \beta_1$ balance with $\hat{\beta}_0 = -10.65$ and $\hat{\beta}_1 = 0.0055$ (per dollar). For a customer with a balance of \$1,500:

1. Compute the log-odds η.
2. Compute the probability of default $p(X)$.
3. Compute the odds of default and relate them to part (1).
4. At what balance does the model give $p(X) = 0.5$?

Solution — Exercise 4.2 (1/2)

Solution

Method. Everything follows one pipeline: balance $\xrightarrow{\text{linear}}$ log-odds η $\xrightarrow{\exp}$ odds e^{η} $\xrightarrow{\text{normalise}}$ probability $p = e^{\eta}/(1+e^{\eta})$.

(1) Log-odds. Substitute the balance into the linear predictor:

$$\eta = -10.65 + 0.0055 \times 1500 = -10.65 + 8.25 = -2.40.$$

Negative log-odds already signal $p < 0.5$; the customer looks safe.

(2) Probability. Push η through the sigmoid (needed because η itself lives on $(-\infty, \infty)$, not $[0, 1]$):

$$p = \frac{e^{\eta}}{1+e^{\eta}} = \frac{e^{-2.40}}{1+e^{-2.40}} = \frac{0.0907}{1.0907} \approx 0.083 \ (8.3\%).$$

Solution — Exercise 4.2 (2/2)

Solution

(3) Odds.

$$\text{odds} = \frac{p}{1-p} = e^{\eta} = e^{-2.40} \approx 0.091 \qquad \left(\text{check: } \tfrac{0.083}{0.917} \approx 0.091 \checkmark\right).$$

The odds are exactly exp(log-odds) from part (1): about one default for every eleven non-defaults at this balance.

(4) Balance giving $p = 0.5$. $p = 0.5 \iff \text{odds} = 1 \iff \eta = 0$ (since $\sigma(0) = \frac{1}{2}$), so solve the linear equation

$$0 = -10.65 + 0.0055\,\text{balance} \implies \text{balance} = \frac{10.65}{0.0055} \approx 1936.4 \approx \$1{,}936.$$

Interpretation. Under a 0.5 threshold the model flags customers above $\approx$ \$1,936; at \$1,500 our customer sits clearly on the safe side with $p \approx 8\%$.

Common mistake

reading $\eta = -2.40$ itself as a probability. Log-odds must first be pushed through the sigmoid — probabilities can never be negative.

Maximum likelihood: the recipe

Given data (x_i, y_i) with $y_i \in \{0, 1\}$, build the likelihood:

Likelihood

$$L(\beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)).$$

How to read this

Each customer who *did* default contributes $p(x_i)$ to the product; each who *did not* default contributes $1 - p(x_i)$. Taking logs turns the product into the sum we actually maximise (numerically — there is no closed form):

$$\log L(\beta) = \sum_{i=1}^{n} \Big[y_i \log p(x_i) + (1 - y_i) \log\big(1 - p(x_i)\big) \Big].$$

Interpreting the coefficients

Three different interpretation scales

- **Sign of** β_j: increasing X_j raises (positive) or lowers (negative) the probability of $Y = 1$.
- $\exp(\beta_j)$: multiplicative effect on the *odds* of $Y = 1$.
- **Effect on probability**: nonlinear — moves probabilities most in the middle of the curve, less at the extremes.

Takeaway

For interpretation, report $\exp(\beta_j)$, the *odds ratio*: a single number with a clean "holding the others fixed" meaning. The probability effect has no such fixed value — it depends on where you sit on the sigmoid.

Worked example: Default vs. balance

Estimated $\hat{\beta}_{\text{balance}} \approx 0.0055$ in the Default data.

Three different scales

Log-odds: a \$1,000 increase in balance increases the log-odds of default by $0.0055 \cdot 1000 = 5.5$.

Odds: the odds multiply by $e^{5.5} \approx 245$.

Probabilities: at a balance of \$1,500 the model assigns about 8 % probability of default; at \$2,500 it assigns about 96 %. Going up just \$1,000 lifts the probability from under 10 % to over 95 % because we are crossing the steep middle of the sigmoid.

In industry — Credit: coefficients become reason codes

A declined applicant must be told *which* factors counted against them; the signed coefficients supply exactly that, which is why scorecards stay logistic. Quote an odds ratio as if it were a change in probability and the notice is simply wrong.

Exercise 4.3 — Coefficients on the odds scale [Math]

Exercise

A *multiple* logistic regression on the Default data gives $\hat{\beta}_0 = -10.87$, $\hat{\beta}_{\text{balance}} = 0.00574$, $\hat{\beta}_{\text{income}} = 0.003$ (per \$1,000), and $\hat{\beta}_{\text{student[Yes]}} = -0.6468$.

1. Interpret $e^{\hat{\beta}_{\text{student}}}$. By what percentage do the odds of default differ for a student versus a non-student at the *same* balance and income?
2. By what factor do the odds of default change for a \$200 increase in balance, holding the other predictors fixed?
3. Why do we report a *constant* multiplicative effect on the odds, but not on the probability?

Solution — Exercise 4.3 (1/2)

Solution

Method. Because the logit is linear, any coefficient converts to an odds statement through one operation: odds ratio $= e^{\beta_j \, \Delta X_j}$, "holding the other predictors fixed".

(1) Student effect.

$$e^{\hat{\beta}_{\text{student}}} = e^{-0.6468} \approx 0.524.$$

Holding balance and income fixed, being a student multiplies the odds of default by 0.524 — i.e. a student's odds are about $1 - 0.524 = 47.6\%$ *lower* than an otherwise identical non-student. (This is the *adjusted* effect; the unadjusted comparison points the other way — the student paradox.)

(2) A \$200 increase in balance. The log-odds shift by $\Delta\eta = 0.00574 \times 200 = 1.148$, so the odds multiply by

$$e^{0.00574 \times 200} = e^{1.148} \approx 3.15,$$

and this factor is the same whether balance goes $500 \to 700$ or $1{,}500 \to 1{,}700$.

Solution — Exercise 4.3 (2/2)

Solution

(3) Odds vs. probability. Because the logit is *linear*, a fixed change ΔX_j shifts the log-odds by the constant $\beta_j \Delta X_j$, so the odds scale by the constant factor $e^{\beta_j \Delta X_j}$ regardless of the starting point. The probability, however, is the nonlinear sigmoid of the log-odds, so no constant probability effect exists.

Seeing it with numbers (odds ratio 3.15 from part 2):

$$p = 0.10 : \text{odds } \frac{0.10}{0.90} = 0.111 \xrightarrow{\times 3.15} 0.350 \Rightarrow p = \frac{0.350}{1.350} \approx 0.26 \ (+16 \text{ pts}),$$

$$p = 0.90 : \text{odds } \frac{0.90}{0.10} = 9 \xrightarrow{\times 3.15} 28.4 \Rightarrow p = \frac{28.4}{29.4} \approx 0.97 \ (+7 \text{ pts}).$$

Same odds ratio, very different probability moves — which is why the odds scale is the one with a clean single-number interpretation.

Common mistake

reporting part (1) as "students are 47.6% less likely to default". The reduction is in the *odds*, not the probability; the two only coincide approximately when the event is rare.

Multiple logistic regression: the model

Formal model

$$\log \frac{p(X)}{1 - p(X)} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

Same machinery as the simple case — and the same warnings as in Chapter 3:

- Univariate slopes can mislead under confounding.
- "Holding others fixed" interpretation applies to each coefficient.
- Confounders can flip the sign of an estimated effect.

Takeaway

Each β_j is now an *adjusted* effect — the impact of X_j with the other predictors held fixed. That adjustment is what lets the model untangle the confounding on the next slide.

The student paradox in the Default data

What looks one way alone, another way together

In a univariate logistic regression, `student` appears *more* default-prone than non-students. But controlling for `balance`, students are in fact *less* likely to default at the same balance — they simply tend to carry larger balances overall.

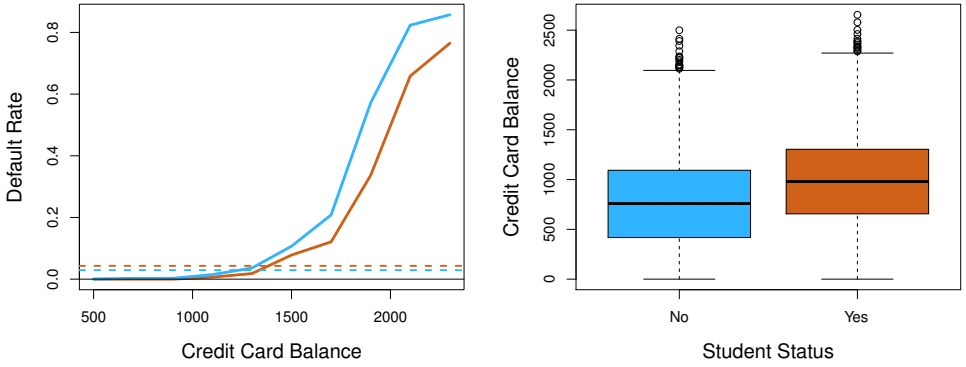
Takeaway

A textbook case of confounding. Always run multiple logistic regression when several predictors might be related.

In industry — Portfolio analytics: mix effects

The same trap sits in every management report: a segment or a channel looks riskier (or more churn-prone) only because it carries larger balances, younger accounts or cheaper tariffs. Repricing a segment on the raw cross-tabulation prices the mix, not the risk.

The student paradox visualised



Conditional on balance, the orange (student) curve sits below the blue (non-student) curve. The univariate marginal told the opposite story.

Source: Figure 4.3 — ISLP, James et al. (2023).

Exercise 4.4 — From probability to decision [Math]

Exercise

Use the simple model $\eta = -10.65 + 0.0055\,\text{balance}$. Three applicants have balances \$1,800, \$2,000 and \$2,200.

1. Compute the predicted probability of default for each.
2. Using the default threshold 0.5 (predict "default" if $p > 0.5$), classify each applicant.
3. The bank now lowers the threshold to 0.2 to catch more defaulters. What is the new decision boundary (in dollars of balance), and which classifications change?

Solution — Exercise 4.4 (1/2)

Solution

Method. For each applicant compute $\eta = -10.65 + 0.0055 \times \text{balance}$, then $p = e^{\eta}/(1 + e^{\eta})$; a threshold on p is equivalent to a threshold on η, i.e. to a dollar boundary on balance.

(1) Probabilities (substituting each balance):

$$\$1{,}800 : \eta = -10.65 + 9.90 = -0.75,\ p = \frac{e^{-0.75}}{1+e^{-0.75}} = \frac{0.472}{1.472} \approx 0.321,$$

$$\$2{,}000 : \eta = -10.65 + 11.00 = 0.35,\ p = \frac{e^{0.35}}{1+e^{0.35}} = \frac{1.419}{2.419} \approx 0.587,$$

$$\$2{,}200 : \eta = -10.65 + 12.10 = 1.45,\ p = \frac{e^{1.45}}{1+e^{1.45}} = \frac{4.263}{5.263} \approx 0.810.$$

Note the balances are evenly spaced ($200 apart) but the probability steps are not ($+0.27$, then $+0.22$) — the sigmoid is nonlinear.

Solution — Exercise 4.4 (2/2)

Solution

(2) Threshold 0.5. $p > 0.5 \iff \eta > 0 \iff \text{balance} > \frac{10.65}{0.0055} \approx \$1{,}936$. So \$1,800 → **No**, \$2,000 → **Yes**, \$2,200 → **Yes** — consistent with the probabilities $0.321 < 0.5 < 0.587 < 0.810$ from part (1).

(3) Threshold 0.2. First convert the probability threshold to the log-odds scale: $p = 0.2 \iff \eta = \log\frac{0.2}{0.8} = \log 0.25 = -1.386$. Then solve for the dollar boundary:

$$\text{balance} = \frac{-1.386 + 10.65}{0.0055} = \frac{9.264}{0.0055} \approx \$1{,}684.$$

Every applicant above \$1,684 is flagged: the \$1,800 applicant flips from **No** to **Yes**; the other two remain **Yes**.

Interpretation. Lowering the threshold slides the dollar boundary left (1,936 → 1,684): sensitivity rises (more true defaulters caught) at the price of more false positives. The right cutoff depends on the bank's cost ratio of the two errors.

Common mistake: applying the new threshold on the wrong scale — requiring $\eta > 0.2$ instead of $\eta > \log(0.2/0.8) = -1.386$.

Mini-check: interpreting logistic coefficients

Self-assessment

A logistic regression on customer-default data gives $\hat{\beta}_{\text{balance}} = 0.0055$ (per dollar of balance).

1. By how much does the *log-odds* change for a \$500 increase in balance?
2. By how much do the *odds* change for a \$500 increase?
3. Why is the change in *probability* not proportional to β?

Answers

1. $0.0055 \cdot 500 = 2.75$ on the log-odds scale. 2. Multiplied by $e^{2.75} \approx 15.6$. 3. The sigmoid is nonlinear — the same $\Delta\beta$ moves the probability differently depending on where on the curve you are.

Extended Exercise 4.1 — Logistic regression end-to-end [Math]

Extended exercise (15 min)

A multiple logistic regression on the Default data estimates the log-odds of default as $\eta = \beta_0 + \beta_1 \,\text{balance} + \beta_2 \,\text{income} + \beta_3 \,\text{student}$, with $\beta_0 = -10.869$, $\beta_1 = 0.00574$ (per \$1 of balance), $\beta_2 = 0.003$ (per \$1,000 of income) and $\beta_3 = -0.6468$ (student = Yes). Two customers: *A* — non-student, balance \$1,000, income \$40,000; *B* — student, balance \$2,000, income \$30,000.

1. Compute the log-odds η and probability $p(X)$ for A and B.
2. For A, compute the odds of default and relate them to η.
3. By what factor do the odds change for a \$100 increase in balance (others fixed)? Interpret e^{β}.
4. Interpret e^{β_3}: how do a student's odds compare with an identical non-student's?
5. Find the balance at which $p = 0.5$ for a non-student with \$40,000 income (the decision boundary); note how it shifts for a student.
6. In one sentence, how are the β's estimated?

Solution — Extended Exercise 4.1 (1/3)

Solution

Method. Assemble η term by term — watching units: income enters in \$1,000s — then map to a probability with the sigmoid.

(1) Log-odds and probabilities:

$$\begin{aligned}\eta_A &= -10.869 + 0.00574(1000) + 0.003(40) = -10.869 + 5.740 + 0.120 = -5.009,\\ \eta_B &= -10.869 + 0.00574(2000) + 0.003(30) - 0.6468\\ &= -10.869 + 11.480 + 0.090 - 0.647 = 0.054.\end{aligned}$$

Then $p = \sigma(\eta) = e^{\eta}/(1+e^{\eta})$ gives $p_A = \frac{e^{-5.009}}{1+e^{-5.009}} = \frac{0.0067}{1.0067} \approx 0.0066$ (0.66%) and $p_B = \frac{e^{0.054}}{1+e^{0.054}} = \frac{1.055}{2.055} \approx 0.514$ (51.4%): A is a very safe customer; B sits right at the knife edge.

Solution — Extended Exercise 4.1 (2/3)

Solution

(2) Odds for A. $\text{odds}_A = \dfrac{p_A}{1-p_A} = e^{\eta_A} = e^{-5.009} \approx 0.0067$ — about one default per 150 non-defaults. The odds are exactly exp(log-odds).

(3) A \$100 increase in balance. $\Delta\eta = \beta_1 \cdot 100 = 0.574$, so the odds multiply by $e^{0.574} \approx 1.78$: every extra \$100 of balance raises the odds of default by about 78%, *regardless* of the starting point (constant on the odds scale because the logit is linear).

(4) Student effect. $e^{\beta_3} = e^{-0.6468} \approx 0.524$. Holding balance and income fixed, a student's odds of default are $0.524\times$ those of an identical non-student — about 47.6% lower.

Solution — Extended Exercise 4.1 (3/3)

Solution

(5) Decision boundary ($p = 0.5 \iff \eta = 0$). For a non-student with \$40,000 income,

$$0 = -10.869 + 0.00574\,\text{balance} + 0.12 \;\Rightarrow\; \text{balance}^{\star} = \frac{10.869 - 0.12}{0.00574} \approx \$1{,}873.$$

Above $\approx$ \$1,873 the model predicts default at a 0.5 cutoff. For a *student* the boundary rises to $\frac{10.869-0.12+0.6468}{0.00574} \approx$ \$1,985: a student needs a larger balance to reach the same 50% risk — the same message as (4).

(6) Estimation. By *maximum likelihood*: choose β to maximise $L(\beta) = \prod_{y_i=1} p(x_i) \prod_{y_i=0} (1 - p(x_i))$ (equivalently minimise the cross-entropy $-\log L$). There is no closed form, so software solves it numerically (Newton / IRLS) — conceptually, pick the coefficients that make the observed Yes/No pattern most probable.

Common mistake

plugging income in dollars. $\beta_2 = 0.003$ is per \$1,000, so use 0.003×40, not $0.003 \times 40{,}000$ — the latter adds 120 to η and drives every probability to 1.

Outline

Why we need more than "accuracy"

Imagine a fraud detector with 99 % accuracy on a stream where 1 % of transactions are fraudulent.

The trivial classifier

The model that simply *always predicts "no fraud"* has accuracy 99 % — and detects zero fraud.

Takeaway

Accuracy alone is uninformative on imbalanced data. We need to decompose it into the four cells of a *confusion matrix*.

In industry — Payments: fraud and AML screening

This is not a toy example — it is how card fraud and money-laundering alerts actually look. A missed fraud costs the chargeback; a false positive blocks a good customer at checkout. Fraud teams therefore price each cell of the matrix, and set the threshold to the number of alerts their reviewers can actually work through in a day.

The confusion matrix

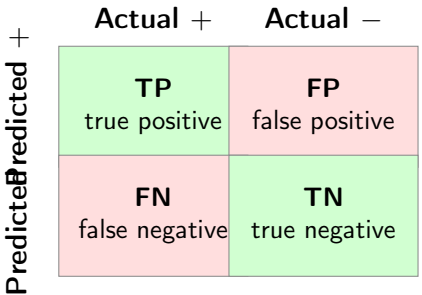
After choosing a threshold (default 0.5), tabulate predicted vs. actual:

	Pred. neg.	Pred. pos.	Total
True neg.	TN	FP	N
True pos.	FN	TP	P
Total	N*	P*	n

How to read this

- Sensitivity / recall: $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ — of true positives, how many did we catch?
- Specificity: $\mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$ — of true negatives, how many did we correctly leave alone?
- Precision: $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$ — of those we flagged, how many were truly positive?

The confusion matrix as a picture



$$\textbf{Sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\textbf{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

$$\textbf{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Takeaway

The diagonal cells — TP and TN — are the correct decisions; FP and FN are the two error types. Sensitivity reads down the true-positive column; specificity down the true-negative column; precision across the predicted-positive row.

Exercise 4.8 — Reading a confusion matrix [Concept]

Exercise

A classifier fitted to the Default data ($n = 10{,}000$), scored at threshold 0.5, gives:

	Pred. No	Pred. Yes	Total
True No	9644	23	9667
True Yes	252	81	333
Total	9896	104	10000

Compute (a) the accuracy and error rate, (b) sensitivity/recall, (c) specificity, (d) precision. (e) Given that only 3.3% of customers default, comment on how good this classifier really is.

Nothing here depends on *which* classifier produced the table — these are the ISLP Table 4.4 counts (LDA; appendix).

Solution — Exercise 4.8 (1/2)

Solution

Method. First pin down the four cells — every metric is a ratio of them: $\mathrm{TN} = 9644$, $\mathrm{FP} = 23$, $\mathrm{FN} = 252$, $\mathrm{TP} = 81$, with $P = 333$ actual defaulters and $N = 9667$ actual non-defaulters.

(a) Accuracy $= \dfrac{9644 + 81}{10000} = \dfrac{9725}{10000} = 0.9725$; error rate $= 1 - 0.9725 = 2.75\%$.

(b) Sensitivity $= \dfrac{\mathrm{TP}}{P} = \dfrac{81}{333} \approx 0.243$: of the 333 customers who really default, the classifier catches only 81 and misses 252.

(c) Specificity $= \dfrac{\mathrm{TN}}{N} = \dfrac{9644}{9667} \approx 0.9976$: non-defaulters are almost never disturbed (only 23 false alarms).

(d) Precision $= \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = \dfrac{81}{104} \approx 0.779$: of the 104 customers flagged, 81 truly default.

Solution — Exercise 4.8 (2/2)

Solution

(e) Interpretation. The trivial classifier that *always predicts "No"* has an error rate of $333/10000 = 3.33\%$ — barely worse than the model's 2.75%. So almost all of the headline accuracy comes from the 96.7% base rate of non-defaulters, not from skill. The high accuracy hides a poor sensitivity: the model *misses* $252/333 = 75.7\%$ *of actual defaulters*. For a bank a missed defaulter (FN) is the expensive kind of error, which motivates lowering the classification threshold — the subject of the next exercise.

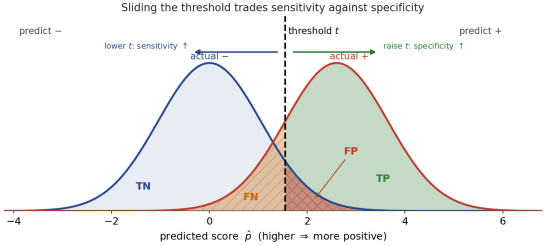
Takeaway. On imbalanced data always report the pair (sensitivity, specificity) — accuracy alone can be matched by a classifier that does nothing.

Common mistake

mixing up sensitivity and precision. Both have TP on top, but sensitivity divides by the *actual* positives ($\mathrm{TP} + \mathrm{FN} = 333$) while precision divides by the *predicted* positives ($\mathrm{TP} + \mathrm{FP} = 104$) — here 24.3% vs. 77.9%, a huge difference.

Why a single threshold is not enough

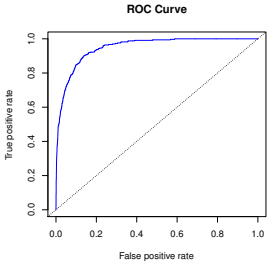
Predicting "positive" at $\hat{p} > 0.5$ is only *one* operating point; any threshold t trades sensitivity vs. specificity:



Takeaway

Lower $t \Rightarrow$ more true positives (sensitivity $\uparrow$) but more false alarms (specificity $\downarrow$); raise t for the reverse. Choose t by the cost of a false negative vs. a false positive — a cancer screen lowers it, a spam filter raises it.

The ROC curve

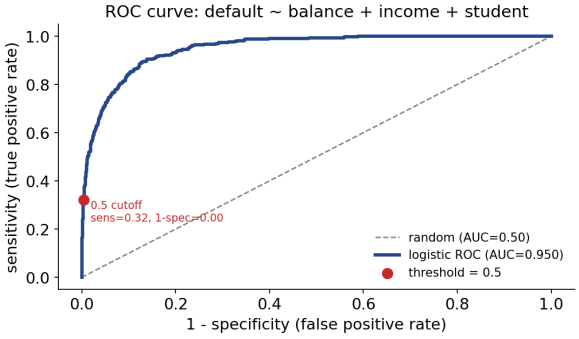


How to read this

y-axis = sensitivity (true-positive rate); x-axis = $1-$ specificity (false-positive rate). Each point is *one* threshold; the 45° diagonal is random guessing — top-left is better.

Source: Figure 4.8 — ISLP, James et al. (2023).

ROC and AUC on the Default data



Takeaway

A high AUC means the model ranks defaulters above non-defaulters. The default 0.5 cutoff sits low on the curve; lowering it trades specificity for sensitivity.

AUC: area under the ROC curve

Probabilistic interpretation

AUC $= \Pr\big(\hat{p}(\text{positive}) > \hat{p}(\text{negative})\big)$ for a randomly chosen positive–negative pair.

How to read this

- AUC $= 0.5$: model is no better than random.
- AUC $= 1$: perfect ranking of positives above negatives.
- Threshold-free summary; widely used in medicine and credit scoring.

Caveat under heavy class imbalance

AUC can look misleadingly good when one class is rare. Use precision-recall curves or balanced accuracy in that regime.

Exercise 4.9 — Threshold, ROC and AUC [Concept]

Exercise

Lowering the threshold of the same classifier to 0.2 on the same Default data gives:

	Pred. No	Pred. Yes	Total
True No	9432	235	9667
True Yes	138	195	333

1. Compute sensitivity and specificity at threshold 0.2 and compare with the 0.5 results (Exercise 4.8).
2. Compute the overall error rate at 0.2.
3. On ROC axes (sensitivity vs. 1−specificity), how does the operating point move as the threshold drops from 0.5 to 0.2?
4. State the probabilistic meaning of the AUC and why it is threshold-free.

Solution — Exercise 4.9 (1/2)

Solution

Method. Same model, same data — only the cutoff moved, so we recompute the two rates from the new cell counts ($\mathrm{TN} = 9432$, $\mathrm{FP} = 235$, $\mathrm{FN} = 138$, $\mathrm{TP} = 195$) and compare operating points.

(1) At threshold 0.2. Sensitivity $= \dfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = \dfrac{195}{333} \approx 0.586$; specificity $= \dfrac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = \dfrac{9432}{9667} \approx 0.9757$. Compared with 0.5 (Exercise 4.8): sensitivity rises from 24.3% to 58.6% (81 $\rightarrow$ 195 defaulters caught), while specificity falls from 99.76% to 97.57% (23 $\rightarrow$ 235 false alarms).

(2) Error rate $= \dfrac{\mathrm{FP}+\mathrm{FN}}{n} = \dfrac{235+138}{10000} = \dfrac{373}{10000} = 3.73\%$ (up from 2.75%) — yet arguably the *better* bank decision, because the errors it now makes are the cheap kind.

Solution — Exercise 4.9 (2/2)

Solution

(3) ROC movement. With y = sensitivity and $x = 1-$specificity:

$$x:\ 1 - 0.9976 = 0.0024 \ \rightarrow\ 1 - 0.9757 = 0.0243, \qquad y:\ 0.243 \ \rightarrow\ 0.586.$$

The operating point moves *up and to the right along the same ROC curve*: a small loss of specificity buys a large gain in sensitivity. The curve itself is fixed — it belongs to the model's ranking, and each threshold merely picks a point on it.

(4) AUC. AUC $= \Pr\big(\hat{p}(\text{a random positive}) > \hat{p}(\text{a random negative})\big)$: the probability the model ranks a random defaulter above a random non-defaulter. It integrates over *all* thresholds, so it summarises ranking quality independently of any single cutoff — 0.5 is random guessing, 1 is a perfect ranking.

Common mistake

expecting the AUC to improve after lowering the threshold. Threshold changes slide the operating point *along* the curve; they can never move the curve — only a better model can.

Extended Exercise 4.5 — Default: logistic, confusion, ROC/AUC [Python]

Extended exercise (15 min)

Using `Default.csv`:

1. Fit a logistic regression of `default` on `balance`, `income` and a 0/1 `student` indicator (scikit-learn).
2. Build the confusion matrix at thresholds 0.5 and 0.2.
3. At each threshold compute accuracy, sensitivity, specificity and precision.
4. Plot the ROC curve, report the AUC, and interpret the whole picture.

Write runnable code and state the numbers you expect.

Solution — Extended Exercise 4.5 (1/3)

Solution

```
import pandas as pd # data handling
from sklearn.linear_model import LogisticRegression # classifier
from sklearn.metrics import (confusion_matrix, roc_auc_score,
                             RocCurveDisplay) # evaluation tools
import matplotlib.pyplot as plt # plotting

df = pd.read_csv("../ALL CSV FILES - 2nd Edition/Default.csv") # load
y = (df["default"] == "Yes").astype(int) # encode response as 0/1
df["stud"] = (df["student"] == "Yes").astype(int) # dummy-code student
X = df[["balance", "income", "stud"]].values # predictor matrix

clf = LogisticRegression(max_iter=10000).fit(X, y) # fit model (MLE)
proba = clf.predict_proba(X)[:, 1] # predicted P(default=Yes) per customer
```

Solution — Extended Exercise 4.5 (2/3)

Solution

```
for t in (0.5, 0.2): # two decision thresholds
    tn, fp, fn, tp = confusion_matrix(y, proba > t).ravel() # cell counts
    sens, spec = tp/(tp+fn), tn/(tn+fp) # sensitivity, specificity
    prec, acc = tp/(tp+fp), (tp+tn)/len(y) # precision, accuracy
    print(t, tn, fp, fn, tp, # report all metrics
          round(acc,3), round(sens,3), round(spec,3), round(prec,3))

print("AUC =", round(roc_auc_score(y, proba), 3)) # threshold-free AUC
RocCurveDisplay.from_predictions(y, proba); plt.show() # draw ROC curve
```

Expected (counts close to):

t	TN	FP	FN	TP	acc	sens	spec	prec
0.5	9627	40	228	105	.973	.315	.996	.724
0.2	9390	277	130	203	.959	.610	.971	.423

AUC ≈ 0.950 — ranking quality, unchanged by the threshold.

Solution — Extended Exercise 4.5 (3/3)

Solution

Interpretation.

- **Threshold** 0.5. Accuracy is high (97.3%) but sensitivity is only 31.5%: the model misses about two-thirds of true defaulters. Precision is decent (72%) only because it flags very few customers — the always-predict-"No" trap on imbalanced data.
- **Threshold** 0.2. Sensitivity nearly doubles (to 61%), catching far more defaulters, at the cost of specificity (99.6% $\rightarrow$ 97.1%) and precision (72% $\rightarrow$ 42%): more false alarms. Accuracy dips slightly — but accuracy is the wrong yardstick here.
- **ROC / AUC.** AUC ≈ 0.95 summarises ranking quality over *all* thresholds: a random defaulter outscores a random non-defaulter about 95% of the time. The 0.5 and 0.2 cut-offs are two operating points on the *same* ROC curve, so the AUC does not change with the threshold.

Choose the operating point by the bank's relative cost of a missed defaulter (FN) versus a false alarm (FP). Note `sklearn` orders the matrix as $[[\mathrm{TN}, \mathrm{FP}], [\mathrm{FN}, \mathrm{TP}]]$.

Outline

Python lab — run it live in the notebook

Companion notebook: `chapter_04_lab.ipynb`

The code in this section is meant to be **demonstrated live**. Open the companion Jupyter notebook and run it cell by cell:

- §1 *Logistic regression*, including *Predicted probabilities*;
- the *ROC curves* subsection of §2;
- *optional:* the rest of §2, *LDA, QDA, naive Bayes, KNN — head-to-head*, and §3 *Poisson regression on Bikeshare* — both appendix material.

Data loads via the `ISLP` package or the bundled CSV files; the notebook ends with hands-on exercises.

Logistic regression with statsmodels

```
import statsmodels.api as sm # GLM fitting with inference
from ISLP import load_data # textbook data loader

Default = load_data("Default") # load Default data
y = (Default["default"] == "Yes").astype(int) # encode response 0/1
X = sm.add_constant(Default[["balance", "income"]]) # add intercept column

res = sm.GLM(y, X, family=sm.families.Binomial()).fit() # logit link fit
print(res.summary()) # coefficients, SEs, z, p-values
```

`statsmodels` reports each coefficient with its standard error, z-statistic, and p-value, plus deviance-based goodness-of-fit measures.

Logistic regression with scikit-learn

```
from sklearn.linear_model import LogisticRegression # classifier
from sklearn.model_selection import train_test_split # data splitting
from sklearn.metrics import classification_report, roc_auc_score

X = Default[["balance", "income"]].values # predictor matrix
# hold out 30% as a test set (fixed seed for reproducibility)
Xtr,Xte,ytr,yte = train_test_split(X, y, test_size=0.3, random_state=0)

lr = LogisticRegression(max_iter=1000).fit(Xtr, ytr) # fit on training set
proba = lr.predict_proba(Xte)[:, 1] # test-set P(default=Yes)
print(classification_report(yte, proba > 0.5)) # metrics at threshold 0.5
print("AUC:", roc_auc_score(yte, proba)) # ranking quality
```

ROC curves

```
from sklearn.metrics import RocCurveDisplay # ROC plotting helper
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(5, 5)) # one axes, all curves
RocCurveDisplay.from_estimator(lr, Xte, yte, ax=ax) # logistic ROC
ax.plot([0,1], [0,1], "k--", linewidth=0.8) # chance line (AUC = 0.5)
ax.set_title("Default: ROC on the held-out 30%"); plt.show()
```

Every extra classifier is one more `RocCurveDisplay.from_estimator(...)` call on the same `ax` — the appendix does exactly that for LDA and QDA.

Exercise 4.10 — Logistic regression + confusion matrix in Python [Python]

Exercise

Using `Default.csv`:

1. Fit a logistic regression of `default` on `balance`, `income` and a 0/1 `student` indicator with scikit-learn.
2. Produce the confusion matrix at thresholds 0.5 and 0.2.
3. Report the sensitivity at each threshold and interpret the trade-off.

Write the code and describe the output you expect.

Solution — Exercise 4.10 (1/2)

Solution

```
import pandas as pd # data handling
from sklearn.linear_model import LogisticRegression # classifier
from sklearn.metrics import confusion_matrix # evaluation

df = pd.read_csv("../ALL CSV FILES - 2nd Edition/Default.csv") # load
y = (df["default"] == "Yes").astype(int) # encode response as 0/1
df["stud"] = (df["student"] == "Yes").astype(int) # dummy-code student
X = df[["balance", "income", "stud"]].values # predictor matrix

clf = LogisticRegression(max_iter=10000).fit(X, y) # fit model (MLE)
proba = clf.predict_proba(X)[:, 1] # predicted P(default=Yes) per customer
```

Solution — Exercise 4.10 (2/2)

Solution

```
for t in (0.5, 0.2): # default and lowered threshold
    tn, fp, fn, tp = confusion_matrix(y, proba > t).ravel() # cell counts
    sens = tp / (tp + fn) # sensitivity = true-positive rate
    print(f"t={t}: TN={tn} FP={fp} FN={fn} TP={tp} sens={sens:.3f}")
```

Expected output (counts close to):

```
t=0.5: TN=9627 FP=40 FN=228 TP=105 sens=0.315
t=0.2: TN=9390 FP=277 FN=130 TP=203 sens=0.610
```

Interpretation. At the default threshold 0.5 the model is very accurate overall but has *low sensitivity* (about 30%): it misses most true defaulters, exactly as in Exercise 4.8. Dropping the threshold to 0.2 roughly doubles sensitivity (to about 60%) by flagging more customers, at the cost of more false positives (lower specificity). The right threshold depends on the bank's relative cost of a missed defaulter versus a false alarm. Note `sklearn`'s `confusion_matrix` orders cells as $[[\mathrm{TN}, \mathrm{FP}], [\mathrm{FN}, \mathrm{TP}]]$.

Extended Exercise 4.6 — Predicting market direction out-of-sample [Python]

Extended exercise (15 min)

Using `Weekly.csv` (weekly S&P 500 returns 1990–2010; `Direction` is Up / Down):

1. Fit a logistic regression of `Direction` on `Lag1`–`Lag5` and `Volume` on the full data. What in-sample accuracy do you get, and what does the confusion matrix reveal?
2. Do an *honest* out-of-sample test: train on Year ≤ 2008, test on 2009–2010, using `Lag2` as the only predictor. Report test accuracy and the confusion matrix for logistic regression — and, as a one-line bonus, for LDA (appendix material; here it is a single `scikit-learn` import).
3. Compare with the naive "always predict Up" baseline and explain why market-direction prediction is so hard.

Write runnable code and give the expected numbers.

Solution — Extended Exercise 4.6 (1/3)

Solution

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.metrics import accuracy_score, confusion_matrix

wk = pd.read_csv("../ALL CSV FILES - 2nd Edition/Weekly.csv") # load
y = (wk["Direction"] == "Up").astype(int) # encode Up=1, Down=0
lags = ["Lag1","Lag2","Lag3","Lag4","Lag5","Volume"] # predictor names

# (1) in-sample logistic on all predictors
full = LogisticRegression(max_iter=10000).fit(wk[lags], y) # fit full model
print("acc:", round(full.score(wk[lags], y), 3)) # training accuracy
print(confusion_matrix(y, full.predict(wk[lags]))) # [[TN,FP],[FN,TP]]
```

Output: `acc: 0.561`; confusion [[54, 430], [48, 557]] in order [[TN, FP], [FN, TP]].

Solution — Extended Exercise 4.6 (2/3)

Solution

```
# (2) train Year<=2008, test 2009-2010, using Lag2 only
tr, te = wk["Year"] <= 2008, wk["Year"] >= 2009 # boolean masks
for nm, mdl in [("logit", LogisticRegression(max_iter=10000)),
                ("LDA", LDA())]: # two linear classifiers
    mdl.fit(wk.loc[tr, ["Lag2"]], y[tr]) # fit on training years
    pred = mdl.predict(wk.loc[te, ["Lag2"]]) # 0/1 test predictions
    print(nm, round(accuracy_score(y[te], pred), 3),
          confusion_matrix(y[te], pred).ravel()) # acc and TN FP FN TP
```

Output: `logit 0.625 [ 9 34 5 56]  LDA 0.625 [ 9 34 5 56]` (TN FP FN TP). Both linear classifiers give the *same* result on the 2009–2010 hold-out ($n = 104$).

Solution — Extended Exercise 4.6 (3/3)

Solution

Reading it.

- The full-data model reaches only 56%, and its confusion matrix shows it predicts "Up" almost always ($430 + 557$ of 1089): the apparent edge is mostly the market's upward drift, not skill — and this is optimistic, being in-sample.
- Out-of-sample (2009–2010) logistic regression and LDA give the *same* 62.5% and confusion matrix (near-identical linear classifiers). Both still predict "Up" almost every week, catching 56/61 up-weeks (sensitivity 0.92) but only 9/43 down-weeks (specificity 0.21).
- **Baseline.** "Always predict Up" already scores $61/104 = 58.7\%$; the models beat it only marginally.

Why it is hard. Weekly returns are close to a random walk: the lagged predictors carry almost no signal, the signal-to-noise ratio is tiny, and any real pattern is arbitraged away (market efficiency). Barely beating the "always Up" baseline — and only by riding the drift — is no basis for trading once transaction costs are included.

Outline

Chapter 4 in one slide

What we accomplished

One classifier done properly — **logistic regression** — plus the tools to judge *any* classifier: the confusion matrix, ROC and AUC.

- Saw why linear regression cannot model a categorical response.
- Modelled $\Pr(Y = 1 \mid X)$ as a logistic curve, linear in the log-odds; fitted it by maximum likelihood.
- Read coefficients on the odds scale, and watched a slope flip sign under confounding (the student paradox).
- Evaluated by confusion matrix, sensitivity / specificity / precision, ROC and AUC — and chose the threshold by cost.
- Ran the whole workflow in `statsmodels` and `scikit-learn`.

Appendix (optional): the generative family — Bayes refresher, LDA, QDA, naive Bayes — their comparison with logistic regression and KNN, and the GLM / Poisson extension.

Vocabulary checklist

Term	Meaning
Odds	$p/(1-p)$; e^{β_j} is the odds multiplier for X_j
Logit / log-odds	$\log[p/(1-p)]$; linear in X
Maximum likelihood	Choose β to maximise $\Pr(\text{data} \mid \beta)$
Discriminative model	Models $\Pr(Y \mid X)$ directly (logistic regression)
Confusion matrix	TP, FP, TN, FN at one chosen threshold
Sensitivity / specificity	TP/(TP+FN) / TN/(TN+FP)
Precision	TP/(TP+FP)
ROC / AUC	TPR vs. FPR / area under that curve
Appendix vocabulary (optional):	
Generative model	Models π_k and $f_k(X)$, applies Bayes
LDA / QDA	Linear / quadratic discriminant analysis
Naive Bayes	Class-conditional independence assumption
GLM / link function	Family of regression-style models / $g(E[Y])$

Key formulas at a glance

Logistic regression

$p(X) = e^{\beta_0+\beta^\top X}/(1+e^{\beta_0+\beta^\top X}), \qquad \log\frac{p}{1-p} = \beta_0 + \beta^\top X = \eta.$

Evaluation: confusion-matrix rates, ROC, AUC

Sensitivity (recall, TPR) $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$; specificity (TNR) $= \mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$; precision $= \mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$; FPR $= 1-\text{spec.} = \mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$. ROC: TPR vs. FPR across thresholds; AUC $= \Pr\big(\hat{p}(\text{pos}) > \hat{p}(\text{neg})\big)$, 0.5 chance, 1 perfect.

Threshold → boundary: $p > t \iff \eta > \log\frac{t}{1-t}$ $(t = 0.5 \iff \eta > 0)$.

Generative models — appendix, kept here for reference

Bayes posterior: $\Pr(Y = k \mid X) = \pi_k f_k(X)/\sum_\ell \pi_\ell f_\ell(X)$.
LDA: $\delta_k(x) = x^\top\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^\top\Sigma^{-1}\mu_k + \log\pi_k$; QDA gives each class its own Σ_k, so the quadratic term survives: $\delta_k(x) = -\frac{1}{2}(x-\mu_k)^\top\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k$.
1-D LDA, common σ^2: $x^\star = \frac{\mu_0+\mu_1}{2} - \frac{\sigma^2}{\mu_1-\mu_0}\log\frac{\pi_1}{\pi_0}$.

Decision rules of thumb

- Binary or categorical response $\Rightarrow$ never ordinary least squares; start with logistic regression.
- Large n, modest p, no idea about the distribution of X $\Rightarrow$ logistic regression is the safe default.
- Report a *probability*, then set the threshold; never let 0.5 be an unexamined default.
- On imbalanced data, judge by sensitivity / specificity / AUC, never by accuracy alone.
- Interpret coefficients on the odds scale, and only ever *holding the other predictors fixed*.
- Highly nonlinear boundary, plenty of data $\Rightarrow$ trees and ensembles (Ch. 8), KNN or kernel SVM (Ch. 9).

Appendix rules of thumb. Small n with roughly Gaussian, equally spread classes $\Rightarrow$ LDA; visibly unequal class covariances with enough data $\Rightarrow$ QDA; many binary / categorical features with $p \gg n \Rightarrow$ naive Bayes. Standardise features before any distance-based model.

Common pitfalls

Don't

1. Compare classifiers by accuracy alone on imbalanced data.
2. Treat logistic-regression probabilities as calibrated without checking.
3. Read e^{β_j} as a change in *probability* — it is a multiplier on the *odds*, and the probability effect depends on where you sit on the curve.
4. Interpret a univariate slope as a causal effect: adding `balance` flips the sign of `student`.
5. Report a confusion matrix without saying which threshold produced it — change the threshold, change every cell.
6. Ignore the base rate — in rare-event detection a "perfect-looking" classifier can be useless.

Appendix pitfalls: LDA on visibly unequal class covariances; forgetting that QDA needs $\sim Kp^2/2$ parameters.

Connections to the rest of the course

- Ch. 2: the Bayes classifier and KNN — the benchmark today's logistic regression is measured against.
- Ch. 3: linear regression is logistic regression's continuous twin (both are GLMs; see the appendix).
- Ch. 5: cross-validation for choosing thresholds and tuning.
- Ch. 6: ridge / lasso regularisation extends to logistic regression.
- Ch. 8: tree ensembles often beat logistic regression on tabular data — but are harder to explain to a regulator.
- Ch. 9: SVMs as another linear / kernelised classifier.
- Ch. 10: deep learning is logistic regression with representation learning baked in (the softmax output layer is the multinomial logit of the appendix).

Self-check questions

1. Why does ordinary linear regression fail for binary Y? (p. 11)
2. Write the logistic model and its log-odds form. What exactly does $e^{\beta_j} = 1.15$ tell a client? (pp. 19–20 and 27)
3. In the Default data, why does `student` flip sign once `balance` is included? (pp. 33–34)
4. Sketch a confusion matrix for a fraud detector with 99 % accuracy on data with 1 % fraud. Is it useful? (pp. 44–46)
5. Explain ROC and AUC. What does a 45° ROC mean? (pp. 51–53)
6. The threshold drops from 0.5 to 0.2. Which way do sensitivity, specificity and the ROC operating point move, and what decides whether that is an improvement? (pp. 50 and 54–56)

Appendix questions (optional). Derive the LDA discriminant from Bayes' rule assuming $X \mid Y = k \sim \mathcal{N}(\mu_k, \Sigma)$ (pp. 90–91; step-by-step pp. 108–111). When does naive Bayes outperform logistic regression even though its independence assumption is wrong? (pp. 104 and 116)

Ten things to remember

1. Linear regression is wrong for a categorical Y.
2. Logistic regression: linear in the log-odds, S-shaped in p.
3. e^{β_j} multiplies the *odds*, not the probability.
4. Estimation is by maximum likelihood; no closed form.
5. Univariate slopes can flip sign — watch for confounding.
6. A model outputs a probability; a *threshold* turns it into a decision.
7. Accuracy alone is meaningless on imbalanced data.
8. Confusion matrix $\Rightarrow$ sensitivity, specificity, precision — always at a stated threshold.
9. ROC sweeps every threshold; AUC $= \Pr(\hat{p}_{\text{pos}} > \hat{p}_{\text{neg}})$ is threshold-free.
10. Threshold choice is a business decision, driven by the relative cost of a false negative and a false positive.

Eleventh, if you read the appendix: LDA, QDA and naive Bayes are all one idea — model π_k and $f_k(x)$, then apply Bayes' rule; they differ only in how they model f_k.

What's next

Chapter 5: *Resampling Methods*.

- Validation set, LOOCV, k-fold cross-validation.
- The bootstrap.
- Why these are the most important tools in your toolbox.

Appendix — what is in here, and why

Nothing here is needed to follow the chapter. The taught thread is **logistic regression and classifier evaluation**; everything below is a side topic, a formal derivation or a longer worked exercise.

Contents of the appendix

- **Generative models — Bayes refresher, LDA, QDA, naive Bayes** (Exercises 4.5–4.7): the whole family, from the prior × likelihood picture to the two discriminants and the naive independence assumption. Read it as one block.
- **Deriving LDA from Bayes' theorem** (Extended Exercise 4.2): where that discriminant comes from, line by line.
- **Naive Bayes by hand** (Extended Exercise 4.3): a longer second pass over Exercise 4.7, and how naive Bayes relates to LDA.
- **Comparing the classifiers** (Extended Exercise 4.4): logistic regression vs. LDA / QDA / naive Bayes / KNN — when each wins, the Bayes optimum, and the head-to-head lab cell.
- **Behind logistic regression:** log-likelihood, deviance and IRLS as three views of one optimisation; the softmax for $K > 2$.
- **GLMs and Poisson regression:** the umbrella over linear, logistic and Poisson regression, with Bikeshare. Not examinable.

Exercise numbers are unchanged: 4.5–4.7 and Extended 4.2–4.4 live here, not in the main flow.

Three equivalent ways to think about the optimisation

- **Maximum likelihood**: find $\hat{\beta}$ that makes the observed data most probable under the model.
- **Cross-entropy minimisation**: $-\log L$ is exactly the binary cross-entropy loss — the same objective deep nets use.
- **Iteratively reweighted least squares (IRLS)**: solve a sequence of weighted regressions. Equivalent to Newton's method on the log-likelihood.

Takeaway

All three give the same $\hat{\beta}$. Which view to take is a matter of taste; deep-learning courses prefer cross-entropy, statistics courses prefer maximum likelihood.

From two classes to K: the softmax

Multinomial logistic model

Pick a baseline class (say K) and set, for $k = 1, \dots, K-1$,

$$\Pr(Y = k \mid X) = \frac{e^{\beta_{k\mathbf{0}} + \beta_k^\top X}}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{\ell\mathbf{0}} + \beta_\ell^\top X}}, \qquad \Pr(Y = K \mid X) = \frac{1}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{\ell\mathbf{0}} + \beta_\ell^\top X}}.$$

How to read this

Coefficients depend on the chosen baseline, but predictions are invariant to the choice. The equivalent symmetric "softmax" parameterisation removes the need for a baseline — this is the form used in deep learning.

Takeaway

The K outputs are non-negative and sum to 1 by construction, so they are genuine probabilities. Predict the class with the largest numerator.

Two philosophies of classification

Discriminative

Model $\Pr(Y \mid X)$ directly. *Logistic regression* does this.

Generative

Model the priors π_k and class-conditional densities $f_k(X)$, then apply Bayes' rule.

Which to prefer?

Discriminative models assume *less* and often predict better when data are plentiful. Generative models (LDA, QDA, naive Bayes) add structure through f_k, which pays off when n is small, when classes are well separated, or when some features are missing — situations where logistic regression can be unstable.

In industry — Screening: the prior is the prevalence

Generative models make the base rate explicit: in a diagnostic or fraud screen, π_k *is* the prevalence. When the condition is rare, even a sharp test flags mostly false positives — which is why a screening result is confirmed, not acted on.

Bayes' theorem in the classification setting

Bayes posterior

$$\Pr(Y = k \mid X = x) = \frac{\pi_k \, f_k(x)}{\sum_{\ell=1}^{K} \pi_\ell \, f_\ell(x)}.$$

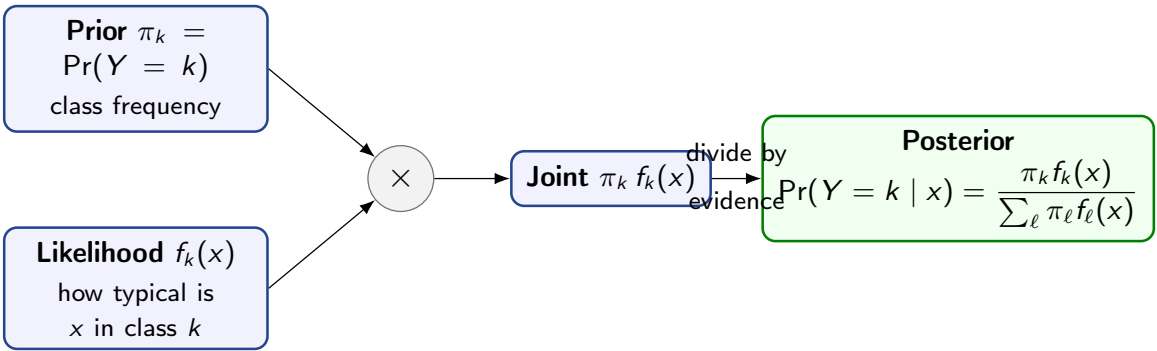
Read it as **posterior** $\propto$ **prior** $\times$ **likelihood**: the numerator scores class k, and the denominator merely renormalises so the K posteriors sum to one.

How to read this

Symbol	Meaning
$\pi_k = \Pr(Y = k)$	Prior class probability (easy: class proportions in the training data)
$f_k(x)$	Class-conditional density of X given $Y = k$

Generative methods differ in *how* they estimate f_k.

Bayes' theorem as a flow: prior $\times$ likelihood $\to$ posterior



Takeaway

Every generative classifier — LDA, QDA, naive Bayes — *is* this same flow. They differ only in *how they model the likelihood* $f_k(x)$. Classify x to the class with the largest posterior.

Linear Discriminant Analysis: the assumption

Assumption

Within each class k, the predictor vector X is multivariate normal with class-specific mean μ_k but a *common* covariance matrix Σ.

Why this assumption is interesting

With a common Σ, the difference between two posterior log-probabilities reduces to a *linear* function of x. So LDA gives linear decision boundaries — by assumption, not by design.

The LDA discriminant

Plugging Gaussian densities into Bayes' rule and dropping common factors leaves the linear discriminant:

LDA discriminant

$$\delta_k(x) = x^\top \Sigma^{-1} \mu_k - \tfrac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \log \pi_k.$$

Classify x to the class k that maximises $\delta_k(x)$: it is a *score*, not a probability — larger means more evidence for class k. Because $\delta_k(x)$ is *linear* in x, the boundary between any two classes is a straight line (hyperplane) — the "linear" in LDA.

Reading the LDA discriminant

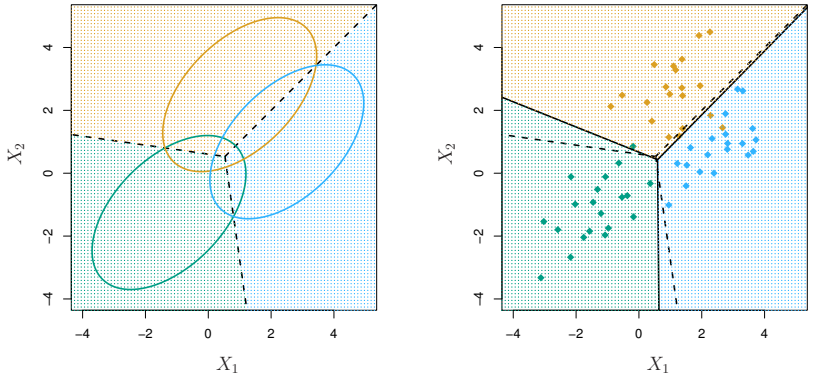
How to read this

Term	Role
$x^\top \Sigma^{-1} \mu_k$	Project x onto the direction of class-k's mean, weighted by Σ^{-1}
$-\frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k$	Penalty for distant means (a constant per class)
$\log \pi_k$	Prior boost: bigger classes win ties

Takeaway

LDA is a weighted nearest-mean classifier, with weights given by the (inverse) covariance and a prior tilt for class size.

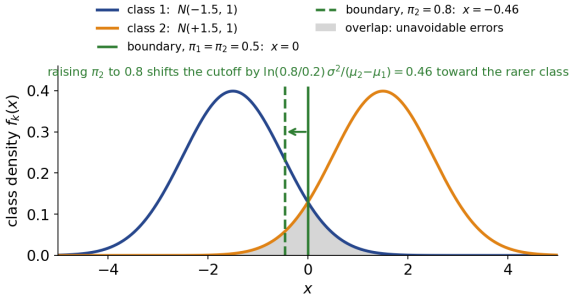
LDA decision boundary on simulated data



Three-class LDA on simulated Gaussians: dashed = Bayes optimum, solid = LDA estimated from training data. The two are very close.

Source: Figure 4.6 — ISLP, James et al. (2023).

LDA with one predictor: two Gaussians and a cutoff



Takeaway

LDA compares the prior-weighted densities $\pi_k f_k(x)$: equal priors put the cutoff at the midpoint 0; $\pi_2 = 0.8$ slides it by $\log(0.8/0.2)\,\sigma^2/(\mu_2-\mu_1) = 0.46$ toward the rarer class. The shaded overlap is error even Bayes cannot remove.

Exercise 4.5 — LDA discriminant in one dimension [Math]

Exercise

Two classes with $X \mid Y = k \sim \mathcal{N}(\mu_k, \sigma^2)$, $\sigma^2 = 1$, $\mu_0 = 0$, $\mu_1 = 2$. Recall $\delta_k(x) = x\dfrac{\mu_k}{\sigma^2} - \dfrac{\mu_k^2}{2\sigma^2} + \log \pi_k$.

1. With equal priors, write δ_0, δ_1 and find the decision boundary.
2. Classify $x = 1.5$.
3. Now take priors $\pi_0 = 0.8$, $\pi_1 = 0.2$. Recompute the boundary and reclassify $x = 1.5$.
4. Give the general boundary formula and check it against part (3).

Solution — Exercise 4.5 (1/2)

Solution

Method. With a shared variance we classify by the larger discriminant score $\delta_k(x)$; the boundary is the x where the two scores tie. Substitute $\mu_0 = 0$, $\mu_1 = 2$, $\sigma^2 = 1$ throughout.

(1) Equal priors. Substituting into $\delta_k(x) = x\mu_k/\sigma^2 - \mu_k^2/(2\sigma^2) + \log \pi_k$:

$$\delta_0(x) = 0 + \log \tfrac{1}{2}, \qquad \delta_1(x) = 2x - \tfrac{4}{2} + \log \tfrac{1}{2} = 2x - 2 + \log \tfrac{1}{2}.$$

The equal $\log \pi_k$ terms cancel in the comparison, leaving $\delta_0 = 0$ vs. $\delta_1 = 2x - 2$. Setting them equal: $0 = 2x - 2 \Rightarrow x = 1$ — exactly the midpoint $(\mu_0 + \mu_1)/2$ of the means, as symmetry demands (equal priors, equal spread). Assign class 1 if $x > 1$.

(2) $\delta_1(1.5) - \delta_0(1.5) = 2(1.5) - 2 = 1 > 0$, so $x = 1.5 \Rightarrow$ class 1 (consistent with $1.5 > 1$).

Solution — Exercise 4.5 (2/2)

Solution

(3) Unequal priors. Now the $\log \pi_k$ terms differ: $\delta_0 = \log 0.8 = -0.223$ and $\delta_1 = 2x - 2 + \log 0.2 = 2x - 2 - 1.609$. Setting equal:

$$-0.223 = 2x - 3.609 \Rightarrow 2x = 3.386 \Rightarrow x = 1.693.$$

Assign class 1 if $x > 1.693$. Now $x = 1.5 < 1.693 \Rightarrow$ class 0: same data point, opposite decision, purely because class 1 became rare — the rarer class requires stronger evidence.

(4) General boundary.

$$x^\star = \frac{\mu_0 + \mu_1}{2} - \frac{\sigma^2}{\mu_1 - \mu_0} \log \frac{\pi_1}{\pi_0}.$$

Check: $\frac{0+2}{2} - \frac{1}{2} \log \frac{0.2}{0.8} = 1 - 0.5(-1.386) = 1.693$. ✓ The prior term shifts the midpoint by $+0.693$ *toward the rare class's mean*, enlarging the common class's region.

Common mistake: shifting the boundary toward the common class. Rarity penalises class 1 *through* $\log \pi_1$, *so the boundary moves toward* μ_1 *and class* 0 *wins more of the axis.*

Quadratic Discriminant Analysis: relaxing the assumption

QDA assumption & discriminant (quadratic in x)

Each class keeps its *own* covariance, $X \mid Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$, giving

$$\boxed{\delta_k(x) = -\tfrac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k) - \tfrac{1}{2}\log|\Sigma_k| + \log \pi_k.}$$

How to read this

Term	Role
$-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)$	Squared (Mahalanobis) distance of x from class-k's centre
$-\frac{1}{2}\log\lvert\Sigma_k\rvert$	Volume penalty: down-weights classes with large spread
$\log \pi_k$	Prior boost: bigger classes win ties

Since each Σ_k differs, the $x^\top \Sigma_k^{-1} x$ term no longer cancels — boundaries become *ellipses or hyperbolas*, not lines.

LDA vs. QDA: a bias-variance choice

LDA

Estimates one Σ ($\sim p^2/2$ parameters). Lower variance; biased if class covariances really differ.

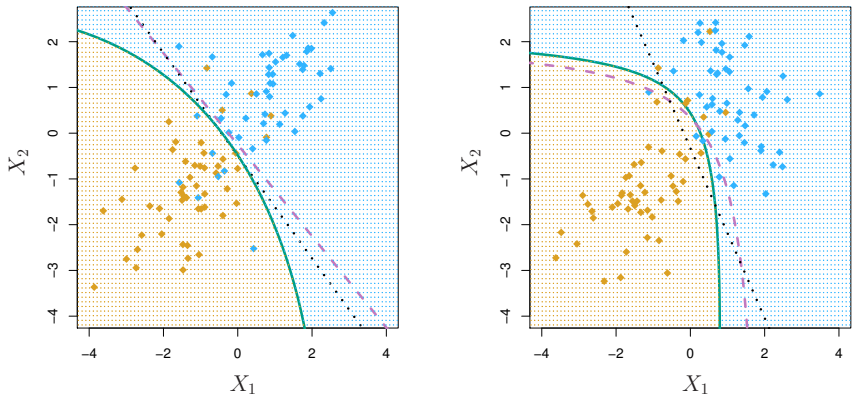
QDA

Estimates K separate Σ_k ($\sim Kp^2/2$ parameters). Lower bias; high variance when n is small.

Practical rule

Use LDA when classes look similar in spread; use QDA when class covariances are visibly different and you have plenty of data per class.

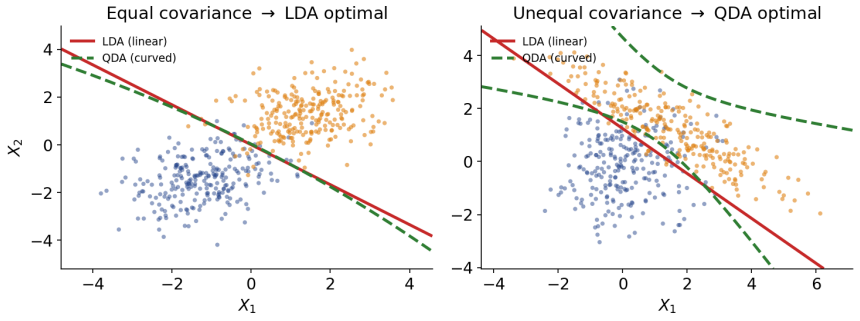
LDA vs. QDA: visual



Equal covariances (left) ⇒ LDA wins. Different covariances (right) ⇒ QDA's curved boundary is closer to optimal.

Source: Figure 4.9 — ISLP, James et al. (2023).

LDA vs. QDA decision boundaries (simulated)



Takeaway

Equal covariances (left): LDA's line all but coincides with QDA. Unequal covariances (right): only QDA's curved boundary tracks the true class structure.

Exercise 4.6 — LDA vs. QDA: parameters and bias-variance [Concept]

Exercise

Consider K classes and p predictors.

1. How many free covariance parameters does LDA estimate? How many does QDA estimate?
2. Evaluate both for $p = 5$, $K = 3$.
3. Which method has higher variance, and which has higher bias? Under what condition is LDA's extra bias zero?
4. State a practical rule for choosing between them.

Solution — Exercise 4.6 (1/2)

Solution

Method. Count the free entries in the covariance objects each method must estimate; the difference is the extra variance QDA pays for its extra flexibility.

(1) Parameter counts. A covariance matrix is symmetric $p \times p$: p variances on the diagonal plus $p(p-1)/2$ distinct covariances off it, i.e.

$$p + \frac{p(p-1)}{2} = \frac{p(p+1)}{2} \text{ free entries.}$$

LDA assumes one *shared* Σ: $p(p+1)/2$ parameters. QDA estimates one Σ_k per class: $K \cdot p(p+1)/2$ parameters — the count scales with the number of classes.

(2) For $p = 5$, $K = 3$. $p(p+1)/2 = 5 \cdot 6/2 = 15$. LDA: 15; QDA: $3 \times 15 = 45$ covariance parameters. Both additionally estimate $Kp = 15$ means and $K - 1 = 2$ free priors, so the totals are $15 + 15 + 2 = 32$ (LDA) vs. $45 + 15 + 2 = 62$ (QDA) — QDA nearly doubles the parameter budget.

Solution — Exercise 4.6 (2/2)

Solution

(3) Bias-variance. QDA is more flexible: *higher variance, lower bias* — its extra 30 parameters (here) buy curved boundaries. LDA is more rigid: *lower variance, higher bias* — but its extra bias is *zero when the true class covariances are actually equal* ($\Sigma_1 = \cdots = \Sigma_K$): then both methods target the same truth and LDA gets there with fewer parameters, so it is preferable.

(4) Rule of thumb. Use LDA when n is small or the classes look similarly spread; use QDA when the class covariances are visibly different and you have plenty of data per class.

Interpretation. This is the Chapter 2 bias-variance trade-off in classifier form: QDA is the "more flexible" model, and whether its lower bias pays depends on how much data must stretch across K covariance matrices.

Common mistake

counting p^2 entries per covariance matrix — symmetry roughly halves the count to $p(p+1)/2$. And remember QDA's count grows with K: many classes with few observations each is its worst case.

Naive Bayes: extreme simplification

The independence assumption

Within each class, the features are independent: $f_k(x) = \prod_{j=1}^{p} f_{kj}(x_j)$.

Takeaway

The assumption is almost always wrong — features are usually correlated. But the simplification massively reduces variance: each f_{kj} is just a one-dimensional density.

Why it works in practice

Even when independence fails, naive Bayes often *ranks* observations correctly — which is all a classifier needs. It routinely beats more sophisticated methods when n is small and p is large.

Exercise 4.7 — Naive Bayes from a small table [Math]

Exercise

A spam filter has priors $\Pr(\text{Spam}) = 0.4$, $\Pr(\text{Ham}) = 0.6$ and two binary features: $F_1 =$ "contains the word *free*", $F_2 =$ "contains a link".

	$\Pr(F_1 = 1 \mid \cdot)$	$\Pr(F_2 = 1 \mid \cdot)$
Spam	0.6	0.7
Ham	0.1	0.2

A new email contains *both* features.

1. Using the conditional-independence assumption, compute the unnormalised score for each class.
2. Compute $\Pr(\text{Spam} \mid F_1 = 1, F_2 = 1)$.
3. Classify the email.
4. What does the naive assumption buy you, and what does it cost?

Solution — Exercise 4.7 (1/2)

Solution

Method. Bayes' rule scores each class by its numerator $\pi_k f_k(x)$; naive Bayes factorises the likelihood as $f_k(x) = \prod_j \Pr(F_j \mid k)$. The denominator is common to both classes, so we compare numerators first and normalise at the end.

(1) Unnormalised scores $(\pi_k \prod_j \Pr(F_j \mid k))$:

$$\begin{aligned} \text{Spam} &: 0.4 \times 0.6 \times 0.7 = 0.4 \times 0.42 = 0.168, \\ \text{Ham} &: 0.6 \times 0.1 \times 0.2 = 0.6 \times 0.02 = 0.012. \end{aligned}$$

(2) Posterior. Normalise by the sum $0.168 + 0.012 = 0.180$:

$$\Pr(\text{Spam} \mid F_1, F_2) = \frac{0.168}{0.180} \approx 0.933, \qquad \Pr(\text{Ham} \mid F_1, F_2) = \frac{0.012}{0.180} \approx 0.067,$$

and the two posteriors sum to 1. ✓

Solution — Exercise 4.7 (2/2)

Solution

(3) Decision. $0.933 > 0.5 \Rightarrow$ classify as **Spam**. In odds form: posterior odds $= 0.168/0.012 = 14$, the product of prior odds $0.4/0.6 = 0.67$ and likelihood ratio $0.42/0.02 = 21$ — the two features overwhelm the prior that favoured Ham.

(4) Trade-off. Assuming the features are independent *within each class* means we only estimate one-dimensional marginals f_{kj} — here just two numbers per class instead of a joint table over all feature combinations — drastically reducing the number of parameters and hence the variance. The cost is bias: the assumption is usually false (e.g. e-mails containing *free* disproportionately also contain links), so the posterior probabilities are miscalibrated — yet the *ranking* of observations is often still good enough to classify well.

Common mistake

dropping the prior and normalising the likelihoods alone — $0.42/(0.42 + 0.02) \approx 0.955$ overstates the spam probability. The prior matters; with rare positives it can flip the decision outright.

Extended Exercise 4.2 — LDA from Bayes' theorem [Math]

Extended exercise (15 min)

Two classes with a single predictor X. Within class k, $X \mid Y = k \sim \mathcal{N}(\mu_k, \sigma^2)$ with a *shared* variance $\sigma^2 = 4$ and means $\mu_0 = 1$, $\mu_1 = 5$. Priors $\pi_0 = 0.7$, $\pi_1 = 0.3$.

1. Starting from Bayes' theorem, show that comparing $\log[\pi_k f_k(x)]$ across classes reduces to the linear discriminant $\delta_k(x) = x\,\mu_k/\sigma^2 - \mu_k^2/(2\sigma^2) + \log \pi_k$.
2. Derive the decision boundary $x^\star$ where $\delta_0 = \delta_1$; give the general formula and evaluate it.
3. For $x = 4$, compute δ_0, δ_1 and the posterior $\Pr(Y = 1 \mid x = 4)$; classify the point.
4. How does QDA change the derivation and the boundary?

Solution — Extended Exercise 4.2 (1/3)

Solution

Method. Everything is Bayes' rule; the only algebra is separating the terms that depend on k from those that do not — possible precisely because σ^2 is shared.

(1) Bayes $\Rightarrow$ linear discriminant. Bayes gives $\Pr(Y = k \mid x) = \pi_k f_k(x) / \sum_\ell \pi_\ell f_\ell(x)$; the denominator is common, and log is monotone, so we classify by the largest $\log[\pi_k f_k(x)]$. With $f_k(x) = (2\pi\sigma^2)^{-1/2} e^{-(x-\mu_k)^2/(2\sigma^2)}$, expanding the square $(x-\mu_k)^2 = x^2 - 2x\mu_k + \mu_k^2$ gives

$$\log[\pi_k f_k(x)] = \underbrace{-\tfrac{1}{2}\log(2\pi\sigma^2) - \tfrac{x^2}{2\sigma^2}}_{\text{same for every class}} + \underbrace{x\tfrac{\mu_k}{\sigma^2} - \tfrac{\mu_k^2}{2\sigma^2} + \log \pi_k}_{=\,\delta_k(x)} .$$

The first group is identical across classes (σ^2 shared, x^2 has no k), so it cancels in any comparison, leaving the *linear* $\delta_k(x)$ — this cancellation is the whole reason LDA's boundary is a line.

Solution — Extended Exercise 4.2 (2/3)

Solution

(2) Boundary. Set $\delta_0(x) = \delta_1(x)$ and solve; substituting $\mu_0 = 1$, $\mu_1 = 5$, $\sigma^2 = 4$ and $\log \frac{\pi_1}{\pi_0} = \log \frac{0.3}{0.7} = -0.847$:

$$x^\star = \frac{\mu_0 + \mu_1}{2} - \frac{\sigma^2}{\mu_1 - \mu_0} \log \frac{\pi_1}{\pi_0} = \frac{1+5}{2} - \frac{4}{4}(-0.847) = 3 + 0.847 = 3.85.$$

Assign class 1 if $x > 3.85$. The prior tilt moves the boundary 0.85 units from the midpoint 3 *toward* the rarer class, which needs stronger evidence.

(3) Posterior at $x = 4$. Substituting into δ_k:

$$\delta_0(4) = \tfrac{4 \cdot 1}{4} - \tfrac{1}{8} + \log 0.7 = 1 - 0.125 - 0.357 = 0.518,$$
$$\delta_1(4) = \tfrac{4 \cdot 5}{4} - \tfrac{25}{8} + \log 0.3 = 5 - 3.125 - 1.204 = 0.671.$$

Since the dropped term cancels, the posterior is the softmax of the δ_k:

$$\Pr(Y = 1 \mid 4) = \frac{e^{0.671}}{e^{0.518} + e^{0.671}} = \frac{1.956}{1.679 + 1.956} \approx 0.538 \ > 0.5 \Rightarrow \text{ class 1},$$

consistent with $4 > 3.85$ — but only barely: $x = 4$ sits close to the boundary, so the posterior is near 0.5.

Solution — Extended Exercise 4.2 (3/3)

Solution

Cross-check via the raw densities. With $\sigma = 2$, $f_k(4) = (2\pi \cdot 4)^{-1/2} e^{-(4-\mu_k)^2/8} = 0.1995\, e^{-(4-\mu_k)^2/8}$:

$$\pi_0 f_0(4) = 0.7 \times 0.1995\, e^{-9/8} = 0.7 \times 0.0648 = 0.0453,$$
$$\pi_1 f_1(4) = 0.3 \times 0.1995\, e^{-1/8} = 0.3 \times 0.1760 = 0.0528,$$

so $\Pr(Y = 1 \mid 4) = \frac{0.0528}{0.0453+0.0528} = \frac{0.0528}{0.0981} \approx 0.538$. ✓ Same answer by the direct route — the discriminants are just a shortcut through Bayes' rule.

(4) QDA. QDA lets each class keep its own variance σ_k^2. Then the $-x^2/(2\sigma_k^2)$ term no longer cancels, so a *quadratic* term in x survives: $\delta_k(x) = -\frac{(x-\mu_k)^2}{2\sigma_k^2} - \frac{1}{2}\log\sigma_k^2 + \log\pi_k$. The boundary becomes quadratic (a conic in higher dimensions). QDA adds a variance per class — lower bias, higher variance — and reduces to LDA exactly when $\sigma_0^2 = \sigma_1^2$.

Common mistake: treating the δ_k as probabilities. They are unnormalised log-scores (here both ≈ 0.5–0.7); only the softmax of the δ_k gives the posterior.

Extended Exercise 4.3 — Naive Bayes by hand [Math]

Extended exercise (15 min)

A lender labels customers Default (D) or No-default (N) using three binary features: F_1 = high balance, F_2 = student, F_3 = a late payment in the past year. Priors $\pi_D = 0.2$, $\pi_N = 0.8$, with estimated class-conditional likelihoods:

$\Pr(\cdot = \text{Yes} \mid \text{class})$	D	N
F_1 high balance	0.8	0.3
F_2 student	0.3	0.4
F_3 late payment	0.6	0.1

A new customer has all three features present ($F_1 = F_2 = F_3 = \text{Yes}$).

1. Under conditional independence, compute the unnormalised score $\pi_k \prod_j \Pr(F_j \mid k)$ for each class.
2. Compute the posterior $\Pr(\text{D} \mid F_1, F_2, F_3)$ and classify.
3. The prior favoured N. Explain why the decision comes out as it does.
4. Compare naive Bayes with LDA: assumptions, parameters, when each is preferable.

Solution — Extended Exercise 4.3 (1/3)

Solution

Method. Bayes' rule scores each class by its numerator $\pi_k f_k(x)$; the *naive* assumption factorises the class-conditional likelihood into one-dimensional marginals, $f_k(x) = \prod_j \Pr(F_j \mid k)$, so we never estimate a joint table over the three features. The evidence denominator is common to both classes, so we compare numerators first and normalise only at the end.

(1) Unnormalised scores $\pi_k \prod_j \Pr(F_j = \text{Yes} \mid k)$:

$$\begin{aligned} \text{D} &: 0.2\,(0.8 \times 0.3 \times 0.6) = 0.2 \times 0.144 = 0.0288, \\ \text{N} &: 0.8\,(0.3 \times 0.4 \times 0.1) = 0.8 \times 0.012 = 0.0096. \end{aligned}$$

(2) Posterior. Normalise by the sum $0.0288 + 0.0096 = 0.0384$: $\Pr(\text{D} \mid x) = \dfrac{0.0288}{0.0384} = 0.75$ (so $\Pr(\text{N} \mid x) = 0.25$) $\Rightarrow$ classify as **Default**.

Solution — Extended Exercise 4.3 (2/3)

Solution

(3) Why the prior is overturned. Write the posterior as prior odds $\times$ likelihood ratio:

$$\frac{\Pr(\mathrm{D} \mid x)}{\Pr(\mathrm{N} \mid x)} = \frac{\pi_D}{\pi_N} \cdot \frac{\prod_j \Pr(F_j \mid D)}{\prod_j \Pr(F_j \mid N)} = \frac{0.2}{0.8} \cdot \frac{0.144}{0.012} = 0.25 \times 12 = 3.$$

The prior favours N by $4\times$, but the features favour D by $12\times$; net, D is $3\times$ as likely, i.e. $\Pr(\mathrm{D}) = 3/(3+1) = 3/4$. The three "Yes" features — above all the high balance and the late payment, each far more common under D — swamp the base rate.

Common mistake

normalising the likelihoods alone and dropping the prior. That gives $0.144/(0.144 + 0.012) \approx 0.92$ instead of the correct 0.75 — overstating default because it forgets that non-defaulters are $4\times$ more common to begin with.

Solution — Extended Exercise 4.3 (3/3)

Solution

(4) Naive Bayes vs. LDA. Both are *generative*: model the priors π_k and class-conditional densities $f_k(x)$, then apply Bayes.

- **Assumption.** LDA: X is multivariate Gaussian with a *common* covariance Σ, so it models linear correlations between features. Naive Bayes: features are conditionally *independent* within a class, i.e. $f_k(x) = \prod_j f_{kj}(x_j)$; it ignores within-class correlations but handles categorical / binary and mixed features naturally (as here).
- **Parameters.** NB estimates only one-dimensional marginals $\Rightarrow$ very few parameters, low variance. LDA estimates the full Σ ($p(p+1)/2$ entries), which fails when p is comparable to n.
- **When.** NB shines when p is large relative to n or the features are discrete / non-Gaussian; LDA is better when features are roughly Gaussian and their correlations carry signal.

Connection: Gaussian naive Bayes is exactly discriminant analysis with a *diagonal* covariance matrix (all cross-correlations set to zero).

Method comparison: when each wins

Method	Boundary	Best when
Logistic regression	linear	no Gaussian assumption needed
LDA	linear	classes share covariance, normal
QDA	quadratic	classes have unequal covariance
Naive Bayes	flexible (per-feature)	many features, n small
KNN	nonparametric	truth is highly nonlinear

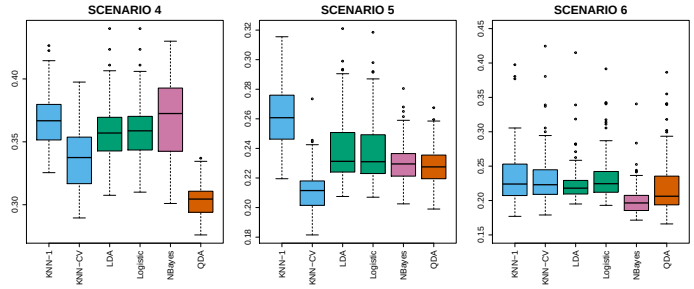
In practice

No method dominates universally. Treat all five as candidates; pick by held-out AUC. With one line of scikit-learn each, the comparison is essentially free.

In industry — Risk and campaign teams: how it is actually reported

Risk teams quote AUC as the Gini coefficient ($2\,\text{AUC} - 1$) and the KS statistic; campaign teams compare models at the *top decile*, because that is where the budget goes. Boosted trees (Chapter 8) often win outright — which is why regulated decisions keep the model they can explain.

Bayes optimum and our methods



Takeaway

Test error per scenario: *no classifier wins everywhere* — the best method in each panel matches its data-generating process. This is why we compare candidates by held-out error.

Source: Figure 4.12 — ISLP, James et al. (2023).

Extended Exercise 4.4 — Choosing a classifier: three cases [Integrative]

Extended exercise (15 min)

For each dataset, pick the most appropriate classifier from {logistic regression, LDA, QDA, naive Bayes, KNN} and justify it in terms of boundary shape, n vs. p, within-class normality, feature type, and number of classes.

A. Credit scoring. $n = 30{,}000$ applicants, $p = 6$ continuous predictors, $K = 2$. The log-odds of default look approximately linear; the two classes have similar spread; income is heavily right-skewed.

B. Tumour diagnosis. $n = 200$ patients, $p = 3$ continuous features that are roughly Gaussian within each class, $K = 2$ (benign / malignant). The malignant class is visibly more variable than the benign class.

C. Spam filtering. $n = 1{,}000$ e-mails, $p = 5{,}000$ binary word-presence indicators, $K = 2$. Features are discrete and many are correlated; $p \gg n$.

Recommend a method for each and name your runner-up.

Solution — Extended Exercise 4.4 (1/2)

Solution

A quick decision checklist. Boundary linear? $\rightarrow$ logistic / LDA. Curved? $\rightarrow$ QDA / KNN. Gaussian within class with equal covariance? $\rightarrow$ LDA; unequal? $\rightarrow$ QDA. $p \gg n$ or discrete features? $\rightarrow$ naive Bayes (or regularised logistic). Highly nonlinear, large n, standardised continuous? $\rightarrow$ KNN.

A — Credit scoring $\Rightarrow$ logistic regression. Large n, small p, binary response, and an approximately linear log-odds make logistic regression the natural first choice. It assumes nothing about the distribution of X, so the skewed income variable is harmless, and its coefficients are interpretable on the odds scale. *Runner-up:* LDA (fine under approximate normality and equal covariance) — but the skew makes logistic safer.

B — Tumour diagnosis $\Rightarrow$ QDA. Features are roughly Gaussian within class, but the covariances clearly *differ*, so the optimal boundary is quadratic — exactly QDA's model. With $p = 3$ each class covariance has only 6 free entries, and $n = 200$ easily supports two of them, so QDA's extra variance is affordable. *Runner-up:* LDA (its lower variance would win if n were much smaller).

Solution — Extended Exercise 4.4 (2/2)

Solution

C — Spam filtering ⇒ naive Bayes. With $p = 5{,}000 \gg n = 1{,}000$, LDA/QDA cannot estimate a $p \times p$ covariance (singular, millions of entries) and unregularised logistic regression would overfit badly. The features are binary, not Gaussian. Naive Bayes assumes conditional independence, needs only one-dimensional marginals, has very low variance, and is the classic strong text baseline. *Runner-up:* L_1/L_2-regularised logistic regression.

Where the remaining methods fit.

- **LDA** is the pick when features are Gaussian with a shared covariance — a middle ground between logistic and QDA; it is the closest call in scenario A.
- **KNN** wins only when the boundary is highly nonlinear, the features are continuous and standardised, and n is large relative to p. None of A–C qualifies; C is catastrophic for KNN (curse of dimensionality at $p = 5{,}000$).

Takeaway

Match the method to the boundary shape, the n-vs-p budget, and the feature type — then confirm the choice with held-out AUC.

LDA, QDA, naive Bayes head-to-head

```
from sklearn.discriminant_analysis import (
    LinearDiscriminantAnalysis as LDA,
    QuadraticDiscriminantAnalysis as QDA) # Gaussian discriminants
from sklearn.naive_bayes import GaussianNB # naive Bayes (Gaussian)

for name, mdl in [("LDA", LDA()),
                  ("QDA", QDA()),
                  ("NB", GaussianNB())]: # three models, same split
    mdl.fit(Xtr, ytr) # estimate means/covariances
    p = mdl.predict_proba(Xte)[:, 1] # posterior P(class 1) on test
    print(name, "AUC:", roc_auc_score(yte, p)) # compare by AUC
```

Continues the Python lab: `Xtr`, `Xte`, `ytr`, `yte` are the same train/test split used for logistic regression, so the AUCs are directly comparable. Add each model to the ROC axes with one more `RocCurveDisplay.from_estimator(...)` call.

The GLM framework: one umbrella

Linear, logistic, and Poisson regression are all special cases of:

Generalised linear model

$$g\big(E[Y \mid X]\big) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p,$$

where g is the *link function*.

Distribution	Link	Use
Gaussian	identity	continuous Y (Ch. 3)
Bernoulli	logit	binary Y (this chapter)
Poisson	log	counts
Gamma	inverse / log	positive continuous

How to read this

The *link* g puts the mean $E[Y \mid X]$ on a scale where a linear model fits; invert g to predict on the original scale. Maximum likelihood fits the whole family.

Poisson regression for counts

Model

$Y \mid X \sim \text{Poisson}\big(\lambda(X)\big)$ with $\log \lambda(X) = \beta_0 + \beta^\top X$.

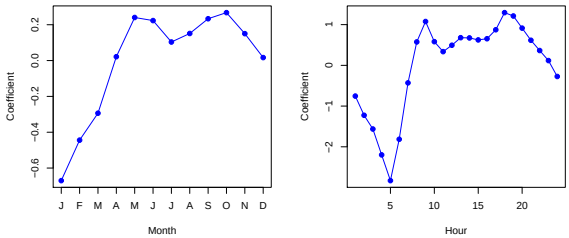
How to read this

Coefficients are on the log-rate scale: $\exp(\beta_j)$ is the multiplicative effect on the expected count. Mean equals variance for Poisson; if your data shows variance much greater than the mean, look at negative binomial instead.

Bikeshare

The Bikeshare data records bikes rented per hour. A Poisson model with hour-of-day, weather, and weekday gives multiplicative effects: "rush-hour increases rental rate by a factor of 4."

Bikeshare: Poisson captures the commute rhythm



Takeaway

Fitted Poisson coefficients by month and hour: ridership rises through the warmer months and peaks at the two commutes. On the *log*-rate scale these act *multiplicatively* on the expected count, so predicted ridership can never go negative.

Source: Figure 4.15 — ISLP, James et al. (2023).

Poisson regression on Bikeshare

```
Bike = load_data("Bikeshare") # hourly bike-rental counts
X = sm.add_constant(Bike[["temp", "hum", "windspeed"]]) # add intercept
res = sm.GLM(Bike["bikers"], X, # count response
             family=sm.families.Poisson()).fit() # log link for counts
print(res.summary()) # coefficients on log-rate scale
```

Coefficients are on the log-rate scale; `exp(coef)` gives the multiplicative effect on hourly ridership.