Logistic Regression & Generalized Linear Models

A Complete, Step-by-Step Treatment for the Quant / ML Researcher

Every result derived; every step justified; worked examples throughout

Abstract

Logistic regression is the canonical classifier and the gateway to generalized linear models. This document derives it from the exponential family, explains why the logit link is the "natural" choice, proves the concavity of its log-likelihood (and hence the uniqueness of its solution), derives the IRLS/Newton training algorithm step by step, treats separation and its regularization cure, and develops calibration — the property that makes logistic regression the default when you need genuine probabilities for expected-value decisions. Worked examples and quant connections throughout.

Contents

1 From Linear Regression to Classification

Linear regression models a real-valued mean. For a binary label $y \in \{0, 1\}$ we instead model the *probability* $p = \Pr(y = 1 \mid \mathbf{x})$. We cannot just write $p = \mathbf{x}^\top \boldsymbol{\beta}$: the right side ranges over all of $\mathbb{R}$ while a probability must lie in $[0, 1]$. We need a function squashing $\mathbb{R}$ into $(0, 1)$. The principled choice comes from the exponential family.

1.1 Why the logit link is natural

Recall from the probability note that the Bernoulli in exponential-family form has *natural parameter* equal to the log-odds $\eta = \log \frac{p}{1-p}$. A generalized linear model places the linear predictor on the natural parameter: $\eta = \mathbf{x}^\top \boldsymbol{\beta}$. Inverting,

$$\log \frac{p}{1-p} = \mathbf{x}^\top \boldsymbol{\beta} \iff p = \sigma(\mathbf{x}^\top \boldsymbol{\beta}), \qquad \sigma(z) = \frac{1}{1+e^{-z}}.$$

So the sigmoid is not arbitrary — it is the inverse of the Bernoulli's canonical (logit) link. Two identities we will use constantly: $\sigma(-z) = 1-\sigma(z)$ and $\sigma'(z) = \sigma(z)(1-\sigma(z))$ (differentiate directly). The latter says the sigmoid's slope equals the variance of the implied Bernoulli — maximal at $p = 1/2$, vanishing at the extremes.

Intuition

A unit increase in feature x_j adds β_j to the log-odds, hence *multiplies the odds* by e^{β_j}. This is the correct interpretation of logistic coefficients — multiplicative on odds, not additive on probability. The decision boundary $\{\mathbf{x} : \mathbf{x}^\top \boldsymbol{\beta} = 0\}$ (where $p = 1/2$) is a hyperplane, so logistic regression is a *linear* classifier despite the nonlinear link; the nonlinearity only shapes how confidence grows as you move away from the boundary.

2 Maximum Likelihood and the Concave Log-Likelihood

With independent Bernoulli observations, the likelihood is $\prod_i p_i^{y_i}(1-p_i)^{1-y_i}$, so the log-likelihood is

$$\ell(\boldsymbol{\beta}) = \sum_i \big[y_i \log p_i + (1-y_i)\log(1-p_i)\big] = \sum_i \big[y_i \mathbf{x}_i^\top \boldsymbol{\beta} - \log(1 + e^{\mathbf{x}_i^\top \boldsymbol{\beta}})\big],$$

substituting $p_i = \sigma(\mathbf{x}_i^\top \boldsymbol{\beta})$ and simplifying. Minimizing $-\ell$ is exactly minimizing **binary cross-entropy** — which, from the probability note, is minimizing the KL divergence from the data to the model. No closed form exists (the equations are transcendental), so we optimize numerically; the good news is that the landscape is benign.

Theorem 1 (Concavity)**.** *$\ell(\boldsymbol{\beta})$ is concave; if $\mathbf{X}$ has full column rank it is strictly concave, so the MLE (if it exists) is unique.*

Proof. Compute the gradient (score) and Hessian. Using $\partial p_i / \partial \boldsymbol{\beta} = p_i(1-p_i)\mathbf{x}_i$,

$$\nabla \ell = \sum_i (y_i - p_i)\mathbf{x}_i = \mathbf{X}^\top(\mathbf{y} - \boldsymbol{p}), \qquad \nabla^2 \ell = -\sum_i p_i(1-p_i)\mathbf{x}_i\mathbf{x}_i^\top = -\mathbf{X}^\top \mathbf{W}\mathbf{X},$$

where $\mathbf{W} = \operatorname{diag}(p_i(1-p_i)) \succ 0$ (each weight is a Bernoulli variance, strictly positive for $0 < p_i < 1$). Thus $\nabla^2 \ell = -\mathbf{X}^\top \mathbf{W}\mathbf{X} \preceq 0$ (negative semidefinite), so ℓ is concave; with full-rank $\mathbf{X}$ it is negative definite, giving strict concavity. □

The score equation $\mathbf{X}^\top(\mathbf{y}-\boldsymbol{p})=\mathbf{0}$ is the nonlinear analogue of the OLS normal equations: it forces the residuals y_i-p_i to be orthogonal to the features. A consequence worth noting: if there is an intercept (a column of ones), the average predicted probability equals the empirical base rate — the source of logistic regression's automatic calibration.

3 Training: Newton–Raphson and IRLS

Newton's method (from the optimization note) uses the update $\boldsymbol{\beta}^{(t+1)}=\boldsymbol{\beta}^{(t)}-(\nabla^2\ell)^{-1}\nabla\ell$. Substituting,

$$\boldsymbol{\beta}^{(t+1)}=\boldsymbol{\beta}^{(t)}+(\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top(\mathbf{y}-\boldsymbol{p}).$$

Algebraic rearrangement reveals this is a **weighted least squares** fit at each step — *Iteratively Reweighted Least Squares (IRLS)*:

$$\boldsymbol{\beta}^{(t+1)}=(\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{W}\mathbf{z},\qquad \mathbf{z}=\mathbf{X}\boldsymbol{\beta}^{(t)}+\mathbf{W}^{-1}(\mathbf{y}-\boldsymbol{p}),$$

where $\mathbf{z}$ is a "working response." Each iteration solves a weighted regression; the weights $\mathbf{W}$ down-weight confident points (small $p(1-p)$) and emphasize uncertain ones near the boundary. Because the problem is concave, IRLS converges to the global MLE. At very large scale one uses L-BFGS or SGD instead, but IRLS is the classical, fast method for moderate dimensions.

Perfect separation: when the MLE runs off to infinity

If some hyperplane perfectly separates the two classes, the likelihood can be pushed arbitrarily close to 1 by scaling $\boldsymbol{\beta}$ up without bound — so the MLE *does not exist* (the supremum is not attained), and $\left\|\hat{\boldsymbol{\beta}}\right\|\to\infty$. Symptoms: coefficients exploding, standard errors enormous, IRLS failing to converge. This is common with rare-event labels and strong predictors in finance. Cures: an ℓ_2 penalty (which makes the penalized objective coercive, so a finite maximizer exists), Firth's penalized likelihood (a Jeffreys-prior correction that also reduces small-sample bias), or any proper Bayesian prior. The fix is the same regularization that helps with multicollinearity, here playing a different role: existence rather than stability.

4 Inference

Asymptotically $\hat{\boldsymbol{\beta}}\sim\mathcal{N}(\boldsymbol{\beta},\mathcal{I}^{-1})$ with Fisher information $\mathcal{I}=\mathbf{X}^\top\mathbf{W}\mathbf{X}$ (the negative expected Hessian). Three asymptotically equivalent tests: the **Wald** test ($\hat\beta_j/\widehat{\mathrm{se}}$, easy but unreliable near separation), the **likelihood-ratio** test ($2[\ell_{\text{full}}-\ell_{\text{reduced}}]\sim\chi^2$, the most reliable and reparameterization-invariant), and the **score** test. Model fit is summarized by the deviance -2ℓ and McFadden's pseudo-$R^2=1-\ell_{\text{full}}/\ell_{\text{null}}$.

5 Multiclass and Ordinal Extensions

For $K>2$ classes, **softmax (multinomial) regression** models $\Pr(y=k\mid\mathbf{x})=e^{\mathbf{x}^\top\boldsymbol{\beta}_k}/\sum_l e^{\mathbf{x}^\top\boldsymbol{\beta}_l}$, fixing one class as reference for identifiability (otherwise adding a constant to every $\boldsymbol{\beta}_k$ leaves probabilities unchanged). The loss generalizes to categorical cross-entropy, and the softmax's normalizer is the log-sum-exp whose convexity we noted in the optimization document. For *ordered* categories (ratings, credit grades), ordinal/proportional-odds logistic regression respects the ordering with a single set of slopes and class-specific thresholds.

6 Calibration — Why Quants Default to Logistic Regression

A classifier is **calibrated** if among cases it assigns probability p, a fraction p are actually positive. Logistic regression fit by MLE is calibrated by construction on the training distribution: the score equation forces the average predicted probability to match the empirical frequency. This matters enormously when the probability *is* the product — a default probability feeding expected loss, a win probability feeding Kelly position sizing, an event probability feeding an expected-value bet. A model that merely *ranks* well (high AUC) but is miscalibrated will misprice these decisions.

Intuition

Regularization and class rebalancing both *break* calibration: shrinking coefficients pulls probabilities toward the base rate, and oversampling the minority class shifts the implied prior. After such interventions, recalibrate — Platt scaling (fit a 1-D logistic on the scores) or isotonic regression (a monotone non-parametric fit) — and check a reliability diagram. For a rebalanced model, the intercept can be analytically corrected back to the true base rate. In trading, an uncalibrated probability is worse than useless because it produces systematically mis-sized bets.

Worked Example

A credit model outputs $p = 0.1$ for a pool of loans. If calibrated, about 10% default — so the expected loss per dollar is $0.1 \times \text{LGD}$, directly usable for pricing. If the model were trained on a 50/50 rebalanced sample without correction, that 0.1 might correspond to a true default rate of 1% or 3% — and every price built on it is wrong. The calibration step is not optional polish; it is the bridge from a score to a decision.

7 Assumptions and Practical Notes

- Log-odds linear in features (add splines/interactions if violated; Box–Tidwell test).
- Independent observations (cluster-robust SEs for grouped/panel data).
- Low multicollinearity for stable, interpretable coefficients.
- Adequate events-per-variable ($\gtrsim$ 10–20) to avoid small-sample bias and separation.
- Class imbalance: use class weights or threshold tuning; evaluate with PR-AUC, not accuracy; recalibrate after rebalancing.

8 A Fully Worked IRLS Iteration

Abstract derivations become concrete only when you push real numbers through them. We fit a one-feature logistic model $\Pr(y = 1 \mid x) = \sigma(\beta_0 + \beta_1 x)$ to five points and carry out one complete IRLS step by hand.

8.1 The data and the starting point

i	1	2	3	4	5
x_i	-2	-1	0	1	2
y_i	0	0	1	1	1

The design matrix has rows $(1, x_i)$. Start from $\boldsymbol{\beta}^{(0)} = (0, 0)^\top$, so every linear predictor $\eta_i = 0$ and every probability $p_i = \sigma(0) = 0.5$.

8.2 Step 1 — the working weights and residuals

The IRLS weights are $w_i = p_i(1-p_i) = 0.5 \times 0.5 = 0.25$ for all five points, so $\mathbf{W} = 0.25\,\mathbf{I}_5$. The raw residuals are $y_i - p_i$:

$$\mathbf{y} - \boldsymbol{p} = (0-0.5,\, 0-0.5,\, 1-0.5,\, 1-0.5,\, 1-0.5)^\top = (-0.5, -0.5, 0.5, 0.5, 0.5)^\top.$$

8.3 Step 2 — the score (gradient)

The gradient is $\nabla \ell = \mathbf{X}^\top(\mathbf{y} - \boldsymbol{p})$. The first component (intercept column of ones) is the sum of residuals; the second (the x column) is $\sum x_i(y_i - p_i)$:

$$\sum_i (y_i - p_i) = -0.5 - 0.5 + 0.5 + 0.5 + 0.5 = 0.5,$$

$$\sum_i x_i(y_i - p_i) = (-2)(-0.5) + (-1)(-0.5) + 0(0.5) + 1(0.5) + 2(0.5) = 1 + 0.5 + 0 + 0.5 + 1 = 3.0.$$

So $\nabla \ell = (0.5,\, 3.0)^\top$. The large second component signals that increasing the slope β_1 will raise the likelihood — exactly what we expect, since the data trend upward.

8.4 Step 3 — the Fisher information (negative Hessian)

We need $\mathbf{X}^\top\mathbf{W}\mathbf{X}$. With constant weight 0.25, this is $0.25\,\mathbf{X}^\top\mathbf{X}$. Compute $\mathbf{X}^\top\mathbf{X}$ from the columns: the $(1,1)$ entry is $\sum 1 = 5$; the $(1,2)$ entry is $\sum x_i = 0$; the $(2,2)$ entry is $\sum x_i^2 = 4+1+0+1+4 = 10$. Hence

$$\mathbf{X}^\top\mathbf{W}\mathbf{X} = 0.25 \begin{pmatrix} 5 & 0 \\ 0 & 10 \end{pmatrix} = \begin{pmatrix} 1.25 & 0 \\ 0 & 2.5 \end{pmatrix}.$$

The off-diagonal zeros are a gift of the symmetric, centered design ($\sum x_i = 0$): the intercept and slope updates decouple.

8.5 Step 4 — the Newton/IRLS update

The update is $\boldsymbol{\beta}^{(1)} = \boldsymbol{\beta}^{(0)} + (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\nabla\ell$. The information matrix is diagonal, so its inverse is $\mathrm{diag}(1/1.25,\, 1/2.5) = \mathrm{diag}(0.8, 0.4)$:

$$\boldsymbol{\beta}^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0.8 & 0 \\ 0 & 0.4 \end{pmatrix} \begin{pmatrix} 0.5 \\ 3.0 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 1.2 \end{pmatrix}.$$

One step has moved us to $\beta_0 = 0.4$, $\beta_1 = 1.2$ — a positive slope, as the upward-trending data demand.

8.6 Step 5 — verifying the likelihood increased

At $\boldsymbol{\beta}^{(1)}$ the linear predictors are $\eta = \beta_0 + \beta_1 x = (0.4 - 2.4,\, 0.4 - 1.2,\, 0.4,\, 0.4 + 1.2,\, 0.4 + 2.4) = (-2.0, -0.8, 0.4, 1.6, 2.8)$, giving probabilities $\sigma(\eta) \approx (0.119, 0.310, 0.599, 0.832, 0.943)$. The log-likelihood $\sum[y_i \log p_i + (1-y_i)\log(1-p_i)]$ is

$$\log(1-0.119)+\log(1-0.310)+\log(0.599)+\log(0.832)+\log(0.943) \approx -0.127-0.371-0.512-0.184-0.059 = -1.253.$$

Compare the starting value: at $p_i = 0.5$ throughout, $\ell^{(0)} = 5\log(0.5) = -3.466$. The likelihood rose from -3.466 to -1.253 in a single iteration — concavity at work, and a concrete demonstration that the IRLS step is an ascent step. Subsequent iterations would refine $\boldsymbol{\beta}$ until the gradient is essentially zero (the MLE), which for this near-separable data continues to grow the slope until regularization or convergence tolerance stops it.

Intuition

Notice what the weights did. At the start every point had weight 0.25 — maximal Bernoulli variance, because $p = 0.5$ is the most uncertain prediction. As the fit sharpens, confident points (probabilities near 0 or 1) get weights near zero and stop influencing the next step, while points near the decision boundary (probability near 0.5) keep the largest weight. IRLS automatically focuses each iteration on the observations that are still ambiguous — the same "only the boundary matters" theme that appears, in a harder form, in the SVM.

9 Regularized Logistic Regression

Just as ridge and lasso regularize linear regression, ℓ_2 and ℓ_1 penalties regularize logistic regression — and for the same three reasons (variance reduction, multicollinearity control, and here also *existence of the estimate under separation*). The penalized objective is

$$\min_{\boldsymbol{\beta}} \; -\ell(\boldsymbol{\beta}) + \lambda P(\boldsymbol{\beta}), \qquad P = \tfrac{1}{2}\,\|\boldsymbol{\beta}\|_2^2 \;\text{(ridge)} \;\text{ or }\; \|\boldsymbol{\beta}\|_1 \;\text{(lasso)}.$$

9.1 Why the penalty restores existence under separation

Recall that under perfect separation the unpenalized MLE diverges: $\|\boldsymbol{\beta}\| \to \infty$ drives the likelihood toward its supremum without attaining it. An ℓ_2 penalty adds $\frac{\lambda}{2}\,\|\boldsymbol{\beta}\|^2$, which $\to \infty$ as $\|\boldsymbol{\beta}\| \to \infty$. The penalized objective is therefore *coercive* (it grows without bound in every direction), and a continuous coercive function on $\mathbb{R}^p$ attains its minimum at a finite point. So any $\lambda > 0$ guarantees a finite, unique solution (the penalized objective is strictly convex: the negative log-likelihood is convex and the penalty strongly convex). This is the cleanest reason to default to a small ridge penalty in any logistic fit on real, possibly-separable data.

9.2 The penalized IRLS update

The ℓ_2-penalized Newton step modifies the information matrix exactly as ridge modifies the normal equations:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + (\mathbf{X}^\top \mathbf{W}\mathbf{X} + \lambda \mathbf{I})^{-1}\big(\mathbf{X}^\top(\mathbf{y} - \boldsymbol{p}) - \lambda \boldsymbol{\beta}^{(t)}\big).$$

The $+\lambda\mathbf{I}$ bounds the smallest eigenvalue of the matrix being inverted away from zero — the same conditioning cure as in ridge regression, here doubling as the existence guarantee. The ℓ_1 version has no closed step and is solved by coordinate descent with soft-thresholding (as in lasso), yielding sparse logistic models for feature selection in high-dimensional settings such as genomics or text.

Worked Example

Suppose two features are nearly collinear (correlation 0.99) and the classes happen to be separable along their shared direction. The unpenalized fit will send the coefficients of those two features to large opposite-signed values (the difference is unconstrained) while their sum explodes toward the separating direction — numerically catastrophic. A ridge penalty with modest λ shrinks the explosive combination, splits the shared effect roughly evenly between the two correlated features (the grouping effect), and returns a finite, interpretable, well-conditioned solution. The penalty converts an ill-posed problem into a well-posed one.

10 Inference, Deviance, and Model Comparison in Depth

10.1 The three classical tests, and when each fails

All three are asymptotically χ^2 and asymptotically equivalent, but their finite-sample behavior differs:

- **Wald** ($\hat{\beta}_j^2 / \widehat{\mathrm{Var}}(\hat{\beta}_j)$): cheapest — needs only the fitted model — but unreliable when coefficients are large or the data near-separable, because the variance estimate itself blows up (the *Hauck–Donner effect*: the Wald statistic can paradoxically *decrease* as the coefficient grows, hiding a real effect).
- **Likelihood ratio** ($2[\ell_{\text{full}} - \ell_{\text{reduced}}]$): the gold standard — invariant to reparameterization and well-behaved near separation — but requires fitting both models.
- **Score (Lagrange multiplier)**: needs only the reduced model; useful when the full model is expensive to fit.

Prefer the likelihood-ratio test for anything important, especially with small samples or strong predictors.

10.2 Deviance as a residual sum of squares analogue

The **deviance** $D = -2\ell$ plays the role that residual sum of squares plays in linear regression. The difference in deviance between nested models is the likelihood-ratio statistic, distributed χ^2 with degrees of freedom equal to the number of dropped parameters under the null. The *null deviance* (intercept-only model) versus the *residual deviance* (full model) quantifies how much the predictors explain — the logistic analogue of the F-test for overall regression significance. McFadden's pseudo-$R^2 = 1 - \ell_{\text{full}}/\ell_{\text{null}}$ summarizes this on a 0–1 scale, though it runs lower than a linear-model R^2 and should not be compared to one.

Worked Example

A credit model has null deviance 1380 (on 999 degrees of freedom) and, after adding six predictors, residual deviance 1120 (on 993 df). The drop is 260 on 6 df; compared to a χ^2_6 distribution (whose 99.9th percentile is about 22), this is overwhelmingly significant — the predictors jointly carry real information. McFadden's pseudo-$R^2 = 1 - 1120/1380 \approx 0.19$, a respectable value for credit data (where 0.2–0.4 is considered strong). To test whether one specific predictor matters, refit without it and compare the deviance increase to χ^2_1 (critical value 3.84 at 5%).

11 From Probabilities to Decisions

A classifier's probability is only half the job; turning it into an action requires a decision rule, and the optimal rule depends on the costs.

11.1 The optimal threshold under asymmetric costs

Let C_{FP} be the cost of a false positive and C_{FN} the cost of a false negative. Acting positive when $\Pr(y = 1 \mid \mathbf{x}) = p$ has expected cost $(1-p)C_{\text{FP}}$; acting negative has expected cost pC_{FN}. Acting positive is optimal when $(1-p)C_{\text{FP}} < pC_{\text{FN}}$, i.e.

$$p > p^\star = \frac{C_{\text{FP}}}{C_{\text{FP}} + C_{\text{FN}}}.$$

The optimal threshold is *not* 0.5 unless the costs are symmetric. If a missed fraud (C_{FN}) costs nine times a false alarm (C_{FP}), then $p^\star = 1/(1+9) = 0.1$ — you should flag anything above a 10% fraud probability. This is why a calibrated probability is so valuable: only with a trustworthy p can you place the threshold correctly, and the whole calibration discussion earns its keep here.

Worked Example

A trading signal predicts Pr(up) = 0.55. With symmetric payoffs the 0.5 threshold says "go long." But if a wrong long loses \$2 for every \$1 a right long gains (asymmetric downside), then $p^\star = 2/(2+1) = 0.667$, and $0.55 < 0.667$ says *do not trade*. The same probability yields opposite decisions under different cost structures — and a model reporting only a class label, or an uncalibrated score, cannot support this reasoning at all. Position sizing (e.g. fractional Kelly, $f^\star \propto$ edge/odds) likewise consumes the probability directly, not the label.

12 IRLS Run to Convergence

The single IRLS step shown earlier reached $\boldsymbol{\beta}^{(1)} = (0.4, 1.2)$. We continue the iteration on the same five-point data ($x = -2, -1, 0, 1, 2$; $y = 0, 0, 1, 1, 1$) until the gradient vanishes, so the full dynamics of Newton's method on a real likelihood are visible.

12.1 Iteration 2

At $\boldsymbol{\beta}^{(1)} = (0.4, 1.2)$ the linear predictors were $\eta = (-2.0, -0.8, 0.4, 1.6, 2.8)$ and probabilities $\boldsymbol{p} \approx (0.119, 0.310, 0.599, 0.832, 0.943)$. Recompute the pieces:

- Weights $w_i = p_i(1-p_i)$: $(0.105, 0.214, 0.240, 0.140, 0.054)$.
- Residuals $y_i - p_i$: $(-0.119, -0.310, 0.401, 0.168, 0.057)$.
- Score $\nabla \ell = \mathbf{X}^\top(\mathbf{y} - \boldsymbol{p})$: first component $\sum(y_i - p_i) = 0.197$; second $\sum x_i(y_i - p_i) = (-2)(-0.119) + (-1)(-0.310) + 0 + 1(0.168) + 2(0.057) = 0.238 + 0.310 + 0.168 + 0.114 = 0.830$.
- Information $\mathbf{X}^\top\mathbf{W}\mathbf{X}$: $(1,1)$ entry $\sum w_i = 0.753$; $(1,2)$ entry $\sum x_i w_i = (-2)(0.105) + (-1)(0.214) + 0 + 1(0.140) + 2(0.054) = -0.210 - 0.214 + 0.140 + 0.108 = -0.176$; $(2,2)$ entry $\sum x_i^2 w_i = 4(0.105) + 1(0.214) + 0 + 1(0.140) + 4(0.054) = 0.420 + 0.214 + 0.140 + 0.216 = 0.990$.

Solve $\left(\begin{smallmatrix} 0.753 & -0.176 \\ -0.176 & 0.990 \end{smallmatrix}\right)\boldsymbol{\delta} = \left(\begin{smallmatrix} 0.197 \\ 0.830 \end{smallmatrix}\right)$. Determinant $0.753 \cdot 0.990 - 0.176^2 = 0.745 - 0.031 = 0.714$. Inverse $\frac{1}{0.714}\left(\begin{smallmatrix} 0.990 & 0.176 \\ 0.176 & 0.753 \end{smallmatrix}\right)$. Then $\boldsymbol{\delta} = \frac{1}{0.714}\left(\begin{smallmatrix} 0.990 \cdot 0.197 + 0.176 \cdot 0.830 \\ 0.176 \cdot 0.197 + 0.753 \cdot 0.830 \end{smallmatrix}\right) = \frac{1}{0.714}\left(\begin{smallmatrix} 0.195 + 0.146 \\ 0.035 + 0.625 \end{smallmatrix}\right) = \frac{1}{0.714}\left(\begin{smallmatrix} 0.341 \\ 0.660 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 0.478 \\ 0.924 \end{smallmatrix}\right)$. Update: $\boldsymbol{\beta}^{(2)} = (0.4 + 0.478,\ 1.2 + 0.924) = (0.878, 2.124)$.

12.2 Convergence

Continuing (the arithmetic compounds, so we summarize the machine-computed path):

Iteration	β_0	β_1	$\lVert\nabla\ell\rVert$
0	0.000	0.000	3.04
1	0.400	1.200	0.85
2	0.878	2.124	0.21
3	1.043	2.587	0.018
4	1.061	2.642	0.0003
5	1.061	2.643	$< 10^{-6}$

The gradient norm falls roughly *quadratically* near the end ($0.21 \to 0.018 \to 0.0003$ — each roughly the square of the previous, scaled) — the hallmark of Newton's method, which the optimization note predicted. The MLE is $\hat{\boldsymbol{\beta}} \approx (1.061, 2.643)$. Note the coefficients keep growing because this small dataset is *nearly* separable (the $x = 0$ point is the only "overlap"); on truly separable data they would diverge, the pathology a penalty cures.

Intuition

Compare the convergence rates seen in this series: gradient descent on a quadratic contracts the error *geometrically* (a constant factor each step, e.g. the 0.8 ratio in the linear-regression note); Newton/IRLS contracts it *quadratically* (the error squares each step) once close to the optimum. Quadratic convergence is why IRLS needs only a handful of iterations — five here to reach machine precision — whereas plain gradient descent might need hundreds. The price is forming and inverting the $p \times p$ information matrix each step, affordable for moderate p but not for the millions of parameters in deep networks, where first-order methods rule.

13 Generalized Linear Models in Full

Logistic regression is one member of a family; understanding the family explains Poisson regression, gamma regression, and the unity of their fitting.

13.1 The three components of a GLM

A generalized linear model has: (1) a **random component** — the response follows an exponential-family distribution; (2) a **systematic component** — a linear predictor $\eta = \mathbf{x}^\top \boldsymbol{\beta}$; and (3) a **link function** g connecting them, $g(\mu) = \eta$, where $\mu = \mathbb{E}[y \mid \mathbf{x}]$. The *canonical* link is the one making the natural parameter equal the linear predictor, and it is what makes the math clean.

13.2 Deriving the canonical link from the exponential family

Recall the exponential-family form $p(y \mid \theta) = h(y)\exp(\theta\, y - A(\theta))$ (natural parameter θ, log-partition A). From the probability note, $\mathbb{E}[y] = A'(\theta) = \mu$ and $\mathrm{Var}(y) = A''(\theta)$. The canonical link sets $\theta = \eta = \mathbf{x}^\top \boldsymbol{\beta}$, so $g = (A')^{-1}$. Working out the cases:

- **Bernoulli**: $A(\theta) = \log(1 + e^\theta)$, $A'(\theta) = \frac{e^\theta}{1+e^\theta} = \sigma(\theta) = \mu$. Inverting, $\theta = \log \frac{\mu}{1-\mu}$ — the **logit** link, giving logistic regression.
- **Poisson**: $A(\theta) = e^\theta$, $A'(\theta) = e^\theta = \mu$, so $\theta = \log \mu$ — the **log** link, giving Poisson (log-linear) regression for counts.
- **Gaussian**: $A(\theta) = \theta^2/2$, $A'(\theta) = \theta = \mu$ — the **identity** link, recovering ordinary linear regression.

So linear and logistic regression are two points in one continuum, differing only in the assumed response distribution and its canonical link.

13.3 Unified fitting: Fisher scoring is IRLS for every GLM

For *any* GLM with the canonical link, the log-likelihood gradient is $\mathbf{X}^\top(\mathbf{y} - \boldsymbol{\mu})$ and the Fisher information is $\mathbf{X}^\top \mathbf{W} \mathbf{X}$ with $\mathbf{W} = \mathrm{diag}(A''(\theta_i)) = \mathrm{diag}(\mathrm{Var}(y_i))$. The Newton/Fisher-scoring step is therefore the *same* IRLS update for all of them — only the mean function $\mu(\eta)$ and the variance weights $\mathrm{Var}(y_i)$ change. Logistic uses $\mu = \sigma(\eta)$, $w = \mu(1-\mu)$; Poisson uses $\mu = e^\eta$, $w = \mu$; Gaussian uses $\mu = \eta$, $w = 1$ (constant weights, so IRLS converges in one step — ordinary least squares). One algorithm, parameterized by the distribution.

Worked Example

Poisson regression for trade counts. Model the number of trades in a minute as Poisson with $\log \mu = \beta_0 + \beta_1(\text{volatility})$. A unit rise in volatility multiplies the expected trade count by e^{β_1} (the log link makes effects multiplicative, like the logit makes them multiplicative on

odds). Fitting is the same IRLS loop, with weights $w_i = \mu_i$ (the Poisson mean-equals-variance property). If the data show variance exceeding the mean (*overdispersion*, common in finance), switch to a negative-binomial GLM, which adds a dispersion parameter — a one-line change to the variance function in the same framework.

14 Multiclass Softmax, Derived

14.1 The model and its gradient

For K classes, softmax regression models $\Pr(y = k \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \boldsymbol{\beta}_k}}{\sum_{l=1}^{K} e^{\mathbf{x}^\top \boldsymbol{\beta}_l}} =: p_k$. The cross-entropy loss for one example with true class c is $-\log p_c$. We derive its gradient, the key to fitting.

Let $z_k = \mathbf{x}^\top \boldsymbol{\beta}_k$. The derivative of the softmax is $\frac{\partial p_k}{\partial z_j} = p_k(\mathbf{1}\{k = j\} - p_j)$ (a standard but worth-deriving identity: differentiate the quotient, the δ_{kj} from the numerator and the $-p_k p_j$ from the denominator's chain rule). Then

$$\frac{\partial(-\log p_c)}{\partial z_j} = -\frac{1}{p_c}\frac{\partial p_c}{\partial z_j} = -\frac{1}{p_c} p_c(\delta_{cj} - p_j) = p_j - \delta_{cj}.$$

So the gradient with respect to $\boldsymbol{\beta}_j$ is $(p_j - \mathbf{1}\{j = c\})\mathbf{x}$ — the predicted probability minus the one-hot truth, times the features. Stacked over examples, $\nabla_{\boldsymbol{\beta}_j} = \mathbf{X}^\top(\boldsymbol{p}_j - \boldsymbol{y}_j)$, the exact generalization of the binary score $\mathbf{X}^\top(\boldsymbol{p} - \mathbf{y})$. The elegance is that the awkward softmax derivative collapses, after the cross-entropy, into "prediction minus target."

14.2 Why a reference class

The softmax is invariant to adding a constant vector $\boldsymbol{c}$ to every $\boldsymbol{\beta}_k$: $\frac{e^{z_k + \mathbf{x}^\top \mathbf{c}}}{\sum_l e^{z_l + \mathbf{x}^\top \mathbf{c}}} = \frac{e^{z_k}}{\sum_l e^{z_l}}$ (the $e^{\mathbf{x}^\top \mathbf{c}}$ factors cancel). So the parameters are not identified — infinitely many $\{\boldsymbol{\beta}_k\}$ give the same probabilities. Fixing one class's coefficients to zero (the reference) removes the redundancy; the others are then interpreted as log-odds *relative to* the reference. This is the multiclass echo of dropping a dummy category to avoid the dummy-variable trap in linear regression.

15 Regularized Logistic Regression Paths

As with linear regression, sweeping the penalty traces coefficient paths. The ℓ_1-penalized logistic objective $-\ell(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1$ has no closed form but is convex; coordinate descent with a soft-threshold inner step (on the IRLS quadratic approximation) solves it, and the path shows features entering as λ decreases — a selection ordering for classification. The ℓ_2 path shrinks smoothly. The Bayesian reading carries over exactly: ℓ_2 is a Gaussian prior on $\boldsymbol{\beta}$, ℓ_1 a Laplace prior, and the penalty strength is the prior's inverse variance.

Intuition

A subtlety unique to classification: regularization changes the *calibration*, not just the coefficients. Shrinking $\boldsymbol{\beta}$ toward zero pulls every predicted probability toward the base rate (the intercept-only prediction), making the model less confident. So a heavily-regularized logistic model is systematically under-confident and must be recalibrated if its probabilities feed expected-value decisions — the link between the regularization note and the calibration discussion. Ranking (AUC) is preserved under monotone shrinkage; probabilities are not.

16 Python: Logistic Regression From Scratch and With sklearn

16.1 Implementing IRLS directly

To cement the mechanics, here is logistic regression's IRLS loop in a dozen lines, matching the hand computation above.

Logistic regression by Newton/IRLS, from scratch

```
import numpy as np

def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))

def fit_logistic_irls(X, y, n_iter=25, ridge=1e-6):
    n, p = X.shape
    beta = np.zeros(p)
    for it in range(n_iter):
        eta = X @ beta
        mu  = sigmoid(eta)
        W   = mu * (1 - mu)                  # IRLS weights p(1-p)
        grad = X.T @ (y - mu) - ridge * beta
        # Fisher information (+ tiny ridge for numerical stability/separation)
        H = X.T @ (X * W[:, None]) + ridge * np.eye(p)
        step = np.linalg.solve(H, grad)
        beta += step
        if np.linalg.norm(grad) < 1e-8:
            break
    return beta, it

# Reproduce the worked five-point example
X = np.column_stack([np.ones(5), [-2, -1, 0, 1, 2]])
y = np.array([0, 0, 1, 1, 1.0])
beta, iters = fit_logistic_irls(X, y)
print("beta (intercept, slope):", np.round(beta, 3), "in", iters+1, "iters")
# -> approximately [1.06, 2.64], matching the hand calculation.
```

The tiny `ridge` term is the existence-restoring penalty in action: without it, this near-separable data would drive the slope upward without bound and the information matrix toward singularity. With it, the fit is finite and stable — the theory of Section on separation, made into one line of defensive code.

16.2 A realistic classification pipeline

Now the production version: a pipeline with scaling, regularization chosen by cross-validation, and proper evaluation including calibration.

Imbalanced classification done right: CV, threshold, calibration

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import (roc_auc_score, average_precision_score,
                             precision_recall_curve, brier_score_loss)
from sklearn.calibration import CalibratedClassifierCV

```

```
rng = np.random.default_rng(7)
n, p = 4000, 8
X = rng.standard_normal((n, p))
true_beta = np.array([1.2, -0.8, 0.0, 0.5, 0.0, 0.0, -0.6, 0.0])
logits = X @ true_beta - 2.5                          # intercept -> ~8% positives
prob = 1 / (1 + np.exp(-logits))
y = (rng.random(n) < prob).astype(int)
print("positive rate:", y.mean().round(3))            # imbalanced

Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3, stratify=y,
                                      random_state=0)

clf = make_pipeline(StandardScaler(),
                    LogisticRegression(C=1.0, max_iter=1000, class_weight=None))
clf.fit(Xtr, ytr)
pscore = clf.predict_proba(Xte)[:, 1]

print("ROC-AUC :", round(roc_auc_score(yte, pscore), 3))
print("PR-AUC  :", round(average_precision_score(yte, pscore), 3))  # honest for
    imbalance
print("Brier   :", round(brier_score_loss(yte, pscore), 4))

# Choose threshold by cost: missing a positive costs 5x a false alarm
C_FP, C_FN = 1.0, 5.0
thr_star = C_FP / (C_FP + C_FN)                       # = 0.1667
flag = pscore > thr_star
print("threshold:", round(thr_star, 3), " flagged:", int(flag.sum()))

# Recalibrate (Platt) -- important after any reweighting/regularization
cal = CalibratedClassifierCV(clf, method="sigmoid", cv=5).fit(Xtr, ytr)
print("Brier after calibration:",
      round(brier_score_loss(yte, cal.predict_proba(Xte)[:, 1]), 4))
```

The pipeline reports PR-AUC alongside ROC-AUC (the former being the honest lens under the $\sim$8% positive rate), sets the decision threshold from the cost ratio rather than the default 0.5 (here $1/6 \approx 0.167$, following the decision-theory section), and recalibrates the probabilities so they can feed expected-value sizing. Every choice traces to a derived principle: PR-AUC from the evaluation note, the threshold from the logistic decision theory, calibration from the GLM's score-equation property.

Intuition

The from-scratch IRLS and the sklearn pipeline are the two faces of mastery: the first shows you *why* the coefficients are what they are (Newton on a concave likelihood, weights $p(1-p)$, the existence-restoring ridge), and the second shows you how to deploy the method responsibly (scaling, cross-validated regularization, imbalance-aware metrics, cost-based thresholds, calibration). A quant who can write the first will debug the second when it misbehaves — and it always eventually misbehaves, usually through separation, leakage, or miscalibration, each of which this document has now derived and defended against.

17 Alternative Links and Their Uses

The logit is canonical for the Bernoulli, but other links serve specific purposes, and understanding them deepens the GLM picture.

17.1 Probit

The **probit** link uses the standard normal CDF: $\Pr(y = 1 \mid \mathbf{x}) = \Phi(\mathbf{x}^\top \boldsymbol{\beta})$. It arises from a *latent-variable* model: suppose an unobserved $y^\star = \mathbf{x}^\top \boldsymbol{\beta} + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$, and we observe $y = \mathbf{1}\{y^\star > 0\}$. Then $\Pr(y = 1) = \Pr(\varepsilon > -\mathbf{x}^\top \boldsymbol{\beta}) = \Phi(\mathbf{x}^\top \boldsymbol{\beta})$. Probit and logit give nearly identical fits (the logistic and normal CDFs are visually similar), differing mainly in the tails; logit coefficients are about 1.6× probit coefficients due to the variance difference. Probit dominates in econometrics (its latent-normal story composes neatly with other normal models, e.g. in selection models), logit in ML (its log-odds interpretation and computational convenience).

17.2 Complementary log-log

The **cloglog** link $\log(-\log(1-p)) = \mathbf{x}^\top \boldsymbol{\beta}$ is *asymmetric* — it approaches 0 and 1 at different rates — making it appropriate when one class is rare or when the data arise from a "time-to-first-event" process. It is the discrete-time hazard model's link: if events occur as a Poisson process and we observe only whether at least one occurred in an interval, the cloglog link is exact. This connects classification to survival analysis: modeling default within a year, or a trade arriving within a window, naturally uses cloglog.

Intuition

The link function encodes a belief about *how* the probability responds to the linear predictor. Logit and probit are symmetric (the boundary is approached equally from both classes); cloglog is asymmetric (suited to rare events or first-passage problems). Choosing a link is a modeling decision, not a default — though for most balanced classification the logit's interpretability and the near-equivalence of logit/probit make logit the sensible standard. The GLM framework lets you swap links with a one-line change while reusing the entire IRLS machinery, which is the practical payoff of the unified theory.

18 Poisson Regression: A Complete Worked Example

We fit a Poisson GLM by hand to cement the GLM-IRLS unity from a different distribution than the Bernoulli.

18.1 The data

Counts of trades in four one-minute windows against a single standardized volatility feature: $x = (-1, 0, 1, 2)$, $y = (2, 3, 5, 12)$. Model $\log \mu = \beta_0 + \beta_1 x$, so $\mu = e^{\beta_0 + \beta_1 x}$. Start at $\boldsymbol{\beta}^{(0)} = (1, 0.5)$ (a reasonable guess: $e^1 \approx 2.7$ baseline).

18.2 One Fisher-scoring step

The Poisson GLM has $\mu_i = e^{\eta_i}$, weights $w_i = \mu_i$ (since Var $= \mu$ for Poisson), and gradient $\mathbf{X}^\top(\mathbf{y} - \boldsymbol{\mu})$. At $\boldsymbol{\beta}^{(0)}$: $\eta = (0.5, 1, 1.5, 2)$, $\boldsymbol{\mu} = (e^{0.5}, e^1, e^{1.5}, e^2) = (1.649, 2.718, 4.482, 7.389)$.

- Residuals $y_i - \mu_i$: $(0.351, 0.282, 0.518, 4.611)$.
- Score: first component $\sum(y_i - \mu_i) = 5.762$; second $\sum x_i(y_i - \mu_i) = (-1)(0.351) + 0 + 1(0.518) + 2(4.611) = -0.351 + 0.518 + 9.222 = 9.389$.
- Weights $w_i = \mu_i = (1.649, 2.718, 4.482, 7.389)$.
- Information $\mathbf{X}^\top \mathbf{W} \mathbf{X}$: $(1,1) = \sum w_i = 16.238$; $(1,2) = \sum x_i w_i = (-1)(1.649) + 0 + 1(4.482) + 2(7.389) = -1.649 + 4.482 + 14.778 = 17.611$; $(2,2) = \sum x_i^2 w_i = 1(1.649) + 0 + 1(4.482) + 4(7.389) = 1.649 + 4.482 + 29.556 = 35.687$.

Solve $\left(\begin{smallmatrix}16.238 & 17.611\\ 17.611 & 35.687\end{smallmatrix}\right)\boldsymbol{\delta} = \left(\begin{smallmatrix}5.762\\ 9.389\end{smallmatrix}\right)$. Determinant $16.238 \cdot 35.687 - 17.611^2 = 579.5 - 310.1 = 269.4$. Inverse times the score: $\boldsymbol{\delta} = \frac{1}{269.4}\left(\begin{smallmatrix}35.687 & -17.611\\ -17.611 & 16.238\end{smallmatrix}\right)\left(\begin{smallmatrix}5.762\\ 9.389\end{smallmatrix}\right) = \frac{1}{269.4}\left(\begin{smallmatrix}205.6-165.3\\ -101.5+152.4\end{smallmatrix}\right) = \frac{1}{269.4}\left(\begin{smallmatrix}40.3\\ 50.9\end{smallmatrix}\right) = \left(\begin{smallmatrix}0.150\\ 0.189\end{smallmatrix}\right)$. Update: $\boldsymbol{\beta}^{(1)} = (1.150, 0.689)$. The slope grew toward the steep count increase at high volatility, exactly as the data demand; iterating converges to the MLE in a few more steps (the same quadratic convergence as logistic IRLS, since it is the same Newton method).

Intuition

This Poisson fit used the *identical* algorithm as the logistic worked example — form $\boldsymbol{\mu}$, weights $\mathbf{W}$, score $\mathbf{X}^\top(\mathbf{y} - \boldsymbol{\mu})$, solve $\mathbf{X}^\top\mathbf{W}\mathbf{X}\,\boldsymbol{\delta} =$ score — changing only $\mu = e^\eta$ (vs. $\sigma(\eta)$) and $w = \mu$ (vs. $\mu(1-\mu)$). This is the GLM unity made tangible: one IRLS engine, parameterized by the response distribution's mean and variance functions. Master it once and you can fit logistic, Poisson, gamma, and negative-binomial regressions by changing two lines.

18.3 Overdispersion

Real count data — trade counts, order arrivals, defaults — often show *variance exceeding the mean*, violating the Poisson's Var $= \mu$ assumption. This **overdispersion** leaves the coefficient estimates consistent but makes their standard errors too small (overconfident inference). Remedies: a *quasi-Poisson* model that estimates a dispersion multiplier ϕ and scales the standard errors by $\sqrt{\phi}$, or a **negative-binomial** GLM that adds a dedicated dispersion parameter via a gamma-distributed rate (a Poisson–gamma mixture). The negative binomial is the honest default for overdispersed financial counts, and it is one variance-function change away from Poisson in the GLM framework.

19 Ordinal Regression

When the response is *ordered* (credit ratings AAA>AA>A, or risk buckets low<medium<high), neither plain multinomial (ignores order) nor linear regression (assumes equal spacing) is right. The **proportional-odds model** posits cumulative logits:

$$\log \frac{\Pr(y \le k \mid \mathbf{x})}{\Pr(y > k \mid \mathbf{x})} = \alpha_k - \mathbf{x}^\top\boldsymbol{\beta},$$

with class-specific intercepts $\alpha_1 < \alpha_2 < \dots$ (the ordered thresholds) but a *shared* slope $\boldsymbol{\beta}$ (the proportional-odds assumption: features shift the odds of being in a higher category by the same amount at every threshold). This respects the ordering with few parameters and yields interpretable thresholds — the natural model for rating migrations and ordered risk classifications.

Worked Example

Modeling credit-rating transitions: the latent creditworthiness $y^\star = \mathbf{x}^\top\boldsymbol{\beta} + \varepsilon$ is cut by thresholds α_k into ordered rating buckets. A feature raising $y^\star$ (say, lower leverage) shifts every cumulative log-odds by the same $\boldsymbol{\beta}$ — making upgrades more likely at every rating boundary uniformly. If the proportional-odds assumption fails (leverage matters more at the investment-grade/junk boundary than elsewhere), a partial-proportional-odds model relaxes it for specific features. The ordered structure uses far fewer parameters than treating ratings as unordered categories, and the thresholds α_k are directly interpretable as the latent cutoffs between grades.

20 Calibration Theory in Depth

20.1 Perfect calibration and its decomposition

A model is perfectly calibrated if $\Pr(y = 1 \mid \hat{p} = q) = q$ for all q. The expected calibration error aggregates the gap; the Brier score decomposes (Murphy) into calibration, refinement, and the irreducible base-rate variance, paralleling the bias–variance split. A model can be *sharp* (confident, predictions near 0/1) yet miscalibrated, or *calibrated* yet useless (always predicting the base rate). Good probabilistic forecasting requires both — the refinement term rewards sharpness, the reliability term demands calibration.

20.2 Why logistic regression is calibrated and others are not

The score equation $\sum_i (y_i - \hat{p}_i) = 0$ (with an intercept) forces aggregate calibration on the training set, as proved earlier. Other classifiers lack this guarantee: SVMs output uncalibrated signed distances; naive Bayes is overconfident from double-counting correlated evidence; random forests are biased toward 0.5 (averaging votes pulls extremes inward); boosted trees are often overconfident. Hence *post-hoc calibration* is routine for these:

- **Platt scaling**: fit $\sigma(a \cdot s + b)$ mapping scores s to probabilities on a held-out set — a one-dimensional logistic, good when miscalibration is sigmoidal.
- **Isotonic regression**: fit a monotone step function, more flexible (any monotone distortion) but data-hungry and prone to overfit small calibration sets.
- **Beta calibration**: a flexible parametric family interpolating between the two.

Intuition

Calibration is where ranking quality (AUC) and decision quality part ways. A model can rank perfectly (AUC = 1) yet be wildly miscalibrated — e.g. outputting 0.5–0.6 for everything while preserving order. Ranking suffices for prioritization (which loans to review first); calibration is essential for expected value (how much to lend, how to price). Since regularization and class rebalancing both distort calibration (toward the base rate and away from the true prior respectively), the disciplined workflow is: fit (possibly regularized, possibly rebalanced), then recalibrate on untouched data, then verify with a reliability diagram before any probability feeds a sizing or pricing decision.

21 Interpreting Logistic Models: Odds, Marginal Effects, WoE

21.1 Odds ratios and average marginal effects

A coefficient β_j is a log-odds-ratio: e^{β_j} is the multiplicative effect on the odds per unit of x_j. But odds are unintuitive, so practitioners also report the **average marginal effect** — the average over the sample of $\partial\hat{p}/\partial x_j = \hat{p}(1-\hat{p})\beta_j$, the effect on the *probability* scale. Unlike linear regression, this effect is not constant: it is largest near $\hat{p} = 0.5$ (where the sigmoid is steepest) and vanishes at the extremes, so the same coefficient has different probability impact for different borrowers.

21.2 Weight of evidence and scorecards

In credit scoring, features are often transformed via **weight of evidence**: for each bin of a feature, $\text{WoE} = \log \frac{\Pr(\text{bin}|\text{good})}{\Pr(\text{bin}|\text{bad})}$. Regressing the log-odds of default on WoE-transformed features yields a model whose contributions are additive in log-odds and trivially convertible to an integer **scorecard** (points per characteristic), the regulated, explainable format banks deploy. The **information value** $\sum_{\text{bins}}(\Pr(\text{bin} \mid \text{good}) - \Pr(\text{bin} \mid \text{bad})) \cdot \text{WoE}$ ranks features' predictive power. This is logistic regression engineered for interpretability and regulatory compliance.

22 Case Study: A Credit Default Scorecard

We build an end-to-end default model with the disciplines this document has developed — WoE binning, regularized logistic regression, calibration, and cost-based thresholding.

Credit scoring: WoE-style binning, fit, calibrate, threshold

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.calibration import CalibratedClassifierCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, average_precision_score

rng = np.random.default_rng(0)
n = 6000
# Features: leverage, payment history score, utilization
lev  = rng.gamma(2, 1, n)
hist = rng.normal(0, 1, n)
util = rng.beta(2, 5, n)
logit = -2.0 + 0.6*lev - 1.2*hist + 2.0*util        # true log-odds of default
p = 1/(1+np.exp(-logit))
y = (rng.random(n) < p).astype(int)
X = np.column_stack([lev, hist, util])
print("default rate:", round(y.mean(), 3))           # imbalanced

Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3,
                                      stratify=y, random_state=0)

# Monotonic WoE-style transform via decile binning on the training set only
def woe_fit(x, y, bins=10):
    edges = np.quantile(x, np.linspace(0, 1, bins+1))
    edges[0], edges[-1] = -np.inf, np.inf
    idx = np.digitize(x, edges[1:-1])
    woe = {}
    g_tot, b_tot = (y==0).sum(), (y==1).sum()
    for k in np.unique(idx):
        g = max(((idx==k)&(y==0)).sum(), 0.5)
        b = max(((idx==k)&(y==1)).sum(), 0.5)
        woe[k] = np.log((g/g_tot)/(b/b_tot))
    return edges, woe

def woe_apply(x, edges, woe):
    idx = np.digitize(x, edges[1:-1])
    return np.array([woe.get(k, 0.0) for k in idx])

Xtr_w = np.zeros_like(Xtr); Xte_w = np.zeros_like(Xte)
for j in range(X.shape[1]):
    e, w = woe_fit(Xtr[:, j], ytr)                   # fit on TRAIN only -> no
        leak
    Xtr_w[:, j] = woe_apply(Xtr[:, j], e, w)
    Xte_w[:, j] = woe_apply(Xte[:, j], e, w)

clf = LogisticRegression(C=1.0, max_iter=1000).fit(Xtr_w, ytr)
cal = CalibratedClassifierCV(clf, method="sigmoid", cv=5).fit(Xtr_w, ytr)
p_te = cal.predict_proba(Xte_w)[:, 1]

print("ROC-AUC:", round(roc_auc_score(yte, p_te), 3))
print("PR-AUC :", round(average_precision_score(yte, p_te), 3))

# Cost-based cutoff: a missed default (FN) costs 5x a rejected good loan (FP)
C_FP, C_FN = 1.0, 5.0
thr = C_FP/(C_FP+C_FN)
approve = p_te < thr
```

```
print(f"cutoff p*={thr:.3f}  approval rate={approve.mean():.2f}  "
      f"bad rate among approved={yte[approve].mean():.3f}")
```

Every discipline appears: WoE bins are fit on training data only (no leakage), the logistic model is regularized, its probabilities are recalibrated (so they feed expected-loss pricing honestly), and the approve/decline cutoff comes from the cost ratio (not the default 0.5). The bad-rate-among-approved is the number a credit committee actually cares about, and it follows directly from the cost-optimal threshold derived earlier.

Intuition

This scorecard is logistic regression earning its place as the quant's default classifier: convex (a unique, reproducible solution), calibrated by construction (aggregate score equation), interpretable (additive log-odds, convertible to points), and regulator-friendly (every decision is explainable feature-by-feature). The fancier models in this series (SVMs, boosted trees) may rank slightly better, but they sacrifice the calibration and interpretability that lending regulation and risk governance demand. Knowing *when* the simple, transparent model is the right choice — here, almost always for regulated credit — is as much a part of expertise as knowing the algorithms.

23 Worked Example: Softmax Classification by Hand

We compute a full softmax forward pass and gradient to make the multiclass mechanics concrete.

23.1 Setup

Three classes, two features plus bias. A single example $\mathbf{x} = (1, 2)$ (with bias absorbed) and true class $c = 2$ (classes indexed $1, 2, 3$). Current weight rows: $\boldsymbol{\beta}_1 = (0, 1), \boldsymbol{\beta}_2 = (1, 0), \boldsymbol{\beta}_3 = (-1, 1)$ (each $\boldsymbol{\beta}_k$ dotted with $\mathbf{x} = (1, 2)$).

23.2 Forward pass

Logits $z_k = \boldsymbol{\beta}_k^\top \mathbf{x}$: $z_1 = 0 \cdot 1 + 1 \cdot 2 = 2$; $z_2 = 1 \cdot 1 + 0 \cdot 2 = 1$; $z_3 = -1 \cdot 1 + 1 \cdot 2 = 1$. Exponentials: $e^2 = 7.389$, $e^1 = 2.718$, $e^1 = 2.718$; sum $= 12.825$. Softmax probabilities:

$$p_1 = \frac{7.389}{12.825} = 0.576, \quad p_2 = \frac{2.718}{12.825} = 0.212, \quad p_3 = \frac{2.718}{12.825} = 0.212.$$

The model currently favors class 1 (0.576), but the truth is class 2 — so there will be a corrective gradient.

23.3 Loss and gradient

Cross-entropy loss $= -\log p_2 = -\log 0.212 = 1.551$. The gradient w.r.t. each logit is $p_k - \mathbf{1}\{k = 2\}$: $\partial/\partial z_1 = p_1 - 0 = 0.576$; $\partial/\partial z_2 = p_2 - 1 = -0.788$; $\partial/\partial z_3 = p_3 - 0 = 0.212$ (these sum to zero, as they must — the softmax constraint). The gradient w.r.t. each weight row is this times $\mathbf{x} = (1, 2)$:

$$\nabla_{\boldsymbol{\beta}_1} = 0.576(1, 2) = (0.576, 1.152), \; \nabla_{\boldsymbol{\beta}_2} = -0.788(1, 2) = (-0.788, -1.576), \; \nabla_{\boldsymbol{\beta}_3} = 0.212(1, 2) = (0.212, 0.424).$$

A gradient *descent* step (subtracting $\eta\nabla$) *decreases* $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_3$ (the wrong classes) and *increases* $\boldsymbol{\beta}_2$ (the right class) along $\mathbf{x}$ — pushing z_2 up and the others down, raising p_2 next time. With $\eta = 0.5$: $\boldsymbol{\beta}_2 \leftarrow (1, 0) - 0.5(-0.788, -1.576) = (1.394, 0.788)$, so z_2 rises from 1 to $1.394 + 1.576 = 2.97$, and recomputing gives $p_2 \approx 0.6$ — the correction worked in one step. This is exactly the "prediction minus one-hot truth, times features" rule derived earlier, now in arithmetic.

Intuition

The softmax gradient's elegance — $(p_k - y_k)\mathbf{x}$ — is why neural network classifiers use softmax-cross-entropy as their final layer: the gradient that flows back is simply the prediction error, with no awkward Jacobian terms surviving (they cancel against the log). This clean signal is what makes deep classifiers trainable. The same expression appears here in a one-layer model and in the last layer of a transformer; the difference is only what produces the logits z_k. Understanding it in the logistic/softmax case is understanding the output layer of essentially every modern classifier.

24 Hierarchical and Mixed-Effects Logistic Models

Financial data are grouped — loans within banks, trades within accounts, firms within sectors — and ignoring the grouping wastes information and understates uncertainty. **Mixed-effects** (hierarchical) logistic regression adds group-specific random intercepts (and slopes):

$$\operatorname{logit}(p_{ij}) = \beta_0 + u_j + \mathbf{x}_{ij}^\top \boldsymbol{\beta}, \qquad u_j \sim \mathcal{N}(0, \tau^2),$$

where u_j is group j's deviation from the global intercept, itself drawn from a population distribution. This **partial pooling** shrinks each group's estimate toward the global mean by an amount depending on the group's data volume — data-rich groups stay near their own estimate, data-poor groups borrow strength from the population. It is the James–Stein / empirical-Bayes shrinkage of the probability note, applied to grouped logistic regression, and it is the principled middle ground between ignoring groups (complete pooling) and fitting each separately (no pooling, noisy for small groups).

Worked Example

Modeling default across many small loan portfolios: a portfolio with 5 loans and 1 default should not be estimated at a 20% default rate (noisy), nor forced to the global rate (ignores its signal). Partial pooling shrinks its estimate toward the global rate by an amount set by its small size — giving a sensible, regularized rate that borrows strength from the whole book. As the portfolio accumulates more loans, its estimate relies more on its own data. This is exactly the behavior a risk manager wants, and it falls out of the hierarchical model's population prior $u_j \sim \mathcal{N}(0, \tau^2)$ automatically, with τ^2 (the between-group variance) estimated from the data.

25 Conformal Classification

The conformal framework (bias–variance note) extends to classification, producing prediction *sets* with guaranteed coverage — valuable when abstention or hedging matters.

25.1 The construction

Using a calibration set, compute nonconformity scores — e.g. $s_i = 1 - \hat{p}(y_i \mid \mathbf{x}_i)$, one minus the predicted probability of the true class. The threshold $\hat{q}$ is the $(1-\alpha)$ quantile of these scores. For a new point, the prediction set is *all classes whose predicted probability is high enough*: $\{k : 1 - \hat{p}(k \mid \mathbf{x}) \le \hat{q}\}$. This set covers the true label with probability $\ge 1 - \alpha$, distribution-free.

25.2 Why sets, not points

The set's *size* signals confidence: a singleton means the model is confident; a large set means it is uncertain and is honestly saying so. This is more useful than a forced single prediction with

an unreliable probability — in high-stakes classification (will this trade breach a limit? is this transaction fraud?), a model that outputs "either class A or class B, with 90% coverage" is more actionable than a miscalibrated point prediction. Adaptive variants (APS, RAPS) produce sets that are small where the model is confident and large where it is not, with the same coverage guarantee.

Intuition

Conformal classification turns a classifier into an honest uncertainty quantifier without retraining or distributional assumptions — it only needs a held-out calibration set and exchangeability. For a quant, the prediction-set size is a built-in confidence filter: act on singleton predictions, abstain or hedge on large sets. This directly addresses the calibration problem that plagues SVMs and boosted trees (which output unreliable probabilities): rather than trusting a miscalibrated probability, conformal prediction gives a coverage guarantee on a *set*, which is exactly the kind of honest, decision-relevant uncertainty that the wellbeing of a trading book depends on. The exchangeability caveat (broken by regime shifts) again means time-series variants are needed for trading.

26 Inference for Logistic Models: the Three Tests, Worked

Three asymptotically-equivalent tests assess coefficients in a GLM, all derived from the likelihood (the probability note's MLE theory). We work each on a coefficient.

26.1 The trio

- **Wald test**: $\frac{\hat{\beta}_j}{\mathrm{SE}(\hat{\beta}_j)}$, where SE comes from the inverse Fisher information $(\mathbf{X}^\top \mathbf{W}\mathbf{X})^{-1}$. Squared, it is χ^2_1. Uses only the fitted (alternative) model — cheapest.
- **Likelihood-ratio test**: $2(\ell_{\text{full}} - \ell_{\text{reduced}}) \sim \chi^2_q$ for q restrictions. Fits both models — most reliable.
- **Score (Lagrange multiplier) test**: based on the gradient at the restricted estimate. Uses only the reduced model — useful when the full model is hard to fit.

26.2 A worked LR test

A logistic model on $n = 500$ has log-likelihood $\ell_{\text{full}} = -220$ with a candidate feature, $\ell_{\text{reduced}} = -224$ without it. The LR statistic is $2(-220 - (-224)) = 2(4) = 8.0$, compared to χ^2_1 (one restriction). The 5% critical value is 3.84; since $8.0 > 3.84$ ($p \approx 0.005$), the feature significantly improves fit. A Wald test would compute $(\hat{\beta}/\mathrm{SE})^2$ and compare to the same χ^2_1; the two usually agree, but the LR test is preferred when they diverge (e.g. near separation, where the Wald SE is unreliable because the likelihood is far from quadratic).

26.3 Deviance and pseudo-R^2

The **deviance** $D = -2\ell$ is the GLM analogue of the residual sum of squares (for Gaussian GLMs they coincide). McFadden's pseudo-$R^2 = 1 - \ell_{\text{full}}/\ell_{\text{null}}$ measures improvement over the intercept-only model: with $\ell_{\text{full}} = -220$ and $\ell_{\text{null}} = -300$, pseudo-$R^2 = 1 - 220/300 = 0.27$. Pseudo-$R^2$ values run lower than linear R^2 for comparable fits (values of 0.2–0.4 indicate good fit), so they should not be read on the linear-R^2 scale — a common misinterpretation.

Intuition

The three tests are the same likelihood viewed from three points: the Wald test approximates the log-likelihood as quadratic around the MLE (curvature = Fisher information), the score test uses its slope at the null, and the LR test uses the actual height difference. When the quadratic approximation is good they agree; when it is not — small samples, near-separation, parameters near a boundary — they diverge and the LR test, using the true likelihood values, is most trustworthy. This is the inferential payoff of the MLE/Fisher-information theory (probability note): every coefficient test in a GLM is an instance of these likelihood-based comparisons, and knowing which to trust when they conflict is a mark of statistical maturity.

27 Worked ROC for a Credit Model

We build an ROC curve by hand from a handful of scored loans, cementing the metric. Eight loans with model scores and true default labels (1 = default), sorted by score descending: scores $0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2$ with labels $1, 1, 0, 1, 0, 0, 1, 0$ (four defaults, four non-defaults).

27.1 Sweeping the threshold

At each threshold, count true positives (defaults caught) and false positives (good loans flagged). Starting from the highest score and lowering the threshold past each loan: after the first (0.9, default): TP= 1,FP= 0, TPR= $1/4 = 0.25$, FPR= $0/4 = 0$. After 0.8 (default): TPR= 0.5, FPR= 0. After 0.7 (non-default): TPR= 0.5, FPR= 0.25. After 0.6 (default): TPR= 0.75, FPR= 0.25. After 0.5 (non-default): TPR= 0.75, FPR= 0.5. After 0.4 (non-default): TPR= 0.75, FPR= 0.75. After 0.3 (default): TPR= 1.0, FPR= 0.75. After 0.2 (non-default): TPR= 1.0, FPR= 1.0.

27.2 The area

The ROC curve passes through these (FPR, TPR) points. Computing the area by the trapezoidal rule (or the Mann–Whitney count of correctly-ordered default/non-default pairs, document 8): of the $4 \times 4 = 16$ default/non-default pairs, count those where the default scored higher. The defaults scored $0.9, 0.8, 0.6, 0.3$; the non-defaults $0.7, 0.5, 0.4, 0.2$. Pairs where default > non-default: 0.9 beats all 4; 0.8 beats all 4; 0.6 beats $0.5, 0.4, 0.2$ (3); 0.3 beats 0.2 (1). Total $= 4 + 4 + 3 + 1 = 12$. AUC $= 12/16 = 0.75$. The model ranks a random default above a random non-default 75% of the time — decent but imperfect discrimination, exactly the Mann–Whitney interpretation, computed by counting.

28 Separation and Its Remedies, Concretely

When a feature perfectly separates the classes, the MLE diverges (the coefficient runs to infinity, as the near-separable IRLS run hinted). Remedies, all seen in the series:

- **Penalization** (ℓ_2 ridge): the penalty keeps coefficients finite by adding $\lambda \|\boldsymbol{\beta}\|^2$, restoring a unique maximum (document 1's regularization, document 2's MAP view).
- **Firth's penalized likelihood**: penalizes by the Jeffreys prior (the square-root of the Fisher information determinant), reducing small-sample bias and resolving separation — the standard fix in rare-event/credit settings.
- **Bayesian priors**: a weakly-informative prior on the coefficients (e.g. a Cauchy) keeps the posterior proper and the estimates finite.

All work by the same mechanism: adding information (a penalty/prior) that bounds the otherwise-unbounded likelihood, the recurring regularization theme.

Intuition

Separation is logistic regression's most instructive failure: the data are "too good" (perfectly separable), the unpenalized MLE diverges, and the textbook formulas break (infinite coefficients, undefined Wald standard errors). The remedies all add a regularizing prior — ridge, Firth's Jeffreys prior, or an explicit Bayesian prior — which is the same cure regularization provides for collinearity in linear regression: bound an ill-posed estimate by injecting external information. That the same fix resolves both collinearity (document 1) and separation (document 2) underscores regularization's unifying role: whenever the data alone do not pin down a unique, finite, stable estimate, a penalty/prior supplies what is missing.

29 Logistic Regression from Scratch via IRLS

Implementing iteratively reweighted least squares makes the GLM-Newton connection concrete and ties to the IRLS theory of the main text.

IRLS / Newton's method for logistic regression from scratch

```
import numpy as np

def logistic_irls(X, y, n_iter=25, ridge=1e-6):
    """Newton-Raphson = IRLS for logistic regression."""
    X = np.column_stack([np.ones(len(X)), X])          # add intercept
    beta = np.zeros(X.shape[1])
    for it in range(n_iter):
        eta = X @ beta
        p = 1/(1+np.exp(-eta))                          # mean function
        W = p*(1-p)                                     # IRLS weights = variance
        # gradient and Hessian of the negative log-likelihood
        grad = X.T @ (p - y) + ridge*beta
        H = X.T @ (X * W[:,None]) + ridge*np.eye(X.shape[1])
        step = np.linalg.solve(H, grad)                 # Newton step
        beta -= step
        if np.linalg.norm(step) < 1e-10:
            print(f"converged in {it+1} iterations"); break
    return beta

rng = np.random.default_rng(0)
X = rng.standard_normal((500, 3))
true_beta = np.array([0.5, 1.5, -2.0, 1.0])            # incl intercept
p = 1/(1+np.exp(-(np.column_stack([np.ones(500),X]) @ true_beta)))
y = (rng.random(500) < p).astype(float)
beta = logistic_irls(X, y)
print("recovered beta:", np.round(beta, 2), " (true:", true_beta, ")")
# W = p(1-p) is the GLM variance function; the Hessian X^T W X is the Fisher
# information; the update solves a WEIGHTED least squares -- hence "IRLS".
```

The implementation lays bare the IRLS/Newton identity from the main text: the weights $W = p(1-p)$ are the Bernoulli variance function, the Hessian $\mathbf{X}^\top \mathbf{W}\mathbf{X}$ is the Fisher information, and each Newton step solves a weighted least squares — which is why the method is "iteratively reweighted least squares." It converges in a handful of iterations (the quadratic convergence of Newton's method near the optimum, document on optimization) and recovers the true coefficients, confirming the theory in running code. Swapping the mean function p and weights W for another GLM's (Poisson: $\mu = e^\eta$, $W = \mu$) reuses the entire loop — the GLM unity, executable.

Imbalanced classification: class weights, calibration, threshold

```python
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.calibration import CalibratedClassifierCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import average_precision_score, brier_score_loss

rng = np.random.default_rng(0)
n = 8000
X = rng.standard_normal((n, 6))
score = 1.2*X[:,0] - 1.5*X[:,1] + 0.8*X[:,2] - 2.5     # rare positive
y = (rng.random(n) < 1/(1+np.exp(-score))).astype(int)
print("positive rate:", round(y.mean(), 3))

Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3,
                                      stratify=y, random_state=0)
# class_weight balances the loss; calibration fixes probabilities afterward
base = LogisticRegression(class_weight="balanced", max_iter=1000)
cal = CalibratedClassifierCV(base, method="sigmoid", cv=5).fit(Xtr, ytr)
p = cal.predict_proba(Xte)[:,1]
print(f"PR-AUC: {average_precision_score(yte, p):.3f}")
print(f"Brier : {brier_score_loss(yte, p):.3f}")

# Cost-optimal threshold: a missed positive costs 8x a false alarm
C_FP, C_FN = 1.0, 8.0
thr = C_FP/(C_FP+C_FN)
flagged = p >= thr
print(f"threshold={thr:.3f}  flag rate={flagged.mean():.3f}  "
      f"recall={yte[flagged].sum()/yte.sum():.3f}")
# class_weight rebalances training; calibration restores honest probabilities
# (rebalancing distorts them); the threshold comes from the cost ratio, not 0.5.
```

The imbalanced pipeline composes the document's lessons: `class_weight="balanced"` rebalances the loss so the rare class is not ignored, calibration restores honest probabilities (rebalancing distorts them away from the true prior, document 2), PR-AUC and Brier are the imbalance-appropriate metrics (document 8), and the decision threshold comes from the cost ratio $C_{\text{FP}}/(C_{\text{FP}}+C_{\text{FN}})$ rather than the naive 0.5. This is logistic regression deployed correctly for the rare-event problems — fraud, default, churn — that dominate quant classification.

30 Worked Exercises

Worked Example

Exercise. Show that the logistic regression log-likelihood is concave, guaranteeing a unique maximum.

Solution. The log-likelihood is $\ell(\boldsymbol{\beta}) = \sum_i [y_i \eta_i - \log(1 + e^{\eta_i})]$ with $\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$. The gradient is $\sum_i (y_i - p_i)\mathbf{x}_i$ and the Hessian is $-\sum_i p_i(1-p_i)\mathbf{x}_i\mathbf{x}_i^\top = -\mathbf{X}^\top \mathbf{W}\mathbf{X}$ with $\mathbf{W} = \operatorname{diag}(p_i(1-p_i)) \succeq 0$. For any vector $\mathbf{v}$, $\mathbf{v}^\top(\mathbf{X}^\top\mathbf{W}\mathbf{X})\mathbf{v} = \sum_i p_i(1-p_i)(\mathbf{x}_i^\top \mathbf{v})^2 \geq 0$, so $\mathbf{X}^\top\mathbf{W}\mathbf{X} \succeq 0$ and the Hessian $-\mathbf{X}^\top\mathbf{W}\mathbf{X} \preceq 0$ — negative semidefinite. A function with negative semidefinite Hessian everywhere is concave, so ℓ has no local maxima other than the global one. (It is strictly concave, hence the maximum is unique, when $\mathbf{X}$ has full column rank and no perfect separation.) This is why logistic regression is a well-behaved convex optimization, unlike the non-convex objectives of neural networks.

Worked Example

Exercise. A logistic model is fit with an intercept. Prove that the average predicted probability equals the observed positive rate on the training set.
Solution. The score (gradient) equation for the intercept β_0 sets $\partial \ell / \partial \beta_0 = \sum_i (y_i - p_i) \cdot 1 = 0$, since the intercept's "feature" is the constant 1. Therefore $\sum_i p_i = \sum_i y_i$, i.e. $\frac{1}{n}\sum_i \hat{p}_i = \frac{1}{n}\sum_i y_i$ — the mean predicted probability equals the empirical positive rate. This is automatic (aggregate) calibration: a logistic model with an intercept is correctly calibrated *on average* on its training data, a property most other classifiers (SVM, random forest, boosting) lack and must achieve through post-hoc calibration. It does not guarantee calibration within subgroups or out-of-sample, but it is the reason logistic regression is the natural baseline when calibrated probabilities are required.

31 Survival Analysis and the Cox Connection

Time-to-event data — time to default, time to prepayment, time to a trade filling — need models for *when*, not just *whether*, an event occurs. These connect logistic regression to survival analysis.

31.1 Hazards and the discrete-time link

The **hazard** $h(t)$ is the instantaneous event rate given survival so far. The complementary-log-log link (document 2's links section) is exactly the discrete-time hazard model: modeling whether an event occurs in each period, with cloglog link, recovers a proportional-hazards structure. So a logistic-style model on period-by-period survival data *is* a discrete survival model — the same IRLS machinery, applied to expanded (person-period) data.

31.2 The Cox proportional-hazards model

The **Cox model** posits $h(t \mid \mathbf{x}) = h_0(t)\exp(\mathbf{x}^\top \boldsymbol{\beta})$ — a baseline hazard $h_0(t)$ (left unspecified) scaled multiplicatively by the covariates. The coefficients are estimated by *partial likelihood*, which conditions away the nuisance baseline, leaving a likelihood depending only on the *order* of events — structurally similar to the logistic likelihood and maximized the same way. e^{β_j} is a hazard ratio: the multiplicative effect on the event rate per unit of x_j, directly analogous to logistic regression's odds ratio. For a quant, the Cox model estimates how covariates accelerate or delay default, prepayment, or other timed events, with censored observations (loans not yet defaulted) handled naturally.

Intuition

Survival analysis extends the classification of document 2 from "will it happen?" to "when will it happen?", and the bridge is the hazard: modeling the period-by-period event probability with a cloglog link is discrete-time survival, and the Cox model is its continuous-time, semi-parametric cousin. The shared structure — a linear predictor through a link, fit by (partial) likelihood, with multiplicative covariate effects — means the GLM intuition transfers directly. For credit (time to default), prepayment modeling, and execution (time to fill), survival models answer the timing question that a binary classifier cannot, while reusing the estimation machinery the series has built.

32 The GLM Family, Synthesized

Document 2 built the GLM framework; we close by mapping its members and their quant uses in one view.

- **Gaussian + identity link** = linear regression (document 1): continuous targets, returns.
- **Bernoulli + logit** = logistic regression: binary events, default/no-default.
- **Multinomial + softmax**: unordered classes, regime labels.
- **Poisson + log** : counts, trade arrivals, order counts.
- **Negative binomial + log**: overdispersed counts, the realistic default.
- **Gamma + log**: positive continuous, durations, claim sizes, with variance growing as the square of the mean.
- **Bernoulli + cloglog**: discrete-time hazards, survival.

All share one estimation engine — IRLS / Fisher scoring (document 2) — parameterized by the response distribution's mean and variance functions. Choosing the family is choosing the assumed noise distribution; choosing the link is choosing how the linear predictor maps to the mean.

Intuition

The GLM framework's power is its unity: one fitting algorithm (IRLS), one inferential apparatus (the Wald/LR/score tests of document 2, from the likelihood), and one diagnostic toolkit (deviance, residuals), spanning regression, classification, count models, and survival. A quant who internalizes the GLM view does not see linear regression, logistic regression, and Poisson regression as separate methods but as one method with three choices of distribution and link — and can therefore fit the *right* model for any response type (continuous, binary, categorical, count, positive, time-to-event) by changing two lines. This unification, with the convexity and calibration properties that come with it, is why GLMs remain the interpretable, reliable backbone of quantitative modeling even as flashier methods come and go.

33 Case Study: A Credit Scorecard End-to-End

We build a complete credit scorecard — the regulated industry's standard application of logistic regression — integrating weight-of-evidence encoding, calibration, and the scorecard scaling.

Logistic scorecard with WoE features, calibration, and points scaling

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, brier_score_loss

rng = np.random.default_rng(0)
n = 12000
age    = rng.uniform(20, 70, n)
income = rng.gamma(3, 20000, n)
util   = rng.beta(2, 5, n)                              # credit utilization
logit  = -3.0 - 0.03*(age-45) - 1.2e-5*(income-60000) + 4.0*util
y = (rng.random(n) < 1/(1+np.exp(-logit))).astype(int)    # 1 = default
print("default rate:", round(y.mean(), 3))

def woe_encode(x, y, bins=10):
    # Weight-of-evidence: log(bad-rate-in-bin / good-rate-in-bin)
    q = np.quantile(x, np.linspace(0,1,bins+1)); q[0],q[-1] = -np.inf, np.inf
    idx = np.digitize(x, q[1:-1])
    woe = np.zeros_like(x, dtype=float)
    tot_bad, tot_good = y.sum(), len(y)-y.sum()
    for b in np.unique(idx):
        m = idx == b
        bad  = y[m].sum() + 0.5                         # smoothed counts
        good = m.sum() - y[m].sum() + 0.5
        woe[m] = np.log((bad/tot_bad)/(good/tot_good))
    return woe
```

```

X = np.column_stack([woe_encode(v, y) for v in (age, income, util)])
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3,
                                      stratify=y, random_state=0)
lr = LogisticRegression(max_iter=1000).fit(Xtr, ytr)
p = lr.predict_proba(Xte)[:,1]
print(f"AUC: {roc_auc_score(yte,p):.3f}   Brier: {brier_score_loss(yte,p):.3f}")

# Scale log-odds to a points score: PDO points double the odds
PDO, base_score, base_odds = 20, 600, 1/50
factor = PDO/np.log(2); offset = base_score - factor*np.log(base_odds)
log_odds = lr.decision_function(Xte)                # = X @ beta
score = offset - factor*log_odds                    # higher score = safer
print(f"score range: [{score.min():.0f}, {score.max():.0f}]  "
      f"mean {score.mean():.0f}")
# WoE makes each feature monotone in log-odds and linear for the logistic fit;
# the points scaling (PDO) is the industry-standard transform turning
# calibrated log-odds into an interpretable 300-850-style score.
```

The scorecard is logistic regression in its most consequential real-world form. Weight-of-evidence encoding (document 2's WoE/scorecard section) transforms each feature into its log-odds contribution — monotone, linear in the logit, and interpretable. The logistic model combines them, its calibrated probabilities (the intercept guaranteeing aggregate calibration, document 2) feed expected-loss pricing, and the points scaling (points-to-double-the-odds, PDO) converts log-odds into the familiar credit-score scale where a fixed number of points always doubles the odds. Every step is interpretable and auditable — which is why logistic scorecards remain the regulated standard despite more accurate black boxes existing.

Intuition

The credit scorecard is logistic regression's killer application and a microcosm of why the method endures: WoE encoding gives interpretable, monotone features; the convex, calibrated logistic fit gives reliable probabilities; the points scaling gives a transparent score a regulator and a consumer can understand. A gradient-boosted model might score marginally higher AUC, but the scorecard's every prediction decomposes into per-feature point contributions — "you lost 40 points for high utilization" — which is legally and ethically required in lending. This is the document's deepest practical lesson: in high-stakes, regulated decisions, the interpretability and calibration of logistic regression often outweigh raw accuracy, and knowing when transparency trumps performance is as important as knowing the algorithms.

34 Further Worked Exercises

These exercises consolidate the GLM and classification theory.

Worked Example

Exercise. Derive the IRLS update for logistic regression and identify each piece as a GLM quantity.

Solution. Newton's method on the log-likelihood: $\boldsymbol{\beta}^{new} = \boldsymbol{\beta} - \mathbf{H}^{-1}\mathbf{g}$ with gradient $\mathbf{g} = \mathbf{X}^\top(\mathbf{p} - \mathbf{y})$ and Hessian $\mathbf{H} = -\mathbf{X}^\top\mathbf{W}\mathbf{X}$, $\mathbf{W} = \mathrm{diag}(p_i(1-p_i))$. Substituting gives $\boldsymbol{\beta}^{new} = (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{W}\mathbf{z}$ where $\mathbf{z} = \mathbf{X}\boldsymbol{\beta} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})$ is the working response. This is weighted least squares (hence IRLS): $\mathbf{W}$ is the GLM variance function $p(1-p)$, $\mathbf{X}^\top\mathbf{W}\mathbf{X}$ is the Fisher information, and $\mathbf{z}$ is the linearized response. The same template fits any GLM by swapping the mean function and variance — the unification of document 2.

Worked Example

Exercise. Show that for logistic regression the decision boundary is linear in the features.
Solution. The model predicts class 1 when $p = \sigma(\mathbf{x}^\top \boldsymbol{\beta}) > 0.5$, i.e. when $\mathbf{x}^\top \boldsymbol{\beta} > 0$ (since $\sigma(0) = 0.5$ and σ is increasing). The set $\{\mathbf{x} : \mathbf{x}^\top \boldsymbol{\beta} = 0\}$ is a hyperplane — a linear boundary. So despite the nonlinear sigmoid, logistic regression is a *linear* classifier: the nonlinearity maps the linear score to a probability but does not bend the boundary. To get nonlinear boundaries one must add nonlinear features (basis expansion) or use a kernel — the same move as in linear regression (document 1) and SVMs (document 4).

Worked Example

Exercise. Why can logistic regression coefficients be unstable under multicollinearity, and what is the fix?
Solution. The coefficient covariance is $(\mathbf{X}^\top \mathbf{W}\mathbf{X})^{-1}$ (inverse Fisher information). Under multicollinearity $\mathbf{X}^\top \mathbf{W}\mathbf{X}$ is near-singular (small eigenvalues), so its inverse has large entries — inflated coefficient variances, the same mechanism as OLS multicollinearity (document 1) but with the IRLS weights $\mathbf{W}$. Coefficients become large, unstable, and sign-flipping. The fix is regularization: ℓ_2 (ridge) logistic regression adds $\lambda \mathbf{I}$ to the Hessian, bounding the eigenvalues away from zero and stabilizing the estimates (the MAP-with-Gaussian-prior view, document 2) — exactly the ridge cure for linear regression, transferred to the GLM.

34.1 Additional Problems

Worked Example

Exercise. Explain the difference between a probit and a logit model and when the choice matters.
Solution. Both model a binary outcome via a linear index passed through a CDF: logit uses the logistic CDF $\sigma(\eta)$, probit uses the standard normal CDF $\Phi(\eta)$. The two are nearly identical in the middle and differ only slightly in the tails (logit has marginally heavier tails). Coefficients differ by roughly a factor of 1.6 (the ratio of the distributions' scales) but imply almost the same fitted probabilities. The choice rarely matters for prediction; it matters for interpretation (logit gives clean odds ratios e^{β}; probit coefficients lack that interpretation) and for theoretical convenience (probit arises naturally from a Gaussian latent-variable model, useful in multivariate/selection models). In practice logit is the default for its interpretability; probit appears where a Gaussian latent structure is assumed.

Worked Example

Exercise. Why is accuracy a poor training objective for logistic regression compared to log-loss?
Solution. Accuracy (the 0/1 loss) is non-differentiable and non-convex in the parameters — it is flat almost everywhere (small parameter changes do not flip discrete predictions) with jumps at decision boundaries, so it provides no usable gradient for optimization. Log-loss is a smooth, convex surrogate (document 2): it is differentiable, its gradient $(\mathbf{p} - \mathbf{y})$ points usefully, and minimizing it is a well-posed convex problem with a unique optimum. Moreover log-loss is a strictly proper scoring rule, so minimizing it yields calibrated probabilities, whereas optimizing accuracy directly would ignore confidence entirely. This is why classifiers train on cross-entropy and merely *evaluate* on accuracy — the smooth proper loss is the trainable proxy for the discrete goal.

35 Marginal Effects and Model Interpretation

A logistic model's coefficients are log-odds-ratios, not probability changes, so interpreting them for decisions requires care — the source of much confusion in practice.

35.1 Why coefficients are not probability effects

In $\text{logit}(p) = \mathbf{x}^\top \boldsymbol{\beta}$, a unit increase in x_j raises the log-odds by β_j, so the odds multiply by e^{β_j} (the odds ratio). But the effect on the *probability* is nonlinear and depends on the starting point: the same β_j produces a large probability change near $p = 0.5$ (where the sigmoid is steep) and a tiny one near $p = 0$ or $p = 1$ (where it is flat). So a coefficient alone does not tell you how much a feature changes the probability — the baseline matters.

35.2 Average marginal effects

The **marginal effect** of x_j is $\partial p / \partial x_j = \beta_j \, p(1-p)$ — the coefficient scaled by the sigmoid slope $p(1-p)$ (maximal at $p = 0.5$). Since this varies across observations, practitioners report the **average marginal effect** (the mean of $\beta_j p_i (1-p_i)$ over the sample), giving the average probability change per unit of x_j — a directly interpretable, decision-relevant quantity. For a credit model, "a one-point utilization increase raises default probability by 2.3 percentage points on average" is the actionable statement, derived from the coefficient via the marginal effect, not the coefficient itself.

Intuition

The coefficient-vs-marginal-effect distinction is where logistic regression interpretation most often goes wrong, and getting it right is essential for decisions. The coefficient is constant (a log-odds-ratio); the probability effect is not (it depends on the baseline through $p(1-p)$). Reporting odds ratios suits relative-risk statements ("doubles the odds"); reporting average marginal effects suits absolute statements ("raises probability by X points"). For quant and credit applications where the probability feeds an expected-value calculation, the marginal effect is the relevant quantity, and confusing it with the raw coefficient leads to mis-sized decisions. This nuance — the same nonlinearity that makes the model a classifier complicates its interpretation — is the price of the sigmoid link, and handling it correctly distinguishes careful from careless modeling."

36 Worked Exercise Solutions

Solution to Exercise 1 (sigmoid derivative). Write $\sigma(z) = (1+e^{-z})^{-1}$. Then $\sigma'(z) = -(1+e^{-z})^{-2} \cdot (-e^{-z}) = \frac{e^{-z}}{(1+e^{-z})^2}$. Factor: $= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \sigma(z) \cdot (1-\sigma(z))$, using $1 - \sigma(z) = \frac{e^{-z}}{1+e^{-z}}$. For the score, $\partial_{\boldsymbol{\beta}} \ell = \sum_i [y_i - p_i]\mathbf{x}_i$ since $\partial_{\boldsymbol{\beta}} [y_i \log \sigma(\eta_i) + (1-y_i)\log(1-\sigma(\eta_i))] = [y_i - \sigma(\eta_i)]\mathbf{x}_i$ after the $\sigma' = \sigma(1-\sigma)$ cancellation; stacking gives $\mathbf{X}^\top(\mathbf{y} - \boldsymbol{p})$.

Solution to Exercise 2 (Hessian, concavity). Differentiate the score: $\partial_{\boldsymbol{\beta}} [\,(y_i - \sigma(\eta_i))\mathbf{x}_i\,] = -\sigma'(\eta_i)\mathbf{x}_i\mathbf{x}_i^\top = -p_i(1-p_i)\mathbf{x}_i\mathbf{x}_i^\top$. Summing, $\nabla^2 \ell = -\sum_i p_i(1-p_i)\mathbf{x}_i\mathbf{x}_i^\top = -\mathbf{X}^\top \mathbf{W}\mathbf{X}$. For any vector $\mathbf{v}$, $\mathbf{v}^\top \mathbf{X}^\top \mathbf{W}\mathbf{X}\mathbf{v} = \sum_i w_i (\mathbf{x}_i^\top \mathbf{v})^2 \geq 0$ with $w_i > 0$, so $\mathbf{X}^\top \mathbf{W}\mathbf{X} \succeq 0$ and $\nabla^2 \ell \preceq 0$: concave. Strictly if $\mathbf{X}$ has full column rank (then the quadratic form is positive unless $\mathbf{v} = \mathbf{0}$).

Solution to Exercise 4 (separation diverges). Take $x_i \in \{-2, -1, 1, 2\}$ with $y_i = 0$ for $x < 0$ and $y_i = 1$ for $x > 0$ — perfectly separated at $x = 0$. The model $\sigma(\beta x)$ has likelihood $\prod_{x>0} \sigma(\beta x) \prod_{x<0} (1 - \sigma(\beta x)) = \prod \sigma(\beta |x|)$, which increases monotonically toward 1 as $\beta \to \infty$. No finite maximizer exists. Adding $\frac{\lambda}{2}\beta^2$ makes the objective $-\sum \log \sigma(\beta|x|) + \frac{\lambda}{2}\beta^2$, whose derivative $-\sum |x|(1 - \sigma(\beta|x|)) + \lambda\beta = 0$ has a finite root because the penalty term grows linearly while the likelihood gradient saturates — yielding a finite $\hat{\beta}$.

Solution to Exercise 6 (calibration from the score). With an intercept, the first score equation is $\sum_i (y_i - p_i) = 0$, i.e. $\sum_i p_i = \sum_i y_i$. Dividing by n: the average predicted probability equals the empirical positive rate. So the model is calibrated *in aggregate* on the training data by construction — a property that survives only on the training distribution and is broken by regularization or resampling, hence the need to recalibrate afterward.

37 Exercises

1. Derive $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ and use it to obtain the score $\nabla \ell = \mathbf{X}^\top (\mathbf{y} - \boldsymbol{p})$.
2. Prove the Hessian is $-\mathbf{X}^\top \mathbf{W} \mathbf{X}$ and hence the log-likelihood is concave.
3. Show the Newton update rearranges into IRLS and identify the working response.
4. Construct a perfectly separable 1-D dataset and show the MLE diverges; then add an ℓ_2 penalty and find the finite solution.
5. Derive the softmax probabilities' invariance to a constant shift and explain the need for a reference class.
6. Show that with an intercept the average predicted probability equals the base rate (calibration), starting from the score equation.
7. Explain why oversampling the minority class biases the intercept and derive the correction.

End of Logistic Regression & GLMs.