L02: Logistic Regression

Full Lecture: Mathematical Foundations, Inference, and Credit Scoring

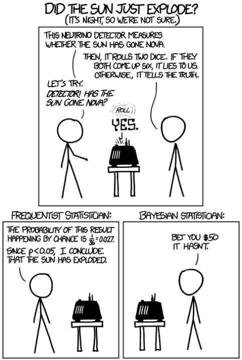
Methods and Algorithms

MSc Data Science

Frequentist vs. Bayesian

- A neutrino detector signals "the sun has gone nova"
- The frequentist: "the p-value is below 0.05, so we reject the null"
- The Bayesian: "given my prior that the sun exists, I bet the detector is lying"

Logistic regression lives in this tension: the model gives you a probability, but how you *interpret* it depends on your framework. Today we build the math that both camps agree on.



XKCD #1132 by Randall Munroe (CC BY-NC 2.5) – "Is the sun going to explode?"

Learning Objectives

After this lecture you will be able to:

1. **Derive** the MLE for logistic regression via the gradient of the log-likelihood (Analyze)
2. **Evaluate** classification performance using ROC, precision-recall, calibration, and Gini (Evaluate)
3. **Analyze** model fit via deviance, Wald test, LRT, AIC/BIC, and Hosmer–Lemeshow (Analyze)
4. **Apply** logistic regression to credit scoring with Basel-compliant PD estimation (Apply)
5. **Compare** regularization strategies (L1, L2, Elastic Net) and their effect on model selection (Evaluate)

Finance application:

Every objective maps to a real credit-scoring task: PD estimation, model validation, regulatory reporting, and scorecard deployment under Basel II/III.

Bloom's taxonomy levels 3–5: Apply, Analyze, Evaluate

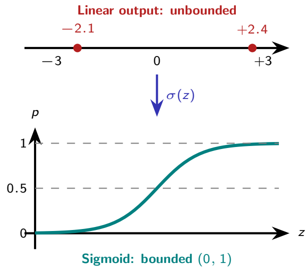
The Core Tension

- A linear model predicts $\hat{y} = \mathbf{w}^\top \mathbf{x} + b$, which can be -3.2 or $+1.7$
- But a bank needs $P(\text{default}) \in [0, 1]$ for capital reserves
- Thresholding a raw score at zero gives a decision, not a probability

The sigmoid function bridges this gap: it maps any real number to $(0, 1)$, turning an unbounded score into a calibrated probability of default.

Insight

Classification demands a probability, but a linear model gives you an unbounded number. The sigmoid is the mathematical bridge, and everything else follows from maximum likelihood.



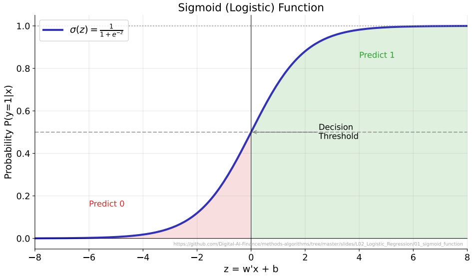
The sigmoid (logistic) function: $\sigma(z) = 1/(1 + e^{-z})$ **maps** $\mathbb{R} \to (0, 1)$

What Happens When You Pass a Linear Combination Through a Sigmoid?

The Sigmoid Function

- **What you see:** The S-curve mapping any real z to a value in $(0, 1)$
- **Key pattern:** The steepest change is at $z = 0$ where $\sigma(0) = 0.5$
- **Takeaway:** Features far from the boundary produce confident predictions near 0 or 1

Key properties: $\sigma(z) = \frac{1}{1+e^{-z}}, \quad \sigma(-z) = 1 - \sigma(z), \quad \sigma'(z) = \sigma(z)(1 - \sigma(z)).$



Insight

The derivative $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ peaks at $z = 0$ and vanishes at extremes – this governs gradient magnitude during training.

The sigmoid is the canonical link function for binomial GLMs – it maps the linear predictor to a probability

How Do Odds and Log-Odds Connect Features to Probabilities?

The Logit Link

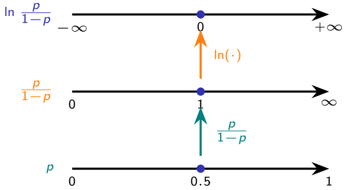
- **Odds:** $\frac{p}{1-p}$ – how much more likely default is than non-default
- **Log-odds (logit):** $\ln\left(\frac{p}{1-p}\right) = \mathbf{w}^\top \mathbf{x} + b$
- **Odds ratio:** e^{w_j} = multiplicative change in odds per unit increase in x_j

Worked example: If $w_{\text{income}} = 0.5$, then OR $= e^{0.5} = 1.65$. A one-unit increase in income multiplies the odds of default by 1.65.

Insight

The logit is the natural parameter of the Bernoulli distribution. This is why logistic regression is a GLM, not an ad hoc formula.

Three Scales



Coefficients are additive in log-odds, multiplicative in odds – this drives the odds-ratio interpretation

How Does Maximum Likelihood Find the Best Coefficients?

The Likelihood Function

For n observations with labels $y_i \in \{0, 1\}$ and predicted probabilities $p_i = \sigma(\mathbf{w}^\top \mathbf{x}_i)$:

Likelihood: $L(\mathbf{w}) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i}$

Log-likelihood: $\ell(\mathbf{w}) = \sum_{i=1}^{n} [y_i \ln p_i + (1 - y_i) \ln(1 - p_i)]$

Why MLE, not OLS? OLS assumes Gaussian errors. Binary outcomes follow a Bernoulli distribution, so MLE is the natural estimator.

Insight

MLE maximizes the probability of observing the actual data. There is no closed-form solution – we must use iterative optimization.

Key insight:

Taking the log converts the product into a sum, making derivatives tractable.

The log-likelihood is **concave** in **w**, guaranteeing a unique global maximum.

No closed form: unlike linear regression, $\hat{\mathbf{w}}_{\text{MLE}}$ cannot be written as $(X^\top X)^{-1} X^\top y$.

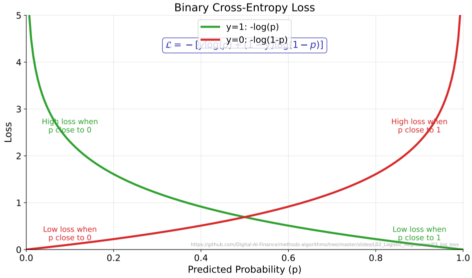
The Bernoulli likelihood is the starting point – everything (gradient, Hessian, inference) derives from it

Binary Cross-Entropy

- **What you see:** The loss curves for $y = 1$ and $y = 0$ as a function of predicted p
- **Key pattern:** Loss explodes as the prediction moves away from the true label
- **Takeaway:** Confident wrong predictions are penalized far more than uncertain ones

$$J(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \ln p_i + (1 - y_i)\ln(1 - p_i)]$$

This is the negative log-likelihood divided by n. Minimizing J = maximizing ℓ.



Insight

Cross-entropy is convex in $\mathbf{w}$ (since $-\ell$ is convex). Every local minimum is the global minimum. No random restarts needed.

Convexity of the loss is the key theoretical advantage of logistic regression over neural networks

What Does the Gradient Look Like in Matrix Form?

Deriving the Gradient

Apply the chain rule to the log-likelihood:

Scalar form: $\frac{\partial \ell}{\partial w_j} = \sum_{i=1}^{n}(y_i - p_i)\, x_{ij}$

Matrix form: $\nabla_{\mathbf{w}} \ell = \mathbf{X}^\top (\mathbf{y} - \mathbf{p})$

For minimizing the loss (negative log-likelihood):

$$\nabla_{\mathbf{w}} J = \frac{1}{n}\mathbf{X}^\top (\mathbf{p} - \mathbf{y})$$

Compare to linear regression: $\nabla J_{\text{LR}} = \frac{1}{n}\mathbf{X}^\top (\mathbf{Xw} - \mathbf{y})$.
The only difference: $\mathbf{p} = \sigma(\mathbf{Xw})$ replaces $\mathbf{Xw}$.

Chain rule steps:

1. $\frac{\partial \ell}{\partial p_i} = \frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}$
2. $\frac{\partial p_i}{\partial z_i} = p_i(1 - p_i)$
3. $\frac{\partial z_i}{\partial w_j} = x_{ij}$
4. Combine: $\frac{\partial \ell}{\partial w_j} = (y_i - p_i)\, x_{ij}$

The $p_i(1 - p_i)$ cancels, yielding a clean form.

Insight

The gradient has the same “residual times features” form as linear regression. The sigmoid is hidden inside $\mathbf{p}$.

Setting the gradient to zero gives the score equations – solved iteratively by gradient descent or Newton’s method

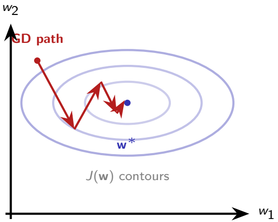
How Does Gradient Descent Update the Weights?

The Update Rule

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{1}{n}\mathbf{X}^\top(\sigma(\mathbf{X}\mathbf{w}^{(t)}) - \mathbf{y})$$

- η = learning rate (too large: diverge; too small: slow)
- Convergence guaranteed for convex loss with appropriate η
- **Feature standardization** essential: unscaled features create elongated contours, slowing convergence

Variants: Batch (full data), mini-batch (subset), stochastic (single sample). Scikit-learn uses L-BFGS by default.



Insight

Gradient descent follows the steepest downhill direction. On elongated loss surfaces (unscaled features), it zigzags. Standardize first.

Convergence rate: $O(1/t)$ for gradient descent vs. quadratic for Newton – but GD avoids Hessian inversion

Why Do Statisticians Prefer Newton-Raphson Over Gradient Descent?

Newton-Raphson (IRLS)

The Hessian of the log-likelihood:
$\mathbf{H} = -\mathbf{X}^\top \mathbf{S}\,\mathbf{X}$
where $\mathbf{S} = \mathrm{diag}(p_i(1-p_i))$ is the weight matrix.

Newton update: $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mathbf{H}^{-1}\nabla\ell$

- Converges in 5–10 iterations (quadratic rate)
- Equivalent to Iteratively Reweighted Least Squares (IRLS)
- Default in R's `glm()` and Python's `statsmodels`

IRLS Pseudocode:

1. Initialize $\mathbf{w}^{(0)} = \mathbf{0}$
2. **Repeat** until convergence:
   a. $\mathbf{p} = \sigma(\mathbf{X}\mathbf{w}^{(t)})$
   b. $\mathbf{S} = \mathrm{diag}(p_i(1-p_i))$
   c. $\mathbf{z} = \mathbf{X}\mathbf{w}^{(t)} + \mathbf{S}^{-1}(\mathbf{y}-\mathbf{p})$
   d. $\mathbf{w}^{(t+1)} = (\mathbf{X}^\top\mathbf{S}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{S}\mathbf{z}$
3. Return $\hat{\mathbf{w}}$

Step (d) is a weighted least squares solve with weights $\mathbf{S}$.

Insight

Newton uses curvature information (Hessian) to take optimal steps. The cost is $O(d^3)$ per iteration for the matrix inversion.

IRLS converges quadratically: each iteration roughly doubles the number of correct digits in $\hat{\mathbf{w}}$

Where Does the Decision Boundary Live in Feature Space?

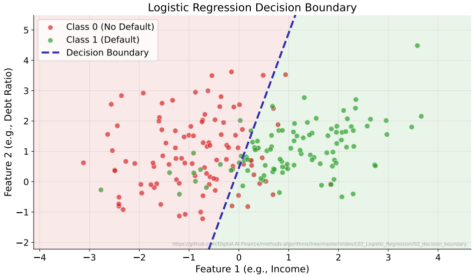
The Linear Boundary

- **What you see:** A line (or hyperplane) separating two classes in feature space
- **Key pattern:** Points far from the boundary have high-confidence predictions
- **Takeaway:** The boundary is where $\mathbf{w}^\top \mathbf{x} + b = 0$, i.e., $P = 0.5$

The threshold $P = 0.5$ is the default, but banks often use $P = 0.3$ or lower for credit scoring where the cost of a missed default is high.

Insight

The model draws a linear boundary. The threshold shifts the boundary parallel to itself – choosing it is a business decision, not a statistical one.



$\mathbf{w}^\top \mathbf{x} + b = 0$ defines a hyperplane in $\mathbb{R}^d$; the normal vector is $\mathbf{w}$

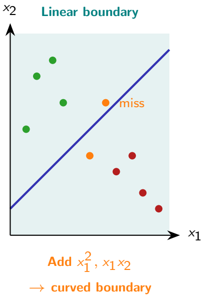
Feature Engineering for Non-Linearity

- **Polynomial features:** Add x^2, $x_1 x_2$ to make the boundary curved
- **Interaction terms:** Capture joint effects (e.g., income $\times$ debt)
- **Binning / splines:** Piecewise-linear approximation of non-linear effects

The model remains linear *in its parameters* – the boundary is non-linear only in the original feature space.

Insight

Feature engineering is how logistic regression competes with non-linear models. The trade-off: flexibility vs. overfitting and loss of interpretability.



Polynomial features of degree d with p variables create $\binom{p+d}{d}$ features – growth is combinatorial

What Changes When There Are More Than Two Classes?

Softmax Regression

For K classes, each has its own weight vector $\mathbf{w}_k$:

$$P(y = k \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})}$$

- Probabilities sum to 1 across all classes
- Loss: categorical cross-entropy $= -\sum_k y_k \ln p_k$
- **One-vs-Rest (OvR):** fit K binary classifiers, cheaper but less principled

For credit scoring, the outcome is usually binary (default/no-default), so softmax is rare. But multi-class extensions appear in rating migration models (AAA, AA, A, BBB, ...).

Insight

Softmax generalizes the sigmoid to K classes. When $K = 2$, softmax reduces exactly to the standard sigmoid.

Softmax vs. Sigmoid:

$$K = 2: \quad \frac{e^{w_1^\top x}}{e^{w_1^\top x} + e^{w_2^\top x}}$$

$$= \frac{1}{1 + e^{-(w_1 - w_2)^\top x}} = \sigma(\Delta w^\top x)$$

One-vs-Rest:

Fit K independent binary models. Faster to train but predictions may not sum to 1.

In practice: Scikit-learn uses OvR by default; statsmodels uses multinomial logit.

Softmax is the canonical link for the multinomial distribution – the multi-class generalization of the logit

How Certain Are We About Each Coefficient?

Standard Errors from the Hessian

The Fisher information matrix equals the negative expected Hessian:

$$\mathcal{I}(\mathbf{w}) = \mathbf{X}^\top \mathbf{S}\,\mathbf{X} \quad \text{where } \mathbf{S} = \mathrm{diag}(p_i(1-p_i)).$$

The variance-covariance matrix of $\hat{\mathbf{w}}$: $\mathrm{Var}(\hat{\mathbf{w}}) = \mathcal{I}(\hat{\mathbf{w}})^{-1} = (\mathbf{X}^\top \hat{\mathbf{S}}\,\mathbf{X})^{-1}$

Standard error of coefficient j: $\mathrm{SE}(\hat{w}_j) = \sqrt{[\mathcal{I}^{-1}]_{jj}}$

- More data $\Rightarrow$ larger $\mathcal{I}$ $\Rightarrow$ smaller SE
- Correlated features $\Rightarrow$ inflated SE (multicollinearity)

Insight

The Hessian serves double duty: it accelerates optimization (Newton) and quantifies uncertainty (standard errors). Both come from the same matrix.

Worked example:

Credit scoring model with 3 features:

Feature	$\hat{w}$	SE
Income	0.50	0.12
Debt ratio	-1.20	0.25
Employment	0.30	0.15

Income: $0.50/0.12 = 4.17 > 1.96$ ✓
Debt: $1.20/0.25 = 4.80 > 1.96$ ✓
Employment: $0.30/0.15 = 2.00 > 1.96$ ✓

All three are significant at $\alpha = 0.05$.

Asymptotic normality: $\hat{w}_j \dot\sim N(w_j,\ [\mathcal{I}^{-1}]_{jj})$ **for large** n

Is This Feature Significant? (Wald Test)

The Wald Test

Null hypothesis: $H_0 : w_j = 0$ (feature j has no effect)

Test statistic: $z = \dfrac{\hat{w}_j}{\text{SE}(\hat{w}_j)} \dot{\sim} N(0, 1)$

Decision rule: Reject H_0 if $|z| > z_{\alpha/2}$ (e.g., $|z| > 1.96$ at 5%).

Confidence interval: $\hat{w}_j \pm z_{\alpha/2} \cdot \text{SE}(\hat{w}_j)$

- For odds ratios: exponentiate the CI endpoints
- Wald test is fast (no model refitting) but unreliable for large $|\hat{w}_j|$
- Hauck–Donner effect: Wald becomes conservative when coefficients are large

Worked example:

Feature: number of late payments
$\hat{w} = 0.85, \quad \text{SE} = 0.18$
$z = 0.85/0.18 = 4.72$
p-value < 0.001

95% CI for w**:**
$0.85 \pm 1.96 \times 0.18 = [0.50, \ 1.20]$

95% CI for OR:
$[e^{0.50}, \ e^{1.20}] = [1.65, \ 3.32]$

Each additional late payment multiplies default odds by 1.65–3.32.

Insight

The Wald test answers: "Is this coefficient significantly different from zero?" For comparing nested models, use the more powerful LRT instead.

The Wald test is the default in statsmodels and R's summary.glm – but LRT is more reliable for small samples

Does Adding These Features Actually Improve the Model?

Likelihood Ratio Test (LRT)

Setup: Compare a reduced model (fewer features) vs. a full model.

$\Lambda = -2\big[\ell(\text{reduced}) - \ell(\text{full})\big] \;\dot{\sim}\; \chi^2_q$
where q = number of additional parameters in the full model.

Example: Adding 3 credit bureau features:
$\Lambda = -2[-1250 - (-1220)] = 60, \quad q = 3$
χ^2_3 critical value at 5% $= 7.81$. Since $60 \gg 7.81$, the bureau features significantly improve the model.

- LRT is more powerful than Wald for multiple coefficients
- Requires fitting both models (more computation)

Insight

LRT asks: "Does the data support the added complexity?" It is the gold standard for nested model comparison in regulatory settings.

AIC and BIC:

For non-nested model comparison:

AIC $= -2\ell + 2k$
BIC $= -2\ell + k \ln n$

where k = number of parameters, n = sample size.

BIC penalizes more for large n (when $\ln n > 2$, i.e., $n > 8$).

Rule: BIC preferred for regulatory models (stronger parsimony).

Lower is better for both. $\Delta > 10$ is strong evidence.

Deviance = -2ℓ; the LRT statistic is the difference in deviances between nested models

How Do You Choose Between Two Non-Nested Models?

Goodness-of-Fit Measures

McFadden pseudo-R^2: $R^2_{\text{McF}} = 1 - \dfrac{\ell(\text{model})}{\ell(\text{null})}$

where $\ell(\text{null})$ = log-likelihood of intercept-only model.

- Values 0.2–0.4 represent excellent fit (not comparable to OLS R^2)
- Adjusted version: $R^2_{\text{adj}} = 1 - \frac{\ell(\text{model}) - k}{\ell(\text{null})}$

Hosmer–Lemeshow test: Group predicted probabilities into deciles, compare observed vs. expected counts with χ^2 test.

Insight

McFadden R^2 measures improvement over the null model. AIC/BIC are better for model selection; pseudo-R^2 is better for reporting overall explanatory power.

Measure	Range	Use
McFadden R^2	$[0, 1)$	Overall fit
AIC	$\mathbb{R}^+$	Model select
BIC	$\mathbb{R}^+$	Regulatory
Deviance	$\mathbb{R}^+$	LRT input
H–L test	p-value	Calibration
Brier score	$[0, 1]$	Probability

Rule of thumb:
McFadden $R^2 \approx 0.2$ is already a well-fitting model. Do not compare to OLS R^2 values.

No single metric suffices – report deviance, AIC, pseudo-R^2, and a calibration test together

How Do You Measure Discrimination Across All Thresholds?

The ROC Curve

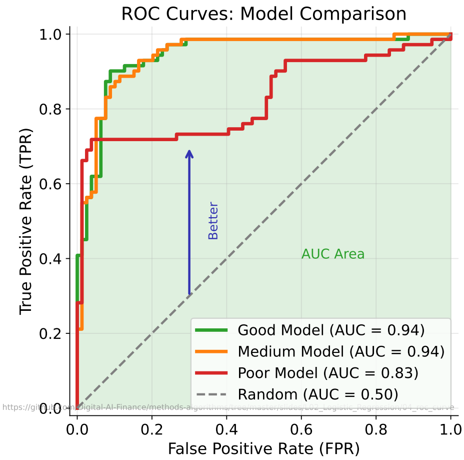
- **What you see:** True Positive Rate vs. False Positive Rate at every threshold
- **Key pattern:** The curve bows toward the top-left corner for good models
- **Takeaway:** AUC summarizes discrimination power in a single number

Interpretation:

- Diagonal = random classifier (AUC = 0.5)
- Perfect classifier hugs the top-left corner (AUC = 1.0)
- AUC = probability that a random positive ranks higher than a random negative

Insight

The ROC curve shows the trade-off between catching defaults (TPR) and raising false alarms (FPR) at every possible threshold.



ROC is threshold-invariant: it evaluates the model's ranking ability, not any specific cutoff

What Does AUC Really Mean, and What Is the Gini Coefficient?

AUC and Gini

AUC (Area Under the ROC Curve):

- AUC $= P(\text{score}(+) > \text{score}(-))$
- Equivalent to the Wilcoxon–Mann–Whitney statistic

Gini coefficient: Gini $= 2 \times$ AUC $- 1$

KS statistic: Maximum vertical distance between CDFs of positive and negative classes.

Insight

Gini is the industry standard in credit scoring. Gini > 0.40 is acceptable; Gini > 0.60 is good. These thresholds appear in Basel validation guidelines.

AUC	**Gini**	**Quality**
0.50	0.00	Random
0.60	0.20	Poor
0.70	0.40	Acceptable
0.80	0.60	Good
0.90	0.80	Excellent
1.00	1.00	Perfect

KS statistic:
KS $= \max_t |F_+(t) - F_-(t)|$
KS > 0.30 is typical for good credit models. Reports the single threshold with maximum separation.

Gini $= 2 \times$ AUC $- 1$: the standard discrimination metric reported to regulators under Basel II/III

When Does ROC Lie, and Why Should You Use Precision-Recall?

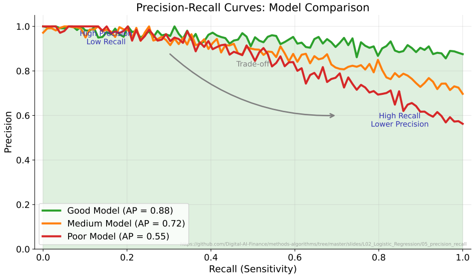
Precision-Recall for Imbalanced Data

- **What you see:** Precision vs. Recall curve – how they trade off
- **Key pattern:** High precision requires sacrificing recall (and vice versa)
- **Takeaway:** PR curves are more informative than ROC when positives are rare

Fraud detection example: With 0.1% fraud rate, a model predicting "no fraud" always achieves 99.9% accuracy and AUC can still look impressive. But precision-recall reveals the model catches zero frauds.

Insight

ROC is optimistic on imbalanced data because TNR stays high. Precision-recall focuses on the rare class. Always report both in fraud and default prediction.



Average Precision (AP) summarizes the PR curve; F1 $= 2 \cdot$ prec $\cdot$ rec$/($prec $+$ rec$)$ at a single threshold

How Do You Know If Your Probabilities Are Truthful?

Calibration and the Confusion Matrix

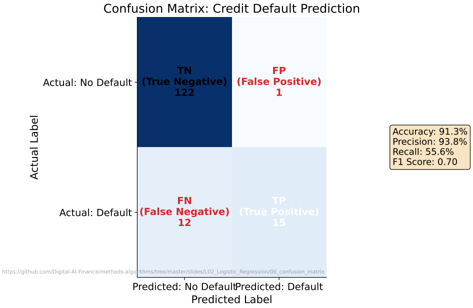
- **What you see:** Counts of TP, FP, TN, FN organized in a 2×2 matrix
- **Key pattern:** Off-diagonal entries are errors; their costs differ in finance
- **Takeaway:** A well-calibrated model means predicted $P = 0.05$ matches observed 5% default rate

Brier score: $\mathrm{BS} = \frac{1}{n}\sum(p_i - y_i)^2$. Lower is better; $\mathrm{BS} = 0$ is perfect.

Hosmer–Lemeshow: Group predictions into deciles, test observed vs. expected with χ^2.

Insight

Discrimination (AUC) and calibration are independent properties. A model can rank well but assign wrong probabilities. Credit scoring requires both.



Calibration is critical for Basel PD: regulators require predicted PDs to match realized default rates

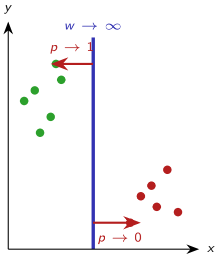
What Could Go Wrong When Data Perfectly Separates the Classes?

Complete and Quasi-Complete Separation

- If one feature perfectly predicts the outcome, the MLE does not exist
- Coefficients diverge to $\pm\infty$ as the optimizer tries to make $p \to 0$ or $p \to 1$
- Standard errors become infinite; Wald tests become meaningless

Solutions:

- **Firth's penalized likelihood:** adds a bias correction term
- **L2 regularization:** shrinks coefficients, prevents divergence
- **Bayesian logistic regression:** prior on coefficients keeps them finite



MLE does not exist!

Insight

Separation is common in small samples or rare-event data (e.g., low-default portfolios). Always check for convergence warnings in your output.

Firth (1993): penalized likelihood reduces bias and handles separation – implemented in R's `brglm2`

Why Would You Penalize Coefficients That Fit the Data Well?

The Overfitting Problem

- With many features and limited data, MLE overfits: $\hat{w}_j$ become large
- The model memorizes training noise instead of learning patterns
- Regularization adds a penalty: $J_{\text{reg}} = J + \lambda R(\mathbf{w})$

Bias-variance trade-off:

- $\lambda = 0$: no penalty, high variance (overfitting)
- $\lambda \to \infty$: all coefficients shrunk to zero, high bias (underfitting)
- Optimal λ: minimizes generalization error

Insight

Regularization is not a hack – it is the Bayesian perspective: the penalty corresponds to a prior belief that coefficients should be small.

Regularized loss:

$$J_{\text{reg}} = \underbrace{-\frac{1}{n}\sum [y_i \ln p_i + (1 - y_i)\ln(1 - p_i)]}_{\text{cross-entropy}} + \underbrace{\lambda\, R(\mathbf{w})}_{\text{penalty}}$$

The penalty $R(\mathbf{w})$ controls model complexity. Three choices: L1, L2, or Elastic Net.

Note: Scikit-learn uses $C = 1/\lambda$, so larger C means *less* regularization.

Regularization strength λ (or $C = 1/\lambda$) is the most important hyperparameter in logistic regression

Should You Shrink All Coefficients or Zero Some Out?

Three Regularization Strategies

L2 (Ridge): $R(\mathbf{w}) = \frac{1}{2}\sum_j w_j^2$
Shrinks all coefficients toward zero but never exactly to zero.

L1 (Lasso): $R(\mathbf{w}) = \sum_j |w_j|$
Drives some coefficients exactly to zero $\Rightarrow$ automatic feature selection.

Elastic Net: $R(\mathbf{w}) = \alpha \sum_j |w_j| + \frac{1-\alpha}{2}\sum_j w_j^2$
Combines L1 sparsity with L2 stability. Parameter α controls the mix.

Insight

L1 is preferred when you suspect many irrelevant features. L2 is preferred when all features may contribute. Elastic Net is the safe default for credit scoring.

	L2	**L1**	**EN**
Sparsity	No	Yes	Yes
Stability	High	Low	High
Groups	All	One	All
Solution	Unique	Not	Unique
Bayesian	Gaussian	Laplace	Mix

Geometric view:
L2 penalty = circle constraint.
L1 penalty = diamond constraint.
Diamond corners touch axes $\Rightarrow$ sparse solutions.

Elastic Net with $\alpha = 0.5$ is a robust default; L1 alone can be unstable with correlated features

How Do You Choose the Regularization Strength?

Cross-Validation for λ (or C)

- Split data into K folds (typically $K = 5$ or 10)
- For each candidate C value, train on $K - 1$ folds, evaluate on the held-out fold
- Average the metric (log-loss or AUC) across folds
- Select C that maximizes average performance

In scikit-learn: `LogisticRegressionCV(Cs=20, cv=5, scoring='roc_auc')`
Automatically searches 20 values of C on a log scale.

One-SE rule: Choose the simplest model within one standard error of the best CV score. This promotes parsimony.

Insight

Never tune λ on the test set. Cross-validation provides an unbiased estimate of generalization performance for each candidate.

CV procedure:

1. Define grid: $C \in \{10^{-4}, \ldots, 10^{4}\}$
2. For each C:
   For each fold $k = 1, \ldots, K$:
      Train on folds $\neq k$
      Score on fold k
   Average scores $\rightarrow \bar{S}(C)$
3. Pick $C^* = \arg\max_C \bar{S}(C)$
4. Refit on all data with C^*

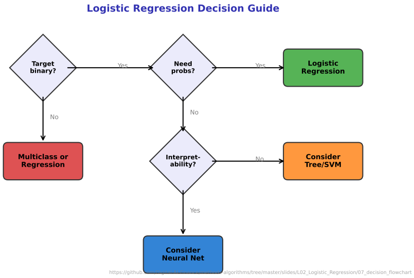
Stratified CV ensures each fold has the same class ratio – critical for imbalanced data.

LogisticRegressionCV is 10x faster than GridSearchCV because it warm-starts along the regularization path

Scorecard Construction

- **What you see:** The decision flowchart from features to approval/denial
- **Key pattern:** Each step adds interpretability and auditability
- **Takeaway:** Scorecards convert log-odds to points using a linear transform

Points-to-Double-the-Odds (PDO):
Score $=$ Offset $+$ Factor $\times \ln(\text{odds})$
where Factor $=$ PDO$/\ln(2)$.
If PDO=20 and base odds 50:1 at score 600, each 20-point drop doubles the default odds.



Insight

Basel II/III requires banks to estimate PD for each rating grade. Logistic regression is the workhorse because every coefficient is auditable and every PD is calibrated.

IRB approach: PD must be validated annually with backtesting, benchmarking, and discrimination analysis

What Features Do Credit Scorecards Actually Use?

Feature Engineering for Credit

Weight of Evidence (WoE) encoding:

$$\text{WoE}_j = \ln\left(\frac{\%\text{non-default}_j}{\%\text{default}_j}\right)$$

Transforms categorical features into a monotonic scale aligned with log-odds.

Common credit features:

- Debt-to-income ratio (DTI)
- Employment stability (years)
- Bureau delinquency count
- Credit utilization ratio
- Time since last delinquency

Insight

WoE encoding is the standard in credit scoring because it preserves the linear relationship with log-odds and handles missing values naturally.

Worked example:

Applicant profile:

Feature	Value
Income	\$55k
DTI	0.32
Employment	6 years
Late payments	1
Utilization	0.45

$$z = -2.1 + 0.5(0.55) + (-1.2)(0.32) + 0.3(0.6) + 0.85(1) + (-0.4)(0.45)$$
$$= -2.1 + 0.275 - 0.384 + 0.18 + 0.85 - 0.18 = -1.359$$

$\text{PD} = \sigma(-1.359) = 0.204$

Decision: PD = 20.4%, above the 15% cutoff → Deny

PD estimation is the foundation: it feeds into EAD and LGD to compute Expected Loss = PD × EAD × LGD

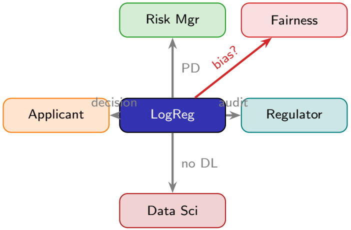
Who Wins and Who Loses When Models Replace Loan Officers?

Stakeholder Analysis

- **Winners:** Risk managers (calibrated PD), regulators (auditable model), applicants (consistent decisions)
- **Losers:** Data scientists wanting complex models, branch managers (discretion reduced)

Fairness considerations:

- Protected attributes (race, gender, age) must not be direct features
- **Disparate impact:** Even without protected features, proxy variables can create bias
- GDPR Article 22: right to explanation of automated decisions



Insight

In regulated finance, interpretability is not optional – it is a legal requirement under Basel II/III and GDPR Article 22. LogReg is the compliance-friendly default.

Fairness audits (disparate impact ratio, equalized odds) are becoming mandatory in EU AI Act regulated use cases

Key Takeaways

Mathematical Foundation

- Sigmoid maps $\mathbb{R} \to (0, 1)$; logit maps $(0, 1) \to \mathbb{R}$
- MLE: maximize $\ell(\mathbf{w}) = \sum[y \ln p + (1 - y) \ln(1 - p)]$
- Gradient: $\nabla J = \frac{1}{n}\mathbf{X}^\top(\mathbf{p} - \mathbf{y})$
- Cross-entropy is convex $\Rightarrow$ global optimum guaranteed

Evaluation and Inference

- Wald test for individual coefficients; LRT for nested models
- ROC/AUC for discrimination; Gini $= 2\text{AUC} - 1$
- Calibration (Brier, Hosmer–Lemeshow) for probability quality

Practical Application

- Regularization: L1 (sparsity), L2 (stability), Elastic Net (both)
- Choose λ via cross-validation; use one-SE rule for parsimony
- Credit scoring: PD estimation, scorecard points, Basel compliance
- Interpretability = regulatory requirement, not just a nice-to-have

One-sentence summary:
Logistic regression turns linear combinations into calibrated probabilities via the sigmoid, estimated by MLE, validated by inference, and deployed as the gold standard in regulated credit scoring.

Logistic regression: simple enough to explain to a regulator, powerful enough to run a bank's credit decisions

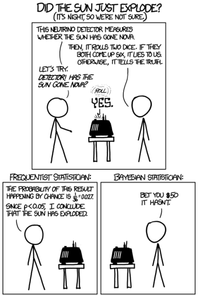
Until Next Time...

Looking Back

We opened with XKCD #1132: the frequentist rejected the null that the sun exploded, while the Bayesian trusted the prior. Logistic regression bridges this divide – it gives you a principled probability.

What you can now do:

- Derive the MLE gradient from first principles
- Evaluate a classifier with ROC, Gini, and calibration
- Build a credit scorecard with interpretable coefficients
- Diagnose separation, imbalance, and overfitting



Next: L03 – KNN & K-Means
From parametric to non-parametric methods.

XKCD #1132 by Randall Munroe (CC BY-NC 2.5) — Next: L03 KNN & K-Means