Lesson 25: Logistic Regression

Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

The Story of Classification

"Can a machine tell species apart?"

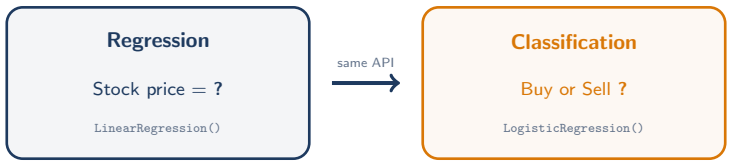
In 1936, Ronald Fisher looked at iris flowers and posed the question that launched an entire field. Two decades later, the tools we still use today were born.

Logistic regression (1958) remains one of the most widely used classification methods today.

From Regression to Classification

The Bridge from Module 4

Regression predicts **numbers**. Classification predicts **categories**. But the sklearn API stays the same: `fit()`, `predict()`.



Logistic regression is a classifier despite having "regression" in the name.

Learning Objectives

The Problem: Linear regression predicts continuous values, but what if we need a yes/no answer? How do we turn a number into a decision?

After this lesson, you will be able to:

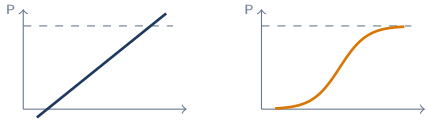
- Understand the sigmoid function and its probability output
- Build binary classifiers with sklearn
- Interpret coefficients as odds ratios
- Predict market direction (up/down) with logistic regression

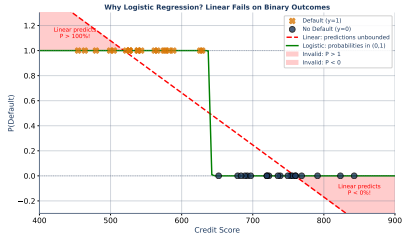
---

Finance Application: Predicting whether tomorrow's return is positive or negative

The Problem: Linear Regression Fails

Discovery: What happens if we use linear regression for binary outcomes? Predictions go below 0 or above 1 – **impossible probabilities!**



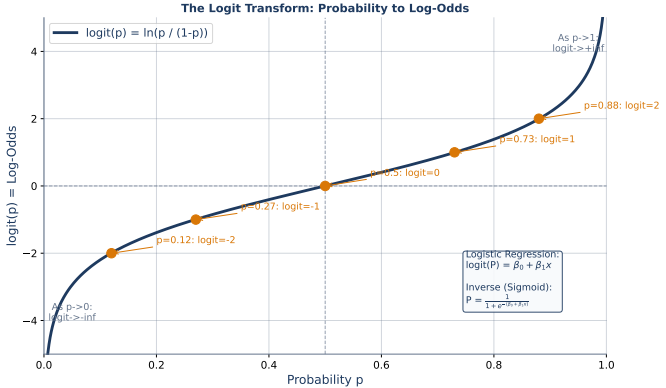


Red regions show impossible predictions. Logistic regression stays neatly in (0, 1).

The Logit Transform

The logit function is the **inverse** of the sigmoid. It maps probabilities from $(0, 1)$ back to the full real line $(-\infty, +\infty)$.

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$



Logit and sigmoid are inverse functions: $\text{logit}(\sigma(z)) = z$.

The Bridge: What IF We Could Squeeze?

We need a function that takes ANY real number and squashes it into (0, 1).

What properties should this magical function have?

- **Smooth:** No sudden jumps – small input changes give small output changes
- **Monotonic:** Always increasing (higher input $\rightarrow$ higher probability)
- **Bounded:** Output stays strictly between 0 and 1
- **Centered:** At input 0, output is exactly 0.5

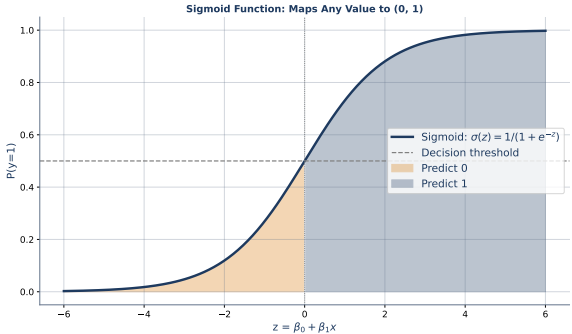
Think for a moment: can you imagine what such a curve looks like?

The answer is one of the most elegant functions in mathematics – coming next.

The Sigmoid: Nature's Squasher

Sigmoid Formula: $\sigma(z) = \frac{1}{1+e^{-z}}$, where $z = \beta_0 + \beta_1 x$

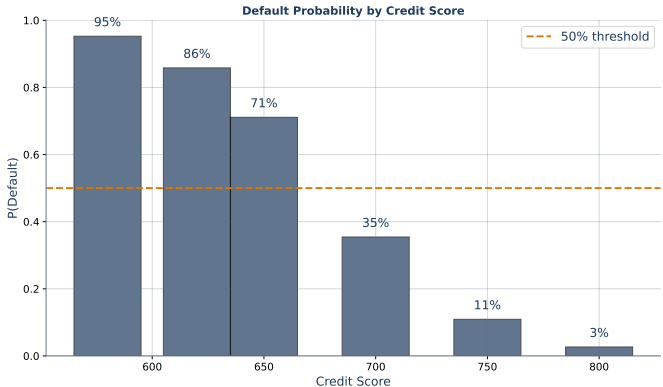
- Input z can be any real number; output is always between 0 and 1
- Logistic regression = linear regression passed through sigmoid



Key insight: Sigmoid squashes $(-\infty, +\infty)$ into valid probabilities $(0, 1)$.

Probability Output

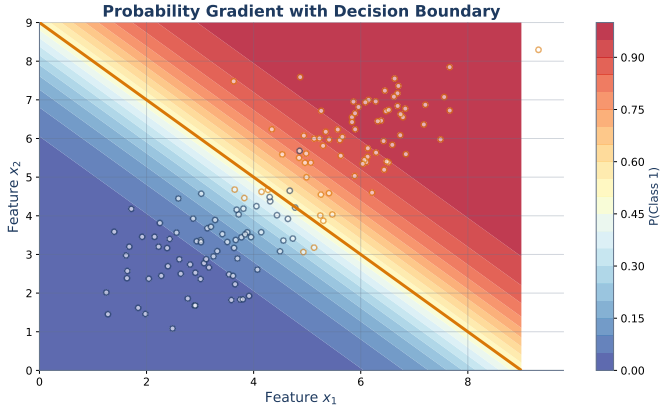
Each point on the sigmoid curve is a **probability**. Higher feature values shift the probability toward 1.



The relationship between features and probability is non-linear – steepest near 0.5.

Decision Boundary: Probability Gradient

The probability transitions smoothly from 0 to 1 across the decision boundary. The gradient is steepest at the boundary itself.



Probability transitions smoothly from 0 to 1 across the boundary.

Mathematical Notation

Key Symbols for Logistic Regression

- σ (sigma) – the sigmoid function
- e – Euler's number (≈ 2.718), the base of natural logarithms
- e^{-z} – "e raised to the power of negative z"
- $P(y=1 \mid x)$ – probability that y equals 1, given features x

The Full Model:

$$P(y = 1 \mid x) = \sigma(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)}}$$

Review L14 for probability basics. The | symbol reads as "given."

Mathematical Notation for Logistic Regression

Key Symbols You'll See

- σ (sigma) – the sigmoid function, a common math symbol
- e – Euler's number (≈ 2.718), the base of natural logarithms
- e^{-z} – "e raised to the power of negative z"
- $P(y = 1|x)$ – "probability that y equals 1, given x" (conditional probability)

Why These Matter

- e^{-z} creates a smooth curve (not a step function)
- Conditional probability lets us estimate "how likely is outcome 1 for this input?"
- The linear combination $z = \beta_0 + \beta_1 x_1 + \ldots$ is identical to linear regression
- The sigmoid wraps this linear output into a valid probability

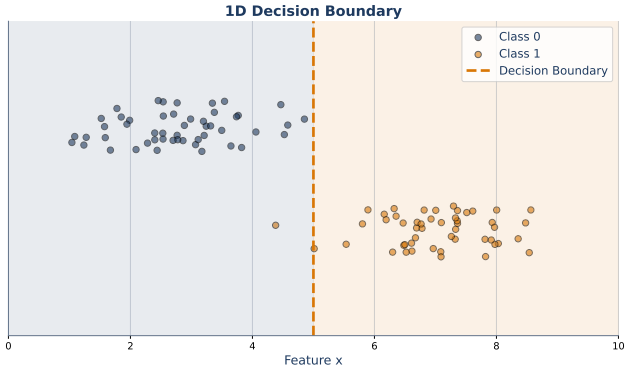
---

Review L14 for probability basics. The — symbol reads as "given."

Decision Boundary: Drawing the Line

Model outputs probability. Where do we cut?

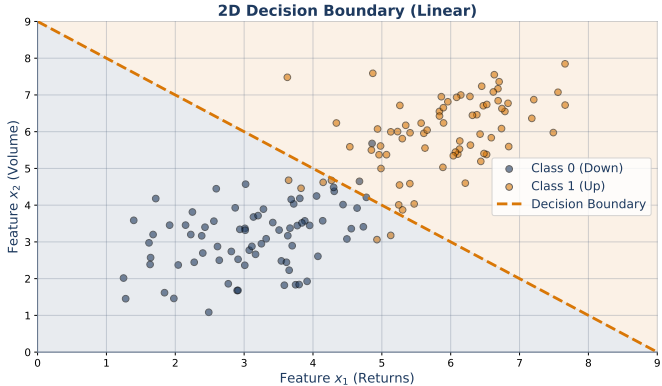
- Default threshold: 0.5. If $P > 0.5 \rightarrow$ class 1, else class 0
- This is **arbitrary** – adjustable based on costs (more in L27)



The threshold is a design choice. We'll revisit threshold tuning in L27.

Decision Boundary: 2D Feature Space

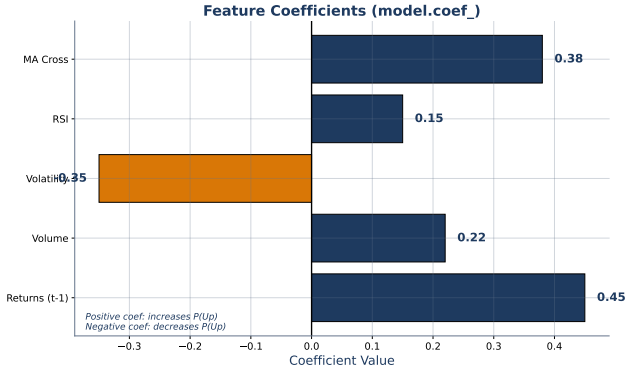
With two features, the decision boundary becomes a **line** in feature space. In 3D it is a plane. In higher dimensions, a hyperplane.



Logistic regression always produces linear boundaries. Non-linear data needs other models.

sklearn: Three Lines of Code

Same API as LinearRegression (L21): `LogisticRegression().fit(X,y).predict(X_test)`

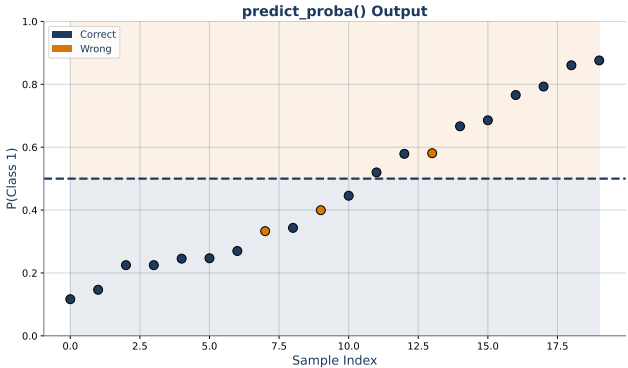


Coefficients show feature importance and direction of effect.

predict_proba: The Confidence Score

Unlike `predict()`, `predict_proba()` gives probabilities:

- Returns $[P(\text{class } 0),\ P(\text{class } 1)]$ – columns sum to 1.0
- Use when you need confidence, not just a label

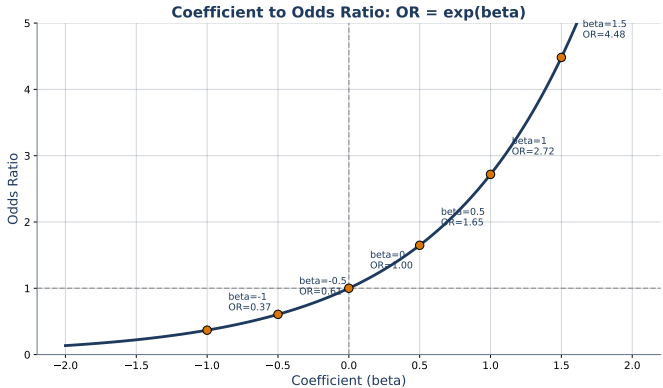


`predict_proba()` is essential for risk-sensitive applications like credit scoring.

Coefficient to Odds Ratio Conversion

Each model coefficient β can be converted to an odds ratio via exponentiation:

$$\text{Odds Ratio} = e^{\beta}$$



Odds Ratio = exp(coefficient) – exponential transformation.

Understanding Odds and Log-Odds

From Probability to Odds to Log-Odds:

- **Probability:** $p = 0.8$ (80% chance)
- **Odds:** $p/(1-p) = 0.8/0.2 = 4$ ("4 to 1")
- **Log-odds:** $\ln(4) \approx 1.39$ (unbounded scale)

Why? Probability is bounded $(0, 1)$. Log-odds is unbounded $(-\infty, +\infty)$ – perfect for linear models!

Odds Ratio: OR $= e^{\beta}$. OR $= 1.5$ means each unit increase multiplies odds by 1.5.

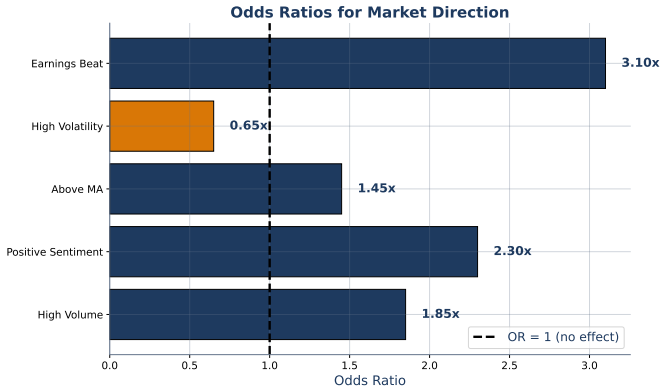
OR > 1: increases odds OR < 1: decreases odds OR $= 1$: no effect

---

Example: OR $= 2$ for income means each \$10K increase doubles the odds of loan approval.

Interpreting Coefficients: Odds Ratios

Odds ratios let us say **"how much does each feature matter?"**



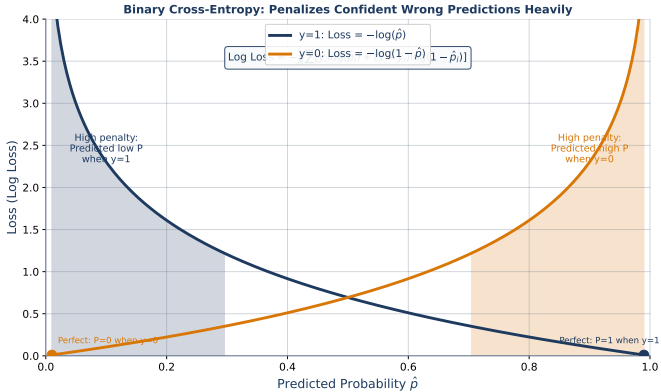
Odds ratio ¿ 1 means the feature increases odds of the positive class.

Log Loss: How the Model Learns

Logistic regression minimizes **log loss** (binary cross-entropy):

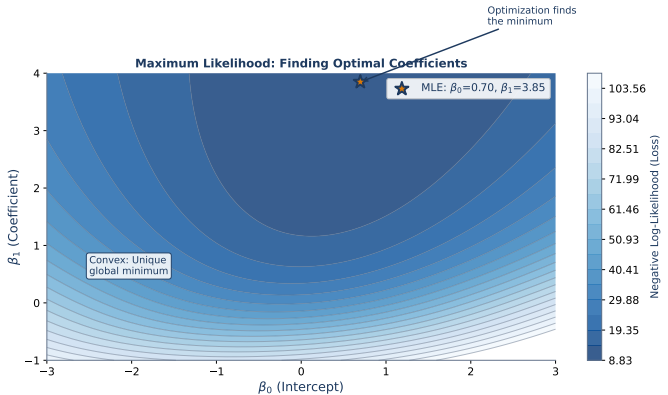
$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \ln(\hat{p}_i) + (1-y_i)\ln(1-\hat{p}_i)]$$

Penalizes confident wrong predictions heavily.



Maximum Likelihood Estimation Surface

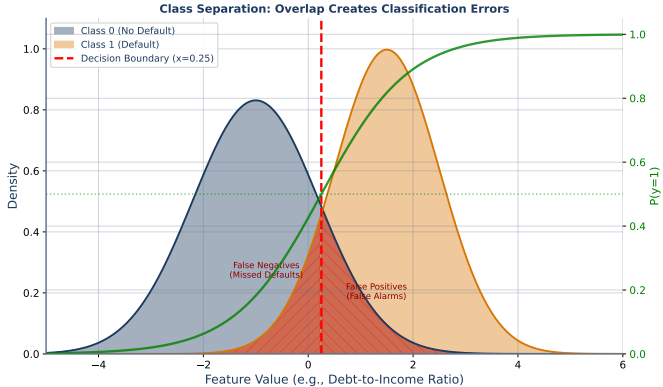
The model finds coefficients that maximize the likelihood of the observed data. This is equivalent to minimizing log loss.



MLE finds the peak of the likelihood surface – the best-fit parameters.

Class Separation

How well a logistic model works depends on how separable the classes are in feature space.



Well-separated classes yield higher accuracy and more confident predictions.

Checkpoint: Test Your Understanding

Q1: What does the sigmoid function output?

Q2: What does an odds ratio of 2 mean?

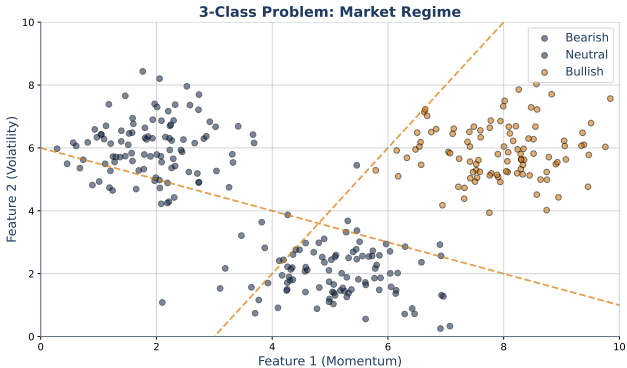
Q3: When would you lower the threshold below 0.5?

A1: Probability between 0 and 1. A2: Each unit increase doubles the odds. A3: When missing positives is very costly (e.g., fraud detection).

Multiclass: Beyond Binary

"Buy / Hold / Sell" requires 3 classes. Two strategies:

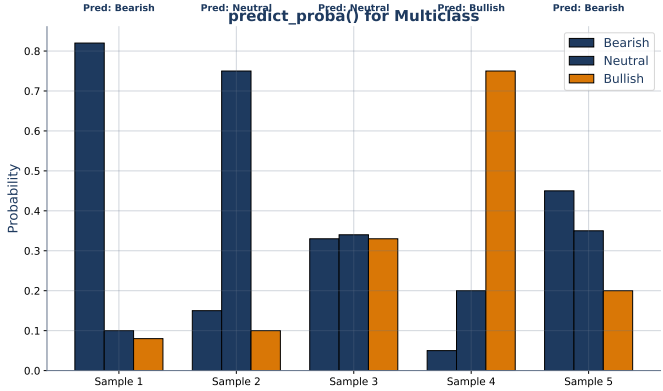
1. **One-vs-Rest (OvR):** Train K binary classifiers, pick highest P
2. **Softmax:** Single model, K outputs summing to 1



sklearn default: OvR. Use `multi_class='multinomial'` for softmax.

Softmax Probabilities

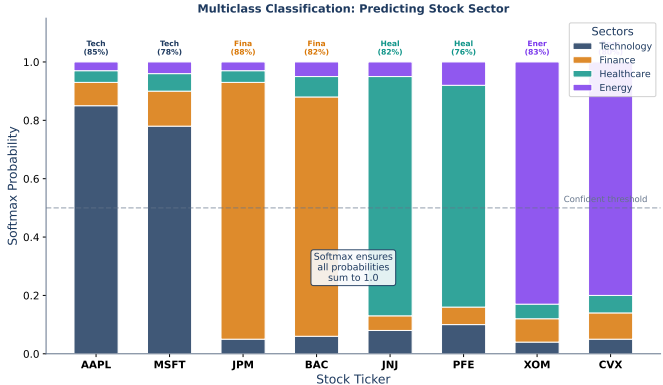
Softmax outputs sum to 1 across all K classes. Each output is the estimated probability for that class.



Softmax generalizes sigmoid to multiple classes.

Sector Classification

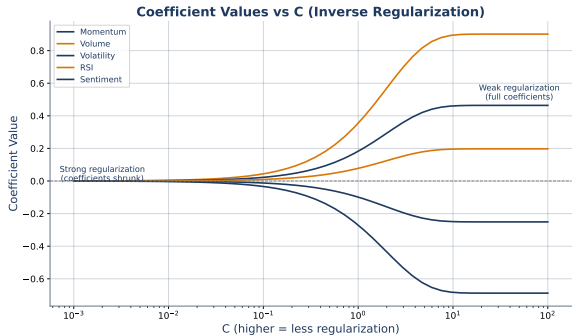
Multiclass logistic regression applied to financial sector prediction.



Real-world multiclass: classifying stocks into industry sectors.

The C Parameter in sklearn (Review L22)

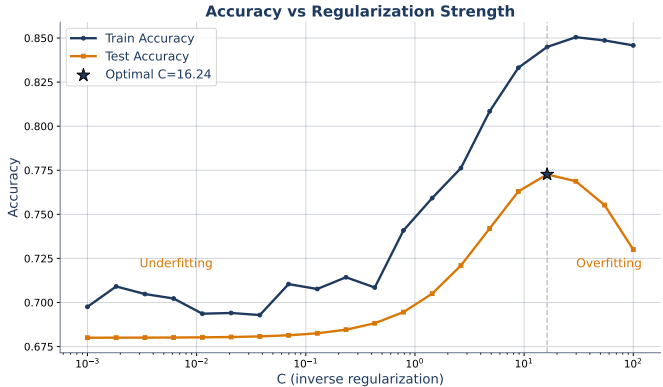
- $C = 1/\lambda$ – **inverse** of regularization strength
- Higher C = less regularization; Lower C = simpler model



C = Complexity allowed. Default $C = 1.0$. Tune via GridSearchCV: $C \in [0.01, 100]$.

Regularization: Train vs Test Accuracy

Finding the right C balances fitting the training data and generalizing to unseen data.

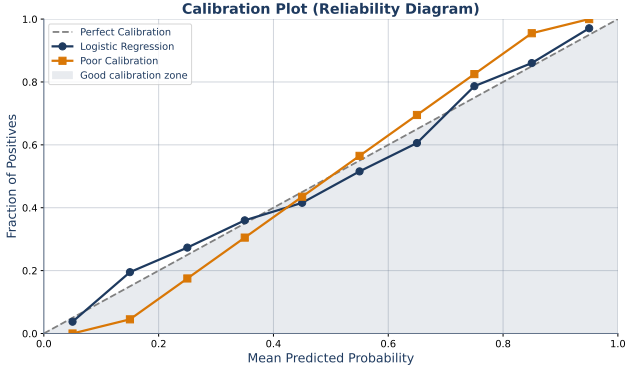


Find C where test accuracy is maximized – balance fit and generalization.

Calibration: Can You Trust the Probabilities?

"Model says 70% – does it really mean 70%?"

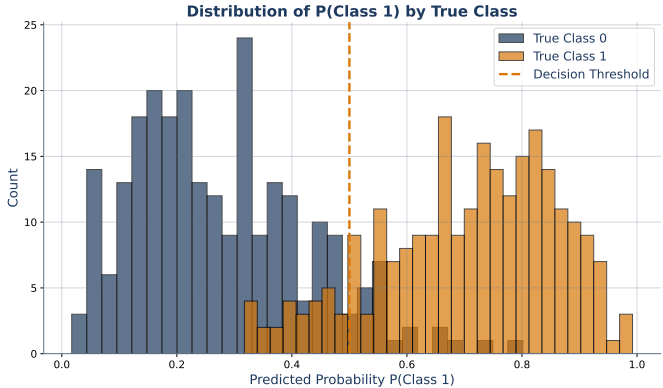
- **Calibrated:** Of all 70% predictions, about 70% are correct
- Above diagonal = underconfident; below = overconfident



Logistic regression is naturally well-calibrated. Trees/forests often need calibration.

Calibration: Probability Distribution

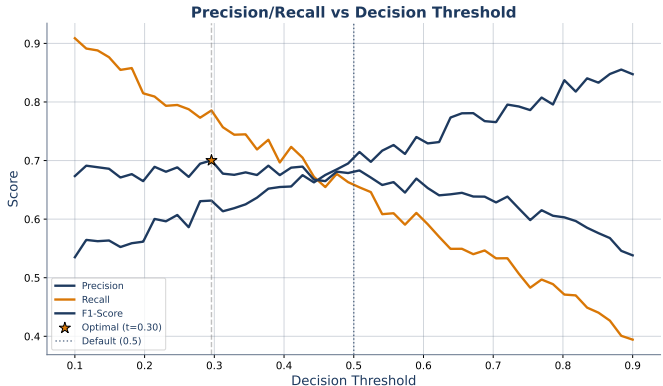
The distribution of predicted probabilities reveals how confident the model is and how well the two classes separate.



Distribution of predicted probabilities for each true class.

Calibration: Threshold Tuning

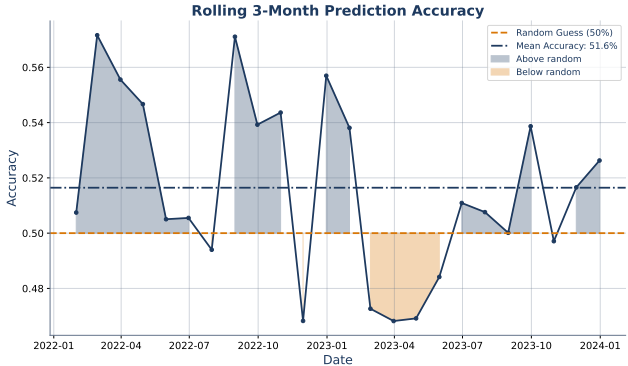
Adjusting the decision threshold trades off precision and recall. The optimal threshold depends on the cost of errors.



Adjust threshold based on costs of false positives vs. false negatives.

Finance: Predicting Market Direction

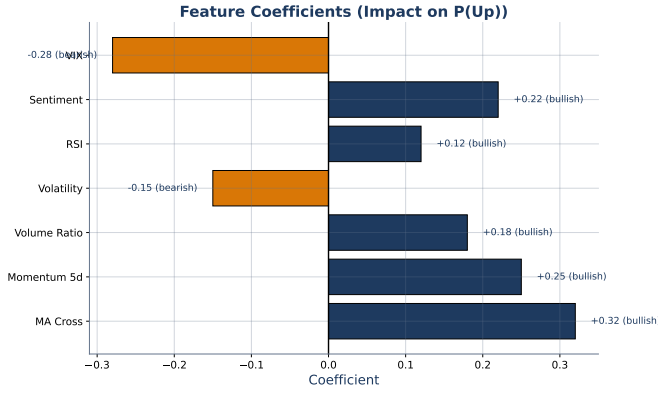
Does the model stay accurate over time? Rolling accuracy reveals whether predictive signals decay.



Monitor prediction accuracy over time – financial signals often decay.

Finance: Feature Importance

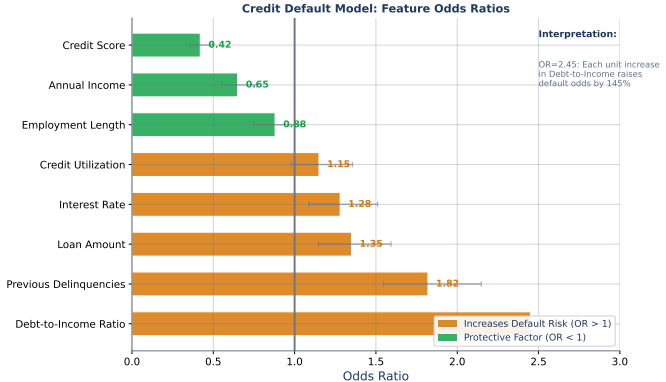
Which features contribute most to predicting market direction? Coefficient magnitude (absolute value) indicates importance.



Which features contribute most to predicting market direction?

Finance: Credit Default Prediction

Logistic regression is widely used in credit scoring. Banks use it to estimate default probability for regulatory capital.



Credit scoring is the classic financial application of logistic regression.

Common Mistakes in Logistic Regression

Using accuracy on imbalanced data
Use F1-score or AUC instead (preview L27).

Forgetting to scale features
Logistic regression is sensitive to feature magnitude. Use StandardScaler.

Ignoring calibration
Uncalibrated probabilities mislead risk models and pricing.

Avoiding these three mistakes will save you hours of debugging.

Hands-On Exercise (25 min)

Task: Predict Stock Direction

Features:

- 5-day momentum: (price today − price 5 days ago) / price 5 days ago
- 20-day volatility: Std. dev. of returns over past 20 days
- Volume ratio: Today's volume / 20-day average volume

Steps:

1. Create binary target: 1 if next-day return > 0, else 0
2. Fit `LogisticRegression()` and examine coefficients
3. Calculate accuracy on test set (80/20 split)
4. Interpret: which features increase odds of positive return?

Deliverable: Coefficient table with odds ratios + test accuracy.

Extension: Try thresholds 0.4 and 0.6 – how does accuracy change? See `inclass/` notebook.

Lesson Summary + Preview

Problem Solved: Predict binary outcomes (up/down, buy/sell) with probability estimates.

Key Takeaways:

- Sigmoid squashes $(-\infty, +\infty)$ into valid probabilities $(0, 1)$
- Same sklearn API: `fit()`, `predict()`, `predict_proba()`
- Coefficients $\rightarrow$ odds ratios via e^{β}
- Regularize with C parameter (C = Complexity allowed)

Next Lesson: Decision Trees (L26)
"Logistic regression draws a LINE. What if the boundary isn't linear? Next: Twenty Questions."

Memory: Logistic = Linear + Sigmoid. Output is probability, not continuous value.