Classification and resampling methods

Beyond linear models and training error

Elodie Chervin Daniel Barbosa

TT 2026

Department of Economics

Roadmap

Lecture roadmap

I. Logistic regression

- Why OLS fails for classification
- Modelling via the sigmoid curve
- Log loss and SGD optimization

II. Naive Bayes

- Generative intuition: authors
- Independence assumptions

III. Evaluation metrics

- Precision, recall and trade-offs
- ROC curves and AUC

IV. Resampling methods

- Overfitting and model complexity
- Cross-validation and bootstrap

The Classification Problem

What is Classification?

In Week 1, we predicted a **quantitative** response (e.g., Salary). Today, we predict a **qualitative** (categorical) response Y.

- **Binary classification**: $Y \in \{0, 1\}$ or {No, Yes}, spam or not spam, default or no default.
- **Multi-class**: $Y \in \{1, 2, \ldots, K\}$ (e.g., credit rating: A, B, C).

The Goal
Instead of predicting Y directly, we model the **probability** that Y belongs to a category:

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

Classification: A supervised learning task where the output variable Y is categorical rather than numerical.

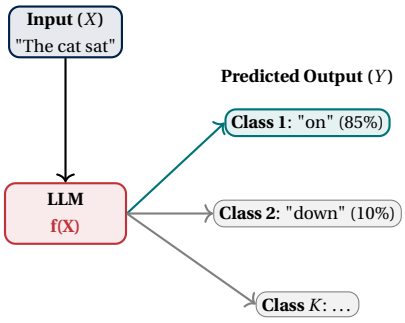
Language Models are just complex classifiers

Large language models (LLMs) predict the most probable next word in a sequence.

It is a classification problem where the "categories" are every word in the dictionary.

We learn the mapping **f(X)** $\rightarrow Y$ by splitting data, much like studying for an exam:

- **Training set** (*the textbook*): the data the model "reads" to learn patterns and vocabulary.
- **Development set** (*the mock exam*): data used to check progress and make improvements before the real test.
- **Test set** (*the final exam*): unseen data used to see how the model performs in the real world.



Classification methods

Our goal is to predict if a customer will default on their loan (Y) based on their credit card balance (X).

1. Discriminative models
Focus entirely on finding the decision boundary. They look at the data and ask: *"At what balance does a customer switch from being likely to pay, to likely to default?"*

2. Generative models
Focus on the underlying groups. They look at the data and ask: *"What is the average balance of a defaulter? What is the average balance of a non-defaulter?"*

When a new customer arrives with a balance of $1,600, it asks: *"Is $1,600 more typical of the defaulter distribution or the non-defaulter distribution?"*

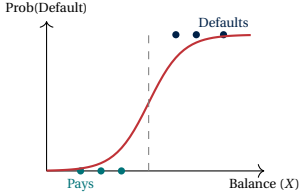
Machine learning tools

Discriminative models

Directly models the decision boundary.

Method: Finds a formula (like an S-curve) that separates defaulters from non-defaulters.

- **Logistic regression**
- **Decision trees**

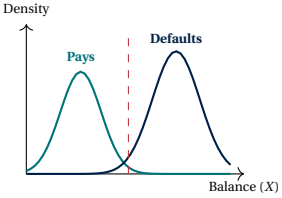


Generative models

Models the distribution of each class.

Note: Once we learn the distributions, we can randomly sample from them.

- **Naive Bayes**
- **Linear discriminant analysis**



Can We Use Linear Regression for Binary Outcomes?

Linear Probability Model (OLS)

Economists often estimate:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \varepsilon_i$$

where $Y_i \in \{0, 1\}$.

Pros:

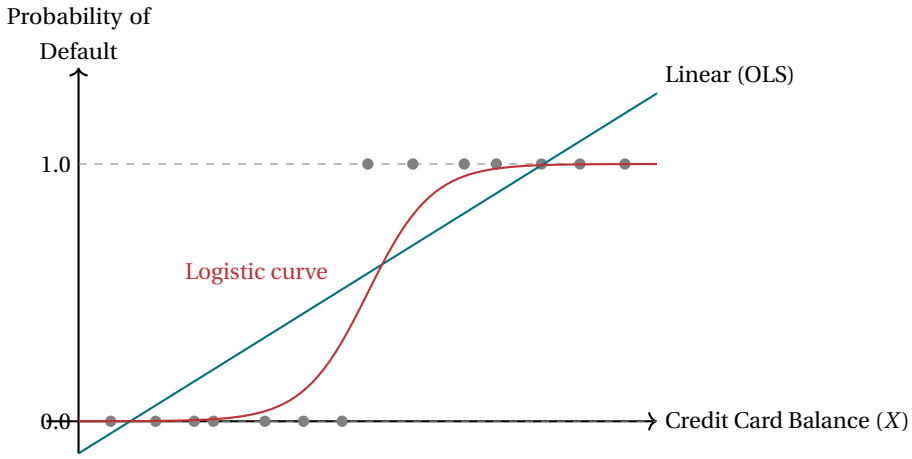
- Extremely easy to interpret.
- β_1 is the *marginal effect* on probability.

Structural Limitations

- **Unbounded**: Can predict probability estimates < 0 or > 1.
- **Heteroskedasticity**: Since Y is binary, the error variance is $p(X)[1 - p(X)]$. Because $p(X)$ changes with X, the variance changes with X. This violates OLS assumptions, rendering the estimator inefficient and standard errors invalid.

Linear probability model (LPM): An OLS regression where the dependent variable is binary, interpreted as modelling the probability of an event.

A solution: Logistic Regression



Another problem with Linear Probability Models: Multiple Categories

Suppose instead of just predicting *if* someone defaults, we must classify the *reason* for the default:

$$Y = \begin{cases} 1 & \text{if Job Loss} \\ 2 & \text{if Medical Emergency} \\ 3 & \text{if Fraudulent Account} \end{cases}$$

This numerical coding suggests an ordering. In fact, it mathematically implies that the difference between *Job Loss* and *Medical Emergency* is the exact same as the difference between *Medical Emergency* and *Fraud*.

Linear regression is fundamentally inappropriate here because these are distinct, unordered categories. **Multiclass Logistic Regression** or **Generative Models** are required.

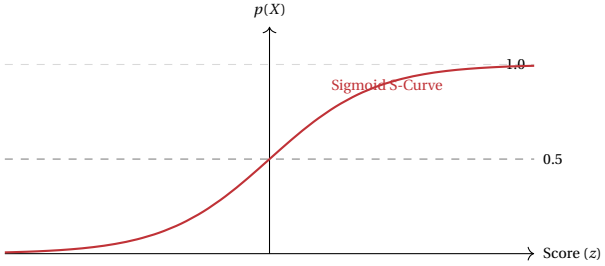
Logistic Regression

Logistic Regression: From Scores to Probabilities

Our goal is to predict the probability $p(X)$ that an event occurs. We do this in two steps:

1. Compute a linear score $z = \beta_0 + \beta_1 X$.
2. Pass the score through the **Sigmoid Function** to "squeeze" it into $[0, 1]$.

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



The model: parameters and the "logit"

- **Steepness (β_1)**: Controls how quickly the probability reacts to changes in X.
- **Centering (β_0)**: Shifts the curve to "anchor" the transition point.

Mathematical structure (The logit): If we rearrange the sigmoid formula, we see the relationship is linear in the log-odds:

$$\ln\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

This allows us to interpret coefficients similar to OLS, but for "log-odds."

Classification and the decision boundary

To turn a probability into a prediction $\hat{Y}$, we choose a threshold (usually 0.5):

$$\hat{Y} = 1 \text{ if } p(X) \geq 0.5$$

Since $\sigma(z) \geq 0.5$ when $z \geq 0$, our rule is:

$$\hat{Y} = 1 \text{ if } \beta_0 + \beta_1 X \geq 0$$

The **decision boundary** is the value of X where we are indifferent:

$$\text{Boundary: } X = -\frac{\beta_0}{\beta_1}$$

Application: The Bank's Decision Rule

Suppose we estimate a model for Default based on Balance (X):

$$z = -10 + 0.005 \times \text{Balance}$$

How does the bank decide? The boundary ($z = 0$) occurs at:

$$-10 + 0.005 \times \text{Balance} = 0 \implies \text{Balance} = \$2,000$$

The Resulting Classifier:

- $\text{Balance} < \$2,000 \implies z < 0 \implies \hat{Y} = 0$ (Accept)
- $\text{Balance} \geq \$2,000 \implies z \geq 0 \implies \hat{Y} = 1$ (Reject)

Naive Bayes

Naive Bayes

Naive Bayes is a generative model that classifies by asking for the probability of class c given observed data X.

Using Bayes' theorem:

$$P(c \mid X) = \frac{P(X \mid c)P(c)}{P(X)}$$

- $P(c \mid X)$: **Posterior**. Probability of class c *given* the observed data X.
- $P(X \mid c)$: **Likelihood**. Probability of data X *if* it were from class c.
- $P(c)$: **Prior**. Base probability of class c before seeing data.
- $P(X)$: **Evidence**. Total probability of the data.

Decision rule: Since $P(X)$ is the same for all classes, we choose the class $\hat{c}$ that maximizes the numerator:

$$\hat{c} = \arg\max_c \underbrace{P(X \mid c)}_{\text{Likelihood}} \times \underbrace{P(c)}_{\text{Prior}}$$

How Naive Bayes works

Imagine filtering **Spam** vs. **Not Spam**:

1. **Look at history:** The model calculates the **prior probability** $P(c)$ of each category based on historical data.
2. **Check the evidence:** It identifies features (e.g., the word "Free") and calculates their probability in each class ($P(X|c)$).
3. **Multiply probabilities:** For a new input, it multiplies the probabilities of all observed features together ($P(X|c)P(c)$).
4. **Make the call:** The category with the highest resulting probability score is the predicted winner ($P(c|X)$).

The "Naive" Assumption in NLP

Consider using a Multinomial Naive Bayes classifier for Sentiment Analysis (predicting if a tweet is positive or negative).

What assumption does the "Naive" in Naive Bayes refer to?

Provide a real-world example of a short phrase or sentence where this "naive" assumption dramatically fails, leading the classifier to make the wrong prediction.

The Naive Bayes Assumptions

1. Conditional Independence (The "Naive" part)
We assume all features X_i are *independent* given the class c.

$$P(X_1, X_2 \mid c) = P(X_1 \mid c) \times P(X_2 \mid c)$$

Why is this "naive"? It ignores word correlations (e.g., "San" and "Francisco").

2. Bag-of-Words Assumption
We assume the *position* and *order* of words do not matter. The sentence is treated as an unordered "bag" of features.

Despite this, Naive Bayes performs well in practice because we care about the *ranking* of classes, not the exact probabilities.

However, it is not always the best model. It is not very good at capturing complex relationships between features, and new words can result in a probability of 0.

Naive Bayes Calculation

Assume the following likelihoods for each word being part of a positive or negative movie review, and equal prior probabilities for each class.

Word	$P(\text{word} \mid \text{pos})$	$P(\text{word} \mid \text{neg})$
I	0.09	0.16
always	0.07	0.06
like	0.29	0.06
foreign	0.04	0.15
films	0.08	0.11

Question: What class will Naive Bayes assign to the sentence:
"I always like foreign films."?

Solution 1: Naive Bayes Calculation

We calculate the score for each class using $P(c) \prod P(\text{word}_i \mid c)$:

1. Positive Class:

$$0.5 \times 0.09 \times 0.07 \times 0.29 \times 0.04 \times 0.08 \approx 2.92 \times 10^{-6}$$

2. Negative Class:

$$0.5 \times 0.16 \times 0.06 \times 0.06 \times 0.15 \times 0.11 \approx 4.75 \times 10^{-6}$$

Decision: Since the Negative score is higher, the sentence is classified as **Negative**.

Worked Example: Why was it Negative?

Let's look at the "Evidence" (Likelihood Ratios $P(w \mid \text{neg}) / P(w \mid \text{pos})$):

Word	Neg Likelihood	Pos Likelihood	Evidence for Neg
I	0.16	0.09	1.77x
foreign	0.15	0.04	3.75x
films	0.11	0.08	1.38x
like	0.06	0.29	0.21x (*Strongly Pos*)
always	0.06	0.07	0.86x (*Slightly Pos*)

Comparing models

Property	Logistic regression	Naive Bayes
Question	*Where is the boundary?*	*What does the class look like?*
Approach	**Discriminative**	**Generative**
Estimation	Iterative optimization (MLE)	Frequency counting
Training speed	Slower (requires optimization)	Very fast
Sample size	Needs more data to stabilize	Robust on small datasets
Correlations	Handles dependencies well	Assumes independence
Probabilities	Often well-calibrated	Often over-confident

Estimation of Logistic Regression Parameters

Estimation of Logistic Regression Parameters: Maximum Likelihood

We cannot use OLS (Minimising Residuals) because the relationship is non-linear. Instead, we use **Maximum Likelihood Estimation (MLE)**.

The Intuition: Find the parameters $(\hat{\beta}_0, \hat{\beta}_1)$ that make the observed data most likely to have occurred.

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

Maximum Likelihood Estimation (MLE): A statistical method for estimating parameters by finding values that maximise the likelihood of the observed data given the model.

From Likelihood to Loss

To find the optimal β, we maximize the Likelihood $\ell(\beta)$. In practice, it is easier to maximize the **Log-Likelihood** (turning products into sums):

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

In Machine Learning, we flip the sign to create a *cost function* to minimize. This is the **Binary Cross-Entropy Loss (or Log Loss)** (J):

$$J(\beta) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)]$$

Intuition: log loss

The loss for a single observation i simplifies based on the true label y_i:

If $y_i = 1$ (Positive)
Loss $= -\log(\hat{p}_i)$
Goal: Push $\hat{p}_i \to 1$.

If $y_i = 0$ (Negative)
Loss $= -\log(1 - \hat{p}_i)$
Goal: Push $\hat{p}_i \to 0$.

Why log?
The log function penalizes mistakes exponentially. If the model is certain of the wrong answer, the loss is massive. This encourages better calibration.

Optimization: stochastic gradient descent

There is no closed-form solution for β. We must numerically minimize the loss using **stochastic gradient descent (SGD)**:

1. **Initialize**: Start with random weights β.
2. **Iterate**: For each observation (or mini-batch):
   - Calculate the gradient (the direction of steepest ascent).
   - **Update**: $\beta \leftarrow \beta - \alpha \cdot \nabla J(\beta)$

Learning rate (α): Controls the step size.

- **Too high**: overshoot or diverge.
- **Too low**: slow convergence.

Multiclass Logistic Regression

Logistic regression generalizes to $K > 2$ classes. One version has the symmetric form:

$$\Pr(Y = k \mid X) = \frac{e^{\beta_{0k} + \beta_{1k} X_1 + \cdots + \beta_{pk} X_p}}{\sum_{K'=1}^{K} e^{\beta_{0K'} + \beta_{1K'} X_1 + \cdots + \beta_{pK'} X_p}}$$

Here there is a linear function for each class. Recognize that some cancellation is possible, and only $K - 1$ linear functions are needed as in 2-class logistic regression.

Multiclass logistic regression is also referred to as **multinomial regression**.

Evaluation Metrics

Preparation Question

The cost of being wrong in medical screening

Consider an AI system deployed in a hospital to screen patients for a rare but aggressive form of cancer.

The model makes two types of mistakes: false positives (an unnecessary biopsy) and false negatives (sending a sick patient home without treatment).

Are these errors equally costly? Why does a single overarching "accuracy" metric fail to capture the real-world utility and ethical implications of this system?

If you were the developer, how would you mathematically "tune" the classifier to prioritize saving lives?

The confusion matrix

		Predicted class	
		Positive ($Y = 1$)	**Negative** ($Y = 0$)
True class	**Positive**	**True positive** (TP)	False negative (FN)
	Negative	False positive (FP)	**True negative** (TN)

- **Sensitivity** (Recall): $TP/(TP + FN)$. Proportion of actual positives caught.
- **Specificity**: $TN/(TN + FP)$. Proportion of actual negatives caught.
- **Precision**: $TP/(TP + FP)$. Of our positive predictions, how many were right?

Evaluation: These metrics apply to any classifier, whether it's a credit model or a text filter.

Decision thresholds

Both logistic regression and Naive Bayes output a probability $p(X)$. To act, we must choose a threshold k:

$$\text{Action} = \text{positive if } p(X) \geq k$$

Sensitive classifier ($k = 0.1$)

"Minimize misses."

- High recall (few misses).
- Low precision (many false alarms).
- Example: Cancer screening.

Conservative classifier ($k = 0.9$)

"Minimize false alarms."

- High precision (few false alarms).
- Low recall (many misses).
- Example: Spam filtering.

ROC curves

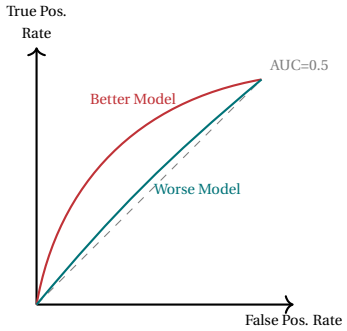
We evaluate a model's ranking ability by plotting performance across all possible thresholds.

ROC curve: plots sensitivity vs. (1, specificity).

AUC (Area Under Curve):

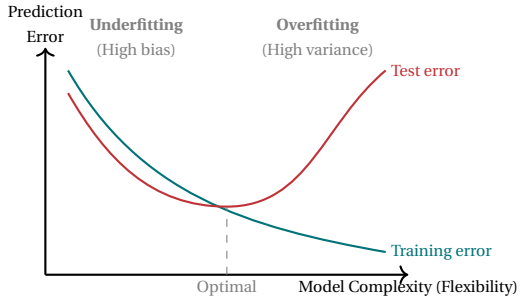
- **1.0**: perfect classifier.
- **0.7, 0.8**: decent performance.
- **0.5**: random guessing.

AUC measures the probability that a random positive observation will be ranked higher than a random negative one.



Resampling Methods

Model complexity: training vs. test performance



The gap between these curves represents **overfitting**.

- **Training error** measures how well the model "fits" the textbook data.
- **Test error** measures how well the model "generalizes" to unseen exams.

As models become more flexible, they stop learning the *signal* (the rule) and start memorizing the *noise* (the specific accidents of the data).

LLMs and Data Leakage

Consider a tech company training a new Large Language Model (LLM) to act as a coding assistant.

They evaluate the model on a test set of challenging 2024 coding problems and achieve a 95% success rate. However, the model was pre-trained on the entire internet, inadvertently including the solutions to those very problems.

If this model is deployed to users writing completely new code, what will happen? Why is it mathematically dangerous to evaluate an AI on data it has already seen?

Validation: estimating test performance

To avoid overfitting, we need to estimate performance on **unseen data**.

Validation set approach: Split data into two parts.

1. **Training set**: The model "studies" this data to learn parameters.
2. **Validation set**: Unseen data used to "grade" performance.

Issues with small samples: On small datasets, a single validation split is unstable (results depend on the draw) and wasteful (you lose observations for training). We need a way to use every observation for both training and testing.

k-Fold Cross-Validation

The gold standard for small-to-medium datasets:

1. Randomly divide data into k equal-sized "folds" (e.g., $k = 5$ or 10).
2. For each fold j:
3. Train on the other $k - 1$ folds.
4. Validate on fold j.

Why this solves the Economist's Problem:

- Every observation gets to be in the test set exactly once.
- We calculate the average error across all k folds, giving a much more stable, less biased estimate of the true test error without wasting data.

k-fold cross-validation: A resampling method that evaluates model performance by averaging results across k different partitions of the data.

Preparation Question

The bootstrap intuition

You have trained a complex statistical model to predict housing prices, but your manager asks: "What is the margin of error for these predictions?"

Unlike linear regression, complex models don't have a mathematical formula for standard errors. Without collecting more data, how could you use the data you have, along with computing power, to estimate this uncertainty?

The bootstrap intuition

Use the computer to mimic the process of drawing new samples from the population.

- Draw a new sample of size n **with replacement** from the original dataset.
- Repeat $B = 1000$ times.
- Calculate the estimator for each sample.
- The standard deviation of these B estimates is the **bootstrap standard error**.

Bootstrap: A resampling technique that involves drawing repeated samples with replacement from a dataset.

Summary

- **Logistic regression** models log-odds directly, making it a baseline for linear classification.
- **Naive Bayes** is a generative model that handles high-dimensional text data efficiently.
- **Accuracy can be misleading**: evaluation requires the confusion matrix, precision, recall and AUC analysis.
- **Resampling is essential**: use **cross-validation** to estimate test performance and the **bootstrap** to quantify uncertainty.
- **Next week**: moving beyond linear models with decision trees and non-linear boundaries.