Quantitative Research Methods

Chapter 5: Resampling Methods

Prof. Dr. Christoph Weisser

HSBI

Summer Semester 2026

The course at a glance

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

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

Contents

Optional and advanced material is collected in the **appendix** at the end of the deck.

Notation in this chapter

Symbol	Meaning
e_i^2	Squared prediction error for held-out observation i
$\mathrm{CV}_{(n)}$	LOOCV estimate of the test MSE (n single-point folds)
h_i	Leverage of observation i (one-fit OLS shortcut)
MSE_j	Held-out mean squared error of fold j
$\mathrm{CV}_{(k)}$	k-fold CV estimate: average of the k fold errors
$I(y_i \neq \hat{y}_i)$	0/1 indicator loss used for classification CV
B	Number of bootstrap replications
Z^{*b}	b-th bootstrap sample: n rows drawn with replacement
$\hat{\alpha}^{*b}$	Statistic $\hat{\alpha}$ recomputed on Z^{*b}
$\widehat{\mathrm{SE}}(\hat{\alpha})$	Bootstrap standard error: SD of the B replicates

Sources and references

Primary textbook

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

About these slides

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

Learning objectives for this chapter

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

- **Apply** the validation-set approach and articulate its weaknesses.
- **Compute** leave-one-out (LOOCV) and k-fold cross-validation test-error estimates.
- **Reason** about the bias–variance trade-off *within* the cross-validation procedure itself.
- **Use** the bootstrap to obtain standard errors and confidence intervals when no closed-form formula exists.
- **Avoid** the common resampling pitfalls (information leakage, naive time-series folds, group-ignoring splits).

Roadmap of this chapter

Two big ideas and four procedures

- **Cross-validation** for estimating test error: validation set, LOOCV, k-fold.
- **The bootstrap** for estimating standard errors and confidence intervals.

Why these matter: they are the practical workhorses you will use in every subsequent chapter to tune hyperparameters and report uncertainty.

Outline

The most important question in machine learning

Every model needs an honest answer to one question:

The single most-asked question

"How well will my model perform on data it has not seen?"

Without an answer to this question, every other claim about a model is unfalsifiable.

Why training error is the wrong answer

The model has been bent to fit the training points; the training error therefore overestimates how well the model would do on new data — often dramatically.

Training error is not test error

A flexible model can drive training error to zero while having arbitrarily bad test error. This is exactly the overfitting pathology of Chapter 2.

Takeaway

What we really want is the *test* error. But how do we estimate it without an external test set? That is what this chapter is about.

Two big ideas in this chapter

Cross-validation

Repeatedly hold part of the data out, fit on the rest, score on the held-out part. Average the scores.

Used to estimate **test error**.

The bootstrap

Sample *with replacement* from the data many times, recompute the statistic each time, look at the spread.

Used to estimate **standard errors** / **confidence intervals**.

Which question does each answer?

Cross-validation answers *"how accurate is my model?"*; the bootstrap answers *"how uncertain is my estimate?"* — two different jobs, two different tools.

Where this chapter is used in industry

Sector	Concrete application	Which decision it drives
Banking	Independent validation of a credit-risk (PD) model	Whether the model may go live
Retail, energy	Walk-forward backtest of a demand or load forecast	Replenishment and hedging settings
Insurance	Bootstrap of a loss-reserve or capital figure	The reserve that gets booked
Pharma, medtech	Patient-grouped CV of a diagnostic or triage model	Whether the model goes into a submission
Tech, platforms	Bootstrap interval for revenue per user in an A/B test	Ship or do not ship
Manufacturing	k-fold CV to tune a quality- or yield-prediction model	Which model complexity goes to production

In industry — The common thread

In every row the number that leaves the room is an *out-of-sample* number. Resampling is what turns a model someone built into a model someone is willing to sign.

Industry case in depth: validating a credit-risk model

In industry — The setting

A bank's retail portfolio needs a probability-of-default (PD) model — the Chapter 4 scorecard, now facing independent review. **Response:** default within the next 12 months (0/1). **Predictors:** months since worst delinquency, revolving-limit utilisation, recent credit enquiries, months on file, instalment-to-income ratio. **Output:** a PD per account, banded into rating grades, with a Gini / AUC reported on *two* hold-outs — one random, one *out-of-time* from a later origination cohort.

Which decision it feeds and who signs it off

The PD drives approve/decline cut-offs, risk-based pricing and the IFRS 9 expected-credit-loss provision. The modelling team does not sign it off: an *independent* validation unit re-runs the tests for a model-risk committee (US: Federal Reserve SR 11-7 guidance; euro area: ECB supervisory model reviews). The person who has to vote on the out-of-time Gini is a committee member from finance or the business, not a modeller.

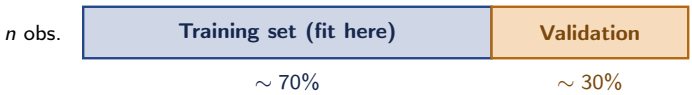
The honest caveat

Random k-fold folds across origination dates let the future inform the past: the Gini looks excellent and then evaporates on the next cohort.

Outline

The validation-set approach: split once, score once

The simplest possible resampling estimator: one random hold-out.



Recipe

1. Randomly split the data into a training set and a validation (or hold-out) set — typically 50/50 or 70/30.
2. Fit the model on the training set; score it on the validation set.

How to read this

The validation MSE estimates the test MSE of a model trained on only n_{train} observations — fewer than the full sample.

Validation set: pros and cons

Pros

- Conceptually simple.
- Very fast — one fit.
- Easy to explain.

Cons

- Estimate is highly variable across random splits.
- Uses only part of the data for fitting — tends to *over*estimate test error.

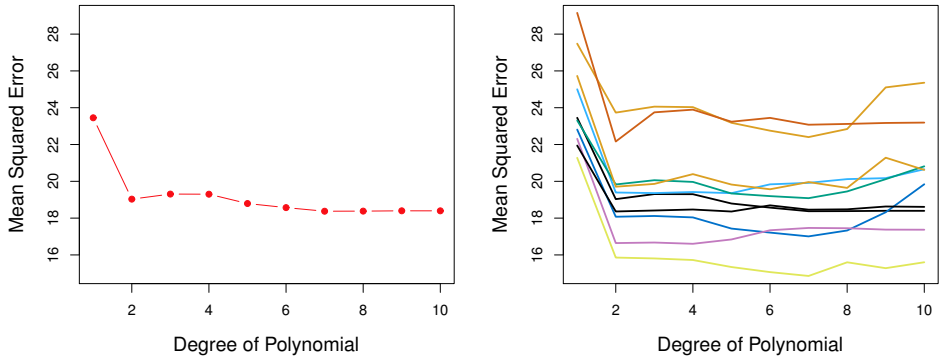
Takeaway

Both weaknesses stem from using a *single* split of only *part* of the data. Averaging over several splits — cross-validation — attacks both at once.

In industry — Banking: a single split is only the floor

A single hold-out is a fine first sanity check, but an independent validation unit will ask for an *out-of-time* hold-out as well: score a later cohort, not just other rows from the same one.

Validation across splits: how erratic is it?



Multiple repetitions on the same data, with different random splits, produce visibly different validation curves — and sometimes different "optimal" degrees.

Source: Figure 5.2 — ISLP, James et al. (2023).

Exercise 5.1 — Drawbacks of the Validation Set [Concept]

Exercise

A colleague splits a data set of $n = 400$ observations once, at random, into a 50/50 training/validation split, fits a model, and reports the validation MSE as "the" test error.

1. Name the **two** principal drawbacks of this validation-set approach.
2. Explain why re-running the split with a different random seed can change which polynomial degree looks "best".
3. Suggest one concrete change to the procedure that reduces both problems, and say why.

Solution — Exercise 5.1

Solution

(1) Two drawbacks.

- *High variability.* The estimate depends on the one random split. Different splits give visibly different validation MSEs, so the number is not reproducible.
- *Pessimistic bias (fewer training observations).* Only half the data ($n_{\text{train}} = 200$) is used to fit. Statistical methods generally perform worse with less training data, so the validation MSE tends to *overestimate* the test error of the model that would be fit on all 400 observations.

(2) Why the "best" degree changes. With only 200 points in training and 200 in validation, the validation curve is noisy. The differences between competing degrees are often smaller than that noise, so the argmin of the validation curve jumps between seeds — exactly the erratic behaviour shown in Figure 5.2.
(3) A concrete fix. Use k-fold cross-validation (e.g. $k = 10$) instead of a single split. Averaging over k folds *reduces variance* (the estimate no longer hinges on one split), and each fit now uses $\frac{k-1}{k} n = 360$ observations, which *reduces the pessimistic bias*. Both drawbacks improve simultaneously.

Common mistake: averaging many repeated 50/50 splits instead. That tames the variance but not the bias — every fit still uses half the data. Only larger training folds (CV) fix both at once.

Outline

Leave-one-out CV: take the validation idea to the limit

Make the validation set as small as possible: one observation.

LOOCV recipe

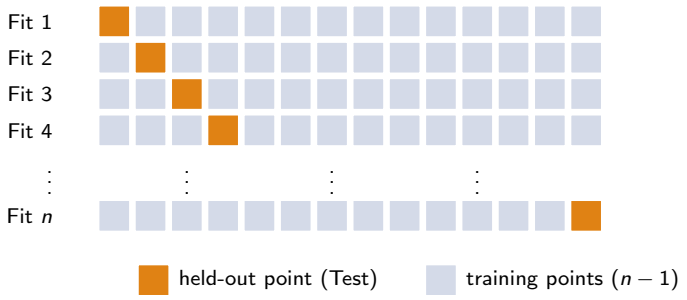
For $i = 1, \ldots, n$:

1. Train on all $n - 1$ observations except i.
2. Predict for i; record the squared error e_i^2.

LOOCV estimate of test MSE:

$$\mathrm{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} e_i^2.$$

LOOCV as the n-fold limit (native diagram)



Takeaway

LOOCV is k-fold with $k = n$: each observation is held out exactly once, giving n fits and $\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} e_i^2$. Lowest bias, but n fits — use the OLS leverage shortcut when it applies.

Reading the LOOCV formula

How to read this

- Each e_i^2 is computed using a model that did *not* see observation i during training.
- Averaging n such squared prediction errors gives an (almost) unbiased estimate of the test MSE.
- Because every fit uses $n-1$ training points, the bias is very small.

Pros

Uses (almost) all data for training; deterministic — no random splits; symmetric — every observation gets the same treatment.

LOOCV's hidden cost

Computational cost

Naive LOOCV requires fitting the model n times. With $n = 10^4$ and a complex model, this is impractical.

The OLS shortcut

For *ordinary least squares* (and ridge regression), there is a one-fit shortcut using the leverage statistic.

In industry — Chemometrics and pilot studies: where n is small

LOOCV earns its keep when observations are expensive: a chemometric calibration built on a few dozen production batches, a pilot study, a handful of clinical sites. Wasting 20% of forty samples on a hold-out is not an option — so hold out one at a time.

The OLS LOOCV shortcut

For OLS regression specifically:

LOOCV formula for OLS

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2,$$

where h_i is the leverage of observation i (the i-th diagonal of the hat matrix, Ch. 3).

How to read this

The denominator $1 - h_i$ corrects for the influence each point has on its own fitted value. High-leverage points get a stronger correction. One fit + the diagonal of H replaces n fits.

Run this live: §4 *LOOCV (leave-one-out)*

The notebook scores degrees 1–5 with `LeaveOneOut` — all $n = 392$ fits — which is the cost this shortcut removes.

Outline

k-fold CV: a practical compromise

Between the (variable) validation set and the (slow) LOOCV.

Recipe

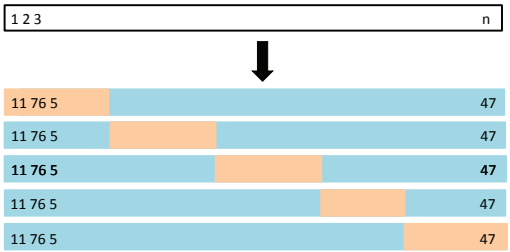
1. Split the data into k roughly equal folds (typically $k = 5$ or $k = 10$).
2. For each fold $j = 1, \ldots, k$: train on the other $k - 1$ folds, validate on fold j.
3. Average the k validation errors.

Every observation appears in the validation fold exactly once and in the training fold $k - 1$ times.

Takeaway

k-fold inherits LOOCV's near-full training sets *and* the validation set's low cost: only k fits, low bias, moderate variance.

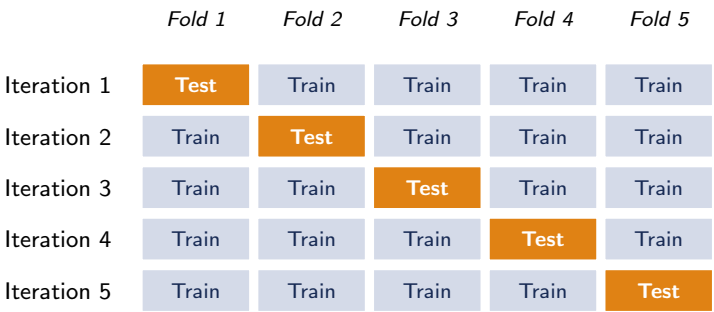
Visualising 5-fold CV



Each row is one of the five fits. The single highlighted block is the validation fold: it is the leftmost fifth of the data in row 1 and slides one fifth to the right in each row below, so every observation is held out exactly once. The rest of each row is the training data.

Source: Figure 5.5 — ISLP, James et al. (2023).

k-fold cross-validation, step by step (native diagram)



Takeaway

Each of the five folds serves as the test set exactly once (orange); the rest train the model. The estimate is the average of the five held-out errors: $\mathrm{CV}_{(5)} = \frac{1}{5}\sum_{j=1}^{5} \mathrm{MSE}_j$.

LOOCV vs. k-fold: the bias-variance trade-off

LOOCV

Lowest bias (training sets nearly the full size) but *highest variance*: the n training sets overlap enormously, so the n errors are highly correlated.

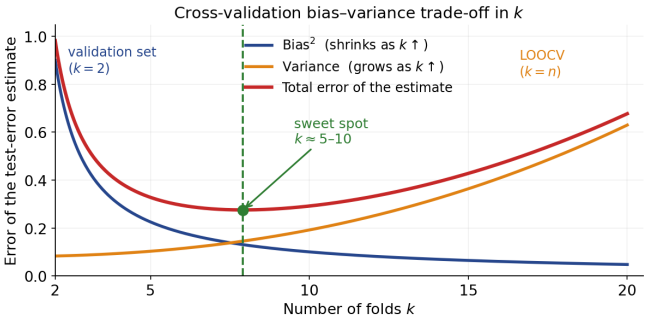
5-fold CV

Slightly upward biased (smaller training sets) but *much lower variance* because the k training sets share less data.

Practical recommendation

The $k = 10$ rule of thumb usually wins on mean-squared error of the test-error estimate. Use LOOCV only when the OLS shortcut applies or n is tiny.

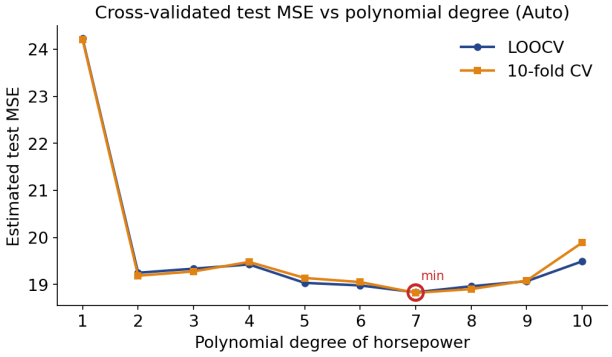
Why $k \approx 10$ wins: visualising the trade-off



Takeaway

More folds $\Rightarrow$ larger training sets (*less bias*) but training sets that overlap more (*more variance*). The total error of the estimate is smallest around $k = 5$–10 — neither the single split ($k = 2$) nor LOOCV ($k = n$).

Cross-validation in practice: choosing the polynomial degree



Takeaway

On Auto (`mpg`∼`poly(horsepower,d)`), LOOCV and 10-fold CV both drop sharply from degree 1 to 2 then stay flat — CV points to a low-degree model.

Cross-validation for classification

For a qualitative response, swap squared error for the 0/1 misclassification loss:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} I\left(y_i \neq \hat{y}_i^{(-i)}\right).$$

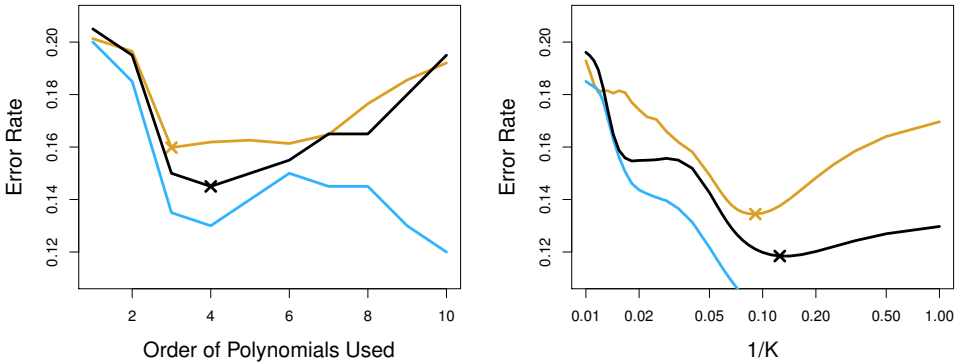
How to read this

- $I(\cdot)$ is the indicator: 1 when the prediction is *wrong*, 0 when it is right.
- $\hat{y}_i^{(-i)}$ is the class predicted for i by a model fit *without* observation i.
- So $\mathrm{CV}_{(n)}$ is just the held-out error *rate* — the fraction of points misclassified.

Takeaway

Same mechanics, different loss. For threshold-dependent metrics (sensitivity at fixed specificity), *nest* the CV: tune the threshold inside, score the model outside.

CV in action: KNN on a synthetic problem



CV closely tracks the true test error — justifying its use for hyperparameter selection.

Source: Figure 5.8 — ISLP, James et al. (2023).

Three pitfalls that quietly invalidate CV

The classic CV mistakes

1. **Information leakage.** Any preprocessing fit on the *full* data and used inside CV (scaling, feature selection, imputation) corrupts the estimate. Always fit preprocessing inside the CV loop.
2. **Time series.** Vanilla random folds destroy temporal order. Use rolling-origin or expanding-window CV.
3. **Grouped data.** CV folds must respect groups (patients, customers) to avoid within-group leakage.

In industry — Forecasting and credit: expensive — not textbook — mistakes

Forecasts and trading strategies are backtested *walk-forward* (train to t, score $t+1$) for reason 2; churn, credit and healthcare models split by *customer* or *patient* for reason 3. Accuracy that came from leakage fails after deployment, not before.

Mini-check: cross-validation

Quick self-assessment

1. Why does 10-fold CV usually beat LOOCV on mean-squared error of the test-error estimate?
2. For a time series, why are random folds wrong?
3. If you standardise predictors on the full data and then run CV, what is the consequence?

Answers

1. The n training sets in LOOCV overlap heavily, so the n errors are correlated — variance of the average is high. 2. They destroy temporal order; future leaks into training. 3. Information leakage: the test fold informed the standardisation.

Exercise 5.3 — k-Fold vs. LOOCV Trade-off [Concept]

Exercise

1. Place validation-set, 10-fold CV, and LOOCV on the *bias* axis and on the *variance* axis of their test-error estimates. Justify each ranking in one sentence.
2. Why are the n error terms in LOOCV so highly correlated, and why does that inflate the variance of the average?
3. Why is $k = 5$ or $k = 10$ the usual recommendation?

Solution — Exercise 5.3

Solution

(1) Bias and variance rankings.

- *Bias* (of the test-error estimate): validation set is *highest* (trains on only $\sim n/2$); 10-fold is intermediate (trains on $0.9n$); LOOCV is *lowest* (trains on $n-1$, essentially the full sample).
- *Variance*: LOOCV is *highest*; 10-fold is *moderate*; the single validation set is also highly variable across splits. So on the bias–variance trade-off, 10-fold sits in the sweet spot.

(2) Correlated errors in LOOCV. Any two LOOCV training sets share $n-2$ of their $n-1$ observations, so the n fitted models are almost identical and their errors are highly positively correlated. The variance of an average of positively correlated terms (each with variance σ^2 and average pairwise correlation ρ), $\mathrm{Var}(\bar{e}) = \frac{1}{n}\sigma^2 + \frac{n-1}{n}\rho\,\sigma^2$, does not shrink toward zero when ρ is large — hence high variance. In k-fold with small k the training sets overlap less, so ρ is smaller.

(3) Why $k = 5$ or 10. These values give training sets of 80–90% of the data (small bias) while keeping the fold errors only moderately correlated (moderate variance), and they cost only 5 or 10 fits. Empirically they minimise the mean-squared error of the test-error estimate — the best of both worlds.

Common mistake: reading "bias" here as model bias. It is the bias of the *test-error estimate*: training on fewer rows makes every model look worse than it will be when fit to all n.

Exercise 5.4 — Computing a k-Fold CV Estimate [Math]

Exercise

A 5-fold cross-validation of a regression model produces the following per-fold mean squared errors:

$$\mathrm{MSE}_1 = 10.2,\ \mathrm{MSE}_2 = 12.4,\ \mathrm{MSE}_3 = 9.6,\ \mathrm{MSE}_4 = 11.8,\ \mathrm{MSE}_5 = 10.0.$$

1. Compute the 5-fold CV error estimate $\mathrm{CV}_{(5)}$.
2. Compute the standard deviation of the five fold errors and the standard error of the mean.
3. If the folds have unequal sizes n_j, how should the average be modified?

Solution — Exercise 5.4 (1/2)

Solution

(1) CV estimate.

$$\mathrm{CV}_{(5)} = \frac{1}{5}\sum_{j=1}^{5}\mathrm{MSE}_j = \frac{10.2 + 12.4 + 9.6 + 11.8 + 10.0}{5} = \frac{54.0}{5} = 10.8.$$

(2) Spread of the fold errors. Deviations from the mean 10.8 are -0.6, 1.6, -1.2, 1.0, -0.8; their squares sum to $0.36 + 2.56 + 1.44 + 1.00 + 0.64 = 6.0$. The sample standard deviation is

$$s = \sqrt{\frac{6.0}{5-1}} = \sqrt{1.5} \approx 1.22,$$

and the standard error of the mean is $s/\sqrt{5} \approx 1.22/2.236 \approx 0.55$. So $\mathrm{CV}_{(5)} \approx 10.8 \pm 0.55$.
Interpretation. The five folds agree to within about 5% of the estimate (0.55/10.8), so the CV number is stable; quoting 10.8 *with* its spread is what turns a point estimate into a defensible claim.

Solution — Exercise 5.4 (2/2)

Solution

(3) Unequal fold sizes. Use a size-weighted average so every observation counts equally:

$$\mathrm{CV}_{(k)} = \frac{1}{n}\sum_{j=1}^{k} n_j \,\mathrm{MSE}_j, \qquad n = \sum_{j=1}^{k} n_j.$$

With equal fold sizes this reduces to the simple mean in part (1).
Why weight? Fold j's error is $\mathrm{MSE}_j = \frac{1}{n_j}\sum_{i\in\text{fold } j} e_i^2$, so $n_j\,\mathrm{MSE}_j$ restores the fold's *total* squared error and dividing by n yields exactly the mean of all n individual squared errors — every observation counts once.
Micro-example. Folds of sizes 90 and 110 with $\mathrm{MSE} = 10$ and 12: unweighted mean 11.0, correct weighted value $\frac{90(10)+110(12)}{200} = \frac{2220}{200} = 11.1$ — the larger fold rightly counts more.

Common mistake

averaging fold MSEs unweighted when fold sizes differ; with equal folds the two coincide, which is why the slip often goes unnoticed.

Outline

Why we need the bootstrap

We often want the standard error or sampling distribution of a statistic $\hat{\alpha}$ for which no closed-form formula exists.

The thought experiment

If only we could see many independent samples! Then we could just compute $\hat{\alpha}$ on each sample and take the empirical SD.

We don't have many samples — we have one. The bootstrap is a trick to simulate "many samples" from the one we have.

In industry — Insurance and platforms: where the formula is missing

Actuaries quote loss-reserve *ranges*; risk teams put error bars on value-at-risk and stress figures; platform teams need a confidence interval for revenue per user, which is heavy-tailed enough that the textbook t-interval is hard to defend. None of these statistics has a usable closed-form standard error — so all of them are bootstrapped.

The bootstrap algorithm

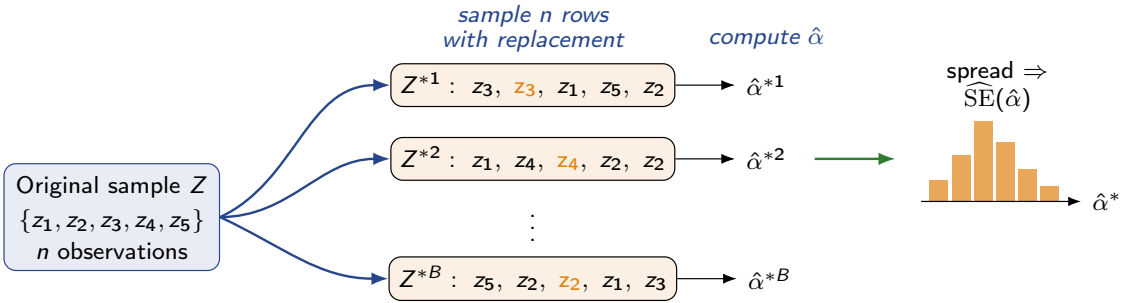
Recipe

1. Draw B bootstrap samples $\mathsf{Z}^{*1}, \ldots, \mathsf{Z}^{*B}$, each by sampling n rows *with replacement* from the data.
2. Compute the statistic $\hat{\alpha}^{*b}$ on each.
3. Estimate the standard error by (with $\bar{\alpha}^* = \frac{1}{B}\sum_b \hat{\alpha}^{*b}$ the mean of the replicates):

$$\widehat{\mathrm{SE}}(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\alpha}^{*b} - \bar{\alpha}^*\right)^2}.$$

4. Build percentile $(1-\alpha)$ confidence intervals as $[\hat{\alpha}^*_{\alpha/2}, \hat{\alpha}^*_{1-\alpha/2}]$.

The bootstrap, schematically (native diagram)



Takeaway

Resample the rows *with replacement* (the repeated observation in each resample is set in bold), recompute $\hat{\alpha}$ on each of the B samples, and read the standard error off the spread of the replicates.

Reading the bootstrap

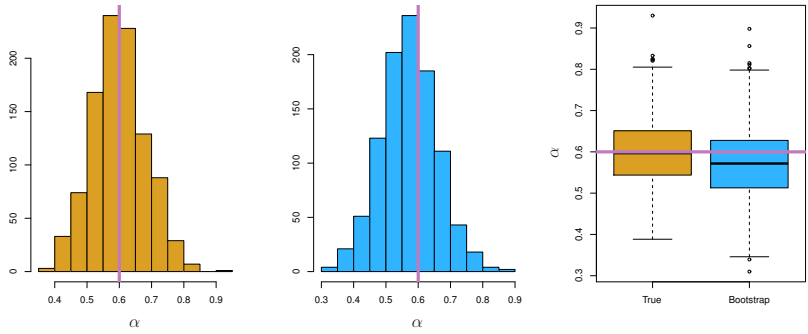
How to read this

- Sampling *with replacement* means some original rows appear multiple times in the bootstrap sample, while others don't appear at all (about 37 % are missing each time).
- Each $\hat{\alpha}^{*b}$ is a draw from an approximation to the sampling distribution of $\hat{\alpha}$.
- The empirical SD of these draws is our estimate of the true SE: in $\widehat{\mathrm{SE}}(\hat{\alpha})$, B is the number of resamples and $\bar{\alpha}^*$ their mean.

Bootstrap of a regression slope

For the Auto data, the OLS slope is $\hat{\beta}_1 = -0.158$. With $B = 1000$ bootstrap samples we get $\widehat{\mathrm{SE}}(\hat{\beta}_1) \approx 0.0075$, somewhat *larger* than the OLS analytical SE of 0.0064 — the bootstrap makes no linear-model assumptions.

The bootstrap sampling distribution, visualised



Left: $\hat{\alpha}$ across 1,000 *simulated* data sets — the true sampling distribution. Centre: $\hat{\alpha}$ across 1,000 *bootstrap* samples drawn from one data set. Right: the same two distributions as boxplots — the bootstrap reproduces the spread, which is why its SD serves as the SE.

Source: Figure 5.10 — ISLP, James et al. (2023).

When the bootstrap works — and when it fails

Works for

- Sample mean, variance, regression coefficients.
- Quantiles in the interior of the distribution.
- Correlation, R^2.
- Bagging in random forests.

Fails or misleads for

- Extreme quantiles (max, min).
- Strongly dependent time series (use *block* bootstrap).
- Hierarchical / clustered data (use cluster bootstrap).
- Small samples with discrete data.

Takeaway

Rule of thumb: the bootstrap is trustworthy when rows are roughly exchangeable and the statistic is smooth. Break either and you need a specialised variant (block, cluster, . . .).

Exercise 5.5 — Bootstrap Inclusion Probability [Math]

Exercise

A bootstrap sample is drawn by sampling n rows *with replacement* from a data set of size n.

1. Show that the probability a specific observation is *not* in a given bootstrap sample is $\left(1 - \frac{1}{n}\right)^n$.
2. Hence write the probability that the observation *is* included, and evaluate it for $n = 5$ and for $n = 1000$.
3. What is the limiting inclusion probability as $n \to \infty$?

Solution — Exercise 5.5 (1/2)

Solution

(1) Probability of exclusion. A single draw picks the given observation with probability $\frac{1}{n}$, hence misses it with probability $1 - \frac{1}{n}$. Because we sample *with replacement*, the data set is restored before every draw, so the n draws are independent and the n miss probabilities *multiply*:

$$P(\text{not included}) = \underbrace{\left(1 - \tfrac{1}{n}\right) \cdots \left(1 - \tfrac{1}{n}\right)}_{n \text{ draws}} = \left(1 - \tfrac{1}{n}\right)^n.$$

(2) Inclusion probability (complement rule): $P(\text{included}) = 1 - \left(1 - \frac{1}{n}\right)^n$.

- $n = 5$: $1 - \frac{1}{5} = 0.8$, so $1 - (0.8)^5 = 1 - 0.3277 = 0.6723$.
- $n = 1000$: $1 - (0.999)^{1000} \approx 1 - 0.3677 = 0.6323$.

Interpretation: the inclusion probability *falls* as n grows — but only from 67% to 63%, so it is already close to its limit for tiny samples.

Solution — Exercise 5.5 (2/2)

Solution

(3) Limit. The standard limit $\lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n = e^x$ with $x=-1$ gives $\left(1-\frac{1}{n}\right)^n \to e^{-1} \approx 0.3679$, hence

$$P(\text{included}) \to 1 - e^{-1} \approx 1 - 0.3679 = 0.6321.$$

So each bootstrap sample contains, on average, about 63.2% of the distinct original observations — the famous 0.632 figure. The remaining $\sim$ 36.8% (the "out-of-bag" points) can serve as a built-in validation set, as used in bagging (Ch. 8).

Common mistake

concluding that a bootstrap sample is "smaller" than the data. It always contains exactly n rows; 63.2% is the expected fraction of *distinct* originals — the remaining slots are filled by duplicates.

Extended Exercise 5.2 — Bootstrap SE: Statistic and Coefficient [Python]

Extended exercise (15 min)

Using `Portfolio` (columns `X`,`Y`) and `Auto`, write Python that:

1. bootstraps ($B = 1000$, resampling rows *with replacement*) the minimum-variance allocation $\hat{\alpha} = \dfrac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$ and reports the bootstrap SE and a 95% *percentile* interval;
2. bootstraps the OLS slope of `mpg` on `horsepower` and compares the bootstrap SE with the textbook (formula-based) OLS standard error.

Report the expected numbers and interpret the comparison in part (2).

Solution — Extended Exercise 5.2 (1/3)

```
import numpy as np, pandas as pd
D = "../ALL CSV FILES - 2nd Edition/"
Port = pd.read_csv(D+"Portfolio.csv") # load Portfolio data
X, Y = Port["X"].values, Port["Y"].values # two asset returns
def alpha(x, y): # min-variance allocation
    sx, sy = np.var(x, ddof=1), np.var(y, ddof=1) # sample variances
    sxy = np.cov(x, y, ddof=1)[0, 1] # sample covariance
    return (sy - sxy) / (sx + sy - 2*sxy)

rng = np.random.default_rng(0) # reproducible RNG
n, B = len(X), 1000 # B bootstrap replications
reps = np.empty(B)
for b in range(B):
    idx = rng.integers(0, n, n) # sample WITH replacement
    reps[b] = alpha(X[idx], Y[idx]) # recompute alpha on resample
print(round(alpha(X, Y), 4), round(reps.std(ddof=1), 4),
      np.round(np.quantile(reps, [.025, .975]), 3)) # SE + 95% CI
```

Solution — Extended Exercise 5.2 (2/3)

```
import statsmodels.api as sm
Auto = pd.read_csv(D+"Auto.csv", na_values="?").dropna() # clean rows
hp = Auto["horsepower"].values.astype(float) # predictor
mpg = Auto["mpg"].values # response

ols = sm.OLS(mpg, sm.add_constant(hp)).fit() # formula-based SE
print(round(ols.params[1], 4), round(ols.bse[1], 5))

m = len(hp); slopes = np.empty(1000) # B = 1000 replications
for b in range(1000):
    idx = rng.integers(0, m, m) # resample rows w/ replacement
    fit = sm.OLS(mpg[idx], sm.add_constant(hp[idx])).fit() # refit OLS
    slopes[b] = fit.params[1] # store bootstrap slope
print(round(slopes.std(ddof=1), 5), # bootstrap SE of slope
      np.round(np.quantile(slopes, [.025, .975]), 4)) # 95% interval
```

Solution — Extended Exercise 5.2 (3/3)

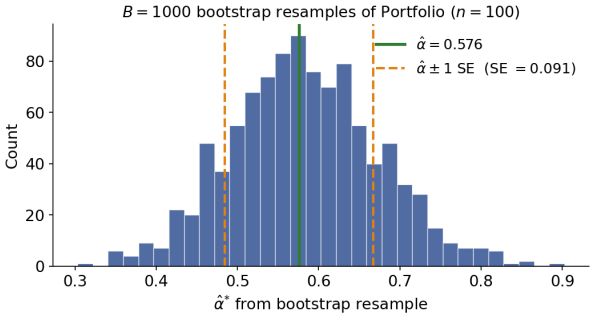
Solution — Expected numbers and interpretation

Portfolio $\hat{\alpha}$. $\hat{\alpha} \approx 0.576$, bootstrap $\widehat{\mathrm{SE}} \approx 0.091$, 95% percentile interval $\approx [0.41, 0.77]$. No closed-form SE exists, so the bootstrap is the natural tool (ISLP Section 5.2).
Auto slope. OLS slope $\hat{\beta}_1 \approx -0.158$ with formula SE ≈ 0.0064; the bootstrap gives SE ≈ 0.0075 and interval $\approx [-0.173, -0.144]$.
Interpretation. The two slope SEs are close, which *validates* the bootstrap. Where they differ the bootstrap is often *more trustworthy*: the formula SE assumes the linear model is correct with constant-variance errors, whereas the bootstrap assumes neither and picks up the mild non-linearity/heteroscedasticity in `Auto` — hence the slightly larger bootstrap SE.

Common mistake

resampling `X` and `Y` as two *independent* columns — that destroys the covariance $\hat{\sigma}_{XY}$ that $\hat{\alpha}$ is built from. Always resample whole *rows* (pairs), i.e. one shared index vector.

The bootstrap in action: the sampling distribution of $\hat{\alpha}$



Takeaway

On `Portfolio.csv` the point estimate is $\hat{\alpha} = 0.576$; $B = 1000$ bootstrap resamples give $\widehat{\mathrm{SE}}(\hat{\alpha}) = 0.091$, so a rough 95 % range is $0.576 \pm 2(0.091) \approx [0.39, 0.76]$ — an uncertainty statement no closed-form formula supplies for α.

Outline

Python lab — run it live in the notebook

Companion notebook: `chapter_05_lab.ipynb`

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

- §1 *The Auto data set*, then §2 *Validation set approach*;
- §3 *k-fold cross-validation* and §4 *LOOCV (leave-one-out)*;
- §5 *The bootstrap*.

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

Validation set in scikit-learn

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from ISLP import load_data

Auto = load_data("Auto") # load the Auto data
X = Auto[["horsepower"]].values; y = Auto["mpg"].values
Xtr,Xte,ytr,yte = train_test_split(X, y, test_size=0.5,
                                   random_state=1) # one 50/50 split
mdl = LinearRegression().fit(Xtr, ytr) # fit on training half only
mean_squared_error(yte, mdl.predict(Xte)) # validation-set MSE
```

Try changing `random_state` and re-running — the validation MSE will jump around noticeably.

k-fold cross-validation

```
from sklearn.model_selection import KFold, cross_val_score
import numpy as np

cv = KFold(n_splits=10, shuffle=True, random_state=0) # 10 folds
mse = -cross_val_score(LinearRegression(), X, y, # 10 fits, one per fold
                       cv=cv, scoring="neg_mean_squared_error")
print(mse.mean(), mse.std()) # CV estimate and spread
```

Always shuffle (unless time-ordered). The `neg_` sign is a scikit-learn quirk: scores are always "larger is better", so MSE is negated.

Bootstrap of a regression coefficient

```python
import numpy as np
rng = np.random.default_rng(0) # reproducible RNG

def boot_one(rng):
    idx = rng.integers(0, len(X), size=len(X)) # rows w/ replacement
    return LinearRegression().fit(X[idx], y[idx]).coef_[0] # refit slope

B = 1000 # bootstrap replications
boots = np.array([boot_one(rng) for _ in range(B)]) # B slope replicates
print("SE:", boots.std(ddof=1)) # bootstrap SE
print("95%:", np.quantile(boots, [0.025, 0.975])) # percentile CI
```

Exercise 5.6 — Bootstrap SE of the Portfolio α [Python]

Exercise

The Portfolio data has returns `X` and `Y`. The minimum-variance allocation is

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$

Write Python that (i) computes $\hat{\alpha}$ on the full sample and (ii) estimates its standard error with $B = 1000$ bootstrap resamples. Report the expected result.

Solution — Exercise 5.6 (1/2)

Solution — Method: why the bootstrap is the right tool

$\hat{\alpha}$ is a nonlinear *ratio* of variances and a covariance, so no textbook SE formula applies. Bootstrap plug-in principle: treat the sample as the population, draw n rows *with replacement* (keeping each (X_i, Y_i) pair together), recompute $\hat{\alpha}$ on every resample, and report the sample SD of the B replicates as $\widehat{\mathrm{SE}}(\hat{\alpha})$.

```
import numpy as np, pandas as pd
Port = pd.read_csv("../ALL CSV FILES - 2nd Edition/Portfolio.csv")
X = Port["X"].values; Y = Port["Y"].values # two asset returns

def alpha(x, y): # min-variance allocation
    sx, sy = np.var(x, ddof=1), np.var(y, ddof=1) # sample variances
    sxy = np.cov(x, y, ddof=1)[0, 1] # sample covariance
    return (sy - sxy) / (sx + sy - 2*sxy)

a_hat = alpha(X, Y) # point estimate
```

Solution — Exercise 5.6 (2/2)

```
rng = np.random.default_rng(0) # reproducible RNG
n, B = len(X), 1000 # B bootstrap replications
reps = np.empty(B)
for b in range(B):
    idx = rng.integers(0, n, size=n) # sample with replacement
    reps[b] = alpha(X[idx], Y[idx]) # recompute alpha on resample

print("alpha_hat:", round(a_hat, 4))
print("bootstrap SE:", round(reps.std(ddof=1), 4)) # SD of replicates
```

Solution — Expected result

You should obtain $\hat{\alpha} \approx 0.58$ and a bootstrap standard error of roughly $\widehat{\mathrm{SE}}(\hat{\alpha}) \approx 0.09$ (matching ISLP Section 5.2). Interpretation: across resamples the estimated optimal allocation to asset X varies by about ± 0.09, which quantifies the uncertainty in $\hat{\alpha}$ with no closed-form formula required.

Common mistake: drawing separate index vectors for `X` and `Y` — that breaks the X–Y covariance the statistic depends on. One shared `idx` keeps the pairs intact.

Extended Exercise 5.3 — Leak-Free CV for Tuning and Evaluation [Integrative]

Extended exercise (15 min)

You must (a) choose a tuning parameter — say the polynomial degree d (or the KNN neighbourhood size K) — *and* (b) report an honest estimate of the final model's test error.

1. A colleague standardises all predictors and selects the best d features on the *full* data set, *then* runs 10-fold CV. Explain precisely why the resulting error estimate is optimistically biased (data leakage).
2. Design the *correct* procedure. Where must scaling, feature selection, and the d/K choice happen relative to the CV split?
3. Sketch a *nested* CV that both tunes d/K and gives an unbiased test-error estimate. What does each loop do?

Solution — Extended Exercise 5.3 (1/3)

Solution — Why full-data preprocessing leaks

(1) The leak. Cross-validation is honest only if each validation fold is treated exactly like genuinely unseen data. If you standardise or select features using *all* n rows first, then every validation fold has already influenced

- the scaling constants (means/SDs use the fold's own rows), and
- which features are kept (the choice saw the fold's response).

The model is then scored on data that helped build it, so the CV error *underestimates* the true test error — sometimes dramatically when p is large and selection is aggressive. The fold is no longer "held out".

Solution — Extended Exercise 5.3 (2/3)

Solution — Correct procedure: everything inside the split

Fit *all* data-dependent steps — scaling, feature selection, and the d/K choice — *inside* each training fold only. A `Pipeline` makes this automatic: the search re-fits every step on each fold's training part.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV, KFold

pipe = Pipeline([("scale", StandardScaler()), # scaling INSIDE each fold
                 ("poly", PolynomialFeatures()), # degree = tuning param
                 ("lm", LinearRegression())])
inner = KFold(10, shuffle=True, random_state=0) # inner tuning loop
search = GridSearchCV(pipe, {"poly__degree": [1, 2, 3, 4, 5]}, # d grid
                      cv=inner, scoring="neg_mean_squared_error")
```

Solution — Extended Exercise 5.3 (3/3)

```
from sklearn.model_selection import cross_val_score
outer = KFold(5, shuffle=True, random_state=1) # outer evaluation loop
test_err = -cross_val_score(search, X, y, cv=outer, # nested CV
                            scoring="neg_mean_squared_error")
print("chosen degree:", search.fit(X, y).best_params_) # tuned d
print("unbiased test MSE:", test_err.mean(), "+/-", test_err.std())
```

Solution — Nested CV: two loops and two jobs

(3) The *inner* loop (inside `GridSearchCV`) selects d/K using only each outer-training part. The *outer* loop scores the whole tuning procedure on data it never touched, giving an *unbiased* test-error estimate. Tuning and evaluation use disjoint data, so neither contaminates the other.

Common mistake

reporting `search.best_score_` as the test error. That score *chose* the winner, so it is optimistically biased — only the outer loop's score is honest.

Outline

Chapter 5 in one slide

What we accomplished

Acquired the most important practical skill in statistical learning: estimating test error and uncertainty without an oracle test set.

- Validation set, LOOCV, k-fold CV — three estimators of test error.
- Bias–variance trade-off *within* the estimator itself.
- The bootstrap for standard errors and confidence intervals.
- Common pitfalls: leakage, time-series CV, grouped data.

Vocabulary checklist

Term	Meaning
Validation set	Single random hold-out split
LOOCV	Leave-one-out CV ($k = n$)
k-fold CV	Average of k disjoint hold-outs
Bootstrap	Resample with replacement, B times
Standard error (SE)	SD of an estimator across samples
Information leakage	Test info bleeding into training pipeline
Nested CV	CV for tuning, second CV for evaluation
Block bootstrap	For time series
Cluster bootstrap	For grouped data

Key formulas at a glance

k-fold and LOOCV estimates

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{j=1}^{k}\mathrm{MSE}_j, \qquad \mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i-\hat{y}_i}{1-h_i}\right)^2 \text{ (OLS)}.$$

Why LOOCV has high variance (mean of n errors, each with variance σ^2, average pairwise correlation ρ)

$$\mathrm{Var}(\bar{e}) = \frac{1}{n}\sigma^2 + \frac{n-1}{n}\rho\,\sigma^2 \;\longrightarrow\; \rho\,\sigma^2 \quad (n\to\infty).$$

Bootstrap SE and inclusion probability ($\approx 37\%$ left out)

$$\widehat{\mathrm{SE}}(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\alpha}^{*b}-\bar{\alpha}^{*})^2}, \qquad 1-\left(1-\tfrac{1}{n}\right)^n \to 1-e^{-1}\approx 0.632.$$

Methods comparison

Procedure	**Bias**	**Variance / Cost**
Single validation set	Upward	High variance
LOOCV	Lowest	Highest variance, n fits
5-fold CV	Slightly upward	Moderate variance, 5 fits
10-fold CV	Smaller bias than 5-fold	Moderate variance
Bootstrap	Estimates SE, not test error	B fits

Decision rules of thumb

- For tuning a hyperparameter: 5- or 10-fold CV.
- For final evaluation: external held-out set or nested CV.
- For SEs and CIs: bootstrap (or theory if available).
- For time series: rolling-origin CV (*never* random folds).
- For grouped data: split at the group level.
- Always fit *all* preprocessing inside the CV loop.

In industry — Governance: what you report to a steering committee

Never the tuned CV score. Searching hyperparameters and reporting on the same folds is optimistic by construction; quote the nested-CV figure or a hold-out the search never touched, with its spread.

Common pitfalls

Don't

1. Standardise / impute / select features on the full data *before* CV.
2. Use random folds for time series.
3. Optimise tuning parameters on the same fold you score them on.
4. Confuse the bootstrap with cross-validation — they answer different questions.
5. Use the bootstrap for extreme quantiles or under strong dependence.
6. Report a single CV score without an estimate of its variability.

Connections to the rest of the course

- Ch. 6: cross-validation chooses λ in ridge / lasso.
- Ch. 7: cross-validation chooses spline df and LOESS span.
- Ch. 8: bagging is the bootstrap; OOB error is built-in CV.
- Ch. 10: tune learning rate, dropout via CV.
- Ch. 13: resampling-based p-values and FDR-control procedures.

Self-check questions

1. Why does the validation-set estimate have such high variance? (pp. 16–17)
2. Show that 5-fold CV has lower variance than LOOCV. (pp. 30–31)
3. How would you cross-validate a model on a time series? (p. 35, and the rules of thumb on p. 74)
4. Describe a scenario in which the bootstrap estimate of SE is wrong. (p. 48)
5. Outline a nested-CV procedure for unbiased evaluation while tuning a hyperparameter. (pp. 66–68)
6. For OLS, derive the LOOCV shortcut formula using leverage. (p. 25; reasoning and practice in the appendix, pp. 82–83)

Five things to remember

1. Resampling lets us estimate test error and uncertainty without a separate test set.
2. Validation set: simple but variable.
3. LOOCV: low bias, high variance, computational shortcuts for OLS.
4. k-fold ($k = 5$ or 10): the workhorse compromise.
5. Bootstrap: resample with replacement to estimate SEs and CIs.

What's next

Chapter 6: *Linear Model Selection and Regularisation*.

- Subset selection.
- Ridge regression and the lasso.
- Dimension reduction: PCR and PLS.
- How to use cross-validation to pick tuning parameters.

Appendix — what is in here, and why

Nothing in the appendix is needed to follow the chapter: each slide is a formal derivation, a longer worked exercise, or a side topic. Read it when you want the full story, or when a later chapter sends you back here.

Contents of the appendix

- **Drilling the leverage shortcut (Exercise 5.2 and Extended Exercise 5.1)** — both work the identity that gives LOOCV for a linear model from a *single* least-squares fit. The identity itself stays in the main deck ("The OLS LOOCV shortcut"); what moves here is the arithmetic practice, which is heavy going in a lecture and ideal as homework. Remember that the shortcut is special: it holds for least squares (and ridge) only.

Exercise 5.2 — LOOCV Fits and the Leverage Shortcut [Math]

Exercise

Consider a data set with $n = 500$ observations.

1. How many separate model fits does *naive* LOOCV require? How many for 10-fold CV?
2. State the OLS leverage shortcut for $\mathrm{CV}_{(n)}$ and explain how many full model fits it needs.
3. For a simple linear regression with $n = 4$ points, one observation has residual $y_i - \hat{y}_i = 2.0$ and leverage $h_i = 0.5$. Compute this point's contribution $\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$ to the LOOCV sum and compare it with the naive squared residual e_i^2.

Solution — Exercise 5.2 (1/2)

Solution

(1) Number of fits. Naive LOOCV leaves out one observation at a time and refits, so it requires $n = 500$ fits. 10-fold CV refits once per fold, so it requires 10 fits.
(2) The OLS shortcut. For ordinary least squares,

$$\mathrm{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2 ,$$

where $\hat{y}_i$ and the leverages h_i (diagonal of the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$) come from a *single* fit on the full data. So it needs only **one** fit instead of n — a dramatic saving.
Why this works: OLS is a linear smoother, so deleting observation i changes its own fitted value in a way captured entirely by h_i; the n leave-one-out residuals can therefore be recovered *exactly* — not approximately — from one fit. No such identity exists for general nonlinear models.

Solution — Exercise 5.2 (2/2)

Solution

(3) Numerical contribution.

$$\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2 = \left(\frac{2.0}{1 - 0.5}\right)^2 = \left(\frac{2.0}{0.5}\right)^2 = 4.0^2 = 16.0.$$

The naive squared residual is $e_i^2 = 2.0^2 = 4.0$. The shortcut inflates this high-leverage point's contribution by the factor $1/(1-h_i)^2 = 1/0.25 = 4$, correctly reflecting that a high-leverage point strongly pulls the fitted line toward itself, so its genuine leave-one-out error is much larger than the in-sample residual suggests.
Interpretation. The in-sample residual understates this point's out-of-sample error by a factor of 4: remove the point and the fitted line springs back, so the prediction misses it far more badly than the training residual admits.

Common mistake

forgetting that $y_i - \hat{y}_i$ and h_i both come from the *full-data* fit (no refitting is involved), or dividing by $1 - h_i$ without squaring — the correction factor on the *squared* error is $1/(1 - h_i)^2 = 4$, not $1/(1 - h_i) = 2$.

Extended Exercise 5.1 — LOOCV Shortcut vs. 5-Fold CV [Math]

Extended exercise (15 min)

A simple linear regression (one predictor, so $p + 1 = 2$ parameters) is fit by OLS to $n = 200$ observations, giving a training residual sum of squares $\mathrm{RSS} = \sum_i e_i^2 = 760$.

1. **Leverage shortcut, per point.** Three representative observations have (residual e_i, leverage h_i): $A = (1.0,\ 0.02)$, $B = (3.0,\ 0.30)$, $C = (-2.0,\ 0.05)$. Compute each LOOCV contribution $\big(e_i/(1 - h_i)\big)^2$ and compare with the naive e_i^2.
2. **Leverage shortcut, whole sample.** The average OLS leverage is $\bar{h} = (p + 1)/n$. Using the approximation that all $h_i \approx \bar{h}$, estimate $\mathrm{CV}_{(n)} \approx \mathrm{MSE}_{\mathbf{train}}/(1 - \bar{h})^2$.
3. **5-fold CV.** A 5-fold run gives per-fold MSEs 4.0, 4.4, 3.8, 4.6, 4.2. Compute $\mathrm{CV}_{(5)}$ and its standard error across folds.
4. **Compare** the two estimates and their bias–variance properties. Which is cheaper *here*, and why does that not usually hold?

Solution — Extended Exercise 5.1 (1/2)

Solution

(1) Per-point leverage correction.

$$A: \left(\tfrac{1.0}{0.98}\right)^2 = 1.04, \quad B: \left(\tfrac{3.0}{0.70}\right)^2 = 18.37, \quad C: \left(\tfrac{-2.0}{0.95}\right)^2 = 4.43,$$

versus the naive $e_i^2 = 1.0,\ 9.0,\ 4.0$. The correction factor $1/(1-h_i)^2$ equals 1.04, 2.04, 1.11: the high-leverage point B is inflated most, because it pulls the fit strongly toward itself, so its true leave-one-out error far exceeds the in-sample residual.

(2) Whole-sample shortcut. $\bar{h} = (p+1)/n = 2/200 = 0.01$ and $\mathrm{MSE}_{\text{train}} = \mathrm{RSS}/n = 760/200 = 3.80$, hence

$$\mathrm{CV}_{(n)} \approx \frac{3.80}{(1-0.01)^2} = \frac{3.80}{0.9801} \approx 3.88.$$

A single OLS fit plus the hat-matrix diagonal replaces $n = 200$ fits. Note that 3.88 sits just *above* $\mathrm{MSE}_{\text{train}} = 3.80$: the $1/(1-\bar{h})^2$ factor always inflates the training error, undoing its optimism.

Solution — Extended Exercise 5.1 (2/2)

Solution

(3) 5-fold estimate. $\mathrm{CV}_{(5)} = \frac{1}{5}(4.0 + 4.4 + 3.8 + 4.6 + 4.2) = \frac{21.0}{5} = 4.20$. Deviations from 4.20 are $-0.2, 0.2, -0.4, 0.4, 0.0$; their squares sum to 0.40, so $s = \sqrt{0.40/4} = \sqrt{0.1} \approx 0.316$ and the standard error is $s/\sqrt{5} \approx 0.14$: $\mathrm{CV}_{(5)} \approx 4.20 \pm 0.14$.

(4) Comparison. Both estimates target the same test MSE and agree closely (3.88 vs. 4.20).

- *Bias*: LOOCV trains on 199 points (almost unbiased); 5-fold trains on 160, so it is slightly pessimistic — consistent with $4.20 > 3.88$.
- *Variance*: LOOCV averages n highly correlated fits (higher variance); 5-fold averages 5 weakly correlated fits (lower variance).
- *Cost*: here LOOCV is *cheaper* — the leverage shortcut is one fit versus five. This reversal is special to OLS/ridge; for a general model LOOCV costs n fits.

Common mistake

reading $3.88 < 4.20$ as "LOOCV is more accurate". Both target the same test MSE; the gap mostly reflects 5-fold's pessimistic bias (training on 160 rather than 199 points), not a defect of either estimate.