Quantitative Research Methods

Chapter 2: Statistical Learning

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
$Y = f(X) + \varepsilon$	general model: systematic part plus noise
$\sigma^2 = \mathrm{Var}(\varepsilon)$	noise variance; the irreducible error
$\hat{f}$	estimate of f learned from the data
$\mathrm{MSE} = \frac{1}{n}\sum_i (y_i - \hat{f}(x_i))^2$	mean squared error of a fit
$E\left[(y_0 - \hat{f}(x_0))^2\right]$	expected test MSE at a new point x_0
$\mathrm{Bias}(\hat{f}(x_0))$	$E[\hat{f}(x_0)] - f(x_0)$: systematic miss
$\mathrm{Var}(\hat{f}(x_0))$	variability of $\hat{f}$ over training sets
$\Pr(Y = j \mid X = x_0)$	conditional class probability (Bayes classifier)
Bayes error	$1 - E\left[\max_j \Pr(Y = j \mid X)\right]$: optimal error rate
K, $\mathcal{N}_0$	KNN: neighbour count; K nearest points to x_0
$I(\cdot)$	indicator: 1 if the condition holds, else 0

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 concept is introduced with motivation, intuition, formal definition, and worked examples. 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:

- **Define** the unknown function f and the irreducible error ε.
- **Decompose** prediction error into reducible and irreducible parts.
- **Distinguish** parametric from nonparametric methods, and discuss their trade-offs.
- **Articulate** the flexibility-interpretability trade-off.
- **Define** mean squared error; **distinguish** training MSE from test MSE.
- **Derive** and **interpret** the bias-variance decomposition.
- **Define** the Bayes classifier and the Bayes error rate.
- **Implement** and **interpret** the K-nearest-neighbours classifier.

Roadmap of this chapter

The conceptual journey

1. Define the unknown f and the irreducible noise ε.
2. Understand why we cannot do better than $\mathrm{Var}(\varepsilon)$.
3. Compare parametric and nonparametric methods.
4. Decompose error into bias, variance, and noise.
5. Move from regression to classification: the Bayes classifier.
6. Meet our first concrete method: KNN.

Where this chapter is used in industry

Sector	Concrete application	Which decision it drives
Telco, SaaS	Churn score per subscriber	A *ranking*: who gets a retention offer
Retail	Weekly demand forecast per article and store	The order quantity, and the spread that sets safety stock
Pricing	Price elasticity estimated for a coefficient, not a ranking	The list price a manager sets
E-commerce	Item–item nearest neighbours ("customers who bought this")	The products shown next to the basket
Marketing	Lookalike audiences: nearest neighbours of known good customers	Whom a campaign is targeted at
Payments, compliance	Nearest-neighbour matching for duplicates, entity resolution, fraud triage	Which cases a human reviewer sees

In industry — The common thread

Before picking a method, each row answers two questions from this chapter: do we need a *rank* or a *coefficient*, and how much of the variation is irreducible?

Industry case in depth: an item-to-item recommender

In industry — The learning problem: this chapter's own method

There is no coefficient to estimate: the "model" is a *nearest-neighbour lookup* in a similarity space built from co-purchase and co-view behaviour, or from learned item embeddings. K is simply how many neighbours are retrieved for the item on the page.

The fitted output — and who signs it off

A *ranked slate* of items, which fills the recommendation module on the product page or the home screen. Only the ordering matters — nothing in this model is interpretable, and nobody asks it to be. Merchandising and personalisation own the module, and the slate is judged by the category merchandiser who carries its revenue target — whose challenge is almost always "why is my range never shown?".

Honest caveats

Wide sparse behaviour vectors put the curse of dimensionality centre stage — hence embeddings and approximate nearest-neighbour search in production. Add popularity bias, cold start for new items, and a feedback loop: showing an item creates the very co-view data the next model trains on.

Outline

The fundamental problem of statistical learning

Almost every method in this course tries to do one thing.

The setup

We observe a quantitative response Y and p predictors $X = (X_1, \ldots, X_p)$, and assume there is some relationship

$$\boxed{Y = f(X) + \varepsilon,}$$

where f is fixed but *unknown* and ε is a random error term, independent of X, with $E[\varepsilon] = 0$.

Reading $Y = f(X) + \varepsilon$ symbol-by-symbol

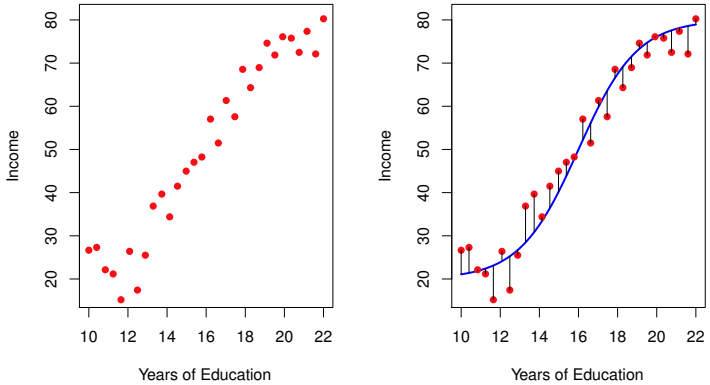
How to read this

Symbol	Meaning
Y	Response, what we want to predict / explain
$X = (X_1, \ldots, X_p)$	Vector of p predictors
f	The *systematic* information X provides about Y
ε	Error: everything in Y that X cannot explain
$E[\varepsilon] = 0$	By assumption, no systematic over/underprediction
$\mathrm{Var}(\varepsilon) = \sigma^2$	Noise floor

Takeaway

Statistical learning = a set of approaches to estimate f.

A simulated example: Income data



Income vs. years of education. The smooth curve added on the right is the *true f* — in real data we never see it.

Source: Figure 2.2 — ISLP, James et al. (2023).

Reducible vs. irreducible error

Suppose we estimate f by some $\hat{f}$ and predict $\hat{Y} = \hat{f}(X)$. Take expectation over a fixed X:

Decomposition

$$E\left[(Y - \hat{Y})^2 \mid X = x\right] = \underbrace{(f(x) - \hat{f}(x))^2}_{\text{reducible}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{irreducible}} .$$

Takeaway

Better data and methods can shrink the reducible part to zero. No method, however clever, can reduce $\mathrm{Var}(\varepsilon)$.

Why irreducible error exists

Even if we knew f exactly, ε may be non-zero because:

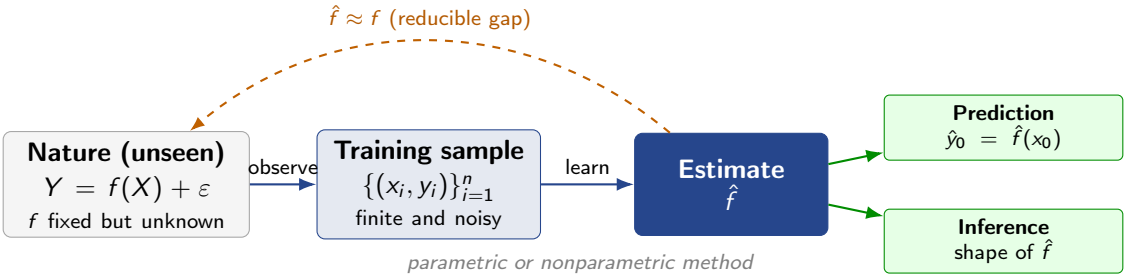
How to read this

- There are *unmeasured* variables that affect Y (luck, family connections, mood — not in our predictors).
- There are *unmeasurable* variables (instrument noise, rounding, sampling).
- Even with all variables, Y may be intrinsically stochastic.

Takeaway

A model that achieves test MSE close to $\mathrm{Var}(\varepsilon)$ is essentially as good as it can get. We should not be disappointed that we cannot do better.

Estimating f: from an unseen truth to a usable model



Takeaway

We never see f or ε — only a finite, noisy sample. A method turns that sample into $\hat{f}$; how close $\hat{f}$ lands to f is the *reducible* error, while ε sets an *irreducible* floor.

Outline

Two reasons to estimate f

In any application, you are doing one (or both) of:

Prediction

Given a new observation $X = x_0$, what value of Y should we expect?

Inference

How does Y depend on $X_1, \ldots, X_p$? Which predictors matter? Are the relationships linear?

Takeaway

Different goals favour different methods. A pure prediction problem often tolerates black-box models; an inference problem demands interpretability.

Worked example: prediction goal

Direct-mail marketing

A company has X = (demographic features of a customer) and wants to predict Y = likelihood the customer responds to a mailing.

- It does not matter *why* a customer responds.
- Whatever model gives the best test-set accuracy wins.
- Random forests or neural networks are perfectly fine here.

In industry — Telco and SaaS: churn scoring

A churn model only has to *rank* subscribers well: the retention team calls the top few per cent, and nobody asks why a customer is on the list — so a black box is acceptable. Rank badly and the discount budget goes to customers who would have stayed anyway.

Worked example: inference goal

Wage data

Researchers want to estimate the *return to education*: how much does an extra year of education raise wages, holding all else equal?

- A black-box prediction is useless — we need a coefficient.
- A linear-regression coefficient (with a confidence interval) answers the question directly.
- Random forests would predict accurately but tell us nothing about the magnitude of effects.

In industry — Pricing and pay equity: the coefficient *is* the deliverable

A price-elasticity model must hand over a signed, quantified effect — "a 1 % price rise costs us this much volume" — because a person sets the price from it. A pay-equity review is the same: only a model whose structure can be reviewed answers it.

Outline

Two strategies to estimate f

Parametric

Assume f has a particular functional form (e.g. linear) governed by a finite set of parameters θ. Estimate θ from data.

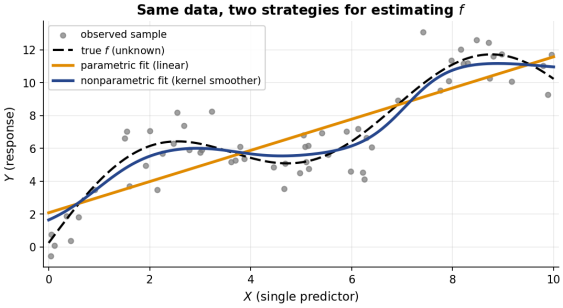
Nonparametric

Make no global assumption about the form of f. Let the data determine f as flexibly as possible.

Takeaway

These two strategies trade off bias against variance and interpretability against flexibility.

Parametric vs. nonparametric: a one-predictor illustration



Takeaway

The straight line (two parameters) *underfits* the curvature — misspecification bias. The kernel smoother assumes no fixed form and tracks f closely. Flexibility buys lower bias, at the cost of higher variance.

The parametric approach: two-step recipe

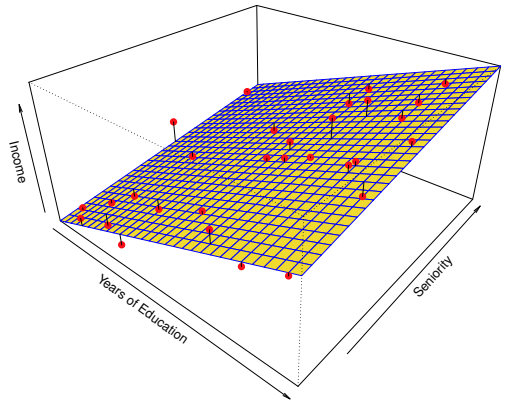
Recipe

1. Choose a functional form. For example, the linear form $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$.
2. Fit the model. Use a procedure (e.g. ordinary least squares) to estimate $\beta_0, \ldots, \beta_p$.

How to read this

This reduces estimating an infinite-dimensional function to estimating $p + 1$ numbers.

Linear fit on Income



The flat plane is the linear regression estimate. It captures the main increasing trends but misses the curvature — this is *bias* from a misspecified functional form.

Source: Figure 2.4 — ISLP, James et al. (2023).

Pros and cons of parametric methods

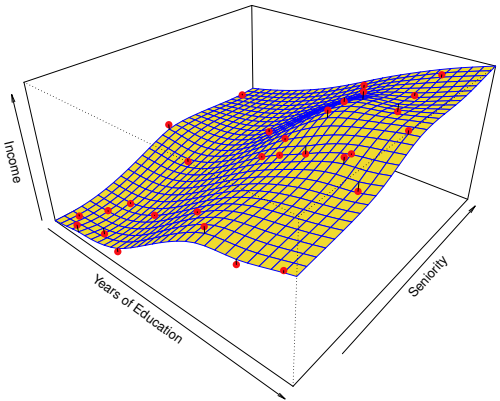
Pros

- Few parameters $\Rightarrow$ low variance.
- Interpretable: parameters *are* the model.
- Inference (SEs, p-values) well-developed.
- Computationally cheap.

Cons

- Bias if the true f is not in the assumed family.
- Sensitive to outliers when the form is wrong.
- Cannot capture interactions unless explicitly added.

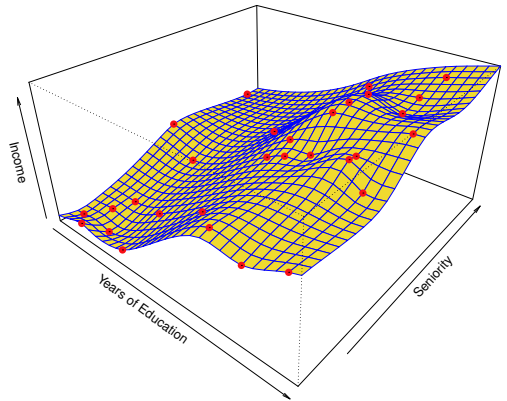
Nonparametric Income fit (smooth thin-plate spline)



The fitted surface is now curved, matching the true f much more closely. No misspecification bias — this is what you want when you do not know the functional form.

Source: Figure 2.5 — ISLP, James et al. (2023).

Nonparametric Income fit (rough thin-plate spline)



This surface passes through nearly every training point — training error is essentially zero. But the bumps respond to *noise*, not signal. Classical *overfitting*.

Source: Figure 2.6 — ISLP, James et al. (2023).

Exercise 2.1 — Parametric vs. Nonparametric [Concept]

Exercise

1. Define, in one sentence each, a *parametric* and a *nonparametric* method.
2. For each, does the number of parameters stay fixed in advance or grow with the sample size n?
3. Classify: (i) OLS linear regression, (ii) KNN, (iii) a fixed cubic polynomial regression, (iv) a thin-plate spline whose smoothing is chosen by cross-validation.
4. You have $n = 40$, $p = 1$; the scatter is clearly curved but you do not know its form. Which family would you choose, and what is the main risk of the *other* family here?

Solution — Exercise 2.1 (1/2)

Solution

1. **Parametric:** assume a fixed functional form for f governed by a finite parameter vector θ, then estimate θ. **Nonparametric:** make no global assumption about the form of f; let the data determine it, seeking a fit close to the points without being too rough.
2. Parametric: the number of parameters is *fixed in advance* (does not grow with n) — the family is chosen before seeing the data, so extra observations only pin down the same few components of θ more precisely. Nonparametric: effective complexity *grows with* n — with more data the fit is allowed to wiggle in more places (think of KNN, where every new training point can reshape the prediction in its neighbourhood).

Solution — Exercise 2.1 (2/2)

Solution

3. (i) OLS linear regression: **parametric** — the form $\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ is fixed, and fitting only estimates $p + 1$ numbers. (ii) KNN: **nonparametric** — no formula for f is written down; the prediction is a local vote that adapts to wherever the data lie. (iii) Fixed cubic polynomial: **parametric** — curved, but still a fixed form with four coefficients $\beta_0, \ldots, \beta_3$ chosen in advance. (iv) Thin-plate spline with CV-chosen smoothing: **nonparametric** — no fixed global form; the data, via cross-validation, determine the effective flexibility.
4. Prefer **nonparametric** (flexible), because a fixed form is unknown and a parametric guess risks large *bias from misspecification*. Caveat: with only $n = 40$ a very flexible fit risks high *variance*/overfitting — so pick the smoothing by cross-validation.

Common mistake

Equating "nonlinear" with "nonparametric". A cubic polynomial is curved yet fully parametric — what matters is whether the form and its parameter count are fixed before seeing the data.

Outline

The flexibility-interpretability trade-off

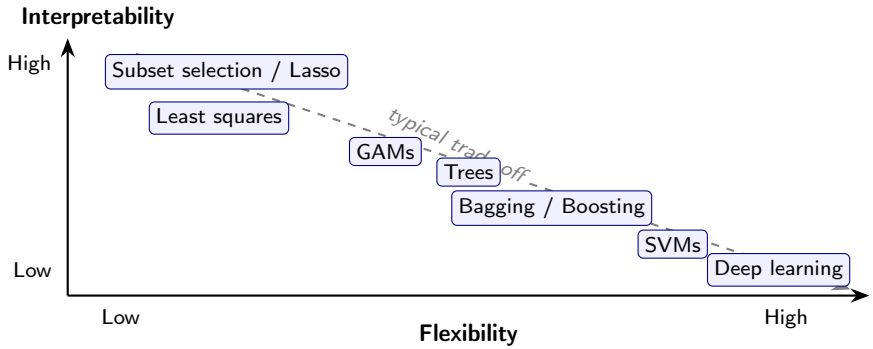
Methods can be ranked along a flexibility spectrum:

Method	Flexibility	Interpretability
Subset selection / Lasso	low	high
Linear regression	low	high
GAMs / splines	medium	medium
Trees	medium	medium
Bagging / random forests	high	low
Boosting	high	low
Deep learning	very high	very low

Takeaway

Reading top to bottom, flexibility rises while interpretability falls — the two trade off. Choose the *least* flexible method that still captures the signal.

The trade-off as a map: where each method sits



Takeaway

Every method buys flexibility by selling interpretability — there is no upper-right corner. Inference pulls you up-left; pure prediction slides you down the diagonal.

Why ever choose a less-flexible method?

Several good reasons:

Takeaway

- **Inference.** Coefficients of a linear regression have meaning; weights of a neural network usually don't.
- **Sample size.** With n small, flexible methods overfit.
- **Stability.** Simple models change little as the data change a little.
- **Communication.** A non-technical audience will not accept a black box.
- **Regulation.** Credit, health, and insurance often require interpretability by law.

Exercise 2.2 — Flexibility vs. Interpretability [Concept]

Exercise

1. Rank these by flexibility, low to high: linear regression, deep neural network, lasso, random forest, GAM/spline.
2. Give *two* distinct reasons to deliberately pick a *less*-flexible method even if a flexible one predicts a little better.
3. True or false, with reason: "More flexible always means lower test error."
4. A credit-scoring model must, by regulation, explain every decision. Which end of the spectrum, and name one suitable method.

Solution — Exercise 2.2 (1/2)

Solution

1. lasso $\approx$ linear regression (low) $<$ GAM/spline (medium) $<$ random forest (high) $<$ deep neural network (very high). Why this order: the lasso is a *constrained* linear model, so it can be at most as flexible as unpenalised least squares; a GAM/spline relaxes linearity to a smooth curve per predictor; a random forest adds interactions and abrupt jumps; a deep network can approximate essentially any function.
2. Any two of: *inference* (coefficients carry meaning); *small n* (flexible methods overfit); *stability* (simple models change little when the data change a little); *communication* to a non-technical audience; *regulation*/legal requirements. Each reason deliberately trades a little predictive accuracy for something the application values more.

Solution — Exercise 2.2 (2/2)

Solution

3. **False.** Test error is U-shaped in flexibility: past the optimum, variance grows faster than bias falls, so test error *rises* (overfitting). Extra flexibility helps only while systematic signal remains to be captured; beyond that point the model starts fitting noise. Only *training* error always falls as flexibility grows.
4. The **low-flexibility, interpretable** end — e.g. a (penalised) logistic/linear regression or a small decision tree, whose decisions can be read off directly. Coefficients (or a short path of splits) provide the per-case justification the regulation demands.

Common mistake

Reading the flexibility ranking in part 1 as a quality ranking. The U-shaped test error means the best method depends on the data and the goal — "more flexible" is not "better" by default.

Outline

Supervised vs. unsupervised learning

Supervised

For each observation $i = 1, \ldots, n$ we have predictors x_i and a corresponding response y_i.

Goal: learn the map $X \mapsto Y$.

Unsupervised

For each observation we have only x_i, no response y_i.

Goal: discover structure in X.

Takeaway

Most of this course focuses on supervised learning. Chapter 12 returns to the unsupervised case.

Outline

Regression vs. classification

The type of Y determines the problem type:

How to read this

- Quantitative $Y \Rightarrow$ *regression problem*.
- Qualitative $Y \Rightarrow$ *classification problem*.

Common confusion

Whether you have a regression or a classification problem depends on Y, *not* on the X_j. A categorical predictor in a regression on a numeric response is still a regression.

Outline

Mean squared error: the central regression metric

Definition

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{f}(x_i)\big)^2.$$

Reading the formula

$y_i - \hat{f}(x_i)$ is the *residual* for point i (truth minus prediction); square it so over- and under-shoots both count and big misses hurt more; then average over all n points. Squaring makes MSE convex and differentiable — friendly to optimisation — and ties it historically to Gaussian errors.

Training MSE vs. test MSE

Training MSE

MSE on the data used to fit the model. Always *over-optimistic* — the model has been bent to fit these points.

Test MSE

MSE on *new*, held-out observations. The honest measure of predictive performance.

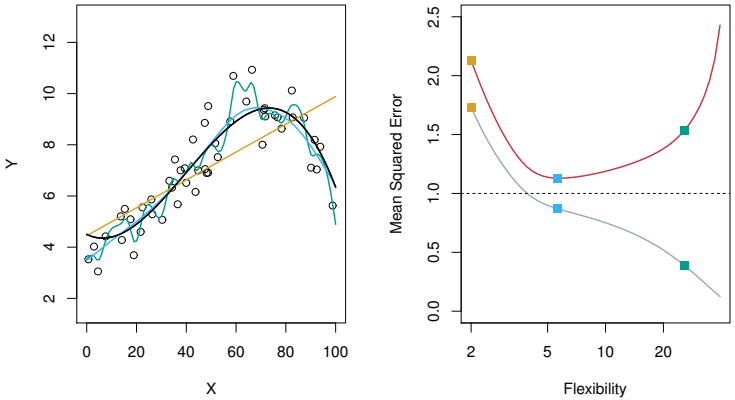
The cardinal sin

Training MSE is essentially never what we care about. We minimise training MSE in order to make test MSE small — not as a goal in itself.

In industry — Procurement: buying a model from a vendor

An accuracy figure quoted on the data the model was fitted to is worth nothing. Ask for performance on a *later* period than the one used for fitting — out-of-time validation is exactly what a model risk function will demand before the model goes live.

Simulated data: candidate fits and their train/test MSE



Source: Figure 2.9 — ISLP, James et al. (2023).

Reading the simulation

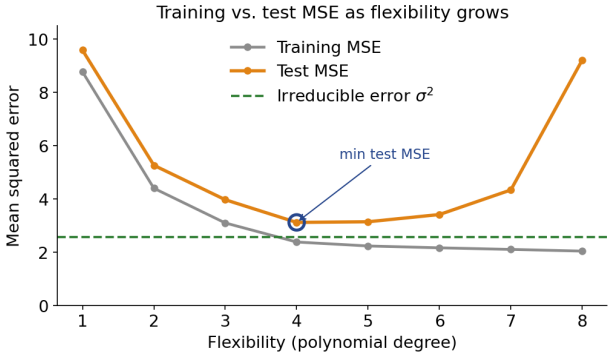
How to read this

Left: simulated data (open circles), the true f (the smooth black curve), and three fits: a straight line (orange), a moderate spline that almost coincides with the truth (blue), and a very flexible spline that wiggles through the noise (green).
Right: the training MSE is the curve that only falls; the test MSE is the U-shaped one.

Three observations

- Training MSE *always* decreases as flexibility grows.
- Test MSE has a U-shape: first decreases, then increases.
- The optimum is where the U bottoms out — which we cannot read off the training MSE.

Simulating the U-shape: training vs. test MSE



Takeaway

Fitting polynomials of growing degree: training MSE falls monotonically while test MSE is U-shaped, minimised at moderate flexibility.

Exercise 2.3 — Why Training MSE Understates Test MSE [Concept]

Exercise

1. Explain in words why, for the same model, training MSE is almost always *smaller* than test MSE.
2. As flexibility increases, describe the behaviour of (i) training MSE and (ii) test MSE.
3. A student fits a degree-15 polynomial to 20 points and gets training MSE ≈ 0, concluding the model is excellent. What is wrong, and what should they report instead?
4. Define *overfitting* in one sentence, in terms of training vs. test error.

Solution — Exercise 2.3 (1/2)

Solution

1. The parameters are chosen to *minimise error on the training data*, so the fit adapts to that sample's specific noise. New observations carry different noise the model has never seen, so error on them is larger. Training error is therefore an *optimistically biased* estimate of test error — the model is graded on the very questions it was tuned to answer.
2. (i) Training MSE decreases *monotonically* as flexibility grows: a more flexible family contains the less flexible fits, so it can only match the training points at least as well. (ii) Test MSE is *U-shaped*: it falls while added flexibility captures genuine signal (bias shrinks), reaches a minimum, then rises once the model starts chasing noise (variance takes over).

Solution — Exercise 2.3 (2/2)

Solution

3. Near-zero training error only means the flexible polynomial *interpolates* the 20 points, noise included — a degree-15 polynomial has 16 coefficients, nearly one per observation, so it can pass through almost every point by construction. That says nothing about new data. They should report **test MSE** on held-out data (or a cross-validated error), which would very likely be far larger than the training MSE.
4. **Overfitting:** when a model has low training error but high test error because it has fit noise rather than signal.

Common mistake

Treating zero training error as proof that the model has "learned the data". It has *memorised* them — interpolation is not generalisation, and only performance on unseen data counts.

Outline

The bias-variance decomposition

Fix a test point x_0 and average over training sets:

Decomposition

$$\boxed{E\left[(y_0 - \hat{f}(x_0))^2\right] = \mathrm{Var}(\hat{f}(x_0)) + \left[\mathrm{Bias}(\hat{f}(x_0))\right]^2 + \mathrm{Var}(\varepsilon).}$$

Reading the three pieces

How to read this

Piece	Meaning
$\mathrm{Var}(\hat{f}(x_0))$	Variability of the estimate across training sets
$\mathrm{Bias}(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0)$	Systematic error from a too-simple model
$\mathrm{Var}(\varepsilon) = \sigma^2$	Irreducible noise (the floor)

Takeaway

Every prediction error has these three sources. The art of statistical learning is balancing the first two.

What is bias?

Definition

Bias is the error introduced by approximating a complicated reality by a simpler model.

Worked example

A linear model has high bias if the truth is curved. A constant model (predicting the global mean) has very high bias unless Y is essentially flat in X.

Takeaway

Bias *decreases* as a method becomes more flexible.

What is variance?

Definition

Variance is how much $\hat{f}$ would change if we re-fit it on a different training set drawn from the same distribution.

Worked example

A flexible method bends to whatever data it sees, so two training sets give very different fits. A linear model is much more stable.

Takeaway

Variance *increases* as a method becomes more flexible.

Bias and variance trade off

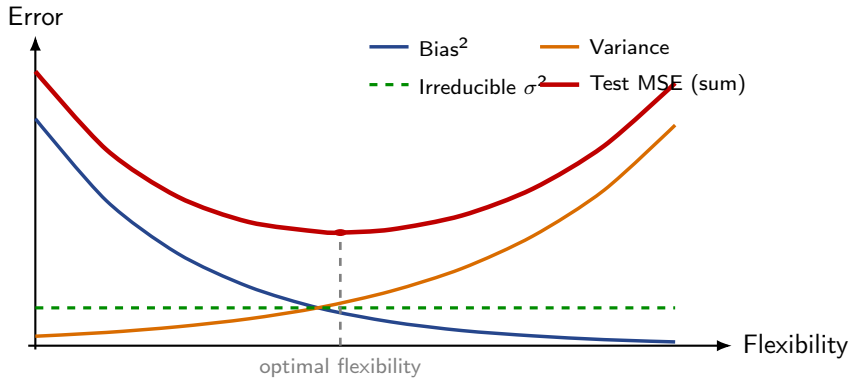
As we increase flexibility:

- Bias goes *down*.
- Variance goes *up*.
- Their sum is a U-shape: the bias-variance trade-off.
- The optimal model lies at the bottom of the U.

In industry — Asset management: why parsimony wins

In settings where the signal is tiny relative to the noise — asset returns are the textbook case — flexible models mostly fit noise, so the variance term dominates and small, heavily constrained models routinely beat them out of sample. The same logic explains why a few well-chosen predictors often outperform a large automated feature set.

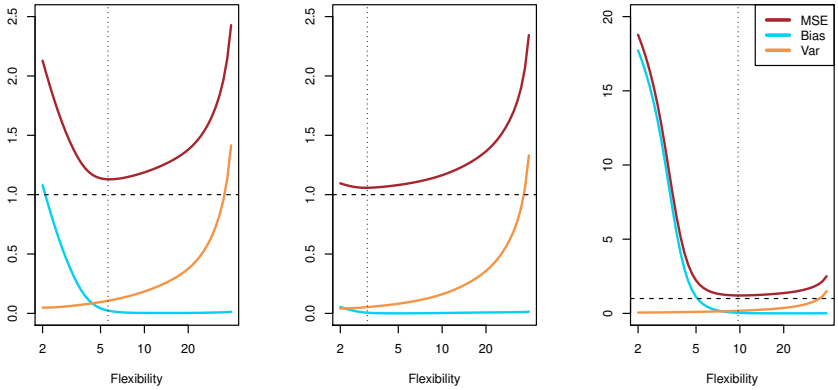
The bias-variance trade-off, schematically



Takeaway

Test MSE $=$ bias$^2+$ variance $+\sigma^2$; it bottoms out where falling bias meets rising variance.

Bias, variance, and test MSE across flexibility



The curve that falls is the squared bias (*Bias* in the legend); the one that rises is the variance (*Var*); their sum plus σ^2 is the U-shaped test MSE (*MSE*). Horizontal dashed line: the irreducible σ^2. Vertical dotted line: the flexibility that minimises test MSE.

Source: Figure 2.12 — ISLP, James et al. (2023).

Mini-check: bias and variance

Match each statement to bias or variance

1. "A linear model on a curved truth." (?)
2. "A KNN classifier with $K = 1$." (?)
3. "Refitting on a different training sample changes the answer a lot." (?)
4. "The model is too simple to capture the relationship." (?)

Answers

1. high bias. 2. high variance. 3. high variance. 4. high bias.

Exercise 2.4 — Bias-Variance Decomposition (numbers) [Math]

Exercise

The expected test MSE at a point x_0 decomposes as

$$E\big[(y_0 - \hat{f}(x_0))^2\big] = \mathrm{Var}(\hat{f}(x_0)) + \big[\mathrm{Bias}(\hat{f}(x_0))\big]^2 + \mathrm{Var}(\varepsilon).$$

1. Which term(s) are reducible and which is irreducible?
2. At x_0: over many training sets $E[\hat{f}(x_0)] = 9.0$ while the truth $f(x_0) = 10.0$; $\mathrm{Var}(\hat{f}(x_0)) = 1.5$; $\sigma^2 = \mathrm{Var}(\varepsilon) = 2.0$. Compute the bias, the squared bias, and the expected test MSE.
3. A more flexible method has $\mathrm{Var}(\hat{f}) = 4.0$ and $\mathrm{Bias} = -0.2$ at x_0. Which method wins at x_0?
4. What is the smallest possible expected test MSE at x_0, and why?

Solution — Exercise 2.4 (1/2)

Solution

1. $\mathrm{Var}(\hat{f})$ and Bias^2 are **reducible**: both depend on the estimator $\hat{f}$, so a better method, more data, or better-tuned flexibility can shrink them. $\mathrm{Var}(\varepsilon) = \sigma^2$ is **irreducible**: it is a property of the data-generating process itself, and no choice of $\hat{f}$ can touch it.
2. The decomposition applies term by term, so we substitute the given quantities. First the bias, from its definition:

$$\mathrm{Bias} = E[\hat{f}] - f = 9.0 - 10.0 = -1.0, \qquad \mathrm{Bias}^2 = (-1.0)^2 = 1.0.$$

Then the expected test MSE is the sum of the three pieces:

$$\mathrm{MSE} = \underbrace{1.5}_{\mathrm{Var}} + \underbrace{1.0}_{\mathrm{Bias}^2} + \underbrace{2.0}_{\sigma^2} = 4.5.$$

Interpretation: almost half of the total (2.0 of 4.5) is noise that no model could ever remove.

Solution — Exercise 2.4 (2/2)

Solution

3. Same recipe for the flexible method: $\mathrm{Bias}^2 = (-0.2)^2 = 0.04$, so MSE $= 4.0 + 0.04 + 2.0 = 6.04$. Since $4.5 < 6.04$, the *less*-flexible method (Ex. part 2) wins here: its lower variance more than offsets its larger bias. In numbers, flexibility cut the squared bias by 0.96 (from 1.00 to 0.04) but raised the variance by 2.5 (from 1.5 to 4.0) — a bad trade at this x_0.
4. The floor is $\sigma^2 = 2.0$, reached only if both variance and squared bias are zero. No method can beat the irreducible noise $\mathrm{Var}(\varepsilon)$: even the true f, known exactly, would still miss new observations by their noise.

Common mistake

Forgetting to square the bias (adding -1.0 or -0.2 directly), or expecting a negative bias to *lower* the MSE. Only Bias^2 enters, so the sign is irrelevant — systematic over- and under-prediction hurt equally.

Exercise 2.5 — Reading the U-shaped Test-Error Curve [Concept]

Exercise

A plot has flexibility on the x-axis and error on the y-axis. It shows a monotonically decreasing curve, a U-shaped curve, and a horizontal dashed line at the bottom.

1. Which curve is training error, which is test error, and what does the dashed line represent?
2. Label the regions left and right of the U's minimum as underfitting or overfitting, and say whether bias or variance dominates in each.
3. Why can the optimal flexibility *not* be found from the training-error curve alone?
4. In practice, how do we locate the test-error minimum without a large held-out set?

Solution — Exercise 2.5 (1/2)

Solution

1. Monotonically decreasing = **training error**: extra flexibility always lets the model fit the training points at least as well. U-shaped = **test error**: it improves while genuine signal is being captured, then worsens. Dashed line = **irreducible error** $\mathrm{Var}(\varepsilon) = \sigma^2$ — the noise floor that expected test error can approach but never fall below.
2. Left of the minimum: **underfitting**, *bias* dominates (model too simple) — systematic structure is being missed, and both curves are still high. Right of the minimum: **overfitting**, *variance* dominates (model too flexible) — the fit tracks the training noise, so training error keeps falling while test error climbs.

Solution — Exercise 2.5 (2/2)

Solution

3. Training error keeps falling with flexibility and never turns up, so it gives no signal about the optimum — it is over-optimistic, and by construction it always prefers the most flexible model on offer. Choosing flexibility by training error would therefore always land at the overfitting end.
4. **Cross-validation** (e.g. k-fold with $k = 5$ or 10): hold out folds in turn, average the validation error across folds, and pick the flexibility that minimises it. This *estimates* the test-error curve from the training data alone, without sacrificing a large held-out set.

Common mistake

Picking the flexibility where the *training*-error curve flattens out. The training curve contains no information about the U's minimum — only validation or test error does.

Outline

Switching to classification

In the classification setting, y_i is qualitative.

How to read this

MSE no longer makes sense (we cannot meaningfully square the difference between "red" and "blue"). We need a notion of accuracy and an analogue of f.

The plan

Replace squared error with the *error rate*, and the regression function with the *class probabilities* $p_k(x) = \Pr(Y = k \mid X = x)$. These lead to the Bayes classifier and to KNN.

Conditional class probabilities

Define

$$p_k(x) := \Pr(Y = k \mid X = x) \quad \text{for each class } k.$$

How to read this

Read $p_k(x)$ as: among all cases whose predictors equal x, the fraction whose true class is k. At each x the $p_k(x)$ sum to 1 across the classes.

Takeaway

These probabilities are the classification analogue of f. Estimating them well is the central task of classification. Logistic regression, LDA/QDA, naive Bayes, KNN, trees, random forests, neural networks all do this — in different ways.

The Bayes classifier

Definition

For each x, predict the class with the largest conditional probability:

$$\hat{y}_{\text{Bayes}}(x) = \arg\max_k p_k(x).$$

Takeaway

The Bayes classifier achieves the minimum possible test error among all classifiers. It requires knowing the true $p_k(x)$ — which we never do. So the Bayes classifier is a *benchmark*, not an algorithm.

The Bayes error rate

At each x, the Bayes error rate is $1 - \max_k p_k(x)$. Averaging over x:

$$\text{Bayes error} = 1 - E\left[\max_k p_k(X)\right].$$

How to read this

This is the irreducible error of classification — analogous to $\mathrm{Var}(\varepsilon)$ in regression. In well-separated classes the Bayes error is near zero; in heavily overlapping classes it can be near 50 % for two roughly balanced classes.

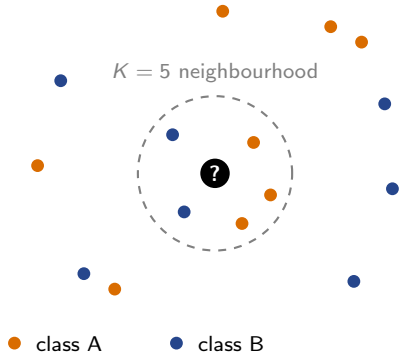
K-nearest neighbours: definition

Algorithm

Given an integer $K \geq 1$ and a test point x_0:

1. Identify the K training points closest to x_0 (by Euclidean distance). Call this set $\mathcal{N}_0$.
2. Estimate
$$\hat{\Pr}(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$
3. Classify x_0 to the class with the largest $\hat{\Pr}$.

How KNN votes: the neighbourhood of a query point

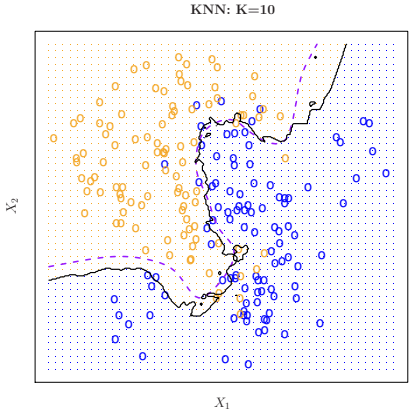


3 orange vs. 2 blue
$\Rightarrow$ **majority vote**:
predict orange

Takeaway

KNN looks at the K closest training points and predicts their majority class — here 3 orange beat 2 blue.

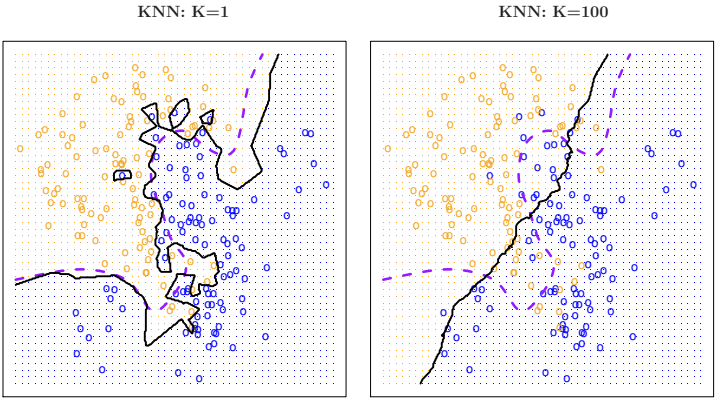
KNN with $K = 10$ vs. Bayes optimum



The KNN decision boundary (black) closely follows the Bayes boundary (purple dashed) — with a well-chosen K.

Source: Figure 2.15 — ISLP, James et al. (2023).

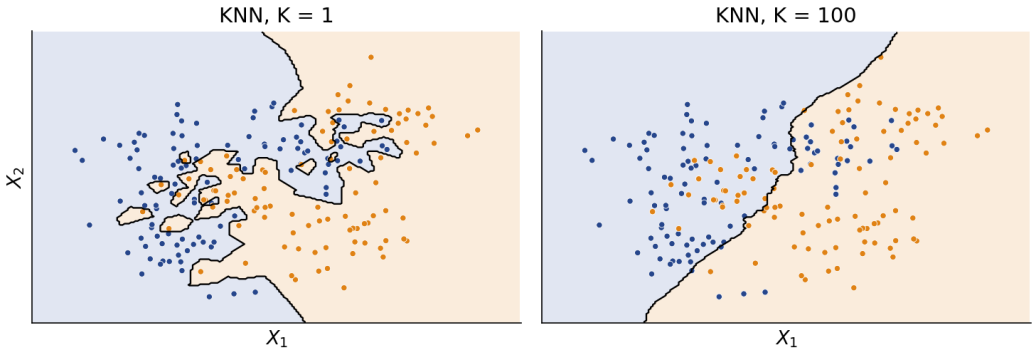
KNN with $K = 1$ (left) and $K = 100$ (right)



$K = 1$: each training point gets its own little island, so the boundary is extremely jagged — *low bias, very high variance*; training error is zero but test error is poor. $K = 100$: an almost straight boundary that smooths the real structure away — *low variance, high bias*.

Source: Figure 2.16 — ISLP, James et al. (2023).

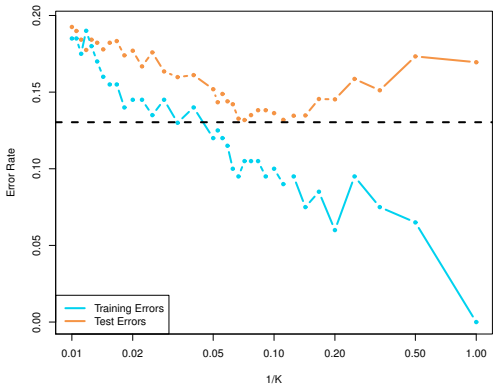
KNN decision boundaries: $K = 1$ vs. $K = 100$



Takeaway

Small K yields a jagged, high-variance boundary that overfits; large K yields a smooth, high-bias boundary — the same trade-off, seen geometrically.

KNN error vs. $1/K$



Training error decreases monotonically; test error is U-shaped with a minimum at moderate K. Cross-validation lets us pick K in practice.

Source: Figure 2.17 — ISLP, James et al. (2023).

Choosing K in practice

Practical advice

- Small $K \Rightarrow$ low bias, high variance, jagged boundary.
- Large $K \Rightarrow$ high bias, low variance, smooth boundary.
- Try $K \in \{1, 3, 5, 10, 25, 50, 100\}$ and pick by cross-validation.
- **Standardise predictors first** — KNN is sensitive to scaling.

In industry — E-commerce and marketing: where nearest neighbours run

Item–item "customers who bought this" recommenders, lookalike audience targeting, duplicate and entity resolution, and k-NN over embeddings for vector search and retrieval. The curse of dimensionality is the practical catch: run naively over a wide customer table, the "nearest" neighbours are not near, so firms reduce dimension or learn embeddings first.

Exercise 2.6 — Bayes Classifier and Bayes Error Rate [Math]

Exercise

A binary problem (classes A, B). X takes three values with the following marginal and true conditional probabilities:

x	$\Pr(X = x)$	$p_A(x) = \Pr(A \mid x)$	$p_B(x) = \Pr(B \mid x)$
1	0.5	0.8	0.2
2	0.3	0.4	0.6
3	0.2	0.5	0.5

1. State the Bayes classifier's prediction at each x.
2. Give the Bayes error at each x, $1 - \max_k p_k(x)$.
3. Compute the *overall* Bayes error rate.
4. Why is the Bayes error rate usually unattainable in practice?

Solution — Exercise 2.6 (1/2)

Solution

Method. At each x the Bayes classifier predicts $\arg\max_k p_k(x)$ — the majority class among cases with that x — because any other choice would err more often at that x.

1. Compare the two conditional probabilities at each x: $x = 1 \to$ **A** ($0.8 > 0.2$); $x = 2 \to$ **B** ($0.6 > 0.4$); $x = 3 \to$ tie ($0.5/0.5$), predict either — the error is 0.5 either way.
2. Local Bayes error $= 1 - \max_k p_k$, i.e. the probability mass of the *losing* class: at $x = 1$, $1 - 0.8 = 0.2$; at $x = 2$, $1 - 0.6 = 0.4$; at $x = 3$, $1 - 0.5 = 0.5$.

Solution — Exercise 2.6 (2/2)

Solution

3. The overall error is the *expectation* of the local error over the distribution of X (law of total probability), so we weight each local error by $\Pr(X = x)$:

$$0.5(0.2) + 0.3(0.4) + 0.2(0.5) = 0.10 + 0.12 + 0.10 = 0.32.$$

   Interpretation: even the *optimal* rule is wrong on 32 % of cases — the classes overlap heavily, so this is an intrinsically hard problem.

4. It requires the *true* conditional probabilities $p_k(x)$, which are unknown; a real classifier only *estimates* them from data, so it does at least as badly. The Bayes error is the classification analogue of the irreducible error $\mathrm{Var}(\varepsilon)$.

Common mistake

Averaging the three local errors without the weights, $(0.2 + 0.4 + 0.5)/3 \approx 0.367$ — wrong, because the three x-values are not equally likely ($x = 1$ occurs half the time).

Exercise 2.7 — KNN by Hand, $K = 1$ and $K = 3$ [Math]

Exercise

Six labelled training points in 2-D:

Point	X_1	X_2	Class
P_1	2	4	Blue
P_2	1	2	Red
P_3	3	2	Red
P_4	4	4	Blue
P_5	1	5	Blue
P_6	4	1	Red

Classify the query $x_0 = (2, 3)$ with Euclidean distance for $K = 1$ and $K = 3$. Show the distances and the estimated class probabilities.

Solution — Exercise 2.7 (1/2)

Solution

Method. KNN ranks the training points by Euclidean distance $d = \sqrt{(X_1 - 2)^2 + (X_2 - 3)^2}$ to the query $x_0 = (2, 3)$. Because the square root is monotone, ranking by d^2 gives the same neighbours — e.g. for $P_2(1, 2)$: $d^2 = (1-2)^2 + (2-3)^2 = 1 + 1 = 2$. Squared distances for all points (then $d = \sqrt{\cdot}$):

Point	d^2	d
$P_1(2, 4)$	$0 + 1 = 1$	1.000
$P_2(1, 2)$	$1 + 1 = 2$	1.414
$P_3(3, 2)$	$1 + 1 = 2$	1.414
$P_5(1, 5)$	$1 + 4 = 5$	2.236
$P_4(4, 4)$	$4 + 1 = 5$	2.236
$P_6(4, 1)$	$4 + 4 = 8$	2.828

$K = 1$: the single nearest neighbour is P_1 (Blue) at $d = 1.0$, so $\hat{\Pr}(\text{Blue}) = 1$ and we predict **Blue**.

Solution — Exercise 2.7 (2/2)

Solution

$K = 3$: the three nearest are P_1 (Blue), P_2 (Red), P_3 (Red). The estimated class probabilities are the vote shares $\hat{\Pr}(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$: votes Red $= 2$, Blue $= 1$, so $\hat{\Pr}(\text{Red}) = \frac{2}{3}$, $\hat{\Pr}(\text{Blue}) = \frac{1}{3}$; we predict **Red**.

Lesson: enlarging K averages over a bigger neighbourhood and can *flip and smooth* the prediction — here from Blue to Red. $K = 1$ trusts a single point (low bias, high variance); $K = 3$ pools three points (more stable, but pulled by farther neighbours).

Common mistake

Mixing scales when ranking — comparing one point's d^2 with another point's d. Rank every point on the same scale; d and d^2 give the identical ordering, so the square roots are only needed if the distances themselves are reported.

Extended Exercise 2.2 — KNN simulation study [Python]

Extended exercise (15 min)

Build a controlled experiment that exposes the bias–variance trade-off in KNN.

1. Simulate a 2-class 2-D problem: class 0 from $\mathcal{N}((0,0), I)$ and class 1 from $\mathcal{N}((2,2), I)$. Draw $n_{\text{train}} = 200$ and a large $n_{\text{test}} = 2000$ (half per class).
2. For $K = 1, 2, \ldots, 50$ fit a KNN classifier and record the *training* and *test* error rates.
3. Plot both errors against $1/K$ (flexibility) and identify the K that minimises test error.
4. Explain each curve via bias and variance, and compare the best test error to the Bayes rate $\Phi(-\sqrt{2}) \approx 0.079$ (means $2\sqrt{2}$ apart, unit variance).

Provide runnable code and state the expected qualitative result.

Solution — Extended Exercise 2.2 (1/2)

Solution

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
rng = np.random.default_rng(1) # seeded RNG (reproducible)
def sample(n): # draw n points, half per class
    n0 = n // 2
    X = np.vstack([rng.normal([0,0], 1, (n0, 2)), # class 0: centre (0,0)
                   rng.normal([2,2], 1, (n-n0, 2))]) # class 1: centre (2,2)
    return X, np.r_[np.zeros(n0), np.ones(n-n0)] # features, labels 0/1
Xtr, ytr = sample(200); Xte, yte = sample(2000) # training and test sets
tr, te = [], []
for K in range(1, 51): # fit KNN for K = 1..50
    m = KNeighborsClassifier(K).fit(Xtr, ytr)
    tr.append(1 - m.score(Xtr, ytr)); te.append(1 - m.score(Xte, yte)) # err = 1 - acc
print("best K =", int(np.argmin(te)) + 1, "min test err =", round(min(te), 3))
```

Solution — Extended Exercise 2.2 (2/2)

Solution

Plot the two error curves against flexibility $1/K$:

```
import matplotlib.pyplot as plt
inv = [1/K for K in range(1, 51)] # x-axis: flexibility = 1/K
# draw the training and test error curves on one plot
plt.plot(inv, tr, label="train"); plt.plot(inv, te, label="test")
plt.xlabel("1/K"); plt.ylabel("error rate"); plt.legend(); plt.show()
```

Expected result (this seed). Best $K \approx 14$ with test error ≈ 0.076 — essentially the Bayes floor ≈ 0.079.

- **Training error** climbs from 0 at $K = 1$ (each point is its own neighbour) to ≈ 0.035; it does *not* track test error.
- **Test error** is U-shaped in $1/K$: worst at $K = 1$ (≈ 0.093, high *variance*), dips near $K \approx 10$–20, then drifts up for very large K (rising *bias*).

Small K = flexible = low bias, high variance; large K = rigid = high bias, low variance. Cross-validation would pick K in the flat basin at the bottom.

Common mistake: over-trusting the exact winner $K = 14$. The basin is flat — any K in 10–20 is statistically equivalent, and the training curve (0 at $K = 1$ by construction) is no evidence at all.

Outline

Python lab — run it live in the notebook

Companion notebook: `chapter_02_lab.ipynb`

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

- §§1–3 *NumPy refresher*, *pandas* and *Plots*;
- §4 *Reproducing Figure 2.9 — bias–variance*;
- §5 *KNN classification*.

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

Python lab: setup

```
import numpy as np # arrays and linear algebra
import pandas as pd # data frames (tables)
import matplotlib.pyplot as plt # plotting
from ISLP import load_data # the book's datasets

rng = np.random.default_rng(2024) # seeded random generator
plt.rcParams['figure.dpi'] = 110 # sharper inline figures
```

Always seed your RNG for reproducible plots.

NumPy basics

```
x = np.array([3, 4, 5]) # 1-D array (a vector)
y = np.array([4, 9, 7])

x.sum(), x.mean(), x.std(ddof=1) # basic stats (sample sd)
A = np.array([[1, 2], [3, 4]]) # 2 x 2 matrix
A.shape, A.T, np.linalg.inv(A), A @ A # shape, transpose, inverse, product
```

KNN classification

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

Default = load_data("Default") # credit-card default data
y = (Default["default"] == "Yes").astype(int).values # 1 = default
X = Default[["balance", "income"]].values # two numeric predictors
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.3,
                                      random_state=0) # 70/30 split
scaler = StandardScaler().fit(Xtr) # KNN needs scaled features
# fit 10-NN on the standardised training data:
knn = KNeighborsClassifier(n_neighbors=10).fit(scaler.transform(Xtr), ytr)
print(knn.score(scaler.transform(Xte), yte)) # test-set accuracy
```

Exercise 2.8 — Scatter and Flexibility on Auto [Python]

Exercise

Load the `Auto` data, make a scatter plot of `mpg` (y-axis) against `horsepower` (x-axis), and comment on what level of *flexibility* a good model of this relationship needs. Note that `horsepower` contains a few "?" missing entries you must handle.

Solution — Exercise 2.8 (1/2)

Solution

```
import pandas as pd
import matplotlib.pyplot as plt

Auto = pd.read_csv("../ALL CSV FILES - 2nd Edition/Auto.csv")
Auto = Auto[Auto["horsepower"] != "?"].copy() # drop 5 missing
Auto["horsepower"] = Auto["horsepower"].astype(float) # str -> num

plt.scatter(Auto["horsepower"], Auto["mpg"], s=10) # small dots
plt.xlabel("horsepower"); plt.ylabel("mpg")
plt.show()

print(Auto.shape[0]) # 392
print(Auto["horsepower"].corr(Auto["mpg"])) # about -0.78
```

Solution — Exercise 2.8 (2/2)

Solution

Interpretation. After dropping the 5 missing rows, $n = 392$. The cloud is clearly *nonlinear*: `mpg` falls steeply as `horsepower` rises and then flattens (a convex, decreasing curve; correlation ≈ -0.78). A straight-line fit would *underfit* this curvature (high bias); a *moderately* flexible fit — a quadratic/polynomial or a spline — captures it well; an extremely flexible fit that wiggles through every point would *overfit* (high variance).
Why the cleaning steps matter. The "?" entries make pandas read `horsepower` as *strings* (dtype `object`); the filter removes those rows and `astype(float)` restores a numeric column — only then are the scatter plot, the correlation, and any later fit meaningful.

Common mistake

Skipping the conversion (the correlation fails or is computed on nonsense) — and note that a strong correlation does *not* certify a linear fit: -0.78 measures linear association, yet the visible curvature still calls for a nonlinear term.

Extended Exercise 2.4 — Linear vs KNN: which wins when? [Integrative]

Extended exercise (15 min)

Six 1-D training points (x, y):

$$(1, 1.0)\ (2, 1.8)\ (3, 3.3)\ (4, 3.9)\ (5, 5.2)\ (6, 5.8).$$

An OLS line fitted to them is $\hat{y} = 0.02 + 0.994\,x$. Query point $x_0 = 3.2$.

1. **KNN regression by hand.** List the distances $|x_i - x_0|$; predict $\hat{y}(x_0)$ for $K = 1$ and $K = 3$ (a KNN regressor *averages* the K nearest responses).
2. Predict $\hat{y}(x_0)$ from the linear model. Are the three predictions close here, and why?
3. For each case name the expected winner (parametric *linear* vs nonparametric *KNN*) with a reason: (a) true f linear; (b) true f strongly nonlinear with large n; (c) p large (say 20) with modest n.
4. Give a one-sentence rule of thumb for choosing between them.

Solution — Extended Exercise 2.4 (1/2)

Solution

1. KNN by hand at $x_0 = 3.2$ (points sorted by distance):

x_i	3	4	2	5	1	6
$\lvert x_i - x_0 \rvert$	0.2	0.8	1.2	1.8	2.2	2.8
y_i	3.3	3.9	1.8	5.2	1.0	5.8

$K = 1$: nearest is $x = 3$, so $\hat{y}(x_0) = 3.3$. $K = 3$: nearest are $x = 3, 4, 2$ with $y = 3.3, 3.9, 1.8$, so $\hat{y}(x_0) = \frac{3.3+3.9+1.8}{3} = 3.0$.

2. Linear model. $\hat{y}(x_0) = 0.02 + 0.994(3.2) = 3.20$. All three (3.30, 3.00, 3.20) are close *because the data are nearly linear*: a global line and a local average agree when f is smooth and roughly straight. ($K = 3$ dips a little, pulled down by the far point $x = 2$.)

Solution — Extended Exercise 2.4 (2/2)

Solution

3. Who wins in general?

- **(a) True f linear $\Rightarrow$ linear wins.** It is correctly specified (no bias) and low-variance (few parameters, uses all n points); KNN only adds variance by averaging K local points and is biased near the edges.
- **(b) Strongly nonlinear f, large n $\Rightarrow$ KNN wins.** The linear model carries irremovable *misspecification bias*; with large n the neighbourhoods are dense, so KNN's variance is small and its flexibility captures the curvature.
- **(c) Large p (≈ 20), modest n $\Rightarrow$ linear wins.** *Curse of dimensionality*: in high dimensions the "nearest" neighbours are actually far, so KNN averages dissimilar points. A parametric model that assumes structure is far more data-efficient.

4. Rule of thumb. Prefer nonparametric KNN when f is unknown/curved, p is small, and n is large; prefer the parametric linear model when n is small, p is large, the functional form is trustworthy, or you need interpretability.

Common mistake: judging KNN by its training error — with $K = 1$ that is exactly zero on *every* data set. Only the test error can compare these models.

Outline

Chapter 2 in one slide

What we accomplished

Established the formal framework: the unknown f, irreducible error ε, and the core trade-offs.

- Decomposed prediction error into reducible + irreducible.
- Compared parametric (linear) and nonparametric (smooth spline) fits.
- Showed why test MSE has a U-shape (bias-variance).
- Defined the Bayes classifier and the Bayes error rate.
- Met our first method: K-nearest neighbours.

Key formulas at a glance

Model $Y = f(X) + \varepsilon, \quad E[\varepsilon] = 0, \;\; \mathrm{Var}(\varepsilon) = \sigma^2.$

Error split $E[(Y - \hat{f})^2] = \underbrace{[f - \hat{f}]^2}_{\text{reducible}} + \underbrace{\sigma^2}_{\text{irreducible}}.$

MSE $\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$ — average squared error on *whichever* sample it is computed on.

Training MSE that average over the n points used to fit $\hat{f}$; always over-optimistic.

Test MSE that average over *unseen* observations; at a new point, $E\left[(y_0 - \hat{f}(x_0))^2\right]$ — the honest one.

Bias $\mathrm{Bias}(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0)$, expectation over training sets; enters *squared* below.

Bias–variance $E\left[(y_0 - \hat{f}(x_0))^2\right] = \mathrm{Var}(\hat{f}(x_0)) + \left[\mathrm{Bias}(\hat{f}(x_0))\right]^2 + \sigma^2.$

Bayes rule assign x_0 to $\arg\max_j \Pr(Y = j \mid X = x_0)$; Bayes error $= 1 - E\left[\max_j \Pr(Y = j \mid X)\right]$.

KNN $\Pr(Y = j \mid X = x_0) = \frac{1}{K}\sum_{i \in \mathcal{N}_0} I(y_i = j).$

Vocabulary checklist

Term	Meaning
$f(X)$	Systematic part of Y given X
ε	Irreducible noise; $\mathrm{Var}(\varepsilon) = \sigma^2$
Reducible / irreducible error	Comes from $\hat{f}$ / from ε
Parametric / nonparametric	Fixed-form f / data-driven f
Flexibility	Effective number of parameters
Training MSE / test MSE	On train data / on held-out data
Bias / variance / irreducible error	Three parts of expected test MSE
Overfitting / underfitting	High variance / high bias
Bayes classifier / Bayes error	Optimal classifier / its error rate
KNN	Local-majority classifier with K neighbours

Decision rules of thumb

Choosing flexibility

- Small n, simple-looking data $\Rightarrow$ low flexibility wins.
- Large n, complex truth $\Rightarrow$ high flexibility wins.
- Inference goal $\Rightarrow$ pay an interpretability premium for low flexibility.
- Pure prediction $\Rightarrow$ choose by cross-validated test error.

Common pitfalls

Watch out for

1. Comparing methods on training MSE.
2. Using KNN without standardising features.
3. Reporting accuracy on imbalanced classes.
4. Confusing the curse of dimensionality with simply "a lot of data" — both can hurt KNN.
5. Treating the Bayes classifier as something you can fit (you can't).
6. Forgetting that bias and variance are computed over many training sets.

Self-check questions

1. Sketch the bias-variance decomposition curves. Where is the optimal flexibility? (p. 58)
2. Why is the irreducible error not a function of $\hat{f}$? (p. 14)
3. Construct an example where a more flexible method is preferable to a linear model. (pp. 23 and 98)
4. For KNN, what happens to bias and variance as K grows? (p. 78)
5. State the Bayes classifier in words; why is it the gold standard? (p. 70)
6. Suppose you have $n = 50$ observations of $p = 1$ predictor and want to estimate $E[Y \mid X]$. Pick a method and justify. (p. 98)

Ten things to remember

1. $Y = f(X) + \varepsilon$: f is what we estimate.
2. Reducible vs. irreducible error.
3. Two goals: prediction and inference.
4. Two strategies: parametric vs. nonparametric.
5. Flexibility vs. interpretability trade-off.
6. Training MSE is too optimistic — only test MSE matters.
7. Bias-variance: $E[(y_0 - \hat{f})^2] = \mathrm{Bias}^2 + \mathrm{Var}(\hat{f}) + \sigma^2$.
8. Optimal flexibility minimises the U-shape of test MSE.
9. Bayes classifier is the gold standard; Bayes error is the floor.
10. KNN: simple, flexible, sensitive to K and to scaling.

What's next

Chapter 3: *Linear Regression*.

- The simplest — and still most-used — statistical learning method.
- Fit, interpret, quantify uncertainty, diagnose problems.
- The benchmark against which every fancier method gets compared.

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

- **Bias–variance from first principles** — Extended Exercise 2.1 derives the decomposition instead of quoting it. The main deck states the result and Exercise 2.4 uses it with numbers — that is what the exam asks for.
- **The Bayes boundary for two Gaussians** — Extended Exercise 2.3 computes the optimal boundary analytically. It is the algebra behind LDA, so it fits better *after* Chapter 4.

Extended Exercise 2.1 — Bias-variance from first principles [Math]

Extended exercise (15 min)

Let $y_0 = f(x_0) + \varepsilon$ with $E[\varepsilon] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$, and ε independent of the training data used to build $\hat{f}$. Expectations are over *both* the random training set and the new noise ε.

1. Show $E\big[(y_0 - \hat{f}(x_0))^2\big] = E\big[(f(x_0) - \hat{f}(x_0))^2\big] + \sigma^2$. Where is independence of ε used?
2. By adding and subtracting $E[\hat{f}(x_0)]$, show $E\big[(f(x_0) - \hat{f}(x_0))^2\big] = \mathrm{Bias}(\hat{f}(x_0))^2 + \mathrm{Var}(\hat{f}(x_0))$.
3. Conclude $E[(y_0 - \hat{f})^2] = \mathrm{Var}(\hat{f}) + \mathrm{Bias}^2 + \sigma^2$, and say which terms are reducible.
4. Apply it at x_0 for three models (with $\sigma^2 = 1.0$): compute each total test MSE, pick the best, and state the lowest MSE attainable here.

Model	Bias	$\mathrm{Var}(\hat{f})$	σ^2
A (low flexibility)	2.0	0.5	1.0
B (medium flexibility)	1.0	1.2	1.0
C (high flexibility)	0.3	3.5	1.0

Solution — Extended Exercise 2.1 (1/3)

Solution

Write $\hat{f} = \hat{f}(x_0)$ and the constant $f = f(x_0)$.

1. Split off the noise. Since $y_0 = f + \varepsilon$,

$$E[(y_0 - \hat{f})^2] = E\left[((f - \hat{f}) + \varepsilon)^2\right] = E[(f - \hat{f})^2] + 2\,E[(f - \hat{f})\varepsilon] + E[\varepsilon^2].$$

Because ε is *independent* of the training data (hence of $\hat{f}$) and $E[\varepsilon] = 0$, the cross term factorises to $2\,E[f - \hat{f}]\,E[\varepsilon] = 0$; and $E[\varepsilon^2] = \mathrm{Var}(\varepsilon) = \sigma^2$. Hence

$$E[(y_0 - \hat{f})^2] = E[(f - \hat{f})^2] + \sigma^2.$$

2. Bias–variance split. Add and subtract $E[\hat{f}]$:

$$E[(f - \hat{f})^2] = E\left[\left((f - E[\hat{f}]) + (E[\hat{f}] - \hat{f})\right)^2\right].$$

The cross term is $2\,(f - E[\hat{f}])\,E\left[E[\hat{f}] - \hat{f}\right] = 0$, leaving

$$(E[\hat{f}] - f)^2 + E\left[(\hat{f} - E[\hat{f}])^2\right] = \mathrm{Bias}(\hat{f})^2 + \mathrm{Var}(\hat{f}).$$

Solution — Extended Exercise 2.1 (2/3)

Solution

3. Decomposition. Combining the two steps,

$$\boxed{E[(y_0 - \hat{f}(x_0))^2] = \mathrm{Var}(\hat{f}(x_0)) + \mathrm{Bias}(\hat{f}(x_0))^2 + \sigma^2.}$$

$\mathrm{Var}(\hat{f})$ and Bias^2 are *reducible* (they depend on the estimator); σ^2 is *irreducible*.
Interpretation: a method can only fight over the first two terms — choosing the flexibility shifts error between variance and squared bias, while the noise floor σ^2 stays fixed no matter what we fit. This is exactly the reducible/irreducible split of the lecture, now derived rather than asserted.

Solution — Extended Exercise 2.1 (3/3)

Solution

4. Apply. Test MSE $= \text{Bias}^2 + \text{Var} + \sigma^2$:

Model	Bias^2	Var	σ^2	Test MSE
A	4.00	0.5	1.0	5.50
B	1.00	1.2	1.0	3.20
C	0.09	3.5	1.0	4.59

Best: model B (test MSE 3.20). Low→medium: falling bias^2 (4.00 $\rightarrow$ 1.00) outweighs rising variance; medium→high: exploding variance (1.2 $\rightarrow$ 3.5) dominates — the classic U-shape. No model can beat the floor $\sigma^2 = 1.0$, reached only if bias *and* variance were both zero.

Common mistake

Judging by a single column — C has the lowest bias and A the lowest variance, yet B wins, because only the sum $\text{Bias}^2 + \text{Var} + \sigma^2$ is the test MSE.

Extended Exercise 2.3 — Bayes boundary for two Gaussians [Math]

Extended exercise (15 min)

A 1-D two-class problem with *common*-variance Gaussian class densities:

$$X \mid Y{=}0 \sim \mathcal{N}(\mu_0, \sigma^2), \qquad X \mid Y{=}1 \sim \mathcal{N}(\mu_1, \sigma^2),$$

priors $\pi_0 = \Pr(Y{=}0)$ and $\pi_1 = 1 - \pi_0$. Use $\mu_0 = 0$, $\mu_1 = 2$, $\sigma = 1$, $\pi_0 = 0.6$, $\pi_1 = 0.4$.

1. Write the Bayes classifier via the posteriors $\propto \pi_k f_k(x)$.
2. Taking logs, derive the boundary and show $x^\star = \dfrac{\mu_0 + \mu_1}{2} + \dfrac{\sigma^2}{\mu_1 - \mu_0} \ln \dfrac{\pi_0}{\pi_1}$; evaluate it.
3. Compute the Bayes error $\pi_0 \Pr(X{>}x^\star \mid Y{=}0) + \pi_1 \Pr(X{<}x^\star \mid Y{=}1)$, using $\Phi(1.20) = 0.885$ and $\Phi(0.80) = 0.788$.
4. Interpret: which way does raising π_0 move $x^\star$, and what does the Bayes error represent?

Solution — Extended Exercise 2.3 (1/2)

Solution

1. Bayes rule. The posterior is $\Pr(Y{=}k \mid x) \propto \pi_k f_k(x)$ with $f_k(x) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\!\big(-\frac{(x-\mu_k)^2}{2\sigma^2}\big)$. Predict class 1 iff $\pi_1 f_1(x) > \pi_0 f_0(x)$.

2. Boundary. Take logs of $\pi_1 f_1(x) = \pi_0 f_0(x)$; the constant $\frac{1}{\sqrt{2\pi}\,\sigma}$ cancels:

$$\ln \pi_1 - \frac{(x-\mu_1)^2}{2\sigma^2} = \ln \pi_0 - \frac{(x-\mu_0)^2}{2\sigma^2}.$$

Multiply by $2\sigma^2$ and expand; the x^2 terms cancel (equal variances $\Rightarrow$ a *linear* boundary), giving

$$2(\mu_1-\mu_0)x - (\mu_1^2-\mu_0^2) = 2\sigma^2 \ln\frac{\pi_0}{\pi_1} \;\Rightarrow\; x^\star = \frac{\mu_0+\mu_1}{2} + \frac{\sigma^2}{\mu_1-\mu_0}\ln\frac{\pi_0}{\pi_1}.$$

Plug in: $x^\star = 1 + \frac{1}{2}\ln(1.5) = 1 + \frac{1}{2}(0.405) = 1.203$.

Solution — Extended Exercise 2.3 (2/2)

Solution

3. Bayes error. With $\sigma = 1$: class 0 is misread when $X > x^\star$, probability $\Pr(Z > 1.203) = 1 - \Phi(1.20) \approx 0.115$; class 1 is misread when $X < x^\star$, probability $\Pr(Z < 1.203 - 2) = \Phi(-0.797) = 1 - \Phi(0.80) \approx 0.212$. Weight by the priors:

$$\text{Bayes error} = 0.6(0.115) + 0.4(0.212) = 0.069 + 0.085 = 0.154.$$

Even the optimal rule misreads about 15% of cases — the price of the overlap between the two bells.

4. Interpretation. The boundary sits at the midpoint 1 only when $\pi_0 = \pi_1$. Raising π_0 makes $\ln(\pi_0/\pi_1) > 0$, sliding $x^\star$ *toward the rarer class* (μ_1): we cede more of the line to the commoner class 0. The Bayes error is the *irreducible* classification error — the analogue of $\mathrm{Var}(\varepsilon)$ in regression — attainable only with the *true* π_k, f_k; any estimated classifier does at least as badly.

Common mistake

Averaging the two tail probabilities without the priors — $(0.115 + 0.212)/2 = 0.164$, not 0.154. Each class's error must be weighted by how often that class occurs.