DegreeLive

Bachelor of Commerce (B2001) | Level 3 • Part B Major | Monash University

Introduction to machine learning

Course Code: ETC3250 • Module 1 Notes

Module 1: Supervised Learning Foundations, Bias–Variance Tradeoff & Resampling

1.1 Statistical Learning Framework: Prediction vs. Inference

Machine learning within econometrics and business analytics formalises the problem of learning an unknown functional relationship between a quantitative or categorical outcome (response) variable Y and a vector of p input variables (predictors or features) $X = (X_1, X_2, \dots, X_p)^\top$. In a general data-generating process, the response is linked to the predictors via an additive stochastic structure:

$$Y = f(X) + \varepsilon$$

Where:

- Y is the scalar observed response variable (for example, quarterly residential property price in Melbourne or corporate default status on an ASX-listed bond).
- $X = (X_1, \dots, X_p)^\top \in \mathrm{R}^p$ is the p-dimensional vector of observable predictors.
- $f : \mathrm{R}^p \to \mathrm{R}$ is the fixed, systematic, yet unknown function capturing the true conditional expectation $\mathrm{E}[Y \mid X = x]$.
- ε is a zero-mean random disturbance term ($\mathrm{E}[\varepsilon \mid X] = 0$) with variance $\mathrm{Var}(\varepsilon) = \sigma_\varepsilon^2$, independent of X, representing unmeasured shocks and inherent randomness.

Statistical and machine learning estimation of $\hat{f}(X)$ from a training sample $\mathrm{D} = \{(x_i, y_i)\}_{i=1}^n$ is motivated by two distinct objectives that govern model selection, interpretability constraints, and diagnostic criteria:

- **Prediction:** The primary goal is to produce accurate out-of-sample forecasts $\hat{Y} = \hat{f}(X)$ for unseen observations where predictors X are observed but Y is unknown. Here, $\hat{f}$ is frequently treated as a black box, provided its out-of-sample generalisation error is minimised. Typical Australian commercial applications include real-time credit card fraud detection by major banks, automated electricity load forecasting in the National Electricity Market (NEM), and dynamic pricing in retail.
- **Inference:** The primary goal is to understand the structural mechanism through which each predictor X_j associates with or influences Y. Analysts ask which variables are statistically important, what sign and magnitude their partial effects carry, and whether the relationship is linear or nonlinear. Examples include evaluating the effect of Reserve Bank of Australia (RBA) cash rate adjustments on household mortgage stress or identifying demographic drivers of labour force participation from Australian Bureau of Statistics (ABS) surveys.

Parametric vs. Non-Parametric Estimation

Methods for constructing $\hat{f}$ divide broadly into parametric and non-parametric paradigms:

- **Parametric Methods:** Assume an explicit functional form for $f(X)$ indexed by a finite parameter vector $\theta \in \mathrm{R}^d$, reducing function estimation to parameter estimation. Ordinary Least Squares (OLS) assumes $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$. Parametric models are computationally efficient, require fewer observations, and yield direct parameter interpretation, but suffer from specification bias if the true f departs substantially from the assumed form.
- **Non-Parametric Methods:** Avoid rigid global functional assumptions and instead allow the data to determine the local shape of $\hat{f}$, subject only to smoothness or regularity conditions. Examples include K-Nearest Neighbours (KNN), smoothing splines, kernel regression, and decision trees. While capable of capturing intricate nonlinearities and interactions, non-parametric estimators require substantially larger sample sizes n to avoid overfitting noise, especially as the feature dimension p grows.

Dimension	Parametric Learning	Non-Parametric Learning
Functional Assumption	Fixed global form (e.g., linear, polynomial of degree d)	No fixed global form; local smoothness or partitioning
Effective Parameters	Fixed and finite, independent of sample size n	Grows flexibly with n and tuning hyperparameters
Sample Size Requirement	Moderate; performs reliably when n is small relative to p	Large n required to populate local neighbourhoods

DegreeLive

Dimension	Parametric Learning	Non-Parametric Learning
Interpretability	High; explicit marginal effects and standard errors	Ranges from moderate (GAMs) to low (deep ensembles)
Primary Risk	High approximation bias if model is misspecified	High estimation variance and overfitting to training noise

1.2 Supervised Learning Tasks and Loss Functions

Supervised learning problems are classified according to the measurement scale of the response variable Y. When $Y \in \mathrm{R}$ is continuous or numerical (such as weekly household expenditure in Australian dollars or ASX 200 daily return), the task is **regression**. When $Y \in \{1, 2, \ldots, K\}$ takes discrete unordered labels (such as loan default versus non-default, or customer churn category), the task is **classification**.

To evaluate how well a candidate prediction $\hat{y} = \hat{f}(x)$ matches the realised outcome y, we specify a non-negative **loss function** $L(y, \hat{f}(x))$ that penalises prediction errors. Minimising the expected loss over the joint distribution $P(X, Y)$ yields the Bayes-optimal theoretical target for $f(x)$.

Regression Loss Functions

- **Squared Error Loss (L_2 Loss):** Penalises errors quadratically, heavily weighting large outliers:

  $$L_2(y, \hat{y}) = (y - \hat{y})^2$$

  Where y is the observed numerical response and $\hat{y} = \hat{f}(x)$ is the predicted value. The function $f^*(x)$ that minimises expected squared error $\mathrm{E}[(Y - f(X))^2 \mid X = x]$ is the conditional mean $f^*(x) = \mathrm{E}[Y \mid X = x]$.

- **Absolute Error Loss (L_1 Loss):** Penalises errors linearly, offering robustness against heavy-tailed disturbances or extreme property valuations:

  $$L_1(y, \hat{y}) = |y - \hat{y}|$$

  Where $|\cdot|$ denotes absolute magnitude. The theoretical minimiser of expected L_1 loss is the conditional median $\mathrm{Median}(Y \mid X = x)$, though the non-differentiability at zero complicates gradient-based optimisation.

- **Huber Loss:** Combines quadratic smoothness near zero with linear robustness in the tails using a threshold parameter $\delta > 0$:

  $$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |y - \hat{y}| \le \delta \\ \delta\,(|y - \hat{y}| - \frac{1}{2}\delta) & \text{if } |y - \hat{y}| > \delta \end{cases}$$

  Where δ defines the residual cutoff separating squared-error behaviour (for small residuals) from absolute-error behaviour (for outliers).

Classification Loss Functions

For binary classification with labels $y \in \{0, 1\}$ and predicted class probability $\hat{p} = \hat{P}(Y = 1 \mid X = x) \in (0, 1)$, or hard label prediction $\hat{y} \in \{0, 1\}$:

- **Zero–One Loss:** Counts raw misclassification events: $L_{0\text{-}1}(y, \hat{y}) = \mathrm{I}(y \neq \hat{y})$, where $\mathrm{I}(\cdot)$ is the indicator function equal to 1 when the condition holds and 0 otherwise. Because $L_{0\text{-}1}$ is discontinuous and piecewise flat, direct numerical optimisation over parameter space is NP-hard, motivating smooth convex surrogates.
- **Binary Cross-Entropy (Log-Loss / Negative Log-Likelihood):** Evaluates calibrated probability estimates by penalising confident wrong predictions logarithmically:

Degree Live

$$L_{\text{CE}}(y, \hat{p}) = -\left[y \ln(\hat{p}) + (1-y)\ln(1-\hat{p})\right]$$

Where $y \in \{0, 1\}$ is the realised binary class and $\hat{p} \in (0, 1)$ is the predicted probability that $Y = 1$. For K-class problems with one-hot indicator $y_{ik} \in \{0, 1\}$ and predicted probabilities $\hat{p}_{ik}$, multinomial deviance equals $-2\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\ln(\hat{p}_{ik})$.

Worked Example 1.1: Comparing Regression and Probability Losses

Suppose an analyst models quarterly retail store revenue (in AUD thousands) across three Australian regional branches and also predicts whether each branch exceeds its sales target ($y_{\text{cls}} \in \{0, 1\}$). Observed revenues are $y = (120, 150, 260)$ and model predictions are $\hat{y} = (115, 160, 200)$, where branch 3 experienced an unusual holiday surge (residual 60). Compute Mean Squared Error (MSE), Mean Absolute Error (MAE), Huber Loss with $\delta = 20$, and Binary Cross-Entropy if branch target exceedances are $y_{\text{cls}} = (1, 0, 1)$ with predicted probabilities $\hat{p} = (0.85, 0.25, 0.40)$.

1. **Residuals $e_i = y_i - \hat{y}_i$:** $e_1 = 120 - 115 = 5$, $e_2 = 150 - 160 = -10$, $e_3 = 260 - 200 = 60$.
2. **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{5^2 + (-10)^2 + 60^2}{3} = \frac{25 + 100 + 3600}{3} = \frac{3725}{3} \approx 1241.67$$

   Notice that the single outlier $e_3 = 60$ contributes $3600/3725 = 96.6\%$ of total squared loss. Taking the square root gives Root Mean Squared Error (RMSE) $\sqrt{1241.67} \approx 35.24$ AUD thousand.

3. **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{|5| + |-10| + |60|}{3} = \frac{5 + 10 + 60}{3} = \frac{75}{3} = 25.00$$

4. **Mean Huber Loss ($\delta = 20$):**
   For $i = 1, 2$, $|e_i| \le 20$, so $L_{20}(e_1) = 0.5(5)^2 = 12.5$ and $L_{20}(e_2) = 0.5(-10)^2 = 50.0$. For $i = 3$, $|e_3| = 60 > 20$, so $L_{20}(e_3) = 20(60 - 0.5 \times 20) = 20(50) = 1000.0$. Averaging gives:

$$\bar{L}_{20} = \frac{12.5 + 50.0 + 1000.0}{3} = \frac{1062.5}{3} \approx 354.17$$

5. **Binary Cross-Entropy (Log-Loss):**
   Evaluating each branch using $L_{\text{CE}}(y_i, \hat{p}_i) = -[y_i \ln(\hat{p}_i) + (1-y_i)\ln(1-\hat{p}_i)]$:
   - Branch 1 ($y_1 = 1, \hat{p}_1 = 0.85$): $-\ln(0.85) \approx 0.1625$.
   - Branch 2 ($y_2 = 0, \hat{p}_2 = 0.25$): $-\ln(0.75) \approx 0.2877$.
   - Branch 3 ($y_3 = 1, \hat{p}_3 = 0.40$): $-\ln(0.40) \approx 0.9163$.

$$\bar{L}_{\text{CE}} = \frac{0.1625 + 0.2877 + 0.9163}{3} \approx 0.4555$$

1.3 Training Error, Test Error, and Generalisation

Given a training dataset $\mathrm{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{n}$, the **training MSE** measures in-sample fit:

DegreeLive

$$\text{MSE}_{\text{train}} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

Where:

- n is the number of observations in the training set.
- y_i is the observed response for training observation i.
- $\hat{f}(x_i)$ is the prediction produced by the model fitted on D_{train}.

Training error systematically underestimates out-of-sample error because $\hat{f}$ is explicitly chosen to fit D_{train}. Every flexible model can drive $\text{MSE}_{\text{train}}$ arbitrarily close to zero simply by interpolating random noise ε_i (for example, fitting a polynomial of degree $n-1$ to n distinct points). What matters for deployment is the **expected test MSE** evaluated on a fresh, independent test observation (x_0, y_0) drawn from the same population distribution:

$$\text{MSE}_{\text{test}} = \text{E}\left[(y_0 - \hat{f}(x_0))^2\right]$$

Where the expectation averages over both the random training sets D_{train} that could be drawn and the unseen test observation (x_0, y_0). As model flexibility increases, $\text{MSE}_{\text{train}}$ declines monotonically, whereas MSE_{test} typically follows a characteristic **U-shaped curve**: initially falling as genuine signal is captured, reaching a minimum at optimal complexity, and then rising sharply as the model overfits idiosyncratic noise.

1.4 The Bias–Variance Decomposition of Mean Squared Error

The U-shaped test error curve is governed mathematically by the **bias–variance decomposition**. Consider a fixed test input point x_0, where the true outcome is $y_0 = f(x_0) + \varepsilon_0$ with $\text{E}[\varepsilon_0] = 0$, $\text{Var}(\varepsilon_0) = \sigma_\varepsilon^2$, and ε_0 independent of the training sample D. Let $\hat{f}(x_0)$ be the random variable representing the prediction at x_0 obtained by fitting our learning algorithm to a randomly drawn training sample D, and write $\bar{f}(x_0) = \text{E}_{\text{D}}[\hat{f}(x_0)]$ for its expected prediction across repeated training samples.

Step-by-Step Mathematical Derivation

Begin by expanding $\text{E}[(y_0 - \hat{f}(x_0))^2]$ around $f(x_0)$:

$$\text{E}\left[(y_0 - \hat{f}(x_0))^2\right] = \text{E}\left[(f(x_0) + \varepsilon_0 - \hat{f}(x_0))^2\right]$$

Since ε_0 is independent of $\hat{f}(x_0)$ and has zero mean, the cross-product term $2\text{E}[\varepsilon_0(f(x_0) - \hat{f}(x_0))] = 2\text{E}[\varepsilon_0]\text{E}[f(x_0) - \hat{f}(x_0)] = 0$. This leaves:

$$\text{E}\left[(y_0 - \hat{f}(x_0))^2\right] = \text{E}\left[(f(x_0) - \hat{f}(x_0))^2\right] + \text{E}[\varepsilon_0^2]$$

Now add and subtract $\bar{f}(x_0) = \text{E}[\hat{f}(x_0)]$ inside the first term:

$$\text{E}\left[(f(x_0) - \bar{f}(x_0) + \bar{f}(x_0) - \hat{f}(x_0))^2\right] = (f(x_0) - \bar{f}(x_0))^2 + \text{E}\left[(\hat{f}(x_0) - \bar{f}(x_0))^2\right] + 2(f(x_0) - \bar{f}(x_0))\text{E}[\bar{f}(x_0) - \hat{f}(x_0)]$$

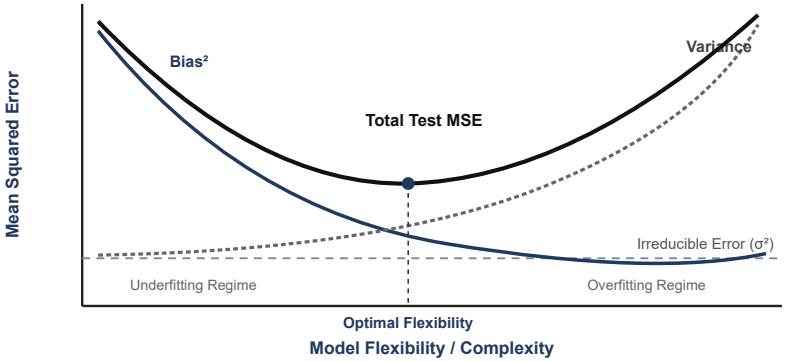
Because $\text{E}[\bar{f}(x_0) - \hat{f}(x_0)] = \bar{f}(x_0) - \bar{f}(x_0) = 0$, the cross-term vanishes completely. Combining all pieces yields the fundamental identity:

DegreeLive

$$\mathrm{E}\left[(y_0 - \hat{f}(x_0))^2\right] = \underbrace{\left(\mathrm{E}[\hat{f}(x_0)] - f(x_0)\right)^2}_{\text{Squared Bias}} + \underbrace{\mathrm{Var}\left(\hat{f}(x_0)\right)}_{\text{Variance}} + \underbrace{\sigma_\varepsilon^2}_{\text{Irreducible Error}}$$

Where:

- $\left(\mathrm{Bias}(\hat{f}(x_0))\right)^2 = \left(\mathrm{E}[\hat{f}(x_0)] - f(x_0)\right)^2$**:** Measures systematic error introduced by approximating a potentially complex real-world relationship with a simplified model structure (for example, fitting a straight line to a quadratic relationship).
- $\mathrm{Var}(\hat{f}(x_0)) = \mathrm{E}[(\hat{f}(x_0) - \mathrm{E}[\hat{f}(x_0)])^2]$**:** Measures the sensitivity of the fitted function to the specific training sample drawn. High-variance algorithms (such as unpruned decision trees or 1-nearest neighbour) fluctuate wildly when a few training observations change.
- $\sigma_\varepsilon^2 = \mathrm{Var}(\varepsilon_0)$**:** The irreducible noise variance inherent in the data-generating process. Regardless of how sophisticated $\hat{f}$ becomes, expected test MSE can never fall below σ_ε^2.



Worked Example 1.2: Computing Exact Bias, Variance, and Test MSE

Suppose the true relationship between an agricultural yield index Y and rainfall deviation X is $f(x) = 2 + 0.5x^2$ with irreducible noise variance $\sigma_\varepsilon^2 = 1.0$. Consider evaluating predictions at the test point $x_0 = 4$, where the true noiseless signal is $f(4) = 2 + 0.5(16) = 10.0$. An analyst compares two competing models fitted across repeated training samples:

- **Model A (Rigid Linear Model):** Produces predictions at $x_0 = 4$ distributed with mean $\mathrm{E}[\hat{f}_A(4)] = 7.5$ and variance $\mathrm{Var}(\hat{f}_A(4)) = 0.40$.
- **Model B (Flexible Cubic Spline):** Produces predictions at $x_0 = 4$ distributed with mean $\mathrm{E}[\hat{f}_B(4)] = 9.8$ and variance $\mathrm{Var}(\hat{f}_B(4)) = 3.20$.
- **Model C (Moderately Flexible Quadratic):** Produces predictions at $x_0 = 4$ distributed with mean $\mathrm{E}[\hat{f}_C(4)] = 9.9$ and variance $\mathrm{Var}(\hat{f}_C(4)) = 0.85$.

Compute each component of the expected test MSE at $x_0 = 4$:

1. **Model A:** $\mathrm{Bias} = 7.5 - 10.0 = -2.5 \implies \mathrm{Bias}^2 = (-2.5)^2 = 6.25$.

$$\mathrm{EPE}_A(4) = 6.25 + 0.40 + 1.00 = 7.65$$

2. **Model B:** $\mathrm{Bias} = 9.8 - 10.0 = -0.2 \implies \mathrm{Bias}^2 = (-0.2)^2 = 0.04$.

$$\mathrm{EPE}_B(4) = 0.04 + 3.20 + 1.00 = 4.24$$

3. **Model C:** $\mathrm{Bias} = 9.9 - 10.0 = -0.1 \implies \mathrm{Bias}^2 = (-0.1)^2 = 0.01$.

$$\text{EPE}_C(4) = 0.01 + 0.85 + 1.00 = 1.86$$

Model A fails because of severe underfitting ($\text{Bias}^2 = 6.25$), whereas Model B captures the curvature almost unbiasedly but incurs excessive sampling variance (3.20). Model C achieves the optimal balance, attaining the lowest expected test MSE of 1.86.

1.5 Resampling Methods for Model Assessment and Selection

In empirical practice, test data are scarce and the true function $f(x)$ is unknown, making analytical computation of test MSE impossible. Resampling methods repeatedly draw subsets from the available training data and refit the model to estimate out-of-sample prediction error and parameter uncertainty without collecting additional data.

The Validation Set Approach

The simplest strategy randomly partitions the n available observations into two disjoint subsets: a training set (typically 70% to 80%) used to fit $\hat{f}$, and a held-out validation set (20% to 30%) used to evaluate MSE or classification error. Although computationally fast, the validation set approach suffers from two major drawbacks: (1) the validation error estimate can exhibit high variance depending on which observations land in the validation split, and (2) fitting the model on substantially fewer than n observations causes the validation error to overestimate the error rate of a model trained on the full dataset.

Leave-One-Out Cross-Validation (LOOCV)

LOOCV resolves the data-reduction bias of a single split by holding out exactly one observation (x_i, y_i) as the validation point and training the model $\hat{f}^{(-i)}$ on the remaining $n-1$ observations. Repeating this across all $i = 1, \dots, n$ yields the LOOCV estimate:

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}^{(-i)}(x_i)\right)^2$$

Where:

- n is the total sample size.
- $\hat{f}^{(-i)}(x_i)$ is the prediction at x_i from the model fitted on all observations except observation i.

Because each training fold contains $n-1$ observations, LOOCV has virtually zero upward bias as an estimate of full-sample test error and exhibits zero randomness across runs (the partition is deterministic). However, fitting n separate models is computationally prohibitive for complex learners such as Random Forests or Neural Networks. Remarkably, for OLS linear regression and linear smoothers (such as polynomial regression and smoothing splines), LOOCV requires only **a single model fit** using the diagonal leverage values h_{ii} of the hat matrix $H = X(X^\top X)^{-1}X^\top$:

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2$$

Where:

- $\hat{y}_i$ is the fitted value from the full-sample OLS regression using all n observations.
- $h_{ii} = x_i^\top (X^\top X)^{-1} x_i \in [1/n, 1)$ is the leverage of observation i, measuring how far x_i sits from the centre of the predictor space.

This shortcut reveals a deep statistical insight: high-leverage points inflate the denominator penalty $(1 - h_{ii})^2$, heavily penalising models whose predictions depend disproportionately on isolated extreme observations.

DegreeLive

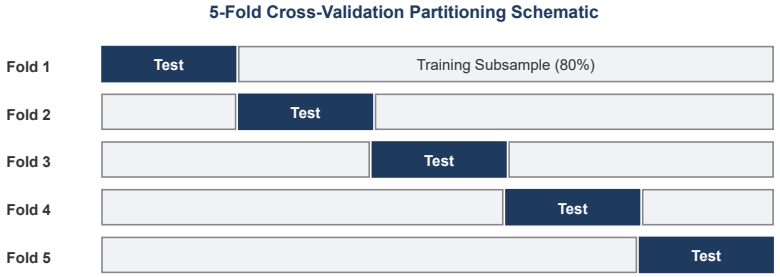
k-Fold Cross-Validation

In k-fold Cross-Validation, the dataset is randomly partitioned into k roughly equal, mutually exclusive folds $C_1, C_2, \ldots, C_k$ of size $n_j \approx n/k$ (commonly $k = 5$ or $k = 10$). For each fold $j \in \{1, \ldots, k\}$, the model is trained on the remaining $k - 1$ folds and evaluated on fold C_j to produce MSE_j. The overall CV estimate averages across folds:

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{j=1}^{k}\text{MSE}_j = \frac{1}{n}\sum_{j=1}^{k}\sum_{i\in C_j}\left(y_i - \hat{f}^{(-j)}(x_i)\right)^2$$

Where:

- k is the number of folds.
- $\hat{f}^{(-j)}$ is the model fitted using all observations outside fold C_j.
- $\text{MSE}_j = \frac{1}{|C_j|}\sum_{i\in C_j}(y_i - \hat{f}^{(-j)}(x_i))^2$ is the validation error on fold j.



Choosing k in cross-validation introduces its own bias–variance trade-off distinct from model complexity:

- **Bias of the CV Estimator:** LOOCV trains on $n - 1$ observations per fold and has minimal bias, whereas $k = 5$ trains on $0.8n$ observations, introducing slight upward bias relative to the full-sample model.
- **Variance of the CV Estimator:** Counterintuitively, LOOCV often exhibits *higher* variance than 5-fold or 10-fold CV. In LOOCV, the n training sets share $n - 2$ identical observations, causing the n fitted models $\hat{f}^{(-i)}$ to be highly positively correlated; averaging strongly correlated quantities yields higher variance than averaging less correlated folds. Empirically, $k = 5$ or $k = 10$ achieves an optimal balance between bias, variance, and computational runtime.

For classification with imbalanced classes (for example, 2% mortgage arrears in an Australian banking portfolio), **Stratified k-Fold CV** ensures every fold preserves the exact population class proportions. For time-series data, standard random folding leaks future observations into the past; instead, **rolling-origin (expanding-window) CV** must be used so training folds always precede validation folds chronologically.

Worked Example 1.3: LOOCV Shortcut vs. Naive Fit

Consider $n = 4$ quarterly observations linking Australian consumer discretionary spending index Y to consumer sentiment X: $(1, 2)$, $(2, 4)$, $(3, 5)$, and $(4, 8)$. An analyst fits simple linear regression $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$.

1. **Sample Means and OLS Fit:** $\bar{x} = (1 + 2 + 3 + 4)/4 = 2.5$ and $\bar{y} = (2 + 4 + 5 + 8)/4 = 4.75$.

$$S_{xx} = \sum_{i=1}^{4}(x_i - \bar{x})^2 = (-1.5)^2 + (-0.5)^2 + 0.5^2 + 1.5^2 = 5.0$$

$$S_{xy} = \sum_{i=1}^{4}(x_i - \bar{x})(y_i - \bar{y}) = (-1.5)(-2.75) + (-0.5)(-0.75) + 0.5(0.25) + 1.5(3.25) = 9.5$$

Hence $\hat{\beta}_1 = 9.5/5.0 = 1.90$ and $\hat{\beta}_0 = 4.75 - 1.90(2.5) = 0.00$, giving $\hat{y}_i = 1.90x_i$.

2. **Residuals and Leverages:** For simple linear regression, leverage is $h_{ii} = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{S_{xx}}$:
   - $i = 1$: $\hat{y}_1 = 1.9, e_1 = 2 - 1.9 = 0.1, h_{11} = 0.25 + 2.25/5 = 0.70$.
   - $i = 2$: $\hat{y}_2 = 3.8, e_2 = 4 - 3.8 = 0.2, h_{22} = 0.25 + 0.25/5 = 0.30$.
   - $i = 3$: $\hat{y}_3 = 5.7, e_3 = 5 - 5.7 = -0.7, h_{33} = 0.25 + 0.25/5 = 0.30$.
   - $i = 4$: $\hat{y}_4 = 7.6, e_4 = 8 - 7.6 = 0.4, h_{44} = 0.25 + 2.25/5 = 0.70$.
3. **LOOCV Error via Leverage Formula:**

$$\mathrm{CV}_{(4)} = \frac{1}{4}\left[\left(\frac{0.1}{1-0.70}\right)^2 + \left(\frac{0.2}{1-0.30}\right)^2 + \left(\frac{-0.7}{1-0.30}\right)^2 + \left(\frac{0.4}{1-0.70}\right)^2\right]$$

$$\mathrm{CV}_{(4)} = \frac{1}{4}\left[(0.333)^2 + (0.286)^2 + (-1.000)^2 + (1.333)^2\right] = \frac{0.111 + 0.082 + 1.000 + 1.778}{4} \approx 0.743$$

Compare this to the in-sample training MSE: $(0.01 + 0.04 + 0.49 + 0.16)/4 = 0.175$. Because observations 1 and 4 carry high leverage ($h = 0.70$), removing them changes the fitted line substantially, pushing LOOCV MSE above four times the training MSE.

The Bootstrap

The **bootstrap** (Efron, 1979) quantifies the sampling distribution and standard error of any estimator $\hat{\theta}$ by repeatedly drawing B datasets of size n *with replacement* from the original sample D. In each bootstrap replicate $b = 1, \ldots, B$, an observation i has probability $1/n$ of being selected on each draw. Consequently, the probability that observation i is omitted completely from a given bootstrap sample D^{*b} is:

$$P(i \notin \mathrm{D}^{*b}) = \left(1 - \frac{1}{n}\right)^n \xrightarrow{n\to\infty} e^{-1} \approx 0.3679$$

Thus each bootstrap sample contains approximately 63.2% unique observations from D, while the remaining 36.8% form an **out-of-bag (OOB)** set. Averaging across B replicates gives the bootstrap standard error estimate:

$$\mathrm{SE}_B(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\theta}^{*b} - \bar{\theta}^*\right)^2}, \qquad \bar{\theta}^* = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^{*b}$$

Where $\hat{\theta}^{*b}$ is the statistic computed on bootstrap sample b, and B is typically 500 to 2000. Notice a subtle warning: if one naively evaluates prediction error by testing on D after training on D^{*b}, overlapping points leak into the test evaluation and severely understate error; valid bootstrap error estimation must evaluate strictly on the OOB observations.

Resampling Scheme	Partition Rule	Primary Use Case	Key Advantage	Key Limitation
Validation Split	Single random split (e.g., 75% / 25%)	Rapid prototyping on very large n	Minimal compute (1 fit)	High variance; upward bias from reduced train size
k-Fold CV ($k = 5, 10$)	k disjoint folds, each held out once	Hyperparameter tuning & model selection	Low variance and low bias	Requires k model fits per candidate
LOOCV ($k = n$)	Each single observation held out once	Small-sample OLS / linear smoothers	Deterministic; nearly unbiased	Slow for nonlinear models; higher variance than 10-fold

Degree Live

Resampling Scheme	Partition Rule	Primary Use Case	Key Advantage	Key Limitation
Bootstrap	B samples of size n with replacement	Standard errors, confidence intervals, bagging	Works for complex non-analytic statistics	Overlap biases naive error unless using OOB points

1.6 The Curse of Dimensionality

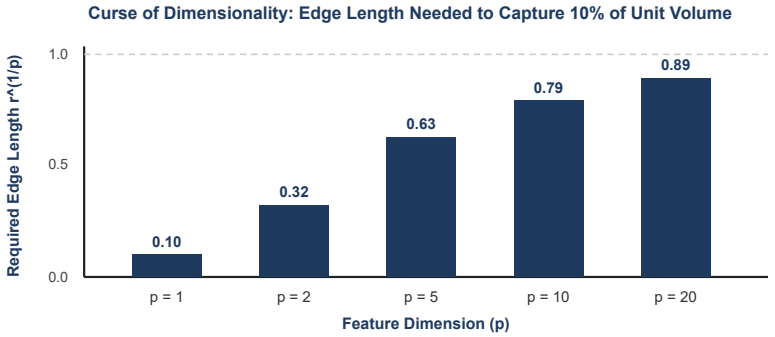
Modern business datasets frequently contain hundreds or thousands of features—such as detailed transaction histories, web clickstreams, or lagged macroeconomic indicators. While adding predictors appears beneficial, high-dimensional feature spaces suffer from the **curse of dimensionality** (Bellman, 1961), which degrades local non-parametric estimators and distance-based algorithms.

Exponential Volume Growth and Sparsity

Consider n observations uniformly distributed on the p-dimensional unit hypercube $[0,1]^p$. Suppose we use a local neighbourhood (as in K-Nearest Neighbours) that captures a target fraction $r \in (0,1)$ of the total volume (and hence roughly r of the observations). A hypercube covering fraction r of unit volume must have edge length $\ell_p(r) = r^{1/p}$. Even when $r = 0.10$ (capturing 10% of the sample):

- In $p = 1$ dimension: $\ell_1(0.10) = 0.10^1 = 0.100$ (truly local, 10% of range).
- In $p = 5$ dimensions: $\ell_5(0.10) = 0.10^{1/5} \approx 0.631$ (spans 63% of every feature).
- In $p = 20$ dimensions: $\ell_{20}(0.10) = 0.10^{1/20} \approx 0.891$ (spans nearly 90% of every axis!).

Thus, in high dimensions, a "nearest neighbour" is no longer local; averaging over such wide regions destroys the assumption that $f(x)$ is approximately constant across the neighbourhood, driving approximation bias dramatically upward.



Distance Concentration and Boundary Dominance

Two additional geometric pathologies emerge as $p \to \infty$:

- **Edge Concentration:** Almost all points sampled uniformly in a hypercube $[-1,1]^p$ lie near its outer shell. The ratio of volume inside an inscribed hypersphere of radius 1 to the enclosing hypercube approaches zero rapidly (already below 2.5% by $p = 10$). Consequently, test observations almost always lie outside the convex hull of training data, turning interpolation into risky extrapolation.
- **Distance Concentration:** For independent coordinates, the ratio between the distance to the nearest neighbour $d_{\min}$ and the farthest neighbour $d_{\max}$ converges in probability toward 1: $(d_{\max} - d_{\min})/d_{\min} \to 0$. Euclidean distances lose discriminative power, motivating dimension reduction (PCA) and regularisation (Ridge, LASSO) covered in Modules 2 and 4.

1.7 Feature Scaling and Leakage-Free Preprocessing Pipelines

Many machine learning algorithms rely explicitly on Euclidean distances or penalty norms across predictors: K-Nearest Neighbours, Ridge, LASSO, Support Vector Machines, Principal Component Analysis, and gradient-based Neural Networks. When predictors sit on vastly different numerical scales—for example, household mortgage balance in dollars ($A\$500{,}000$) versus loan-to-income ratio (4.5)—unscaled Euclidean distance or L_2 penalties are completely dominated by the variable with the largest numerical magnitude rather than predictive relevance.

Three standard feature transformations address scale heterogeneity:

- **Standardisation (z-Score Scaling):** Centres each feature to zero mean and unit sample standard deviation:

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

  Where $\bar{x}_j$ is the training sample mean of predictor j and s_j is its training sample standard deviation. Preferred for Ridge, LASSO, SVM, PCA, and neural networks.

- **Min–Max Normalisation:** Rescales features linearly into the bounded interval $[0, 1]$:

$$\tilde{x}_{ij} = \frac{x_{ij} - x_{j,\min}}{x_{j,\max} - x_{j,\min}}$$

  Where $x_{j,\min}$ and $x_{j,\max}$ are the minimum and maximum values of feature j in the training fold. Sensitive to extreme outliers.

- **Robust Scaling:** Uses median and interquartile range (IQR) to mitigate outlier distortion:

$$x_{ij}^{\text{rob}} = \frac{x_{ij} - \text{Median}(x_j)}{Q_{0.75}(x_j) - Q_{0.25}(x_j)}$$

Preventing Data Leakage in Cross-Validation

A critical methodological error in applied analytics is computing scaling parameters ($\bar{x}_j, s_j$), imputation medians, or variable selection filters on the *entire* dataset before splitting into cross-validation folds. Doing so allows information from the validation fold to influence the training transformation—a form of **data leakage** that produces optimistically biased CV error estimates. In reproducible R workflows (for example, using `recipes` inside `tidymodels`), every preprocessing step must be wrapped inside the resampling loop: means and standard deviations are estimated strictly on the $k - 1$ training folds and then applied identically to transform the held-out validation fold.

Worked Example 1.4: Leakage-Free Scaling Inside a CV Fold

Suppose a small credit dataset has $n = 5$ loan applicants with annual income (in AUD thousands): training fold $\text{D}_{\text{train}} = \{60, 80, 100, 120\}$ and held-out validation observation $x_{\text{val}} = 180$. Contrast proper pipeline scaling with leaky global scaling:

1. **Proper (Leakage-Free) Pipeline:** Compute statistics strictly from D_{train}: $\bar{x}_{\text{train}} = (60 + 80 + 100 + 120)/4 = 90.0$, and sample variance $s^2_{\text{train}} = [(-30)^2 + (-10)^2 + 10^2 + 30^2]/3 = 2000/3 \approx 666.67$, so $s_{\text{train}} \approx 25.82$. Applying these training parameters to the validation applicant gives:

$$z_{\text{val}}^{\text{proper}} = \frac{180 - 90.0}{25.82} \approx 3.49$$

2. **Improper (Leaky) Global Scaling:** Using all five points gives $\bar{x}_{\text{all}} = 108.0$ and $s_{\text{all}} \approx 46.04$, yielding $z_{\text{val}}^{\text{leaky}} = (180 - 108)/46.04 \approx 1.56$. The leaky transformation pulls the outlier closer to the centre by borrowing knowledge of its own magnitude, artificially softening prediction errors on unusual test cases.

Reproducible Pipeline Architecture in R and Model Complexity Control

In Monash ETC3250 empirical practice using R and the `tidymodels` ecosystem, feature engineering and model fitting are unified via a `workflow()` combining a preprocessing `recipe()` with a model specification. Whenever k-fold cross-validation is invoked through `fit_resamples()` or `tune_grid()`, the recipe automatically invokes `prep()` exclusively on each training analysis split and applies `bake()` to the corresponding assessment fold. Furthermore, to ensure reproducibility across stochastic resampling splits, analysts set an explicit pseudorandom seed (`set.seed()`) prior to partitioning, log exact software package versions, and reserve a final untouched test split that is evaluated strictly once after all hyperparameters—such as K in KNN or regularisation penalty λ—have been selected via cross-validation.

DegreeLive

KNN Regression Bias–Variance Behaviour

To observe the bias–variance trade-off concretely in a non-parametric learner, consider K-Nearest Neighbours regression, which estimates $f(x_0)$ by averaging the responses of the K training observations closest to x_0 in scaled Euclidean distance:

$$\hat{f}_K(x_0) = \frac{1}{K} \sum_{i \in \mathrm{N}_K(x_0)} y_i$$

Where:

- $\mathrm{N}_K(x_0)$ denotes the index set of the K training observations with smallest Euclidean distance $\|x_i - x_0\|_2$.
- $K \in \{1, 2, \dots, n\}$ is the smoothing hyperparameter controlling neighbourhood size.

Under homoskedastic noise $\mathrm{Var}(\varepsilon_i) = \sigma_\varepsilon^2$ and fixed training predictors, the conditional variance of the KNN estimator is exactly $\mathrm{Var}(\hat{f}_K(x_0)) = \sigma_\varepsilon^2 / K$. When $K = 1$, $\hat{f}_1(x_0)$ interpolates every training point ($\mathrm{MSE}_{\text{train}} = 0$) with very low bias, yet suffers maximal variance σ_ε^2. Conversely, as K increases toward n, variance shrinks inversely with K, while squared bias $\left(\frac{1}{K}\sum_{i \in \mathrm{N}_K(x_0)} f(x_i) - f(x_0)\right)^2$ rises because distant points from flatter or differently sloped regions of f are averaged into the prediction.

Worked Example 1.5: Exact Bias, Variance, and Optimal K in KNN Regression

Suppose the true relationship between Australian regional store weekly sales Y (in AUD thousands) and foot-traffic index X is $f(x) = 10 + 2x$ with irreducible noise variance $\sigma_\varepsilon^2 = 16$. We wish to predict at $x_0 = 5$ (where true noiseless value $f(5) = 20$) using training observations sampled at integers $x \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Compare $K = 1$, $K = 3$, and $K = 5$ (where the 5 nearest neighbours are $\{3, 4, 5, 6, 7\}$) versus a biased asymmetric neighbourhood $K = 7$ at boundary point $x_0 = 2$ (where true $f(2) = 14$ and the 7 nearest neighbours are $\{1, 2, 3, 4, 5, 6, 7\}$):

1. **Interior Point $x_0 = 5$ (Symmetric Neighbourhood):**
   - For $K = 1$: $\mathrm{N}_1(5) = \{5\}$, so $\mathrm{E}[\hat{f}_1(5)] = f(5) = 20$. Bias $= 0$, Variance $= 16/1 = 16.00$, Test MSE $= 0 + 16 + 16 = 32.00$.
   - For $K = 3$: $\mathrm{N}_3(5) = \{4, 5, 6\}$, $\mathrm{E}[\hat{f}_3(5)] = (18 + 20 + 22)/3 = 20$. Bias $= 0$, Variance $= 16/3 \approx 5.33$, Test MSE $= 0 + 5.33 + 16 = 21.33$.
   - For $K = 5$: $\mathrm{N}_5(5) = \{3, 4, 5, 6, 7\}$, $\mathrm{E}[\hat{f}_5(5)] = 20$. Bias $= 0$, Variance $= 16/5 = 3.20$, Test MSE $= 0 + 3.20 + 16 = 19.20$.
2. **Boundary Point $x_0 = 2$ (Asymmetric Neighbourhood):**
   - For $K = 1$: $\mathrm{N}_1(2) = \{2\}$, Bias $= 0$, Variance $= 16.00$, Test MSE $= 32.00$.
   - For $K = 3$: $\mathrm{N}_3(2) = \{1, 2, 3\}$, $\mathrm{E}[\hat{f}_3(2)] = (12 + 14 + 16)/3 = 14$. Bias $= 0$, Variance $= 5.33$, Test MSE $= 21.33$.
   - For $K = 7$: $\mathrm{N}_7(2) = \{1, 2, 3, 4, 5, 6, 7\}$, whose mean true value is $f(4) = 18$. Here Bias $= 18 - 14 = +4$, so Bias$^2 = 16.00$, Variance $= 16/7 \approx 2.29$, and Test MSE $= 16.00 + 2.29 + 16 = 34.29$.

   This demonstrates why boundary points suffer severe smoothing bias as K grows too large: the neighbourhood expands asymmetrically into higher-value regions.

1.8 Diagnosing Overfitting, Underfitting, and Resampling Uncertainty

In empirical machine learning projects, practitioners examine training curves, cross-validation error curves, and residual diagnostics jointly to determine whether poor out-of-sample performance stems from high bias or high variance—because the corrective actions are exact opposites.

Diagnostic Regime	Training Error	CV / Validation Error	Primary Source of Error	Remedial Engineering Action
Underfitting	High	High (close to training error)	Excessive Bias	Add polynomial/spline terms, include domain interactions, reduce regularisation λ, switch to flexible learner (GAM, RF, GBM)

DegreeLive

Diagnostic Regime	Training Error	CV / Validation Error	Primary Source of Error	Remedial Engineering Action
Overfitting	Very Low (near 0)	High (wide train–CV gap)	Excessive Variance	Gather more training observations n, increase regularisation λ, prune trees, reduce feature dimension p, apply CV early stopping
Optimal Generalisation	Low to Moderate	Minimal (small train–CV gap)	Approaches Irreducible σ_ε^2	Validate once on untouched test split; lock pipeline and deploy with drift monitoring

Nested Cross-Validation for Unbiased Performance Reporting

When cross-validation is used simultaneously to select hyperparameters (such as choosing optimal K or λ) and to report final expected test performance, a subtle selection bias arises: picking the minimum across M noisy CV estimates yields an optimistically downward-biased estimate of true test error. In rigorous Monash ETC3250 workflows with moderate sample sizes, analysts resolve this via **Nested Cross-Validation**: an outer k_{out}-fold loop splits data into outer training and outer test folds strictly for unbiased error estimation, while an inner k_{in}-fold loop operates independently inside each outer training set to tune hyperparameters without ever touching the outer test fold.

DegreeLive