Solution 1. A regression tree leaf: the optimal constant

The target values are

$$2,\ 2,\ 3,\ 8,\ 9,\ 9.$$

Their mean is

$$\bar{y} = \frac{2+2+3+8+9+9}{6} = \frac{33}{6} = 5.5.$$

For the prediction $c = 5.5$,

$$\begin{aligned}\mathrm{RSS} &= (2-5.5)^2 + (2-5.5)^2 + (3-5.5)^2 \\ &\quad + (8-5.5)^2 + (9-5.5)^2 + (9-5.5)^2 \\ &= 12.25 + 12.25 + 6.25 + 6.25 + 12.25 + 12.25 \\ &= \mathbf{61.5}.\end{aligned}$$

For comparison,

$$\mathrm{RSS}(c=4) = 75, \qquad \mathrm{RSS}(c=7) = 75.$$

Thus among these three choices, $c = 5.5$ is best. More generally, the constant c minimizing

$$\sum_i (y_i - c)^2$$

is the sample mean. Therefore a regression-tree leaf using squared-error loss predicts the **mean of the target values in that leaf**.

Extension. If the last 9 is replaced by 15, then

$$\bar{y} = \frac{2+2+3+8+9+15}{6} = \frac{39}{6} = 6.5.$$

The single changed observation moves the leaf prediction by 1. Every observation has the same algebraic weight $1/6$ in the mean, but an extreme observation can move the numerical value of the mean substantially; this is why squared-error regression trees can be sensitive to outliers.

Solution 2. Choosing the best regression split

For each threshold t, the left leaf contains $x < t$ and the right leaf contains $x \geq t$. Each leaf predicts its own mean.

t	left mean	right mean	RSS left	RSS right	total RSS
1.5	2	6.2	0	46.8	46.8
2.5	2	7.25	0	24.75	24.75
3.5	7/3	26/3	2/3	2/3	4/3
4.5	3.75	9	24.75	0	24.75
5.5	4.8	9	46.8	0	46.8

The minimum is attained at

$$\boxed{t = 3.5}, \qquad \mathrm{RSS}_{\min} = \frac{4}{3} \approx 1.333.$$

The depth-1 tree is therefore

$$\hat{y}(x) = \begin{cases} \dfrac{7}{3}, & x < 3.5, \\ \dfrac{26}{3}, & x \geq 3.5. \end{cases}$$

The unsplit tree from Exercise 1 had RSS 61.5. Thus the first split reduces the training RSS from

$$61.5 \quad \text{to} \quad \frac{4}{3}.$$

This numerical example illustrates exactly what the regression-tree splitting rule does: it searches over candidate thresholds and chooses the split with the smallest sum of the two within-leaf squared errors.

Solution 3. Growing the regression tree to depth 2

The first split is $x < 3.5$, so the two children are

$$L: \ (2, 2, 3), \qquad R: \ (8, 9, 9).$$

Left child. The admissible thresholds are 1.5 and 2.5.

- At 1.5: the leaves are (2) and $(2, 3)$, giving total RSS $0 + 0.5 = 0.5$.
- At 2.5: the leaves are $(2, 2)$ and (3), both constant, so the total RSS is 0.

Hence the best left split is **t=2.5**.

Right child. The admissible thresholds are 4.5 and 5.5.

- At 4.5: the leaves are (8) and $(9, 9)$, so the total RSS is 0.
- At 5.5: the leaves are $(8, 9)$ and (9), giving total RSS 0.5.

Hence the best right split is **t=4.5**.

The resulting prediction rule is

$$\hat{y}(x) = \begin{cases} 2, & x < 2.5, \\ 3, & 2.5 \le x < 3.5, \\ 8, & 3.5 \le x < 4.5, \\ 9, & x \ge 4.5. \end{cases}$$

Therefore the depth-2 tree has **training RSS** 0. It exactly reproduces every training target, even though two leaves still contain two observations each.

If splitting continues until every leaf contains one observation, the training RSS also remains 0. The important point is that zero training error does *not* imply good prediction on unseen data. A very deep tree can adapt to noise or isolated observations and therefore have high variance: small changes in the training sample can produce a substantially different tree.

Solution 4. Classification leaves and class probabilities

The labels are

$$0, 0, 0, 1, 0, 1, 1, 1.$$

There are four zeros and four ones, so

$$\widehat{P}(Y = 1) = \frac{4}{8} = 0.5.$$

With majority voting there is a **tie**; a software implementation must use some tie-breaking rule.

If the last label is changed from 1 to 0, there are five zeros and three ones. Thus

$$\widehat{P}(Y = 1) = \frac{3}{8} = 0.375,$$

and the leaf predicts class 0.

If instead the first label is changed from 0 to 1, there are three zeros and five ones. Thus

$$\widehat{P}(Y = 1) = \frac{5}{8} = 0.625,$$

and the leaf predicts class 1.

For the small tree, follow each observation from the root:

(age, income)	prediction	path
$(30, 2500)$	0	age < 35, income < 3000
$(30, 4200)$	1	age < 35, income ≥ 3000
$(50, 2400)$	1	age ≥ 35
$(40, 6000)$	1	age ≥ 35

Thus a classification leaf naturally contains two pieces of information: an estimated class probability (the empirical class proportion) and, after applying a decision rule such as majority vote, a predicted class.

Solution 5. Gini impurity and classification splits

For binary labels,

$$G = 1 - p_0^2 - p_1^2.$$

For the three sample nodes:

$$G_A = 1 - 1^2 - 0^2 = \mathbf{0},$$

$$G_B = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = \mathbf{0.5},$$

$$G_C = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = \mathbf{0.375}.$$

Thus a pure node has Gini impurity 0, while a balanced binary node has the maximum value 0.5.

Now use the labels $(0, 0, 0, 1, 0, 1, 1, 1)$.

Threshold $t = 3.5$. Left: $(0, 0, 0)$, so $G_L = 0$. Right: $(1, 0, 1, 1, 1)$, so

$$G_R = 1 - \left(\frac{1}{5}\right)^2 - \left(\frac{4}{5}\right)^2 = 0.32.$$

Hence

$$G_{\text{split}} = \frac{3}{8}(0) + \frac{5}{8}(0.32) = \mathbf{0.20}.$$

Threshold $t = 4.5$. Both children have proportions $3/4$ and $1/4$, so

$$G_{\text{split}} = \mathbf{0.375}.$$

Threshold $t = 5.5$. Left: $(0, 0, 0, 1, 0)$ gives $G_L = 0.32$; right: $(1, 1, 1)$ gives $G_R = 0$. Thus

$$G_{\text{split}} = \frac{5}{8}(0.32) + \frac{3}{8}(0) = \mathbf{0.20}.$$

Therefore the best value is 0.20, attained by **two equally good splits: $t = 3.5$ and $t = 5.5$**. This is a useful reminder that optimal tree splits need not be unique.

Low Gini impurity means that a node is close to containing only one class. A tree therefore chooses splits that make the children as pure as possible, using the weighted impurity so that the sizes of the children are taken into account.

Solution 6. Tree depth, perfect fitting, and overfitting

The labels are

$$0, 0, 0, 0, 1, 0, 1, 1, 1, 1.$$

A simple split is

$$\boxed{x < 4.5}.$$

The left leaf contains $(0, 0, 0, 0)$ and predicts 0. The right leaf contains

$$(1, 0, 1, 1, 1, 1)$$

and predicts 1. Only the observation at $x = 6$ is misclassified, so

$$\text{training accuracy} = \frac{9}{10} = \mathbf{90\%}.$$

A deeper tree can isolate the unusual point. For example,

$$\hat{y}(x) = \begin{cases} 0, & x < 4.5, \\ 1, & 4.5 \le x < 5.5, \\ 0, & 5.5 \le x < 6.5, \\ 1, & x \ge 6.5. \end{cases}$$

This gives 100% **training accuracy**.

If new observations near $x = 6$ are normally class 1, the simple tree is more plausible: the deep tree has constructed a narrow special region solely to reproduce one unusual training label.

Training error measures performance on observations used to fit the model. **Generalization error** measures performance on new observations drawn from the underlying population. Increasing tree depth almost always makes training error smaller, but after some point it can make generalization worse.

If the label at $x = 6$ is a measurement error, the deep tree reacts much more strongly because it creates extra splits specifically to fit that one bad label.

Solution 7. Bootstrap samples and bagging

A bootstrap sample of size five consists of five draws from $\{A, B, C, D, E\}$ *with replacement.*

Therefore:

- A, B, C, D, E is valid. A bootstrap sample can, by chance, contain every observation exactly once.
- B, B, E, A, E is valid.
- A, C, D, E is not valid for this exercise because it has only four draws.
- C, C, C, C, C is valid, although very unlikely.

In B, B, E, A, E, the absent observations are C **and** D.

Many answers are possible for two new bootstrap samples, for example

$$A, A, C, D, E \qquad \text{and} \qquad B, C, C, E, E.$$

For regression, bagging averages the tree predictions:

$$\hat{y}_{\text{bag}} = \frac{7 + 10 + 8}{3} = \frac{25}{3} \approx \mathbf{8.33}.$$

For classification, the votes are $(0, 1, 1, 1, 0)$, so class 1 receives three of the five votes. The bagged classifier therefore predicts **class** 1.

A deep tree is an unstable estimator: changing the data slightly can change its first few splits and therefore many downstream predictions. Bootstrap trees fluctuate in different ways. Averaging their numerical predictions, or taking a majority vote, tends to cancel part of this fluctuation.

The main target of bagging is therefore not primarily lower bias for each individual tree; it is **lower variance and greater stability of the ensemble prediction**.

Solution 8. From bagging to random forests

At the displayed split, Tree 3 is allowed to use only age and debt, so it is **forced to ignore income**.

If income is the strongest predictor, ordinary bagged trees may repeatedly choose income near the root. Their structures then become similar, and their prediction errors can be strongly correlated. Averaging highly correlated errors gives less variance reduction than averaging errors that fluctuate more independently.

Random forests introduce extra diversity by allowing only a random subset of the features to compete at each split. Some trees are therefore forced to use a weaker feature locally. An individual tree may become slightly worse, but the trees become less correlated, and the *average* can improve.

The requested statements are

- Bagging = bootstrap samples + **decision trees trained independently** + averaging/voting.
- Random forest = bagging + **random subsets of candidate features** at each split.

Conceptually:

- A **single decision tree** is one recursive sequence of data-dependent splits and is easy to inspect directly.
- **Bagging** fits many trees to different bootstrap samples and combines their predictions.
- A **random forest** adds feature randomization to bagging in order to decorrelate the trees further.

A single tree is the easiest model to interpret directly. Bagged ensembles and random forests sacrifice that direct structural interpretability for stability. Among the three, a random forest is typically expected to be the most stable when the feature-randomization step successfully reduces correlation between trees, although the exact comparison is data dependent.