Yale University

Decision Trees, Random Forests, and Gradient Boosting

S&DS 265 · Introductory Machine Learning · Lecture 10

Zhuoran Yang

Fall 2026

AI usage: figures were generated, and these slides polished, with the help of AI tools.

Outline

Learning Objectives

In this lecture you will learn:

1. Why threshold rules are a useful model class when global smooth models miss interactions;
2. How a greedy search over one feature and one threshold at a time grows a tree;
3. Why growth must be stopped, and how pruning chooses the size on held-out data;
4. How bagging averages unstable trees, and what a random forest adds to it;
5. How to compare tree procedures on future data without treating importance as causality.

Outline

Motivating Example: Orange-Juice Purchases

A retailer observes loyalty, prices, discounts, store, and week for 1,070 shopping occasions. It wants the probability that a future customer buys Citrus Hill rather than Minute Maid.

What if a useful rule is mostly thresholds, exceptions, and interactions?

Example and data: ISLP Chapter 8 (James et al., 2023).

The Orange-Juice Data

purchase	LoyalCH	PriceCH	PriceMM	PriceDiff
CH	.500	1.75	1.99	.24
CH	.600	1.75	1.99	-.06
CH	.680	1.86	2.09	.40
MM	.400	1.69	1.69	.00

1,070 shopping occasions: 653 CH purchases and 417 MM purchases. Loyalty and relative price are predictive but do not perfectly separate the classes.

Source: OJ data from James et al., *ISLP* (2023); rows displayed are the first four observations.

Reading One Row

purchase	LoyalCH	PriceCH	PriceMM	PriceDiff
CH	.500	1.75	1.99	.24

On this shopping occasion, loyalty to Citrus Hill was 0.500; Citrus Hill cost $1.75, Minute Maid cost $1.99, and Citrus Hill was $0.24 cheaper. The observed purchase was Citrus Hill.

Problem Setup

Nothing about the problem is new. As in Lectures 3–9, we have $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{\text{CH}, \text{MM}\}$, and we estimate

$$p(x) = \mathbb{P}(\text{Purchase} = \text{CH} \mid X = x)$$

by minimizing empirical risk over a class $\mathcal{F}$.

Problem Setup

Nothing about the problem is new. As in Lectures 3–9, we have $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{\text{CH}, \text{MM}\}$, and we estimate

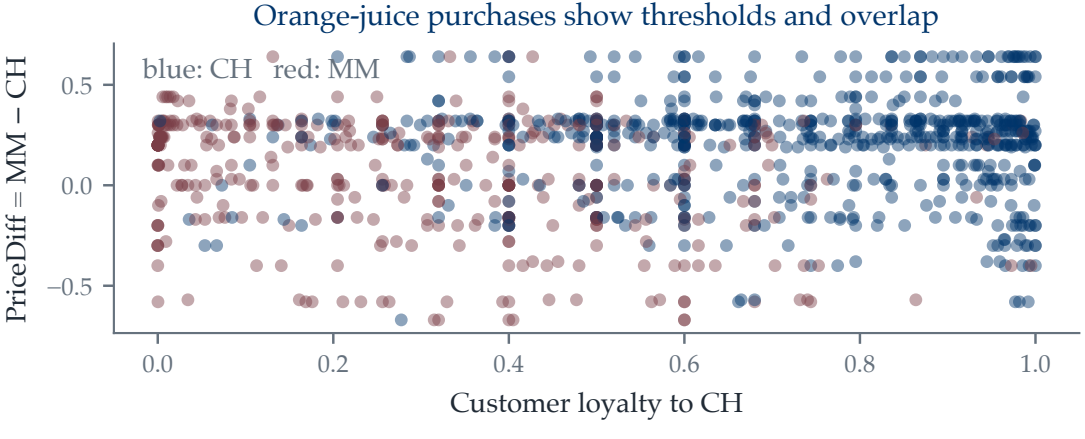
$$p(x) = \mathbb{P}(\text{Purchase} = \text{CH} \mid X = x)$$

by minimizing empirical risk over a class $\mathcal{F}$.

What changes today is only $\mathcal{F}$. Linear scores, polynomials, and kernels all assumed a smooth global form. A **decision tree** assumes something else: a short list of threshold questions.

Protocol: fit on weeks ≤ 260 (629 occasions), choose complexity on weeks 261–266 (138), report once on weeks > 266 (303).

Orange-Juice Purchase Data

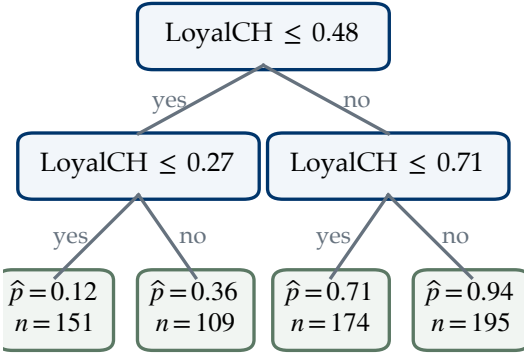


No single line separates the brands: loyalty matters, relative price matters, and their effect changes across the plane.

A Tree Fitted to These Data

One member of that class, fitted on the 629 training occasions:

leaf	n	$\hat{p}$(CH)
LoyalCH $\leq$ .27	151	.12
.27 $<$ LoyalCH $\leq$ .48	109	.36
.48 $<$ LoyalCH $\leq$ .71	174	.71
LoyalCH $>$ .71	195	.94



Source: Depth-2 tree on LoyalCH and PriceDiff; ISLP `OJ`, training weeks $\leq$ 260.

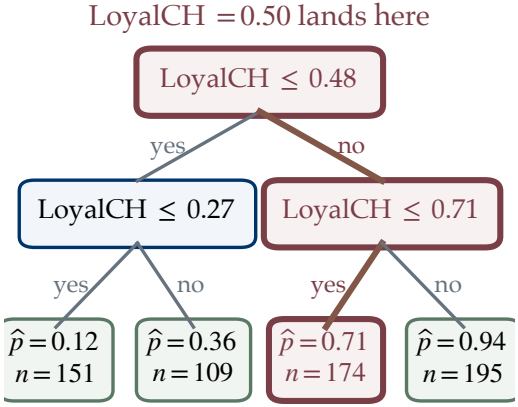
Running One Occasion Through the Tree

Take the row from earlier, LoyalCH = .500:

1. Is LoyalCH $\leq .48$? No — go right.
2. Is LoyalCH $\leq .71$? Yes — go left, and stop.

That leaf holds 174 training occasions, 71% of which bought CH:

$$\hat{p}(\text{CH} \mid x) = .71, \quad \hat{y} = \text{CH}.$$



Does the Fitted Tree Predict Well?

	training (wk $\leq$ 260)	validation (wk 261–266)	future weeks (wk $>$ 266)
four-leaf tree	.811	.848	.789
always predict CH	.579	.696	.637

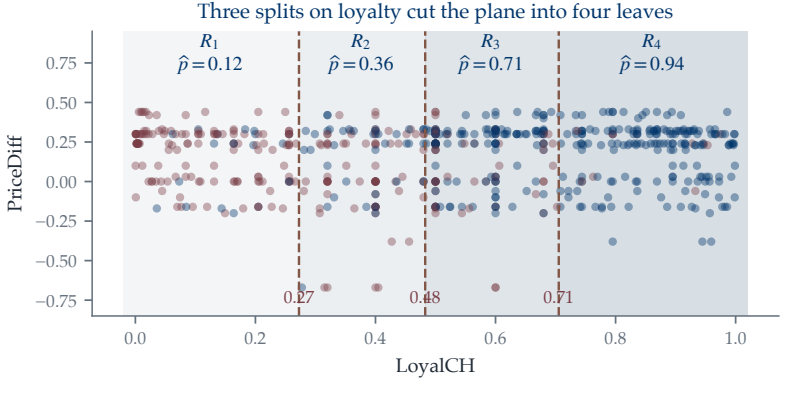
Does the Fitted Tree Predict Well?

	training (wk $\leq$ 260)	validation (wk 261–266)	future weeks (wk $>$ 266)
four-leaf tree	.811	.848	.789
always predict CH	.579	.696	.637

Three questions about one variable already beat the majority rule by 15 points on weeks the tree never saw. The rest of the lecture asks how the questions were chosen, how many to ask, and how to do better.

Source: Accuracy of the depth-2 tree above; course calculation on the ISLP `OJ` data.

The Same Tree, Seen in the Data Plane



The three thresholds are vertical cuts, and each band is one leaf: a region of feature space on which the prediction is constant. Blue points are CH purchases, red are MM, and the shading is the fitted $\hat{p}$.

Training weeks of the ISLP `OJ` data; the depth-2 tree of the previous slides.

What a Tree Is, Precisely

A tree is a set of nested questions. Each **internal node** carries a test $x_j \leq s$; each **leaf** carries a constant.

What a Tree Is, Precisely

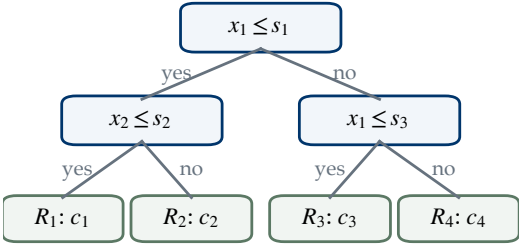
A tree is a set of nested questions. Each **internal node** carries a test $x_j \le s$; each **leaf** carries a constant. Following the tests down to a leaf m intersects finitely many half-spaces, so each leaf is an axis-aligned box R_m, and the leaves $R_1, \dots, R_M$ are disjoint and cover $\mathbb{R}^d$: they partition the feature space. The fitted function is

$$f(x) = \sum_{m=1}^{M} c_m \, \mathbf{1}\{x \in R_m\}.$$

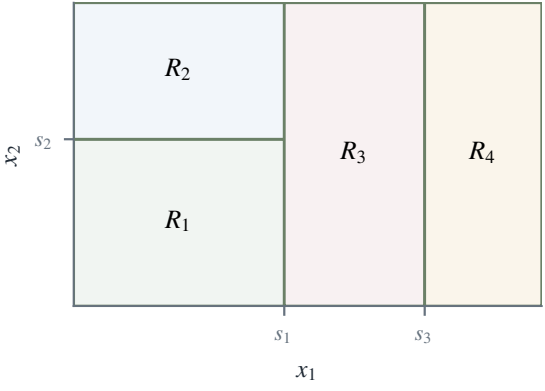
Every x lands in exactly one leaf, so the sum always has exactly one nonzero term.

Any Tree Encodes a Partition into Boxes

Internal nodes ask; leaves answer



Leaves partition the space



The questions on the left and the boxes on the right are the same object.

The Tree Model Class

Collecting all such functions gives the model class

$$\mathcal{F}_M^{\text{tree}} = \Big\{ x \mapsto \sum_{m=1}^{M} c_m \mathbf{1}\{x \in R_m\} : \{R_m\} \text{ a tree partition, } c_m \in \mathbb{R} \Big\}.$$

The Tree Model Class

Collecting all such functions gives the model class

$$\mathcal{F}_M^{\text{tree}} = \Big\{ x \mapsto \sum_{m=1}^{M} c_m \mathbf{1}\{x \in R_m\} : \{R_m\} \text{ a tree partition, } c_m \in \mathbb{R} \Big\}.$$

- Every earlier class fixed the shape and fitted coefficients; here the regions themselves are fitted.
- Complexity is the number of leaves M, not a degree or a penalty.

The Tree Model Class

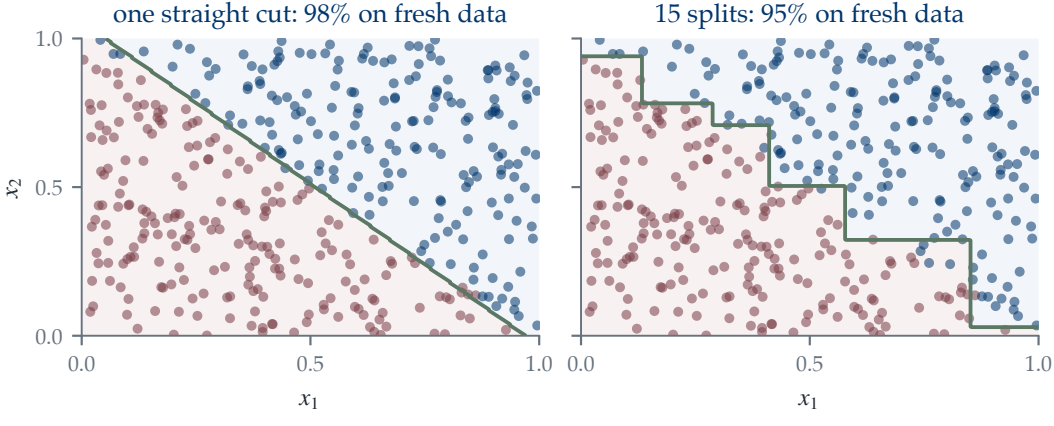
Collecting all such functions gives the model class

$$\mathcal{F}_M^{\text{tree}} = \Big\{ x \mapsto \sum_{m=1}^{M} c_m \mathbf{1}\{x \in R_m\} : \{R_m\} \text{ a tree partition, } c_m \in \mathbb{R} \Big\}.$$

- Every earlier class fixed the shape and fitted coefficients; here the regions themselves are fitted.
- Complexity is the number of leaves M, not a degree or a penalty.

Two consequences follow, one bad and one good — the next two slides show each on simulated data.

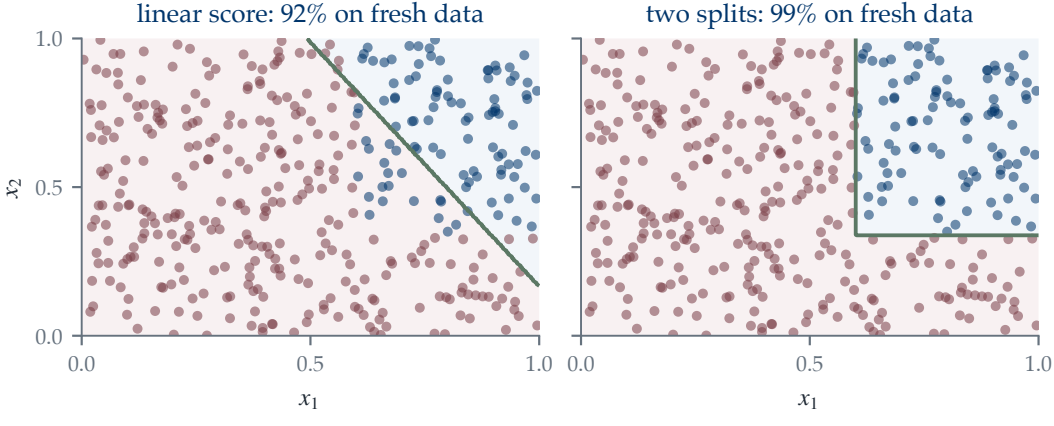
The Price: Boundaries Must Be Axis-Aligned



The truth is the diagonal $x_1 + x_2 > 1$. A linear score states it in one cut; the tree can only approximate it with a staircase, and after 15 splits it is still the worse rule on fresh data.

Simulated data, $n = 400$ training points, accuracy measured on 4,000 fresh points.

The Gain: Thresholds and Interactions Are Free



Here the response is on only when both $x_1 > .6$ and $x_2 > .35$: a sharp threshold, and an effect of x_2 that exists only inside one branch. Two splits capture it exactly; no straight cut can.

Simulated data, $n = 400$ training points, accuracy measured on 4,000 fresh points.

Why the Gain Is Not Free for a Linear Model

A linear score $\beta_1 x_1 + \beta_2 x_2$ adds the two effects: the influence of x_2 is the same at every x_1. To express the previous slide it needs a product term $x_1 x_2$, written down in advance by the analyst.

Why the Gain Is Not Free for a Linear Model

A linear score $\beta_1 x_1 + \beta_2 x_2$ adds the two effects: the influence of x_2 is the same at every x_1. To express the previous slide it needs a product term $x_1 x_2$, written down in advance by the analyst.

A tree writes nothing down. Its second question is asked inside a branch, so the rule for x_2 may differ from branch to branch — that is what **interaction** means, and the greedy search finds it.

Similarly a threshold: a linear score changes gradually with loyalty, while one split states "above .48 is a different regime".

Estimating a Regression Leaf

derive

With the partition held fixed, the constants separate: leaf R only sees its own rows. Minimize

$$Q(c) = \sum_{i:x_i \in R} (y_i - c)^2, \qquad Q'(c) = -2 \sum_{i:x_i \in R} y_i + 2|R|c.$$

Estimating a Regression Leaf

derive

With the partition held fixed, the constants separate: leaf R only sees its own rows. Minimize

$$Q(c) = \sum_{i:x_i \in R} (y_i - c)^2, \qquad Q'(c) = -2 \sum_{i:x_i \in R} y_i + 2|R|c.$$

Setting $Q'(c) = 0$ gives the local sample mean,

$$\boxed{\hat{c}_R = \frac{1}{|R|} \sum_{i:x_i \in R} y_i}.$$

Estimating a Classification Leaf

For class k the same argument returns the local proportion,

$$\hat{p}_{Rk} = \frac{1}{|R|} \sum_{i : x_i \in R} \mathbf{1}\{y_i = k\}, \qquad \hat{y}_R = \arg\max_k \hat{p}_{Rk},$$

which is exactly where the .12, .36, .71, .94 above came from.

Estimating a Classification Leaf

For class k the same argument returns the local proportion,

$$\hat{p}_{Rk} = \frac{1}{|R|} \sum_{i : x_i \in R} \mathbf{1}\{y_i = k\}, \qquad \hat{y}_R = \arg\max_k \hat{p}_{Rk},$$

which is exactly where the .12, .36, .71, .94 above came from.

Few, large leaves

each constant averages many rows: steady, but too crude to follow real structure — bias

Many, small leaves

each constant averages few rows: follows the sample closely, and moves with it — variance

Leaf Size Is the Bias–Variance Knob

A leaf estimate is an average over $|R|$ rows, so its standard error is of order $1/\sqrt{|R|}$: for a proportion,

$$\mathrm{Var}(\hat{p}_R) = \frac{p(1-p)}{|R|}.$$

Leaf Size Is the Bias–Variance Knob

A leaf estimate is an average over $|R|$ rows, so its standard error is of order $1/\sqrt{|R|}$: for a proportion,

$$\mathrm{Var}(\hat{p}_R) = \frac{p(1-p)}{|R|}.$$

At $|R| = 200$ that is about .035; at $|R| = 10$, about .16. Splitting always makes leaves smaller, so every split buys detail with noise.

Leaf Size Is the Bias–Variance Knob

A leaf estimate is an average over $|R|$ rows, so its standard error is of order $1/\sqrt{|R|}$: for a proportion,

$$\mathrm{Var}(\hat{p}_R) = \frac{p(1-p)}{|R|}.$$

At $|R| = 200$ that is about .035; at $|R| = 10$, about .16. Splitting always makes leaves smaller, so every split buys detail with noise.

Training error only ever falls, so it cannot say where the trade sits.
Generalization error decides — which is what the validation weeks are for.

Half the Problem Is Still Open

Given the boxes, the constants are immediate. But the boxes were the model:

$$\min_{\{R_m\},\{c_m\}} \sum_{i=1}^{n} \Big(y_i - \sum_m c_m \mathbf{1}\{x_i \in R_m\} \Big)^2.$$

Half the Problem Is Still Open

Given the boxes, the constants are immediate. But the boxes were the model:

$$\min_{\{R_m\},\{c_m\}} \sum_{i=1}^{n} \Big(y_i - \sum_{m} c_m \mathbf{1}\{x_i \in R_m\} \Big)^2 .$$

Searching over all tree partitions is computationally hopeless — the number of candidate partitions explodes with n and d.

Half the Problem Is Still Open

Given the boxes, the constants are immediate. But the boxes were the model:

$$\min_{\{R_m\},\{c_m\}} \sum_{i=1}^{n} \Big(y_i - \sum_m c_m \mathbf{1}\{x_i \in R_m\} \Big)^2 .$$

Searching over all tree partitions is computationally hopeless — the number of candidate partitions explodes with n and d.

So the partition is built greedily, one question at a time, which is what we develop next.

Outline

What a Node Is

A **node** is a region of feature space together with the training rows that fall in it:

$$R \subseteq \mathbb{R}^d, \qquad S = \{\, i \in \{1, \dots, n\} \;:\; x_i \in R \,\}.$$

What a Node Is

A **node** is a region of feature space together with the training rows that fall in it:

$$R \subseteq \mathbb{R}^d, \qquad S = \{\, i \in \{1, \dots, n\} \;:\; x_i \in R \,\}.$$

So S is a set of row indices, and $|S|$ counts the observations that reached the node. At the root, $R = \mathbb{R}^d$ and $S = \{1, \dots, n\}$: every row.

The region is what the model uses at prediction time; the rows are what the fitting uses. Splitting cuts both at once.

What a Split Is

A **split** adds one interval constraint on one feature. Choosing an entry j and a threshold s cuts the node in two:

$$S_L = \{i \in S : x_{ij} \leq s\}, \qquad S_R = \{i \in S : x_{ij} > s\},$$

with regions $R \cap \{x_j \leq s\}$ and $R \cap \{x_j > s\}$.

What a Split Is

A **split** adds one interval constraint on one feature. Choosing an entry j and a threshold s cuts the node in two:

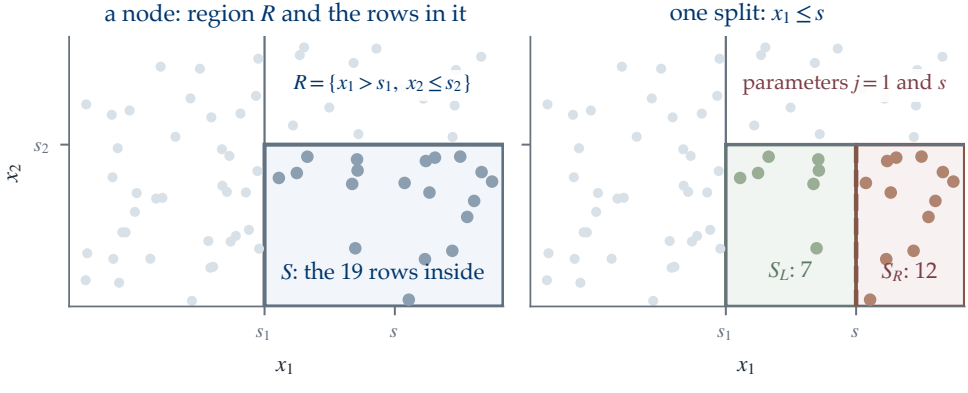
$$S_L = \{i \in S : x_{ij} \le s\}, \qquad S_R = \{i \in S : x_{ij} > s\},$$

with regions $R \cap \{x_j \le s\}$ and $R \cap \{x_j > s\}$.

Growing a tree asks the same two questions over and over: which entry j, and which threshold s.

Each cut is one-dimensional, but the tests stack: a node reached through cuts on x_1 and then x_2 owns a box in both features.

A Node, and One Split of It



Left: two earlier tests, at s_1 and s_2, leave the box R; the 19 dark points inside it are S. Right: one more test, with parameters $j = 1$ and s, sends 7 rows left and 12 right. Regions are multi-dimensional; every cut is one-dimensional.

70 points drawn uniformly, seed 26515.

Purity of a Node

Write the class proportions in a node S as

$$p_k = \frac{1}{|S|} \sum_{i \in S} \mathbf{1}\{y_i = k\}, \qquad p_k \geq 0, \quad \sum_k p_k = 1.$$

Purity of a Node

Write the class proportions in a node S as

$$p_k = \frac{1}{|S|} \sum_{i \in S} \mathbf{1}\{y_i = k\}, \qquad p_k \geq 0, \quad \sum_k p_k = 1.$$

The node is **pure** when one class takes all the mass, $p_k = 1$ for some k, and **maximally impure** when the classes are even, $p_k = 1/K$ for all k. We need one number that is 0 in the first case and largest in the second.

The Gini Index Measures Impurity

Draw two rows from the node independently at random. The chance that their labels differ is

$$G(S) = \sum_k p_k(1 - p_k) = 1 - \sum_k p_k^2$$

The Gini Index Measures Impurity

Draw two rows from the node independently at random. The chance that their labels differ is

$$G(S) = \sum_k p_k(1 - p_k) = 1 - \sum_k p_k^2$$

- $G = 0$ precisely when some $p_k = 1$: a pure node;
- G is largest, at $1 - 1/K$, when $p_k = 1/K$ for all k;
- With two classes $G = 2p(1 - p)$: zero at $p = 0$ or 1, and .5 at $p = \frac{1}{2}$.

So "impurity" is not a metaphor: it is the probability that two members of the node disagree.

Scoring a Split by the Impurity It Removes

The split replaces the node by two children holding $|S_L|$ and $|S_R|$ rows, with $|S_L| + |S_R| = |S|$. Draw a row of the node at random: it lands left with probability $|S_L|/|S|$, right with probability $|S_R|/|S|$.

Scoring a Split by the Impurity It Removes

The split replaces the node by two children holding $|S_L|$ and $|S_R|$ rows, with $|S_L| + |S_R| = |S|$. Draw a row of the node at random: it lands left with probability $|S_L|/|S|$, right with probability $|S_R|/|S|$.

So the impurity a random row now sits in, on average, is

$$\frac{|S_L|}{|S|}\, G(S_L) + \frac{|S_R|}{|S|}\, G(S_R),$$

and a split is worth what that drops below the parent:

$$\Delta G(j, s) = G(S) - \frac{|S_L|}{|S|} G(S_L) - \frac{|S_R|}{|S|} G(S_R) \;\geq 0$$

Why the Children Are Weighted by Size

Without weights, a child holding three rows would count as much as one holding two hundred, and the rule would chase tiny pure corners.

Why the Children Are Weighted by Size

Without weights, a child holding three rows would count as much as one holding two hundred, and the rule would chase tiny pure corners.

With weights, a child of 3 rows out of 200 carries weight $3/200 = .015$, so making it perfectly pure lowers the average by at most $.015\, G(S)$ — while a split that separates the whole node moves both terms at once.

Why the Children Are Weighted by Size

Without weights, a child holding three rows would count as much as one holding two hundred, and the rule would chase tiny pure corners.

With weights, a child of 3 rows out of 200 carries weight $3/200 = .015$, so making it perfectly pure lowers the average by at most $.015\,G(S)$ — while a split that separates the whole node moves both terms at once.

Both choices — entry and threshold — are now decided by one number, and we can go looking for the largest.

A Complete Gini Calculation

Ordered labels $(0, 0, 0, 1, 1, 1)$, so $G(S) = 1 - (.5^2 + .5^2) = .5$.

A Complete Gini Calculation

Ordered labels $(0, 0, 0, 1, 1, 1)$, so $G(S) = 1 - (.5^2 + .5^2) = .5$.

Split after the fourth point: left $(0, 0, 0, 1)$, right $(1, 1)$, giving $G(S_L) = 1 - (.75^2 + .25^2) = .375$ and $G(S_R) = 0$, so

$$\Delta G = .5 - \tfrac{4}{6}(.375) - \tfrac{2}{6}(0) = .25.$$

A Complete Gini Calculation

Ordered labels $(0, 0, 0, 1, 1, 1)$, so $G(S) = 1 - (.5^2 + .5^2) = .5$.

Split after the fourth point: left $(0, 0, 0, 1)$, right $(1, 1)$, giving $G(S_L) = 1 - (.75^2 + .25^2) = .375$ and $G(S_R) = 0$, so

$$\Delta G = .5 - \tfrac{4}{6}(.375) - \tfrac{2}{6}(0) = .25.$$

Split after the third point: both children pure, $\Delta G = .5$. The larger number wins.

Choosing the Threshold s

Fix the entry j, write $v_i = x_{ij}$ for that entry of row i, and sort the values held at the node:

$$v_{(1)} \leq v_{(2)} \leq \cdots \leq v_{(|S|)}.$$

Choosing the Threshold s

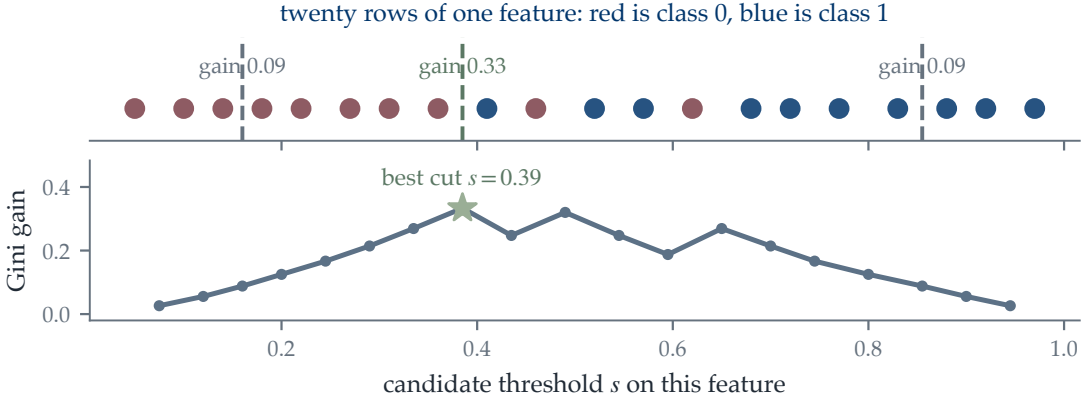
Fix the entry j, write $v_i = x_{ij}$ for that entry of row i, and sort the values held at the node:

$$v_{(1)} \le v_{(2)} \le \cdots \le v_{(|S|)}.$$

A threshold matters only through the set S_L it creates, and every s in the same gap $[v_{(m)}, v_{(m+1)})$ creates the same one. With $|S| - 1$ such gaps leaving both children non-empty, at most $|S| - 1$ distinct splits need trying — one per gap, at its midpoint.

Fewer when values are tied. Evaluate ΔG at each and keep the largest: the search is finite and exact.

Where the Best Cut Sits



Cutting far to either side leaves both children mixed, so the gain is small. The sweet spot is where the two colours separate best, and the curve below reports the gain at every candidate.

Original course calculation on twenty declared points.

Choosing the Entry j

Nothing said which column to cut. So run the same scan on every feature and compare the winners:

$$(\hat{j}, \hat{s}) = \underset{j \le d,\ s \text{ a candidate}}{\arg\max} \quad \Delta G(j, s),$$

a search over d ($|S| - 1$) pairs in total.

Choosing the Entry j

Nothing said which column to cut. So run the same scan on every feature and compare the winners:

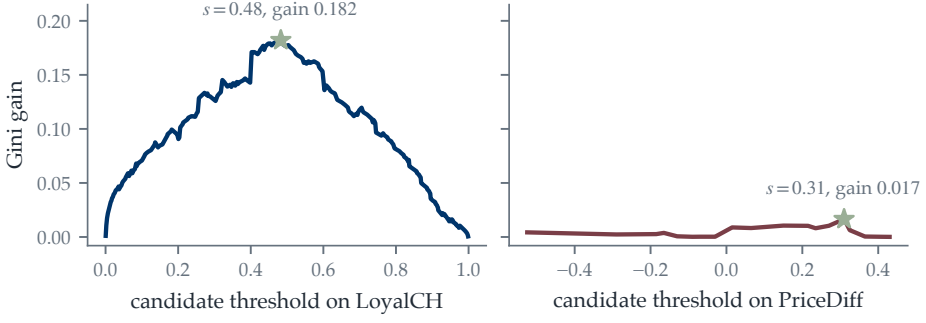
$$(\hat{j}, \hat{s}) = \underset{j \le d,\ s \text{ a candidate}}{\arg\max} \quad \Delta G(j, s),$$

a search over d ($|S| - 1$) pairs in total.

The chosen feature is simply the one whose best cut removes the most impurity — which is why some features are asked about repeatedly and others never.

Scanning Every Feature and Every Threshold

the root split is the largest point on either curve: LoyalCH $\leq$ 0.48



Each curve is the exact gain at every candidate threshold of one feature. The root split is the highest point on either curve: loyalty offers ten times the gain of price difference, so it is asked first.

Course computation on the ISLP `OJ` training weeks.

The Criterion Is Greedy

$\Delta G(j, s)$ measures only what this one split gains right now. It knows nothing about the splits that will follow it.

The Criterion Is Greedy

$\Delta G(j, s)$ measures only what this one split gains right now. It knows nothing about the splits that will follow it.

A cut with a small immediate gain can open up two children that split beautifully, and the rule will never find it.

The Criterion Is Greedy

$\Delta G(j, s)$ measures only what this one split gains right now. It knows nothing about the splits that will follow it.

A cut with a small immediate gain can open up two children that split beautifully, and the rule will never find it.

This is what greedy means: best now, never reconsidered. It is a computational bargain — searching all trees is infeasible — and the reason the fitted tree need not be the best tree of its size.

Greedy Can Miss: A Numerical Example

Twenty rows, two features, five in each quadrant, and $y = 1$ exactly when the two features agree:

	$x_2 \leq .5$	$x_2 > .5$
$x_1 \leq .5$	5 of class 1	5 of class 0
$x_1 > .5$	5 of class 0	5 of class 1

Greedy Can Miss: A Numerical Example

Twenty rows, two features, five in each quadrant, and $y = 1$ exactly when the two features agree:

	$x_2 \le .5$	$x_2 > .5$
$x_1 \le .5$	5 of class 1	5 of class 0
$x_1 > .5$	5 of class 0	5 of class 1

The root has $p = .5$ and $G(S) = .5$. Cut anywhere, on either feature: both children come out 5–5, so

$$\Delta G(j, s) = 0 \qquad \text{for every candidate split.}$$

The Split That Pays Nothing Pays Everything

Take the worthless split $x_1 \leq .5$ anyway. Inside each child, cutting at $x_2 \leq .5$ gives two pure leaves:

$$\Delta G = .5 - \tfrac{1}{2}(0) - \tfrac{1}{2}(0) = .5 \qquad \text{in } \textbf{both} \text{ children.}$$

The Split That Pays Nothing Pays Everything

Take the worthless split $x_1 \leq .5$ anyway. Inside each child, cutting at $x_2 \leq .5$ gives two pure leaves:

$$\Delta G = .5 - \tfrac{1}{2}(0) - \tfrac{1}{2}(0) = .5 \qquad \text{in } \textbf{both} \text{ children.}$$

So a two-level tree classifies all twenty rows correctly, while the criterion scored its first move at exactly zero.

The Split That Pays Nothing Pays Everything

Take the worthless split $x_1 \leq .5$ anyway. Inside each child, cutting at $x_2 \leq .5$ gives two pure leaves:

$$\Delta G = .5 - \tfrac{1}{2}(0) - \tfrac{1}{2}(0) = .5 \qquad \text{in } \textbf{both} \text{ children.}$$

So a two-level tree classifies all twenty rows correctly, while the criterion scored its first move at exactly zero.

A rule that stops when no split helps stops at the root and predicts one class, at 50% accuracy. Depth caps and pruning exist partly to keep such splits alive.

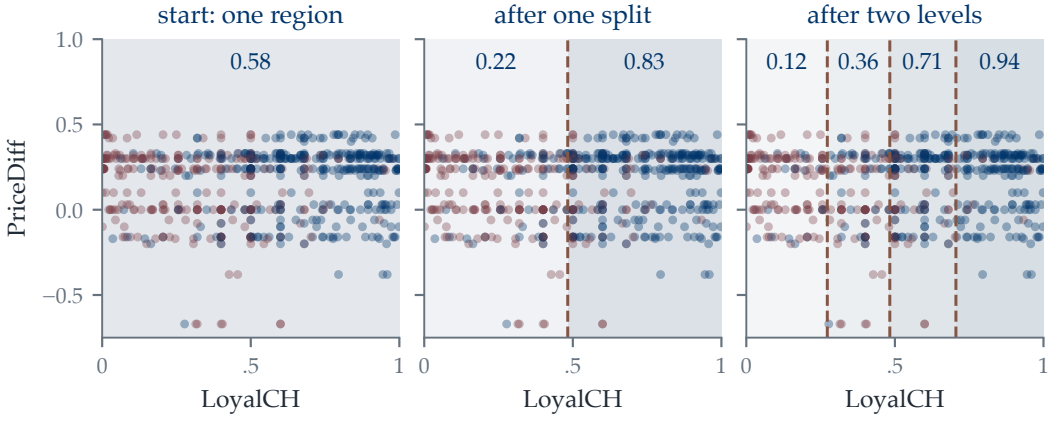
Source: The pattern is the classic exclusive-or example.

The Greedy Growth Algorithm

1. Start with the whole training data in one node: $R = \mathbb{R}^d$, $S = \{1, \dots, n\}$.
2. At a node, evaluate $\Delta G(j, s)$ over all $j \le d$ and all $|S| - 1$ candidate thresholds.
3. Take $(\hat{j}, \hat{s}) = \arg\max_{j,s} \Delta G(j, s)$ and form S_L, S_R.
4. Recurse on S_L and on S_R.
5. Stop when $G(S) = 0$, or $|S| < n_{\min}$, or the depth reaches $L_{\max}$; then set the leaf constant $\hat{c}_S$.

Sorting each feature once makes step 2 cost $O(d|S|)$ per node, so a tree of depth L costs about $O(dnL)$ — trees are cheap to fit, which is what makes the ensembles that follow affordable.

Growing the Tree, One Split at a Time



Each panel is the fit after one more round of step 2–4. The number in a band is that leaf's $\hat{p}(\text{CH})$, and every split makes the bands more homogeneous than the one it replaced.

Course fits on the ISLP `OJ` training weeks; splits chosen by Gini gain.

The Same Search for a Numeric Response

derive

With a numeric y the leaf constants are means, so a candidate split is scored by the squared error it leaves behind:

$$\min_{j,s}\Big\{\sum_{i\in R_L(j,s)}(y_i-\bar{y}_L)^2+\sum_{i\in R_R(j,s)}(y_i-\bar{y}_R)^2\Big\}.$$

The Same Search for a Numeric Response

derive

With a numeric y the leaf constants are means, so a candidate split is scored by the squared error it leaves behind:

$$\min_{j,s}\Big\{\sum_{i\in R_L(j,s)}(y_i-\bar{y}_L)^2+\sum_{i\in R_R(j,s)}(y_i-\bar{y}_R)^2\Big\}.$$

Everything else — finite candidate set, greedy recursion, stopping rules — is unchanged. Only the score differs.

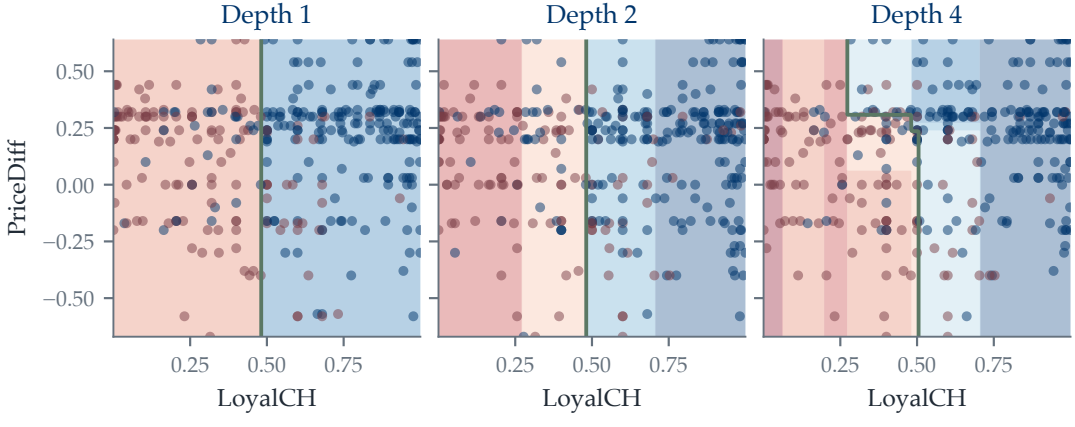
Other Impurity Measures

Gini is the default, not the only choice. Any function of $p_1, \ldots, p_K$ that is zero at a pure node and largest at a balanced one will serve.

- **Entropy**, $H(S) = -\sum_k p_k \log p_k$: the gain is then called information gain. It tracks Gini closely, and the two rarely choose different splits.
- **Misclassification rate**, $1 - \max_k p_k$: the quantity we report, but too blunt to grow with — it can be flat across splits that clearly differ in purity.

Both alternatives leave the algorithm untouched; only the number inside the search changes.

Depth and Partition Complexity



Every extra level doubles the number of regions available. Nothing in the greedy rule ever says stop: it keeps splitting while any gain remains.

Course fits to the same real OJ training data using LoyalCH and PriceDiff.

Outline

Growth Must Be Stopped

Left alone, the recursion runs until every leaf is pure — in the limit, one leaf per observation. The tree then classifies the training data perfectly, and has learned the sample rather than the population.

Growth Must Be Stopped

Left alone, the recursion runs until every leaf is pure — in the limit, one leaf per observation. The tree then classifies the training data perfectly, and has learned the sample rather than the population.

This is the bias–variance trade, with tree size as the knob:

- Too few leaves: each averages many rows, and the fit is too crude — bias;
- Too many leaves: each averages a handful, and the fit moves with the sample — variance.

Growth Must Be Stopped

Left alone, the recursion runs until every leaf is pure — in the limit, one leaf per observation. The tree then classifies the training data perfectly, and has learned the sample rather than the population.

This is the bias–variance trade, with tree size as the knob:

- Too few leaves: each averages many rows, and the fit is too crude — bias;
- Too many leaves: each averages a handful, and the fit moves with the sample — variance.

Training error falls monotonically with size, so it cannot locate the trade. Only held-out error can.

Cost-Complexity Pruning

Write $Q(S)$ for the impurity of a leaf holding rows S:

$$Q(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \hat{c}_S)^2 \ \text{(numeric)}, \qquad Q(S) = G(S) \ \text{(classes)}.$$

Cost-Complexity Pruning

Write $Q(S)$ for the impurity of a leaf holding rows S:

$$Q(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \hat{c}_S)^2 \ \text{(numeric)}, \qquad Q(S) = G(S) \ \text{(classes)}.$$

Grow one large tree T_0, then choose among its subtrees by penalizing the number of leaves:

$$C_\alpha(T) = \underbrace{\sum_{m=1}^{|T|} |S_m| \, Q(S_m)}_{\text{fit}} + \alpha \underbrace{|T|}_{\text{size}}$$

One Number Chooses the Size

This is ridge and lasso logic in a new setting: fit plus a penalty on complexity, with a single number left to choose.

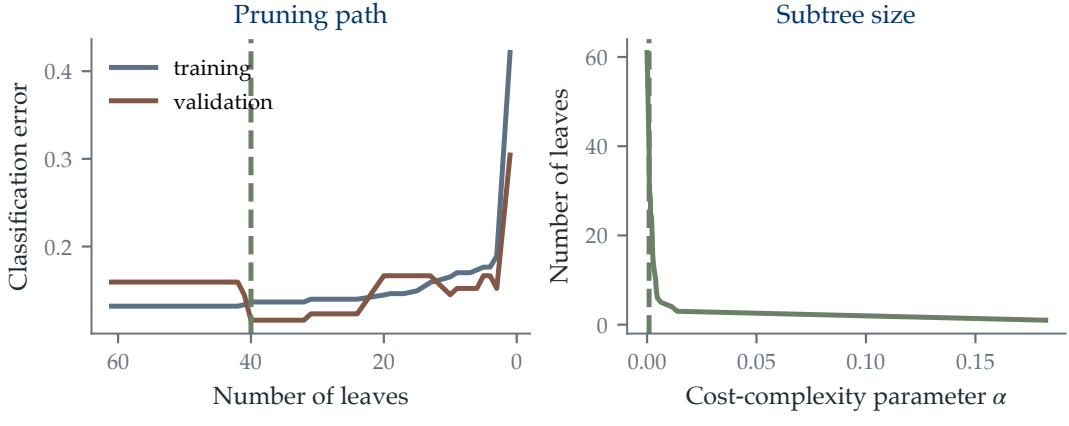
One Number Chooses the Size

This is ridge and lasso logic in a new setting: fit plus a penalty on complexity, with a single number left to choose.

As α grows the chosen subtree only shrinks, so one pass over α produces a nested sequence of candidate trees — and α is selected by validation, exactly as λ was in Lecture 4.

$\alpha = 0$ returns T_0; large enough α collapses the tree to its root.

Pruning on a Chronological Validation Period



Training error falls with every extra leaf; validation error does not. The selected tree is the one minimizing error on weeks the fit never saw.

Course fit on OJ weeks ≤ 260; validation error uses weeks 261–266.

Building a Tree, End to End

The complete recipe

1. **Grow.** Score every candidate split by impurity gain, take the best, recurse.
2. **Stop** at a depth cap or minimum leaf size: a large T_0.
3. **Prune.** Minimize $C_\alpha(T) = \text{fit} + \alpha|T|$ over subtrees of T_0.
4. **Select** α by validation or cross-validation.
5. **Refit** at that α; each leaf reports a mean or a class proportion.
6. **Predict** by dropping x down the questions to its leaf.

Steps 1–2 fit the partition, 3–4 choose its size, 5 fits the constants.

Outline

What a Single Tree Does Badly

A pruned tree is interpretable and cheap, and it has four known weaknesses.

- Unstable: resample the data and an early split can flip, changing every region below it.
- Coarse: predictions are constant on boxes, so a smooth trend becomes a staircase.
- Axis-aligned: a diagonal boundary needs many splits.
- Greedy: each question ignores the questions it makes possible later.

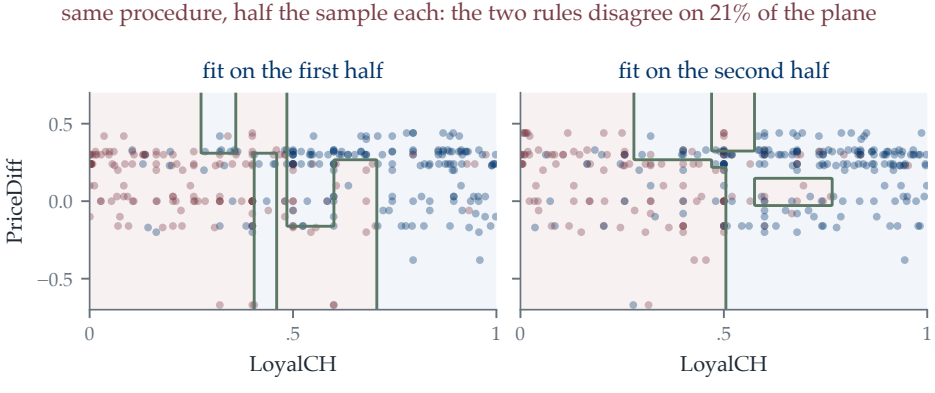
What a Single Tree Does Badly

A pruned tree is interpretable and cheap, and it has four known weaknesses.

- Unstable: resample the data and an early split can flip, changing every region below it.
- Coarse: predictions are constant on boxes, so a smooth trend becomes a staircase.
- Axis-aligned: a diagonal boundary needs many splits.
- Greedy: each question ignores the questions it makes possible later.

The first weakness is the useful one: many unstable fits can be averaged.

The Same Procedure on Two Halves of the Data



Split the training weeks at random into two halves and grow the same depth-5 tree on each. The two fitted rules classify a fifth of the plane differently, though nothing about the population changed.

Course computation on the ISLP `OJ` training weeks; random halves, seed 26515.

Bagging: Refit and Average

Bootstrap aggregation

1. For $b = 1, \dots, B$: draw n data points from the training data with replacement, giving a bootstrap sample $\mathcal{D}_b^\star$.
2. Grow a tree $\hat{f}_b$ on $\mathcal{D}_b^\star$ by the method of the last two sections, deep and unpruned.
3. Predict by averaging: $\hat{f}_{\text{bag}}(x) = \dfrac{1}{B} \displaystyle\sum_{b=1}^{B} \hat{f}_b(x)$.

Bagging: Refit and Average

Bootstrap aggregation

1. For $b = 1, \dots, B$: draw n data points from the training data with replacement, giving a bootstrap sample $\mathcal{D}_b^\star$.
2. Grow a tree $\hat{f}_b$ on $\mathcal{D}_b^\star$ by the method of the last two sections, deep and unpruned.
3. Predict by averaging: $\hat{f}_{\text{bag}}(x) = \dfrac{1}{B} \displaystyle\sum_{b=1}^{B} \hat{f}_b(x)$.

Tree growing is a black box here: nothing about it changes.

Source: Variance of an average of correlated fits: Appendix.

Why the Trees Are Left Unpruned

Averaging does not move the mean: $\mathbb{E}[\hat{f}_{\text{bag}}(x)] = \mathbb{E}[\hat{f}_b(x)]$. It removes variance and leaves bias exactly where it was.

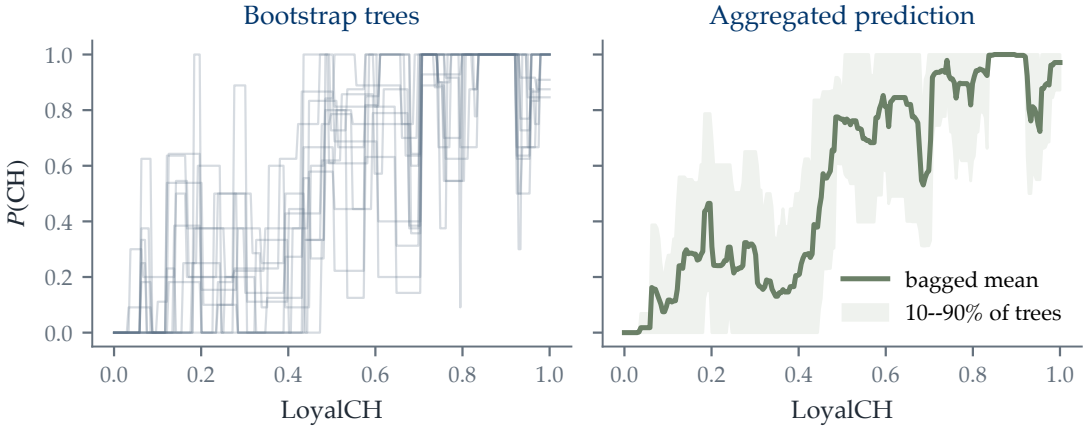
Why the Trees Are Left Unpruned

Averaging does not move the mean: $\mathbb{E}[\hat{f}_{\text{bag}}(x)] = \mathbb{E}[\hat{f}_b(x)]$. It removes variance and leaves bias exactly where it was.

Pruning makes the opposite trade — less variance, more bias. Since the average already handles variance, and can do nothing about bias, each tree is grown deep: low bias, high variance, and let B clean up.

Source: ISL §8.2.1: the trees are grown deep and are not pruned.

Bootstrap Trees and Their Average



Course computation with 40 bootstrap trees on the real OJ training period; PriceDiff is fixed at its median.

Why the Bagged Trees Come Out Alike

The growth algorithm is deterministic: given a sample it returns one tree.
Resampling is the only source of difference between $\hat{f}_1$ and $\hat{f}_2$ — and it is a weak one:

Why the Bagged Trees Come Out Alike

The growth algorithm is deterministic: given a sample it returns one tree. Resampling is the only source of difference between $\hat{f}_1$ and $\hat{f}_2$ — and it is a weak one:

a bootstrap sample contains $1 - (1 - 1/n)^n \approx 63\%$ of the distinct data points, so two bootstrap samples share about

$$(1 - (1 - 1/n)^n)^2 \approx 40\% \text{ of the data.}$$

Fit the same deterministic rule to two samples that overlap this much, and the two trees will mostly agree.

A Dominant Feature Makes It Worse

Suppose one feature has by far the largest gain at the root. Then it wins that comparison under almost every resample, so almost every tree opens with the same question, and the trees below inherit it.

A Dominant Feature Makes It Worse

Suppose one feature has by far the largest gain at the root. Then it wins that comparison under almost every resample, so almost every tree opens with the same question, and the trees below inherit it.

On the OJ data, all 400 bagged trees split on `LoyalCH` first.

A Dominant Feature Makes It Worse

Suppose one feature has by far the largest gain at the root. Then it wins that comparison under almost every resample, so almost every tree opens with the same question, and the trees below inherit it.

On the OJ data, all 400 bagged trees split on `LoyalCH` first.

Averaging strongly correlated predictions leaves a floor that more trees cannot lower. To go further we must make the trees differ.

Source: ISL §8.2.2; the variance floor $\rho\sigma^2$ is derived in the Appendix.

Random Forests: Restrict the Candidates

One line of the growth algorithm changes. At each split, draw a random subset $\mathcal{M} \subseteq \{1, \dots, d\}$ of size m and search only those entries:

$$(\hat{j}, \hat{s}) = \underset{j \in \mathcal{M},\ s \text{ a candidate}}{\arg\max} \Delta G(j, s).$$

Random Forests: Restrict the Candidates

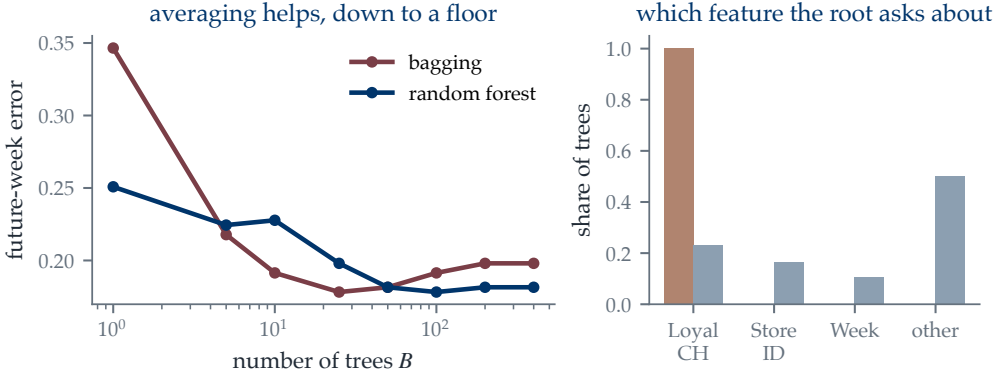
One line of the growth algorithm changes. At each split, draw a random subset $\mathcal{M} \subseteq \{1, \dots, d\}$ of size m and search only those entries:

$$(\hat{j}, \hat{s}) = \underset{j \in \mathcal{M},\ s \text{ a candidate}}{\arg\max} \Delta G(j, s).$$

Bootstrap the data points, average the predictions: everything else is bagging. A tree denied the strongest feature must use another, so the trees stop agreeing.

Usual defaults: $m = \lfloor \sqrt{d} \rfloor$ for classification, $m = \lfloor d/3 \rfloor$ for regression, and $m = d$ is bagging again. Each tree is a little worse on its own; the average is better.

Restricting Features Diversifies the Trees

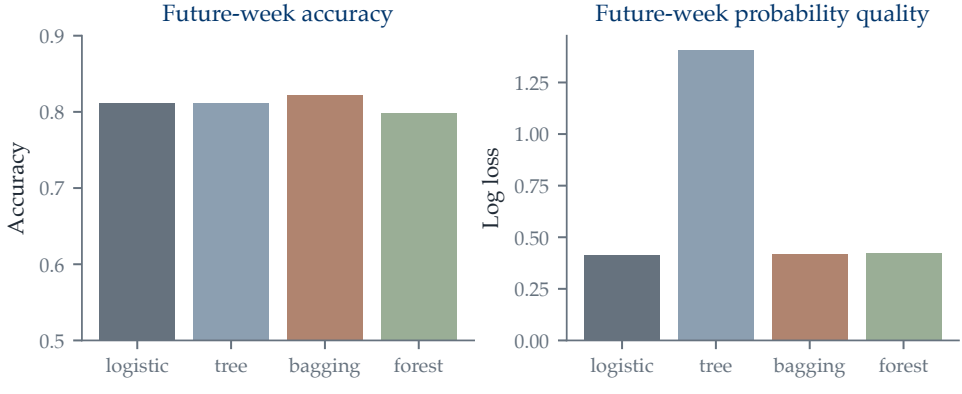


Right: every bagged tree opens with `LoyalCH`, while the forest's trees open with a spread of features. Left: both improve with B, and the forest settles lower — .18 against .20 on the future weeks.

Course computation on the ISLP `OJ` data; $d = 17$ features after encoding, so $m = 4$.

Outline

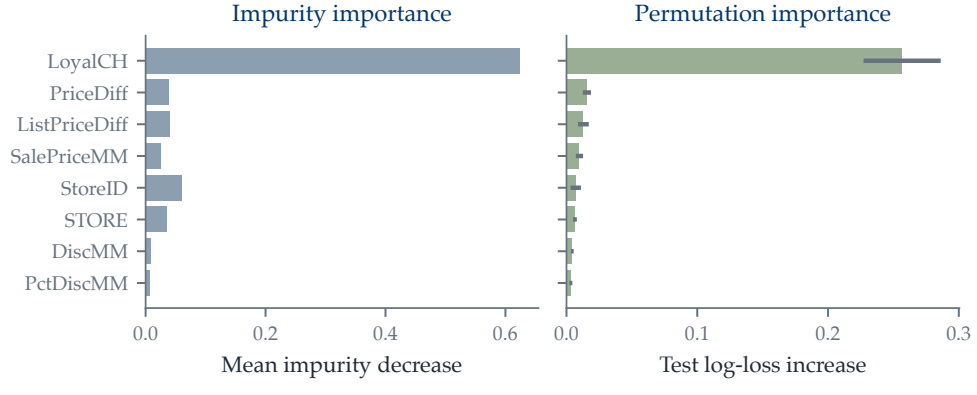
Four Methods on Future Shopping Weeks



All four land within two points of accuracy, but the single tree's probabilities are far worse: its leaves report values like 0 and 1, and log loss punishes that. Both averages fix it, and here bagging and the forest are hard to tell apart.

Course fits to the ISLP `OJ` data; fitted on weeks ≤ 260, tuned on weeks 261–266, reported on weeks > 266.

Which Features the Forest Leans On



Impurity importance adds up the gain a feature won while the trees were grown; permutation importance scrambles one column and measures what held-out loss does. Both describe this fitted model, not a cause.

Course random forest on OJ; permutation importance over 20 repeats; definitions in the Appendix.

What to Use in Practice

- **Random forest** as the default on tabular data: little tuning, handles thresholds and interactions, indifferent to feature scaling and to monotone transforms.
- **One pruned tree** when the rule itself must be read aloud — a forest of 400 trees cannot be.
- **Linear or logistic** when the signal is smooth, coefficients are wanted, or the data are small and wide.

What to Use in Practice

- **Random forest** as the default on tabular data: little tuning, handles thresholds and interactions, indifferent to feature scaling and to monotone transforms.
- **One pruned tree** when the rule itself must be read aloud — a forest of 400 trees cannot be.
- **Linear or logistic** when the signal is smooth, coefficients are wanted, or the data are small and wide.

Neither trees nor forests extrapolate: outside the range of the training data they repeat the nearest leaf. And feature importance describes the fitted model, not a cause.

Summary: Trees and Forests

$$\hat{f}_T(x) = \sum_{m=1}^{|T|} \hat{c}_m \mathbf{1}\{x \in R_m\}, \qquad \hat{f}_{\mathrm{RF}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{(b)}(x).$$

1. A tree is a partition into boxes, grown by greedily maximizing impurity reduction one split at a time.
2. Leaf size is the bias–variance knob, and cost-complexity pruning chooses it on held-out data.
3. Bagging averages trees fitted to bootstrap samples; a forest also restricts the features offered at each split, so the trees differ more.

Next Lecture

Trees choose predictions through recursive partitions, but each split still works with the supplied features. Modern neural networks instead learn a hierarchy of features jointly with the prediction rule.

Lecture 11 — Multilayer Perceptrons, Backpropagation, and Training builds multilayer networks, derives reverse-mode differentiation, and uses initialization, residual paths, and dropout to make training reliable.

Reading for this lecture: ISLP Chapter 8.

Part A

Why Averaging Trees Helps

A.1 Variance of a Correlated Average

derive

Let $Z_1, \dots, Z_B$ each have variance σ^2, with $\mathrm{Cov}(Z_b, Z_{b'}) = \rho\sigma^2$ for $b \neq b'$. Expanding the variance of the average into its B diagonal and $B(B-1)$ off-diagonal terms,

$$\mathrm{Var}\Big(\frac{1}{B}\sum_{b} Z_b\Big) = \frac{1}{B^2}\Big[\underbrace{B\sigma^2}_{\text{diagonal}} + \underbrace{B(B-1)\rho\sigma^2}_{\text{off-diagonal}}\Big] = \sigma^2\Big[\rho + \frac{1-\rho}{B}\Big].$$

A.1 Variance of a Correlated Average

derive

Let $Z_1, \dots, Z_B$ each have variance σ^2, with $\mathrm{Cov}(Z_b, Z_{b'}) = \rho\sigma^2$ for $b \neq b'$. Expanding the variance of the average into its B diagonal and $B(B-1)$ off-diagonal terms,

$$\mathrm{Var}\Big(\frac{1}{B}\sum_{b} Z_b\Big) = \frac{1}{B^2}\Big[\underbrace{B\sigma^2}_{\text{diagonal}} + \underbrace{B(B-1)\rho\sigma^2}_{\text{off-diagonal}}\Big] = \sigma^2\Big[\rho + \frac{1-\rho}{B}\Big].$$

Setting $\rho = 0$ recovers the familiar σ^2/B. Note the mean is unchanged: averaging predictors with a common expectation reduces variance without introducing bias, which is why it is worth doing at all.

A.2 Why About a Third Is Left Out

derive

A bootstrap sample draws n data points with replacement. Data point i escapes a single draw with probability $1 - 1/n$, and the draws are independent, so

$$\mathbb{P}(\text{point } i \text{ omitted}) = \left(1 - \frac{1}{n}\right)^n \longrightarrow e^{-1} \approx 0.368.$$

A.2 Why About a Third Is Left Out

derive

A bootstrap sample draws n data points with replacement. Data point i escapes a single draw with probability $1 - 1/n$, and the draws are independent, so

$$\mathbb{P}(\text{point } i \text{ omitted}) = \left(1 - \frac{1}{n}\right)^n \longrightarrow e^{-1} \approx 0.368.$$

So each tree is fitted without roughly a third of the sample, and each data point is left out by roughly a third of the trees. Averaging the trees that omitted point i gives a prediction that used none of its own label — a held-out estimate obtained without a separate split.

A.3 Out-of-Bag Error Comes Free

Each bootstrap sample leaves out about a third of the data points, since $(1 - 1/n)^n \to e^{-1} \approx 0.368$. Let $\mathcal{B}_i$ be the trees that omitted data point i, and predict it with those alone:

$$\hat{p}_i^{\text{OOB}} = \frac{1}{|\mathcal{B}_i|} \sum_{b \in \mathcal{B}_i} \hat{p}^{(b)}(x_i)$$

A.3 Out-of-Bag Error Comes Free

Each bootstrap sample leaves out about a third of the data points, since $(1 - 1/n)^n \to e^{-1} \approx 0.368$. Let $\mathcal{B}_i$ be the trees that omitted data point i, and predict it with those alone:

$$\hat{p}_i^{\text{OOB}} = \frac{1}{|\mathcal{B}_i|} \sum_{b \in \mathcal{B}_i} \hat{p}^{(b)}(x_i)$$

Data point i's own label never entered those trees, so this is a genuine held-out prediction, obtained without a validation set.

A.3 Out-of-Bag Error Comes Free

Each bootstrap sample leaves out about a third of the data points, since $(1 - 1/n)^n \to e^{-1} \approx 0.368$. Let $\mathcal{B}_i$ be the trees that omitted data point i, and predict it with those alone:

$$\hat{p}_i^{\text{OOB}} = \frac{1}{|\mathcal{B}_i|} \sum_{b \in \mathcal{B}_i} \hat{p}^{(b)}(x_i)$$

Data point i's own label never entered those trees, so this is a genuine held-out prediction, obtained without a validation set.

It assumes the data points are exchangeable: with time-ordered data, out-of-bag error can look better than the future does.

Part B

What Feature Importance Measures

B.1 Impurity Importance

While the trees were grown, every split on feature j removed some impurity. Add those gains up, weighting each node by how many data points reached it, and average over the forest:

$$I_j^{\text{imp}} = \frac{1}{B} \sum_{b=1}^{B} \sum_{\substack{\text{nodes } t \text{ of tree } b \\ \text{that split on } j}} \frac{|S_t|}{n} \Delta G_t.$$

B.1 Impurity Importance

While the trees were grown, every split on feature j removed some impurity. Add those gains up, weighting each node by how many data points reached it, and average over the forest:

$$I_j^{\text{imp}} = \frac{1}{B} \sum_{b=1}^{B} \sum_{\substack{\text{nodes } t \text{ of tree } b \\ \text{that split on } j}} \frac{|S_t|}{n} \Delta G_t.$$

It characterizes how much work the feature did during fitting, on the training data.

It favours features with many distinct values — a continuous column offers hundreds of candidate thresholds, an indicator offers one, so the first gets far more chances to win a split.

B.2 Permutation Importance

Leave the fitted model alone. On held-out data, shuffle column j across data points — keeping its marginal distribution, destroying its link to y — and see what the loss does:

$$I_j^{\text{perm}} = \mathbb{E}[L(\text{model},\ \text{data with column } j \text{ shuffled})] - L(\text{model},\ \text{data}).$$

B.2 Permutation Importance

Leave the fitted model alone. On held-out data, shuffle column j across data points — keeping its marginal distribution, destroying its link to y — and see what the loss does:

$$I_j^{\text{perm}} = \mathbb{E}[L(\text{model},\ \text{data with column } j \text{ shuffled})] - L(\text{model},\ \text{data}).$$

It characterizes how much the fitted model's out-of-sample accuracy depends on that column.

With two strongly correlated columns, shuffling one leaves the other to carry the information, so both can look unimportant.