AI and Machine Learning

Trees, Forests and Boosting

Fatih Kansoy

Oxford
fatih.ai/machinelearning

2026

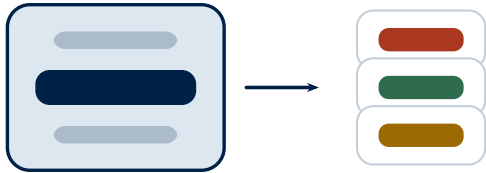
Today's outline

1. When one global rule is not enough
2. A tree builds its own comparison groups
3. How much flexibility, and who chooses it?
4. Many trees instead of one
5. One evolving score
6. Does the flexibility pay?
7. What the score does not promise

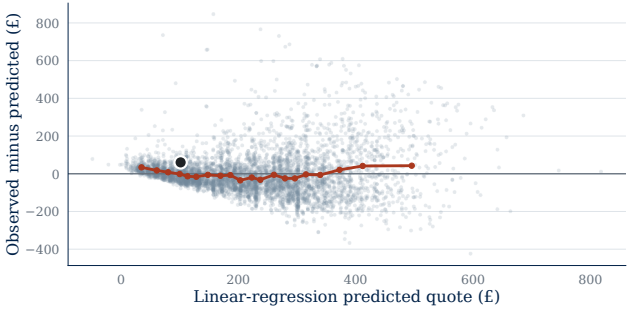
Part 1

When one global rule is not enough

One linear formula priced the whole London market. We start from the misses it left behind, and ask a different question: which past listings should set the price of the next one?



Where the global model left us: honest, with a pattern in the misses



One dot per checking listing (5,000 of 11,063 shown): prediction across, real quote minus prediction up. The red trace averages the misses in twenty price bands, sign kept.

- The trace should sit on zero. Instead: £34 too low at the cheap end, £35 too high near £200, £43 too low at the top.
- Nothing is broken. Right on average, still patterned. A pattern is learnable.
- The **black dot** is one listing we follow all day.

The listing we will follow all day

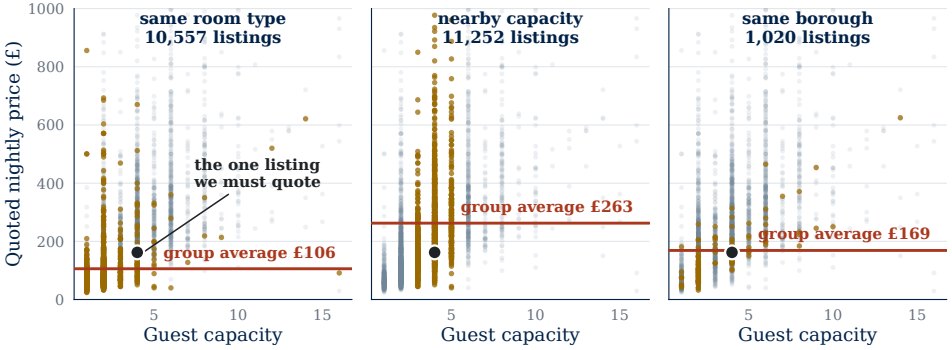
What we know before quoting	Barnet; private room; sleeps four; one bedroom; three beds; one night minimum.
What it was really quoted	£162 for one night, 20 June 2026.
What linear regression predicts	£101.66, so it is £60 too low.

Linear regression adds separate effects, each estimated once across the whole market: a starting level, so much per guest, so much for a private room, so much for the borough.

- A private room sleeping four is not a whole flat sleeping four. One shared capacity effect cannot say so.
- The combination never becomes a case of its own.

So the question is not which formula to use. It is which past listings should set this price.

Three honest rules for who counts as comparable



Every dot is a fitting listing: guests it sleeps, price it was quoted.

- **Black dot**: our Barnet flat, the same in all three panels.
- **Gold**: whoever that rule counts as comparable. Pale: the rest of the market.
- **Red line**: the gold group's average, which is the price that rule would quote.

3,500 listings drawn per panel; counts and averages use all 32,749.

Same flat, three honest rules, three very different prices

Who is comparable	Listings used	Its answer	What it ignores
same room type: any private room	10,557	£106	size: ours sleeps four
nearby capacity: sleeps 3 to 5	11,252	£263	room type: whole flats are mixed in
same borough: anything in Barnet	1,020	£169	size and room type together

Nothing changed but the choice of neighbours, and the answer moved from £106 to £263. The real quote: **£162**.

- Too wide, and you average in listings with little to do with this one.
- Too narrow, and there is almost nothing left to average.
- Borough lands closest here, on one listing. That is luck, not evidence.

The rule we want combines conditions, and combinations run out of evidence

Rule	Listings used	Its answer
every fitting listing	32,749	£228
private room	10,557	£106
private room **and** sleeps 3 to 5	1,202	£159
private room **and** sleeps 3 to 5 **and** in Barnet	25	£166

Two conditions reach £159 against a real £162, on 1,202 listings. The third barely moves the price and leaves 25, where one unusual host swings the average.

- Good rules are combinations, not single conditions.
- Each condition buys sharpness and spends evidence.

We need a procedure that searches these combinations for us, and that knows when to stop.

Three questions we have to answer today

1. **Useful: can flexibility find what one formula missed?** Bends, thresholds, combinations. Parts 2 to 5.
2. **Controlled: how much do we allow?** Depth, group size, rounds, all chosen on fitting data alone. Parts 3 to 5.
3. **Valuable: does it pay on unseen cases?** A smaller quoting error in London, a better probability for the bank. Parts 6 and 7.

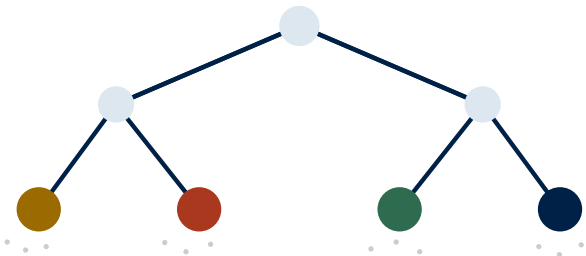
Never whether linear regression is "wrong", but whether a stable local distinction improves it.

Learn the pattern $\longrightarrow$ control the flexibility $\longrightarrow$ measure the payoff.

Part 2

A tree builds its own comparison groups

Start with one number for the whole market. Split it in two only when two averages beat one. Then ask the same question again inside each group.



A tree sorts a listing into a group; it does not score it

From **scoring** to **sorting**:

Model	Route from inputs to prediction
Linear or logistic	weight the inputs into one score $\longrightarrow$ use it, or bend it through the S-curve
Decision tree	answer yes/no questions $\longrightarrow$ land in one group $\longrightarrow$ take its average

> **Our Barnet flat:** sleeps four $\longrightarrow$ sleeps 3 or fewer? no $\longrightarrow$ sleeps 5 or fewer? yes $\longrightarrow$ the group sleeping four or five $\longrightarrow$ quote its average.

- That final group is a **leaf**: in London its average quote, for the Bank its subscription rate.
- Nobody types the questions in. Features and cut points are learned from past quotes.

Start with no questions at all

One quote for everyone. Under squared loss the best single number is the average:

$$\widehat{c}_{\text{root}} = \arg\min_{c} \sum_{i \in \mathcal{D}_{\text{dev}}} (y_i - c)^2 = \bar{y}_{\text{dev}} = £228.38 \qquad (n = 32{,}749).$$

It misses by £172.53 in the squared sense (the typical miss, RMSE) and £129.03 on average (MAE).

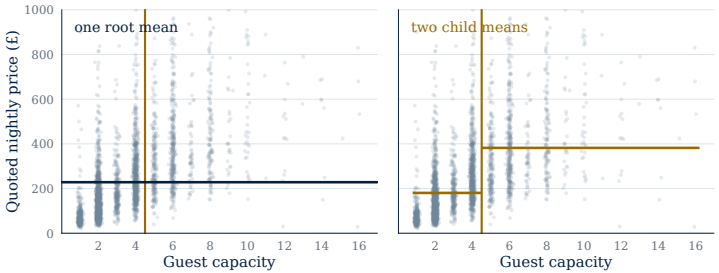
Not a model, the bar: any question must beat £172.53 to earn a place.

It also fixes what every leaf does: predict the average of the cases routed there.

Fitting and checking are what later parts call development and validation. Nothing is refitted on the checking listings today.

A1 · What a leaf predicts under other losses →

Try one question by hand: does it sleep four or fewer?



Rule	Groups and their averages	Typical miss
no question	32,749 listings ⟶ £228.38	£172.53
sleeps 4 or fewer	24,964 ⟶ £180.44 and 7,785 ⟶ £382.10	£149.66

One general average becomes two relevant ones, and the typical miss falls by almost £23. The steps are not drawn by eye: each is the average of its own side.

Software puts a cut half way between two observed values, so “sleeps four or fewer” is written capacity ≤ 4.5.

Score the question: how much of the miss does it remove?

Give each side its average, add up how far the listings sit from it, keep the question that leaves least:

$$\text{Gain}(j, s) = \underbrace{\sum_{i \in P} (y_i - \bar{y}_P)^2}_{\text{one parent average}} - \underbrace{\left[\sum_{i \in L} (y_i - \bar{y}_L)^2 + \sum_{i \in R} (y_i - \bar{y}_R)^2 \right]}_{\text{two child averages}}$$

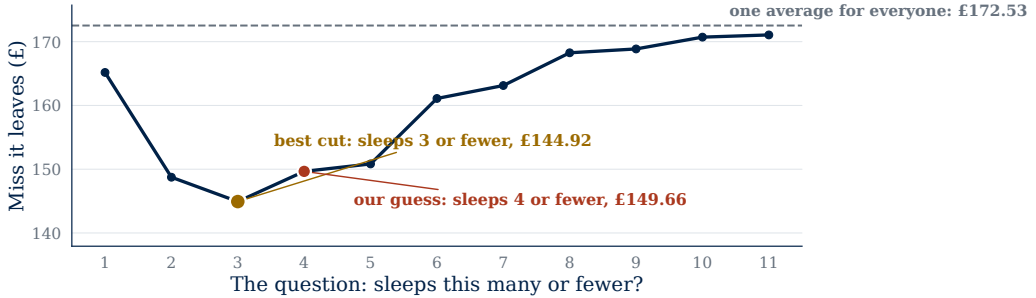
j the feature, s its cut, P the group split, L and R its halves, $\bar{y}_{\cdot}$ a group average. Squares because sums split cleanly across sides; the square root returns pounds.

"Sleeps four or fewer" moves the typical miss **£172.53** $\longrightarrow$ **£149.66**, about £23 a listing.

1. Measured on fitting data, so it proves nothing yet. Part 6 tests honestly.
2. A minimum group size vetoes questions that win on a few odd listings.

A2 · The same score in three algebraic forms →

Let the machine try every cut: it does not share our intuition



Every candidate question, scored the same way on the same 32,749 listings. Lower is better; the dashed line is where we started.

- Our guess loses. Sleeps 3 or fewer leaves £144.92, splitting 18,294 listings at £145 from 14,455 at £334.
- One floor, rising on both sides. Past six guests too few listings remain on the right to separate anything.
- A cut is not a truth about London, only where this sample separates most. Arithmetic, not judgement.

The same search runs over every feature, not only capacity

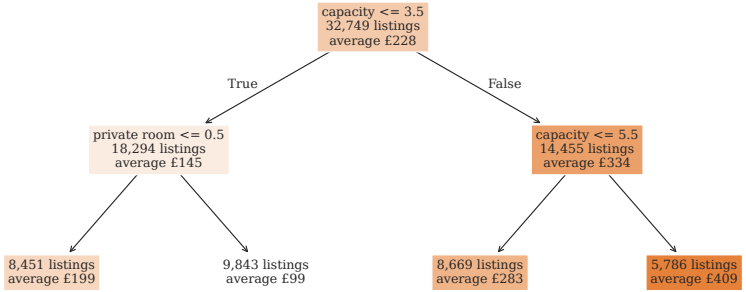
Best question this feature can ask	Groups it makes	Typical miss it leaves
sleeps 3 or fewer?	18,294 | 14,455	**£144.92**
one bedroom or fewer?	21,216 | 11,533	£145.31
a private room?	10,557 | 22,192	£150.37
one bed or fewer?	15,902 | 16,847	£152.66
in Westminster?	4,198 | 28,551	£165.24
one night minimum?	16,502 | 16,247	£172.14

Scored against the £172.53 one average leaves. Part 1's hand-made "private room" rule is one candidate here.

- Capacity beats bedrooms by four pence: two ways of measuring size. Remember that near tie; Part 3 shows its cost.
- Minimum nights earns four pence, so the tree ignores it unprompted.

Exhaustive scan on the fitting listings; a candidate cut must leave at least 250 listings on each side.

Repeat the search inside each group

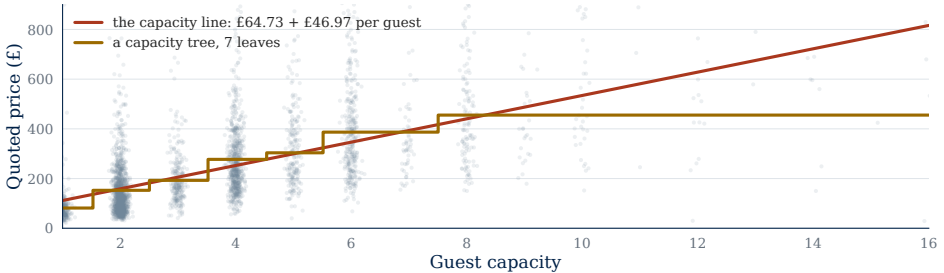


The search runs again inside each half. This tree stops after two questions.

- Among smaller listings, room type wins next: £199 for other rooms, £99 for private rooms. Among larger ones, capacity again at 5.5.
- Each leaf carries its own average and its own support. Nothing outside a leaf touches its prediction.

A3 · Why the tree never revisits an earlier question →

Steps where a line could not bend



A tree on capacity alone, seven leaves, against the capacity line.

- The tree predicts in **steps**: £81 for one guest, £153 for two, £455 for eight upward.
- The line has one slope, £46.97 a guest, so at sixteen it asks £816. The tree stays at £455, which is what the market paid.
- The steps are also the limit: inside one, eight guests and sixteen are the same listing, and past the data it repeats.

A4 · Trees outside the observed range →

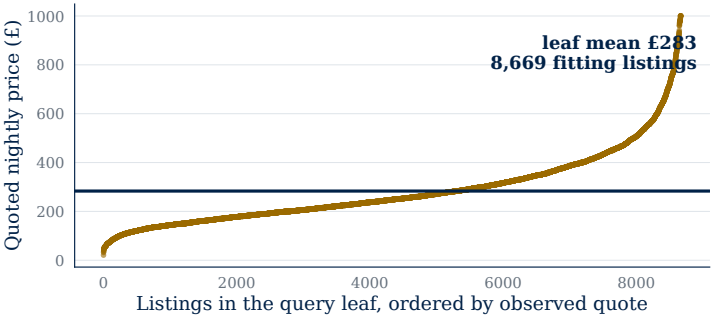
Follow our Barnet flat to its leaf

Step	What the tree does
First question	capacity ≤ 3.5? **No**, four guests, so it takes the larger branch.
Second question	capacity ≤ 5.5? **Yes**, into the leaf sleeping four or five.
Prediction	8,669 listings reached that leaf, averaging **£283.29**.

	Real quote	Linear said	This tree says
our Barnet flat	£162	£102 (£60 low)	£283 (£121 high)

Two questions were not enough. This route never asks about room type, so our private room is quoted like a whole flat sleeping four. The tree used room type on the other branch; this path ran out of questions.

What is actually inside a leaf



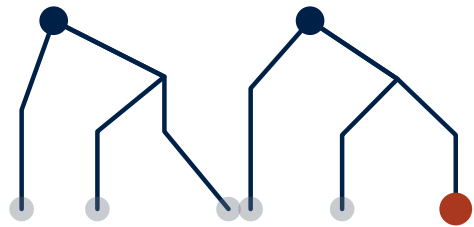
Every quote in our flat's leaf, cheapest to dearest.

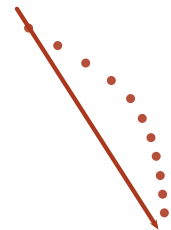
- From £21 to £1,000. Half sits between £183 and £347; only 39% lies within a quarter of the leaf's own average.
- So the prediction means: listings like this were quoted around £283, and "like this" still means only "sleeps four or five".
- More questions would tighten the leaf. Part 3 shows what that costs.

Part 3

How much flexibility, and who chooses it?

A longer path asks a sharper question of fewer listings. Depth and leaf size decide how sharp, and a rule that never touches the checking data decides how much.





Three jobs, three sets of listings

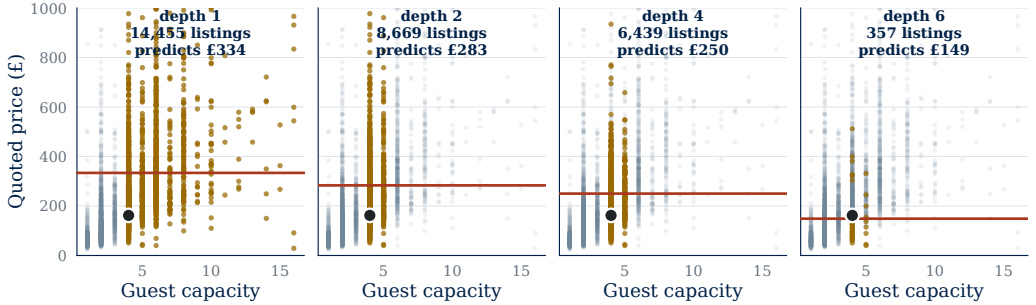
	London listings	Bank contacts
Fitting	32,749: learn the model, choose every setting	24,712: the same job
Checking	11,063 from unseen hosts: compare finished models, once each	8,238 held-out contacts
Final audit	10,824: opened once, at the end	8,238
Currency	average miss per night, in pounds	log loss
The mark to beat	£75.48	0.2718

Every choice today lives in the fitting column. Choose a setting because it scored well on the checking listings, and that score stops being a test.

The Bank is this course's second task: 41,188 telephone contacts from a Portuguese bank selling term deposits, 11.3% of them subscriptions. We predict before the call, so its duration stays forbidden.

Fitting chooses the settings ⟶ **checking** compares finished models ⟶ the **audit** opens once.

A longer path asks a sharper question of fewer listings



Four trees, refitted at four depths on the same listings. Gold is the leaf our Barnet flat lands in; the brick line is that leaf's average.

- Depth one: compared with 14,455 listings, quoted £334. Depth six: compared with 357, quoted £149, against a real £162.
- Each question sharpens the comparison and thins the evidence. Those 357 are one listing in ninety.
- Getting one flat right is not proof. The next two slides choose the depth without looking at this listing.

The second dial: how small may a group get?

Depth limits how many questions a listing faces. This dial limits how few listings a price may rest on.

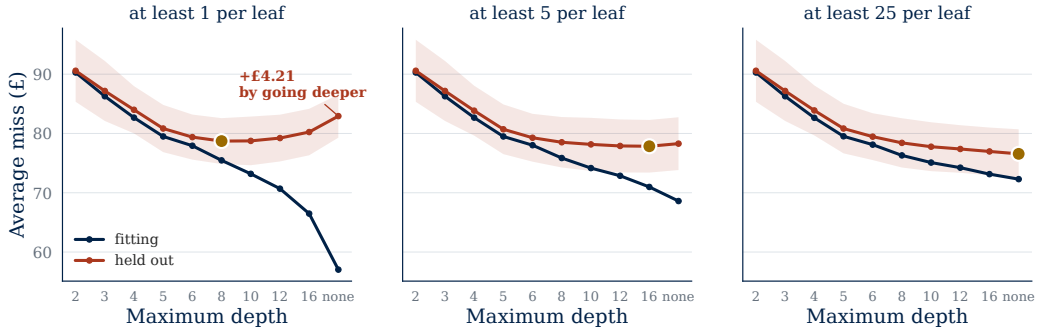
The rule we set: never price a group smaller than	Groups the tree ends up with	Listings behind each price, on average
1 listing	6,743	5
5 listings	2,094	16
25 listings	602	54

The same tree each time, no depth limit at all. Only the first column changes.

- With no rule, each price averages almost nobody. The rule stops the carving; depth did nothing here.
- What it cannot fix: 100 listings from one host are not 100 facts. It counts records, not information.

A5 · Depth, leaf size, and pruning →

Deeper always fits better. It does not always predict better

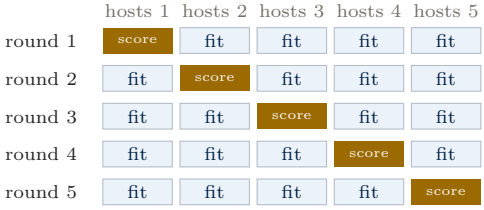


Navy: the miss on listings we fitted. Brick: the miss on listings held out, averaged over five folds, spread shaded. Gold marks the best held-out depth.

- With no floor the curves separate: fitting falls to £57.03 while held-out bottoms at £78.72, then climbs £4.21. That gap is **overfitting**.
- With a floor of 25 the held-out curve never turns up, £90.58 to £76.56.
- Depth alone does not overfit. Depth with nothing in the leaves does.

Five rounds, one score, and the checking listings stay shut

Split the fitting listings into five groups of *hosts*. For one candidate setting, fit five times: each round fits on four groups and scores the one left out.



$$\overline{\mathrm{MAE}}(\text{setting}) = \frac{1}{5}\sum_{j=1}^{5} \mathrm{MAE}_j(\text{setting}), \qquad \widehat{\text{setting}} = \arg\min \ \overline{\mathrm{MAE}}.$$

- Whole hosts move together, so no host sits on both sides of a round. That is deployment's question: a listing from a host we never saw.
- The five trees are thrown away. Keep the winning setting, refit once on all 32,749, then open the checking listings.

A6 · The five rounds, with their numbers →

Five rounds produce one score for each candidate

Round j leaves out one fold and scores it. The five scores differ, because the folds do. Average them, and the candidate has one number:

$$\overline{\mathrm{MAE}}(c) = \frac{\mathrm{MAE}_1(c) + \mathrm{MAE}_2(c) + \mathrm{MAE}_3(c) + \mathrm{MAE}_4(c) + \mathrm{MAE}_5(c)}{5}, \qquad \widehat{c} = \arg\min_{c \in \mathcal{C}} \overline{\mathrm{MAE}}(c).$$

Then repeat for every candidate in the declared grid $\mathcal{C}$, twelve of them:

Candidate	Five round scores	Its one number
depth 3, floor 25	£81 to £95	£87.17
depth 7, floor 25	£74 to £85	£78.77
depth 9, floor 25	**£73 to £85**	**£77.93**
depth 9, floor 150	£74 to £86	£79.36

Sixty trees were fitted to fill that last column, and all sixty are now thrown away. Refit depth 9 with floor 25 once on all 32,749, and only then open the checking listings.

A6 · The five rounds, with their numbers →

What we declared, and what we could have found

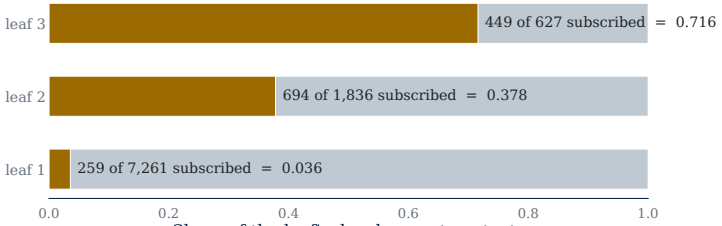
	Settings	Five-fold average miss
The grid we declared	depths 3, 5, 7, 9 by floors 25, 75, 150. Winner: depth 9, floor 25	£77.93
The sweep two slides ago	no depth limit, floor 25	£76.56

The grid we wrote down first did not contain the sweep's best tree, about £1.40 a night better.

- We could add deeper candidates now. Widen a grid after seeing scores and the number we report grows kinder to us than the truth.
- So declare the grid, live with it, and say what was declared. That habit is what makes the audit worth anything.

A3 · Greedy growth is not a global search →

The Bank's leaves are subscription rates

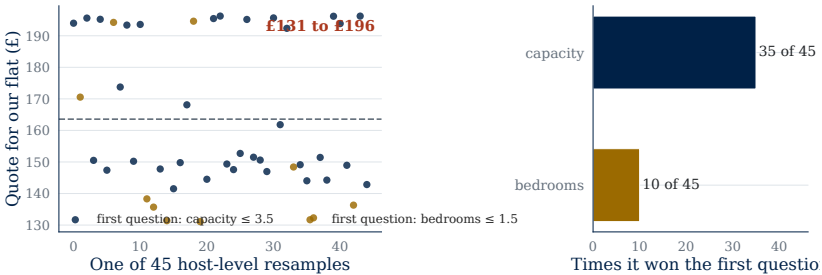


Same machinery, new outcome: will this client subscribe? A tree stopped after three questions; gold is the share who subscribed.

- Its eight leaves run from 3.6% to 71.6%: one tree has split the call list into very different groups.
- A leaf probability is a count over a denominator, 449 of 627. Read the denominator before trusting the rate.
- Bank folds are stratified by outcome, not grouped by client. No client identifier exists, so client-disjoint folds are unavailable.

A7 · What a classification split minimises →

Two size measures, one coin toss



The same recipe refitted 45 times, each on a different four-fifths of the hosts.

- Capacity asks first in 35 trees, bedrooms in ten, always at the same cut. Part 2's near tie: £144.92 against £145.31.
- The quote wanders from £131 to £196. Only the sample changed.

A readable path that changes with the sample is worth less than it looks. So stop trusting any single tree.

Tree vocabulary, in one place

	What it is	Ours
Question	a rule that sends a listing left or right	capacity ≤ 3.5
Depth	the longest chain of questions one listing answers	9, so up to nine
Leaf	the group a listing ends in; its quote is that group's average	170 of them, from 25 listings to 4,768
Leaf floor	the fewest listings a quote may rest on	25

Depth is not the number of questions in the tree. Ours asks 169 of them; any one listing answers at most nine.

- Too few questions and real differences stay averaged together.
- Too many and a quote rests on a handful of listings that happened to be in the sample.

A5 · Depth, leaf size, and pruning →

Cross-validation vocabulary, in one place

	What it is	Ours
Fold	one of five groups of *hosts*, so no host is split	about 6,550 listings
Round	one temporary fit, scored on the fold left out	fit 26,199, score 6,550
Candidate	one setting under test, here a depth with a floor	twelve of them
Its score	the mean of that candidate's five round scores	£77.93 for the winner

Twelve candidates, five rounds each: 60 temporary trees, one winner, and no checking listing touched.

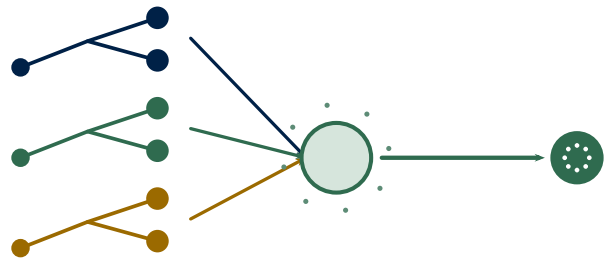
- The score belongs to the *setting*, not to the model. It ranks candidates and nothing else.
- Whole hosts move together because that is deployment's question: a listing from a host we have never seen.

A6 · The five rounds, with their numbers →

Part 4

Many trees instead of one

If one sample gives one answer, take many samples, fit a tree to each, and average the answers. No single tree, and no single early split, gets the last word.



A forest makes trees different, then averages them

Part 3 showed one tree whose first question changed when the sample did. The answer is not a better single tree. It is 400 of them, deliberately unlike each other.

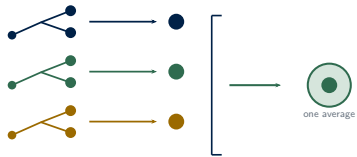
1. **Change the listings.** Draw 32,749 with replacement, once per tree, so each tree sees a slightly different London.
2. **Change the questions on offer.** At every split, show the tree only a random handful of the 39 prepared columns, about six, so no one feature can win everywhere.
3. **Average.** Quote the mean of what the 400 trees say: $\widehat{f}_{\text{forest}}(\mathbf{x}) = B^{-1} \sum_{b=1}^{B} T_b(\mathbf{x})$.

Each tree grows deep, stopped only by a floor of five listings a leaf, a fifth of the single tree's floor of 25. In a forest the averaging does the steadying, so each tree can afford to be sharper than one tree alone dares to be.

Step 1 is the bootstrap, step 2 is feature subsampling. The next three slides take them one at a time, on our own data.

If one sample gives one answer, ask many samples

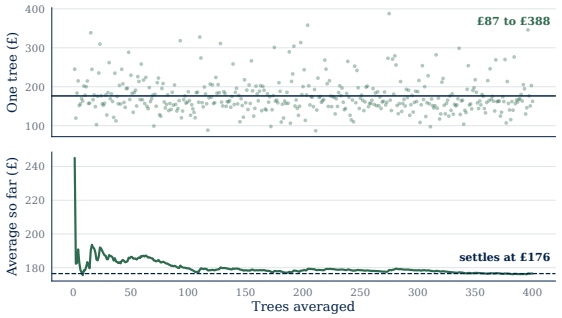
Draw 32,749 listings *with replacement*: some arrive twice, others not at all, and about 63.2% appear. Fit a tree on that, 400 times over.



- Each tree sees a slightly different market, so each asks different questions.
- Not 400 studies of London: one sample shaken 400 times, which is why the trees can share a blind spot.
- Resampling stays inside the fitting listings. The checking hosts are untouched.

A8 · Where 63.2% comes from → **A9 · Bootstrap against cross-validation →**

Four hundred answers, one average

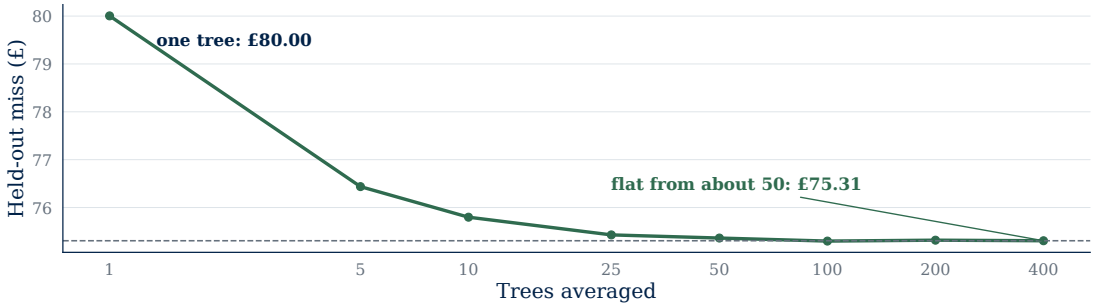


Each dot is one tree's quote for our Barnet flat. Below, the average so far.

- The trees are wild: £87 to £388 for the same flat. Any one of them could have been Part 2's tree.
- The average settles by about 150 trees at £176: $\widehat{f}_B(\mathbf{x}) = B^{-1} \sum_b T_b(\mathbf{x})$.
- For the Bank it averages probabilities, keeping their magnitude instead of voting it away.

A10 · Bagging, forests, and their variants →

How many trees is enough



The same five host-grouped folds as Part 3, now scoring forests of different sizes.

- One tree misses by £80.01. Fifty trees bring it to £75.36, and 400 get no further than £75.31.
- Most of the gain arrives in the first ten. Past fifty, more trees buy computing time and nothing else.
- Note what never happens: more trees never made this recipe worse. Remember that when Part 5 adds rounds instead.

A11 · The forest's own internal estimate →

Averaging only helps if the trees disagree

If one strong feature wins the first question in nearly every tree, the samples differ but the trees barely do.

same first question $\implies$ misses in the same places $\implies$ little left for averaging to cancel.

So the forest handicaps the strongest feature: each node sees only a random handful of columns. Ours offers the square root of 39, about six candidates a question.

Fewer candidates	trees differ more, because the dominant feature is often not on offer.
Too few	trees become weak for no good reason.
Honest wording	subsampling *encourages* disagreement. It does not make the mistakes independent.

Source: Bagging as a remedy for instability: Breiman (1996); strength and correlation: Breiman (2001).

A12 · Strength against correlation →

Why averaging works, and where it stops working

Let σ^2 be one tree's error variance and ρ the correlation between two trees' errors on the same listing. Averaging B of them gives

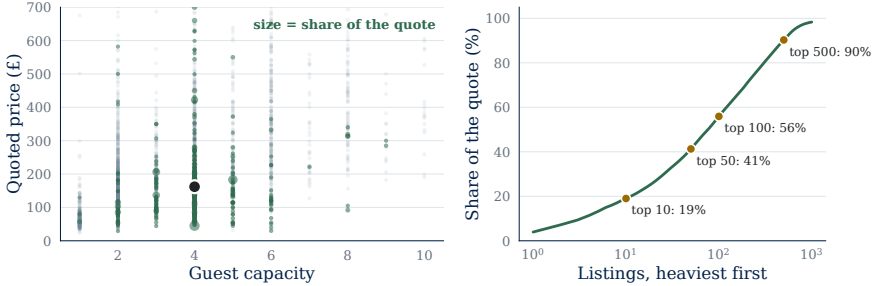
$$\text{Var(forest error)} = \underbrace{\rho\sigma^2}_{\text{shared: stays}} + \underbrace{\frac{(1-\rho)\sigma^2}{B}}_{\text{tree-specific: shrinks with } B} .$$

- Only the second term answers to more trees, which is why our curve fell fast to fifty and then stopped.
- The first term is the floor. Trees that share a blind spot share their misses, and no amount of averaging touches that.
- So a forest needs *different* trees as well as many: ρ is what bootstrap samples and feature subsampling are there to lower.

Equal variances and one common correlation are an idealisation; the point survives without them. This bounds variance, and says nothing about bias.

A13 · The derivation, term by term →

What the forest actually averaged for our flat



How often each fitting listing shares a leaf with our flat, across 400 trees, is the weight it carries: $\widehat{f}(\mathbf{x}_0) = \sum_i w_i(\mathbf{x}_0)\, y_i$.

- Only 2,935 of 32,749 listings carry any weight. The ten heaviest carry 19% of the quote; the heaviest 500 carry 90%.
- All ten are private rooms sleeping two to five. Part 1 built that group by hand, 1,202 listings at £159. The forest found it unprompted.

A14 · The exact weights, with bootstrap counts →

What the forest bought, and what it cost

	Average miss	Gain per listing	Our flat
linear regression	£75.48		£102
one tuned tree	£73.90	£1.58	£175
forest of 400	**£70.87**	**£4.61**	**£176**

Gains are paired listing by listing against linear regression; the forest's 95% host-clustered interval is £3.27 to £5.97. Our flat was really quoted £162.

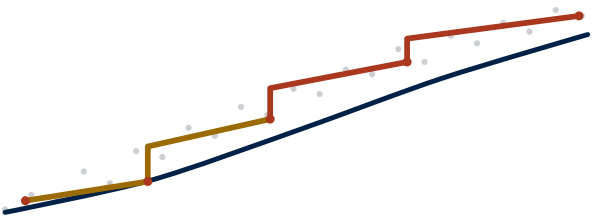
- The forest removes about £4.61 a night, with an interval well clear of zero.
- The price is explanation: one readable path becomes 400, with no single reason. Hence the weights on the last slide.

Averaging steadied the answer. It did not go looking for what the model still gets wrong.

Part 5

One evolving score

A forest grows many trees in parallel and averages them. Boosting keeps one running score and adds a small tree each round to repair what that score still misses.



Two ways to use many trees

	Random forest	Gradient boosting
How they are built	400 trees in parallel, each on its own resample	small trees in sequence, each fitted to what is left
What each aims at	the price	what the current score still misses
How they combine	a plain average	a running sum, each term shrunk
The problem solved	one tree is unstable	one model leaves a pattern

Part 1 ended with a pattern in the misses: too low at the cheap end, too high near £200, too low again at the top. A forest never looks for it. Boosting does nothing else.

Averaging steadies an answer; boosting corrects one.

Start from the best constant, then repair what is left

Begin at Part 2's root, $F_0 = £228.38$. Each round m, look at what the score still gets wrong,

$$r_{im} = -\left.\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right|_{F=F_{m-1}} \overset{\text{squared loss}}{=} y_i - F_{m-1}(\mathbf{x}_i),$$

fit a small tree h_m to those leftovers, add a fraction: $F_m = F_{m-1} + \nu\,\gamma_m h_m$.

r_{im} is case i's leftover at round m, h_m the small tree fitted to it, γ_m the fitted step, ν the learning rate.

- Under squared loss the derivative *is* the residual, so "fit the residual" is a special case, not the definition.
- Gradient earns its keep when the loss changes: the Bank's leftovers are not pounds.

A15 · Boosting as descent in function space →

One correction, worked by hand

Four listings, ordered by one input. The score currently says £100 for all four:

$$x = (1, 2, 3, 4), \qquad F = (100, 100, 100, 100), \qquad y = (80, 90, 140, 150).$$

The leftovers are $(-20, -10, +40, +50)$. Consider the cut $x \le 2.5$: the two sides have leftover averages -15 and $+45$, so the correction tree is

$$h(\mathbf{x}) = \begin{cases} -15, & x \le 2.5, \\ +45, & x > 2.5. \end{cases}$$

With a learning rate of $\nu = 0.5$ we accept half of it:

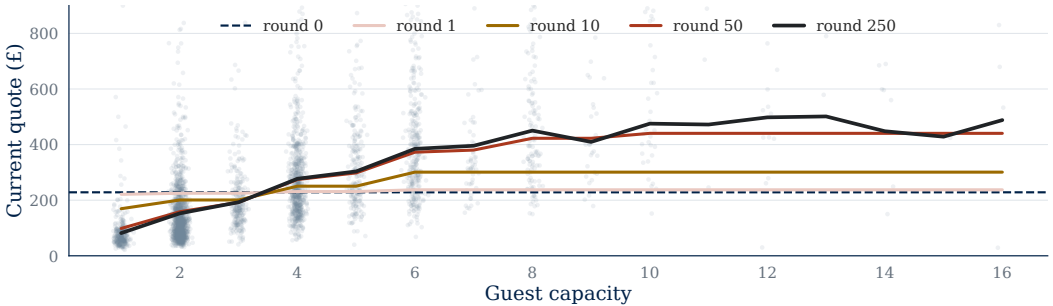
$$F_1 = (92.5,\ 92.5,\ 122.5,\ 122.5).$$

Nothing was replaced. The score moved towards the two groups it was missing; the next round starts from what is left.

Candidate cuts come from the inputs before any leftover is scored; no checking case takes part.

A16 · One full squared-loss round →

Watch the score grow into the shape



Capacity only, so the whole model fits on one pair of axes. Each line is the score after m rounds, not the tree added at m.

- Round 0 is the flat £228. After one round almost nothing moves: at $\nu = 0.05$ each tree may shift the score only slightly.
- By round 50 it has found the staircase Part 2's tree drew in one step, reached by accumulation.
- At round 250 it wobbles above ten guests, where few listings support each step. More rounds fit whatever is left, noise included.

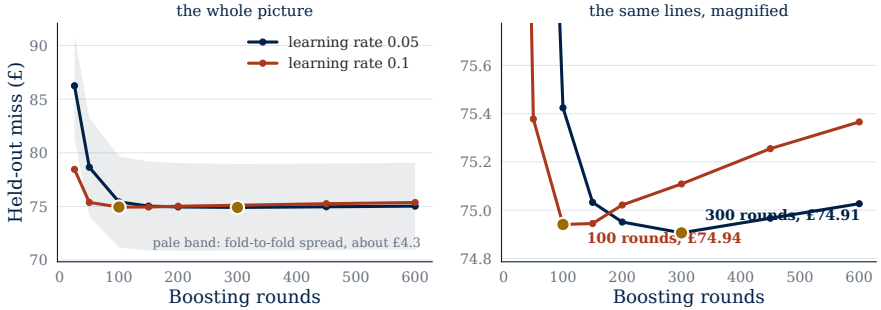
Shrinkage makes each repair deliberately incomplete

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \underbrace{\nu}_{\text{learning rate}} \quad \underbrace{\gamma_m h_m(\mathbf{x})}_{\text{the fitted correction}} \quad .$$

Small ν	each tree nudges, so more rounds are needed.
Large ν	fitting error falls fast, and the score can overshoot what it was chasing.
Why hold back	many small corrections trace a stable pattern more carefully than a few bold ones.

No learning rate is best on its own. It means something only alongside the number of rounds, which is next, and the size of each tree. **A17 · Rate, size, rounds, and early stopping →**

Rounds and learning rate are one dial, not two



The same five folds, scoring every round count at two learning rates.

- The cautious rate needs 300 rounds to reach £74.91; the brisk one arrives by 100, then drifts up. Extra rounds, unlike extra trees, can hurt.
- The minima differ by three pence against a fold spread of £4. We take 300 at 0.05 because the grid says so, not because it matters.
- The alternative is early stopping: watch a held-out slice and stop when it stops improving.

For the Bank, the leftovers are not pounds

The Bank score is log-odds, its probability $p_i = \sigma(F_i)$. Under log loss the leftover is the gap between what happened and what was claimed:

$$-\frac{\partial L(y_i, F_i)}{\partial F_i} = y_i - p_i.$$

What happened	We claimed	Leftover $y - p$, and what the next tree does
subscribed	0.10	$+0.90$: push this kind of contact up hard
did not subscribe	0.10	-0.10: ease it down a little
subscribed	0.80	$+0.20$: the claim was already close

Corrections are added on the score scale and only then squeezed through the S-curve, which is why boosting a probability is not fitting leftover percentages. **A18 · The Bernoulli leaf update →**

What boosting bought

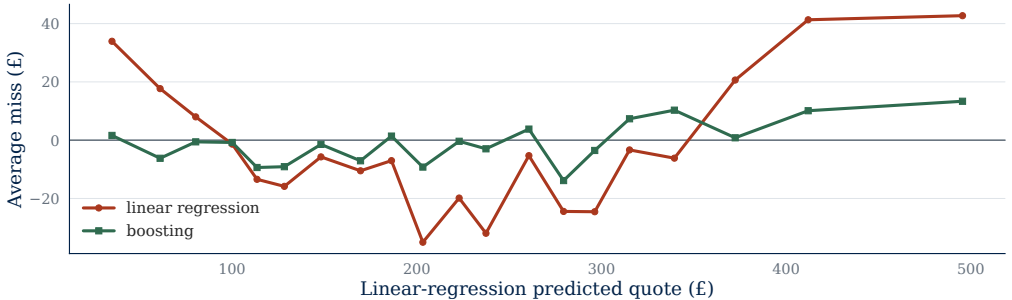
	London miss	Bank log loss	Our flat
linear and logistic	£75.48	0.2718	£102
forest of 400	£70.87	0.2658	£176
boosting	**£69.60**	**0.2652**	**£150**

- London: another £1.27 a night off the forest. Bank: 0.0006 of log loss, which is almost nothing.
- Same code, two tasks, payoffs an order of magnitude apart. The machinery cannot tell us which number matters.

A19 · What XGBoost and friends change →

Three families now beat the global scores on the checking data. Part 6 asks whether any of that is worth having.

The bend we opened with, and what is left of it



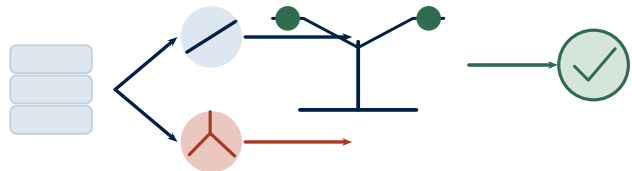
Part 1's twenty price bands, on the same checking listings. Each point is a band's average miss, sign kept, so zero is the target.

- The line's trace is the pattern we started the day with: £34 too low at the cheap end, £35 too high around £200, £43 too low at the top.
- Boosting stays inside £14 everywhere, and is closer in 18 of the 20 bands. Averaged over bands, £5.66 against £18.44.
- Flatter, not flat. A pattern this visible was learnable; what remains is either noise or something these six inputs cannot see.

Part 6

Does the flexibility pay?

Four families, one contract, one set of checking listings. Then the harder question: a better score on paper is not the same thing as a better decision.



One contract for four families

Across all four families, fixed before any result is read:

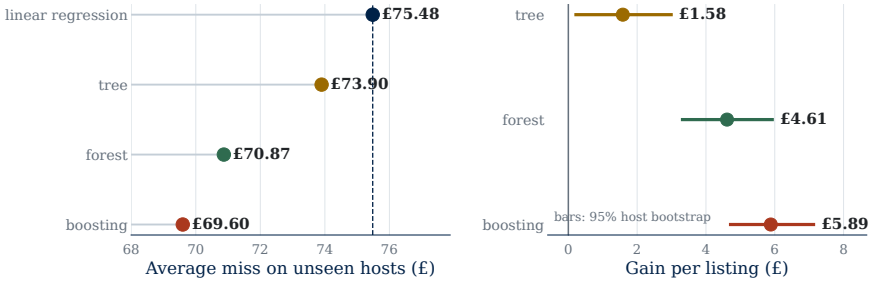
1. the case, target, population, moment of prediction, and forbidden variables;
2. which listings sit in fitting, checking and the audit;
3. the loss we report, and the metrics beside it;
4. that every setting is chosen inside the fitting data.

Each family may prepare the data its own way, provided the pipeline is refitted inside every fold. Fairness is not pretending a tree and a regression want the same matrix.

Compare working systems under one contract, not algorithms under shifting conditions.

A20 · Every grid declared, every setting chosen →

The London verdict, and how sure we are of it



11,063 listings from 5,461 unseen hosts. Gains are measured listing by listing, then resampled by host, because one host's listings miss together.

- Boosting removes £5.89 a night (interval £4.67 to £7.17), the forest £4.61. Both clear of zero.
- The tree's £1.58 is not: its interval nearly touches zero and it wins on only 52% of listings. A coin toss.
- Even boosting is closer on only 58 listings in 100. An average gain is not a promise about the next quote.

A21 · Paired losses and host clustering →

The ranking survives a change of metric

Checking listings	Average miss	Typical miss	R^2
linear regression	£75.48	£114.63	0.549
one tuned tree	£73.90	£114.27	0.552
forest of 400	£70.87	£109.94	0.585
boosting	**£69.60**	**£108.82**	**0.594**

Average miss = MAE, the metric we tuned on. Typical miss = RMSE, harsher on large errors. R^2: the share of the mean baseline's squared error removed.

- All three metrics give the same order, so nothing hangs on which we declared.
- R^2 rises from 0.549 to 0.594: a tenth more of the baseline's squared error removed.
- Remember how ordinary that sounds. Two slides on, the Bank ranks these two ways at once.

A22 · The exact London ledger →

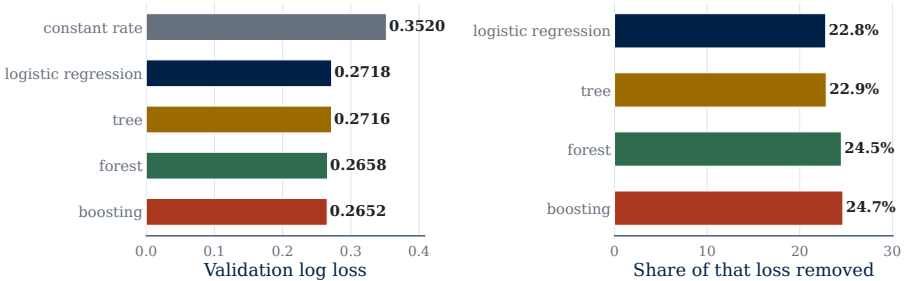
Our flat, judged

	Its quote	Miss
what the flat really charged	£162	
linear regression (Part 1)	£102	£60 low
the depth-two tree (Part 2)	£283	£121 high
the tuned tree	£175	£13 high
the forest	£176	£14 high
boosting	**£150**	**£12 low**

The listing linear regression could not place, and our first shallow tree placed badly. Every tuned family now quotes it within £14.

- Nothing about the flat changed. What changed is which past listings counted as comparable.
- One listing illustrates; it does not prove. The evidence was the last two slides.

The Bank, measured against a constant



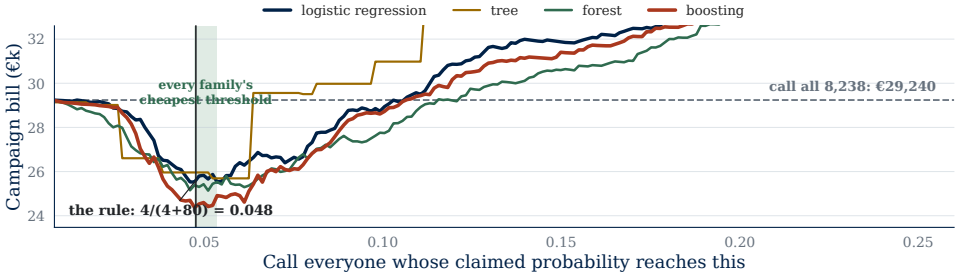
8,238 untouched contacts. The constant claims 11.3% for everybody, and that is the honest thing to beat.

- Logistic regression removes 22.8% of the constant's loss. Today's flexibility adds 1.9 points on top.
- Read the ranking carefully: the tree beats logistic by 0.0002, and forest and boosting differ by 0.0006. Both are nothing.
- In London flexibility was worth a tenth of the baseline error. Here, a rounding error. Same code, different value.

A23 · The exact Bank ledger →

The threshold is a cost decision, not a software default

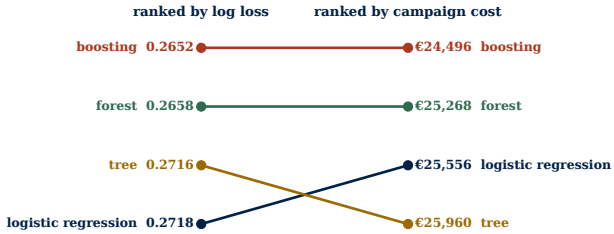
A wasted call costs €4, a missed subscriber €80. Calling costs €4(1-p) in expectation, not calling €80p, so call when the claim reaches $\tau^* = c_{\mathrm{FP}}/(c_{\mathrm{FP}} + c_{\mathrm{FN}}) = 4/84 = 0.048$.



- Every family's own cheapest threshold lands between 0.047 and 0.054. The algebra found the floor of the valley without seeing one outcome.
- Moving the threshold costs more than picking the wrong family: twice τ^* adds €2,100 to €4,100 to the bill.

A24 · The campaign ledger →

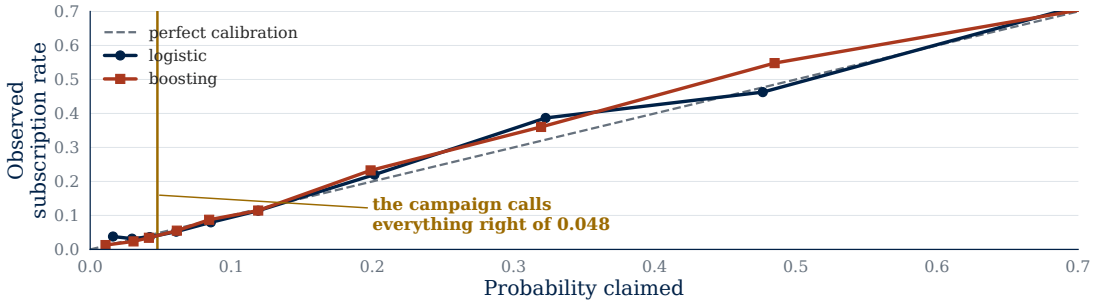
A better score is not a cheaper campaign



The same four families ranked twice: by the score we tuned on, and by the money the campaign pays at τ^*.

- Boosting wins both, beating a blanket campaign by €4,744.
- But the tree beats logistic on log loss and costs €404 more. Two defensible rankings, disagreeing where the scores were nearly tied.
- Move the threshold and the ranking moves: at 0.02 or 0.10 the forest is cheapest. There is no best model, only the best model for a stated decision.

Does the probability mean what it says?

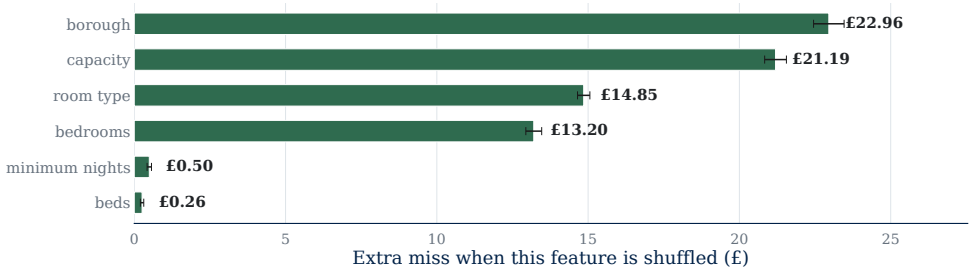


Each point is a bin: probability claimed against share who subscribed. The diagonal is calibration.

- Both hug the diagonal, so 0.30 does broadly mean thirty in a hundred. That is what licenses the campaign arithmetic.
- Gold is the call threshold. Nearly every contact we would ring sits in the leftmost bins, where the plot has fewest events.
- A calibration curve cannot rank models. Keep a proper score for that.

A25 · Bins, counts, and recalibration →

What the fitted model leans on

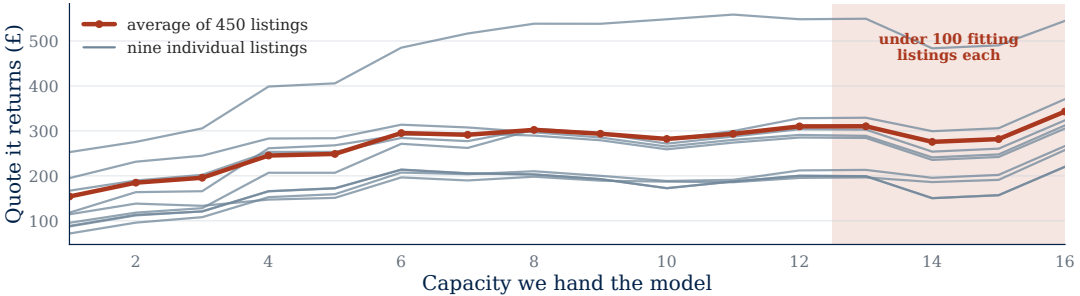


Shuffle one feature among the checking listings and record how much worse the quotes get.

- Scrambling borough costs £22.96, capacity £21.19, room type £14.85, bedrooms £13.20. Beds and minimum nights cost almost nothing.
- Reliance, not effect. Bedrooms looks small partly because capacity stands beside it carrying the same information.
- Not causal, and not a licence to delete a feature. It tells us what to monitor first if a feed breaks.

A26 · Grouped permutation, done properly →

What it does with capacity, and where it stops knowing



Set 450 checking listings to each capacity in turn; average what comes back.

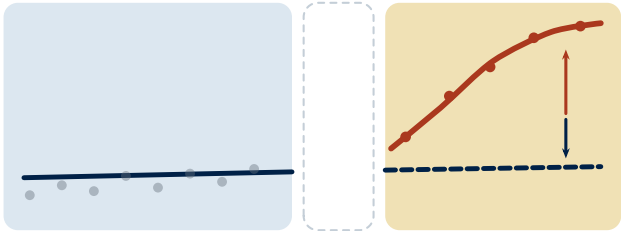
- The average climbs to about six guests, then wanders. Individual listings fan out around it.
- In the shaded tail the fitting listings run thin: the model answers for sixteen guests with almost nothing to answer from.
- A probe of a fitted function, not an experiment. Nobody added guests and watched the price move.

A27 · Partial dependence, ICE, and support →

Part 7

What the score does not promise

Every number so far answers one question: can we predict another case from the same pool? Change the question to a later period and the answers change with it.



Which question is the split answering?

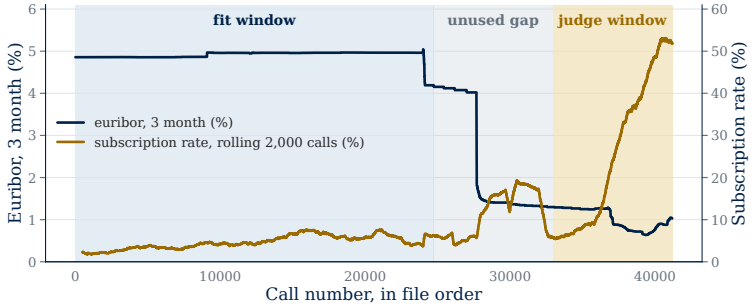
	London listings	Bank contacts
How we split	whole hosts kept together	at random, keeping the outcome mix
Why	one host owns several listings, and we quote for unseen hosts	no client identifier, so clients cannot be kept together
So it answers	quoting for a host we never met	scoring another contact from the same history
And it does not	next season	next quarter

Both are legitimate; neither is a forecast. A precise answer under the wrong split answers a question nobody asked.

The rest of Part 7 uses the Bank: London is one snapshot, 19 June 2026, with no later London to judge against.

Choose the split from the job, before any score.

The Bank file moves under the model



No date column, but the rows are in call order, May 2008 to November 2010. Fit on the first 24,712 calls, leave 8,238 unused, judge the last 8,238.

- Euribor falls from about 4.9% to 1.0%, while the subscription rate rises from 4.8% in the fitting window to 30.8% in the judging window.
- Inputs and outcomes moved together: evidence of a changed environment, not proof that rates caused subscriptions.
- Row order is enough to ask the deployment question, not enough to claim a calendar-accurate forecast test.

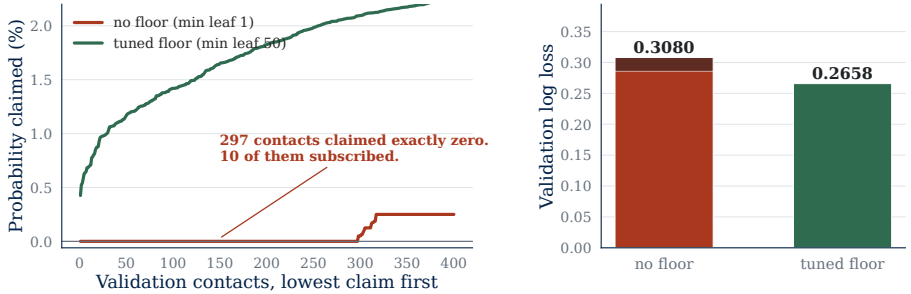
The same models, judged on a later window

	Same pool	Later window	Its average claim
constant rate	0.3520	0.9699	0.048
logistic regression	**0.2718**	**3.6295**	**0.001**
tree	0.2716	1.1621	0.075
forest	0.2658	**0.7880**	0.116
boosting	0.2652	1.0274	0.075

Log loss, lower is better. Same recipes and settings; only the contacts being judged have changed. In the later window 30.8% subscribed.

- Every model is three to thirteen times worse. Logistic keeps claiming one in a thousand while three in ten subscribe.
- Only the forest beats a constant. Our winner, boosting, does not.
- The ranking that decided the contest reverses. The contest was not wrong; it answered the question its split posed.

A floor under the leaves is what keeps a probability honest



The same forest with the tuned floor of 50 per leaf, and with none. The darker band is what ten contacts cost.

- With no floor, 297 contacts are told exactly zero and ten of them subscribed. Log loss punishes a confident zero without limit: those ten add 0.0224.
- The tuned forest never claims below 0.004 and scores 0.2658, not 0.3080. Cross-validation bought that quietly, three parts ago.
- Our tuned *tree* still hands out 47 zeros, one a subscriber. One unlucky contact would have decided its ranking.

Match the split to the job

What you will predict	How to split	Today's example
another case from the same pool	at random, keeping the outcome mix	the Bank contest
a case from a group you have never served	keep whole groups together	London hosts
a case from a later period	fit on the past, judge the future, with a gap if outcomes arrive late	the Bank stress test

- Say who receives the prediction, when, and what it decides. The split follows.
- Fix it before any score is read, and keep it fixed while the families compete.

A model is only tested against the future if the test was built out of the future.

A ledger, not a single winning metric

Family	What it predicts with	What explaining it costs	What it takes to run
linear or logistic	one additive rule	compare coefficients	low
single tree	one set of nested questions	read the path and its support	low to moderate
forest	averaged local groups	weights, probes, no single path	moderate
boosting	a sum of small corrections	the loss path and probes	moderate to high

Record the rest of the bill too: latency, retraining stability, calibration drift, subgroup performance, and what an error costs the person on the other end.

£5.89 a night can be worth a great deal, or less than the governance it adds. That judgement is yours.

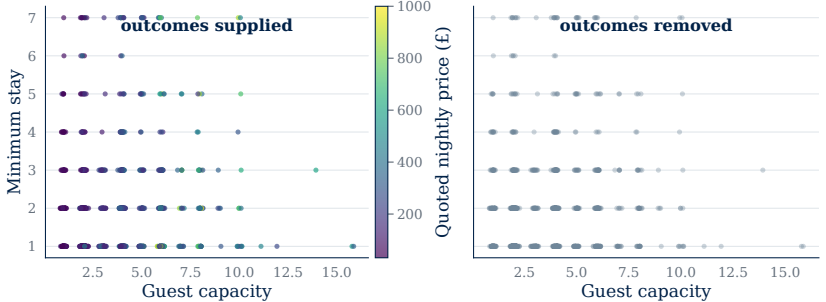
Freeze, refit, and open the audit once

1. tune settings by five-fold resampling inside the fitting listings;
2. choose the family on the checking listings and the ledger;
3. freeze features, preparation, settings, metrics;
4. refit that recipe on fitting plus checking;
5. score once on the audit set, and report it.

	Global baseline	Selected boosting
London audit, average miss	£77.88	£73.06
Bank audit, log loss	0.2733	0.2661

Both are slightly worse than the checking numbers, which is what an honest audit looks like. Report it anyway, and never reopen the audit to repair a story. **A29 · Selection, refitting, and one audit →**

Remove the labels and the question changes



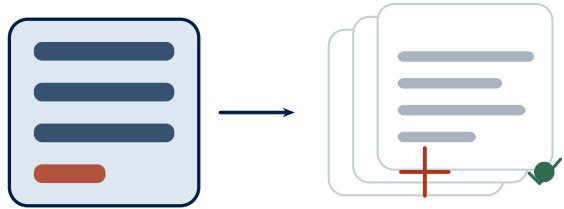
Everything today rested on a supplied outcome, a price or a yes, and was judged by how much of its loss it removed.

- Take the outcome away and “useful” cannot mean lower loss. Nothing is left to be wrong about.
- Next: structure without a target. Which cases resemble each other, and along which directions the data vary.
- Bring today’s discipline. Checking an answer is harder when nothing says it was wrong.

Appendix

Technical appendix

Every claim in the lecture has a page here: the derivation behind it, the exact numbers it quotes, or the assumption it depends on. Each page returns to the slide that sent you.



Technical appendix: map

A1	what a leaf predicts	A11	out-of-bag evaluation	A21	paired losses, host clusters
A2	the squared-error split gain	A12	feature subsampling	A22	the exact London ledger
A3	greedy, not global, search	A13	variance of an average	A23	the exact Bank ledger
A4	outside observed support	A14	exact forest weights	A24	the campaign ledger
A5	depth, leaf size, pruning	A15	boosting in function space	A25	calibration and bin counts
A6	the five rounds, with numbers	A16	one squared-loss iteration	A26	grouped permutation
A7	classification split criteria	A17	shrinkage, size, rounds	A27	partial dependence and ICE
A8	where 63.2% comes from	A18	Bernoulli gradient and leaf	A28	versions, seeds, hashes
A9	bootstrap against folds	A19	what modern variants change	A29	selection, refit, one audit
A10	bagging and forest algorithms	A20	every grid and every setting		

Pages run in the order the lecture sends you to them. They use the formal names for the three sets of cases: *development* is the fitting data, *validation* is the checking data, and the final audit is the sealed set.

Appendix A1: what a leaf predicts

For observations $i \in R$, a terminal node predicts the constant that minimises the declared within-leaf loss.

$$\widehat{c}_R = \arg\min_c \sum_{i \in R} L(y_i, c).$$

Squared loss $\widehat{c}_R = |R|^{-1} \sum_{i \in R} y_i$, the leaf mean.

Absolute loss any leaf median minimises $\sum_{i \in R} |y_i - c|$.

Bernoulli log loss on the probability scale, $\widehat{p}_R = |R|^{-1} \sum_{i \in R} y_i$; practical implementations may regularise or smooth score updates.

Thus "a tree predicts averages" is loss-specific. Change the loss and the loss-optimal leaf summary can change.

← **Return to the root prediction**

Appendix A2: the full squared-error split gain

For parent node $P = L \cup R$, define

$$\mathrm{SSE}(A) = \sum_{i \in A} (y_i - \bar{y}_A)^2.$$

The candidate split gain is

$$\Delta(P \to L, R) = \mathrm{SSE}(P) - \mathrm{SSE}(L) - \mathrm{SSE}(R).$$

Using the within–between decomposition,

$$\Delta = |L|(\bar{y}_L - \bar{y}_P)^2 + |R|(\bar{y}_R - \bar{y}_P)^2 = \frac{|L||R|}{|P|}(\bar{y}_L - \bar{y}_R)^2 \geq 0.$$

Gain grows when the child means are far apart and when both children contain substantial mass. A minimum-leaf constraint is still needed because a rare, extreme subgroup can generate a large development gain.

← **Return to split gain**

Appendix A3: greedy growth is not global optimisation

The space of feature choices, thresholds, and tree topologies grows combinatorially. Standard CART therefore chooses the best immediate split, conditions on that choice, and continues recursively.

Greedy question	which admissible split gives the largest loss reduction now?
Global question	among all trees satisfying a size constraint, which complete partition minimises the objective?
Consequence	a modest first split that enables excellent later splits can lose to a strong immediate split that leads to a poor final partition.

Post-pruning partly revisits size, but it does not undo the fact that the maximal candidate tree was grown through greedy decisions.

← **Return to recursive fitting**

Appendix A4: trees outside observed support

A regression tree is piecewise constant:

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

M is the number of terminal leaves; R_m is leaf m; and $\widehat{c}_m$ is that leaf's fitted constant prediction.

If a new value lies beyond every observed value on a feature, it is still routed left or right by the fitted thresholds and receives a terminal constant.

What continues	the routing rule and its final leaf value.
What does not exist	an estimated slope describing how the outcome should continue beyond support.
Operational requirement	flag support violations or constrain the use population; do not mistake a mechanically available prediction for empirical evidence.

← **Return to tree capabilities and limits**

Appendix A5: depth, leaf size, and cost–complexity pruning

Pre-pruning constrains growth through maximum depth, minimum leaf size, minimum split size, or minimum gain. Cost–complexity pruning instead considers subtrees of a grown tree:

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}|,$$

where $R(T)$ is training leaf loss and $|\widetilde{T}|$ is the number of terminal nodes. Larger α pays a higher price for each leaf, producing a nested pruning sequence. The choice among constrained trees belongs inside development resampling. Validation and final audit data do not select α. Depth controls path length; leaf size controls recorded local support; pruning controls whether a fitted split earns its complexity cost. They are related but not interchangeable.

← **Return to the leaf-size dial**

Appendix A6: the five rounds, with their numbers

The 32,749 development listings belong to 16,381 hosts, dealt whole into five folds. One candidate, depth 9 with a leaf floor of 25, is fitted five times, each round on 26,199 listings and scored on the fold left out:

Round	1	2	3	4	5
listings scored	6,550	6,550	6,550	6,550	6,549
hosts scored	3,273	3,277	3,277	3,277	3,277
their average price	£239	£225	£233	£222	£223
average miss	**£84.60**	**£76.33**	**£77.42**	**£77.89**	**£73.45**

Its score is the mean of the last row, £77.93, with a spread of £4.10.

- The rounds differ because the folds do: the dearest fold, at £239 a night, is also the hardest to quote.
- Twelve candidates were scored this way, so 60 trees were fitted and all 60 discarded. The winner refitted on all 32,749 misses £73.90 on the checking listings.
- Each round fits four fifths of the data, so the average reads a little pessimistically against that refit.
- Folds inherit the split design: grouped folds ask deployment's question, shuffled folds on calendar data flatter the model, and neither detects leakage.

← **Return to the five rounds**

Appendix A7: classification split criteria

For a binary node with event proportion p:

$$\text{Gini}(p) = 2p(1-p), \qquad \text{Entropy}(p) = -p\log p - (1-p)\log(1-p).$$

For impurity I, the reduction from parent P is

$$\Delta_I = I(P) - \frac{|L|}{|P|}I(L) - \frac{|R|}{|P|}I(R).$$

Both criteria are smallest for pure nodes and largest near $p = 1/2$, but their numerical scales differ. They choose partitions; the leaf probability remains an event-rate estimate unless a later calibration or smoothing step changes it. Probability quality must finally be judged by a proper scoring rule such as log loss or Brier score, not by training impurity alone.

← **Return to the Bank leaf**

Appendix A8: why a bootstrap sample contains about 63.2% distinct cases

For a particular record, the probability it is omitted on one of n draws is $1 - 1/n$. The probability it is omitted from all n draws is

$$\left(1 - \frac{1}{n}\right)^n \longrightarrow e^{-1} \approx 0.368.$$

Therefore the probability it appears at least once tends to

$$1 - e^{-1} \approx 0.632.$$

The sample still contains n draws; repeated records account for the difference between n draws and roughly $0.632n$ distinct records. Under clustered data, record-level resampling does not preserve a cluster-level deployment design unless the ensemble procedure is deliberately changed.

← **Return to bootstrap perturbations**

Appendix A9: bootstrap and cross-validation do different jobs

Both resample the same 32,749 development listings, for opposite reasons.

	Bootstrap, inside a forest	Five-fold cross-validation
how it draws	32,749 draws with replacement, once per tree	five host-disjoint folds; nothing is drawn twice
cases per fit	32,749 positions holding about 20,702 distinct listings, with about 12,047 absent	26,199 listings, each appearing once
what it is for	make the trees disagree, so that averaging has something to cancel	estimate what one candidate setting scores on unseen hosts
afterwards	keep every tree and average them	average the scores and discard the models

A forest uses both at once: the bootstrap builds its 400 trees, and the rotation outside it chooses the leaf floor and the feature rule. Record-level resampling does not preserve the host-level design, which is why the bootstrap is a fitting device here and never the evidence.

← **Return to bootstrap perturbations**

Appendix A10: bagging and random-forest algorithms

	Bagged trees	Random forest
for $b = 1, \ldots, B$	draw a bootstrap sample	draw a bootstrap sample
at each node	search all eligible features	sample m_{try} features, then search them
tree growth	usually deep, subject to leaf rules	usually deep, subject to leaf rules
combine	average values or probabilities	average values or probabilities

Extra-trees add further threshold randomisation. Honest forests separate data used to choose splits from data used to estimate leaf values. These are meaningful algorithmic variants, not synonyms for any collection of trees.

← **Return to forest averaging**

Appendix A11: out-of-bag evaluation is useful, but it is not the final audit

For record i, let $\mathcal{B}_{-i}$ be trees whose bootstrap samples omitted that record. Its out-of-bag prediction is

$$\widehat{f}_{\mathrm{OOB}}(\mathbf{x}_i) = \frac{1}{|\mathcal{B}_{-i}|} \sum_{b \in \mathcal{B}_{-i}} T_b(\mathbf{x}_i).$$

OOB predictions provide an efficient development-stage estimate and diagnostics without fitting a separate forest for every fold. They do not automatically reproduce host-grouped, household-grouped, or future-time deployment. They are also repeatedly consulted during modelling. The untouched final audit remains the answer to the frozen external evaluation contract.

← **Return to how many trees is enough**

Appendix A12: feature subsampling trades strength for correlation

If p eligible features exist, a random forest exposes a node to $m_{\text{try}} \leq p$ sampled candidates.

Large m_{try}	stronger individual splits, but similar dominant features can make trees more correlated.
Small m_{try}	more varied paths, but some nodes may be forced to choose weak candidates.
Grouped encodings	sampling one-hot columns independently can give high-cardinality raw features more opportunities; implementations and grouped-feature strategies differ.
Selection rule	choose the complete forest recipe inside development resampling, not from a universal default.

← **Return to feature randomisation**

Appendix A13: variance of a correlated average

Assume $\mathrm{Var}(e_b) = \sigma^2$ and $\mathrm{Cov}(e_b, e_{b'}) = \rho\sigma^2$ for $b \neq b'$. Then

$$\begin{aligned}\mathrm{Var}(\bar{e}) &= \frac{1}{B^2}\left[B\sigma^2 + B(B-1)\rho\sigma^2\right] \\ &= \sigma^2\left[\rho + \frac{1-\rho}{B}\right].\end{aligned}$$

$\bar{e} = B^{-1}\sum_{b=1}^{B} e_b$ is the mean error across the B trees for the same new case.

If $\rho = 0$, variance falls as $1/B$. If $\rho = 1$, averaging changes nothing. Real trees need not have equal variance or equal pairwise correlation; the exact identity is then the sum of their individual variances and covariances. This calculation explains the ensemble intuition but is not a predictive-risk guarantee and does not include bias.

← **Return to the variance identity**

Appendix A14: exact forest weights with bootstrap multiplicities

Let N_{bi} be the number of times training case i appears in bootstrap sample b, and let $R_b(\mathbf{x})$ be the query leaf. Then

$$T_b(\mathbf{x}) = \frac{\sum_i N_{bi}\mathbf{1}\{\mathbf{x}_i \in R_b(\mathbf{x})\}y_i}{\sum_k N_{bk}\mathbf{1}\{\mathbf{x}_k \in R_b(\mathbf{x})\}}.$$

Hence

$$\widehat{f}_B(\mathbf{x}) = \sum_i w_i(\mathbf{x})y_i, \quad w_i(\mathbf{x}) = \frac{1}{B}\sum_b \frac{N_{bi}\mathbf{1}\{\mathbf{x}_i \in R_b(\mathbf{x})\}}{\sum_k N_{bk}\mathbf{1}\{\mathbf{x}_k \in R_b(\mathbf{x})\}}.$$

The weights are non-negative and sum to one whenever each tree leaf contains at least one bootstrap observation.

← **Return to the adaptive-weight view**

Appendix A15: boosting as gradient descent in function space

Boosting seeks a function F that minimises empirical risk

$$\mathcal{R}(F) = \sum_{i=1}^{n} L(y_i, F(\mathbf{x}_i)).$$

At round m, the vector of negative gradients evaluated on the training cases is

$$r_{im} = - \left. \frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)} \right|_{F=F_{m-1}}.$$

The base learner h_m approximates this direction within a restricted function class. A line search can choose

$$\gamma_m = \arg\min_{\gamma} \sum_{i} L(y_i, F_{m-1}(\mathbf{x}_i) + \gamma h_m(\mathbf{x}_i)) \,.$$

Shrinkage then takes only a fraction $\nu\gamma_m$ of the fitted step.

Source: Gradient boosting: Friedman (2001).

← **Return to the negative gradient**

Appendix A16: a complete squared-loss boosting iteration

With $L(y, F) = \frac{1}{2}(y - F)^2$:

1. initialise $F_0(\mathbf{x}) = \bar{y}$;
2. compute residuals $r_{im} = y_i - F_{m-1}(\mathbf{x}_i)$;
3. fit a regression tree that partitions the residuals into leaves R_{jm};
4. in each leaf, the optimal correction is the residual mean $\gamma_{jm} = |R_{jm}|^{-1} \sum_{i \in R_{jm}} r_{im}$;
5. update

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \sum_j \gamma_{jm} \mathbf{1}\{\mathbf{x} \in R_{jm}\}.$$

After M rounds, F_M is an additive sum of shallow, piecewise-constant corrections. The ensemble can represent a smooth-looking fitted response even though each component tree is stepped.

← **Return to the worked correction**

Appendix A17: shrinkage, tree size, rounds, and early stopping

These controls interact:

Learning rate ν	fraction of each fitted correction admitted to the evolving score.
Tree size	order and number of interactions one correction can express.
Rounds M	how long the model continues repairing development loss.
Leaf regularisation	minimum support and/or penalty attached to local updates.
Early stopping	selects a round using a development-only monitoring split or fold scheme.

A smaller learning rate generally requires more rounds; a larger tree can express more structure per round. Early stopping data become part of model selection and must not be the final audit set.

← **Return to shrinkage**

Appendix A18: Bernoulli gradient and optimal leaf update

With score F_i, probability $p_i = \sigma(F_i)$, and Bernoulli loss $L_i = -y_i \log p_i - (1 - y_i) \log(1 - p_i)$,

$$\frac{\partial L_i}{\partial F_i} = p_i - y_i, \qquad -\frac{\partial L_i}{\partial F_i} = y_i - p_i.$$

For a tree leaf R, a Newton approximation gives the score update

$$\gamma_R \approx \frac{\sum_{i \in R} (y_i - p_i)}{\sum_{i \in R} p_i (1 - p_i) + \lambda},$$

where λ represents optional second-order regularisation conventions. The numerator is remaining signed probability error; the denominator scales the step by local curvature. Software variants differ in exact regularisation, constraints, and leaf optimisation.

← **Return to the Bank pseudo-response**

Appendix A19: what modern boosting variants change

Family	Distinctive engineering or statistical choices
histogram GB	bins continuous values to accelerate split search; implementations differ in missing-value and category support.
XGBoost	regularised second-order objective, shrinkage, column subsampling, sparse-aware split search, and systems optimisation.
LightGBM	histogram methods with leaf-wise growth and specialised sampling/category strategies; powerful but capable of rapid overfit on small data.
CatBoost	ordered target statistics and ordered boosting designed to reduce target leakage and prediction shift with categorical inputs.

This deck's empirical ledger uses scikit-learn histogram gradient boosting, not a claim that all four systems are equivalent. Product choice also depends on data types, scale, reproducibility, latency, and governance.

Source: XGBoost: Chen and Guestrin (2016).

← **Return to what boosting bought**

Appendix A20: every grid declared, every setting chosen

	Declared before any score was read	Chosen by the folds
London · five host-grouped folds, scored by average miss		
linear	nothing to tune	ordinary least squares
tree	depths 3, 5, 7, 9 by leaf floors 25, 75, 150 (12)	depth 9, floor 25
forest	floors 5, 25, 75 by `max_features` sqrt or 0.65 (6)	floor 5, sqrt; 120 trees inside folds, 400 in the fitted forest
boosting	150 or 300 rounds by rates 0.05, 0.10 by 7 or 15 leaves (8)	300 rounds, rate 0.05, 15 leaves
Bank · five stratified folds, scored by log loss		
logistic	nothing to tune	unpenalised, lbfgs
tree	depths 2 to 6 by leaf floors 50, 150, 300 (15)	depth 6, floor 50
forest	floors 10, 50, 150 by `max_features` sqrt or 0.65 (6)	floor 50, 0.65
boosting	the same eight candidates as London	150 rounds, rate 0.05, 15 leaves

Boosting also declares a leaf floor of 40, an L2 term of 1.0, and early stopping switched off, so its round count is chosen by the folds rather than by an internal holdout. Choosing the tree cost 60 fits, the whole London grid 130, and the Bank grid 145.

← **Return to the one contract**

Appendix A21: paired losses and host-cluster uncertainty

For two fitted systems A and B, define the paired loss difference on the same validation listing:

$$d_i = L(y_i, \widehat{f}_A(\mathbf{x}_i)) - L(y_i, \widehat{f}_B(\mathbf{x}_i)).$$

The mean $\bar{d}$ estimates the validation-set advantage of B. Because listings from the same host may be dependent, resample or aggregate at host level:

$$d_h = \frac{1}{n_h} \sum_{i:\, host(i)=h} d_i.$$

n_h is the number of validation listings belonging to host h; d_h is their mean paired loss difference.

A host bootstrap or cluster-robust calculation respects the unit used in the London split. Independent listing-level error bars would overstate effective information when within-host losses move together. This uncertainty describes the declared validation population and split design; it is not automatically uncertainty for a future market regime.

← **Return to the London comparison**

Appendix A22: exact London result ledger

Inside Airbnb London snapshot dated 19 June 2026: 92,638 raw rows; 54,636 eligible under the frozen eligibility rules. Deterministic host-disjoint split: development 32,749; validation 11,063; final audit 10,824.

Validation family	MAE	RMSE	R^2
linear regression	75.482	114.628	0.549
tree	73.899	114.271	0.552
random forest	70.869	109.944	0.585
histogram boosting	69.597	108.822	0.594

Frozen winner refitted on development plus validation: final boosting MAE 73.062, RMSE 111.900, $R^2 = 0.579$; refitted global baseline MAE 77.882, RMSE 117.029, $R^2 = 0.540$. Primary selection metric: MAE. All figures use quoted nightly listing price for the declared 1–7-night query, not completed transaction price.

← **Return to the three-metric table**

Appendix A23: exact corrected Bank result ledger

UCI Bank Marketing file: 41,188 cases; 4,640 subscriptions. Stratified split with seeds 42/43: development 24,712; validation 8,238; final audit 8,238. Prediction is made before the recorded outcome; call duration is forbidden. The raw `pdays`=999 sentinel is represented as `pdays_is_999`=1 and `pdays_days`=0; raw `pdays` is dropped.

Validation family	Log loss	Brier
constant development rate	0.352018	0.099959
logistic regression	0.271754	0.077051
tree	0.271558	0.076178
random forest	0.265809	0.075349
histogram boosting	0.265221	0.075288

Frozen winner refitted on development plus validation: final boosting log loss 0.266113, Brier 0.075146; logistic baseline 0.273256, Brier 0.077216. Because no stable client ID or exact use-time date is supplied, claims remain random-case, not client-disjoint or future-time.

← **Return to the Bank validation evidence**

Appendix A24: the campaign ledger

Declared costs: €4 for a wasted call (c_{FP}) and €80 for a missed subscriber (c_{FN}), so $\tau^* = c_{\mathrm{FP}}/(c_{\mathrm{FP}} + c_{\mathrm{FN}}) = 0.0476$. These costs are an assumption of this lecture, not an estimate. The rule follows from comparing the expected cost of calling, $c_{\mathrm{FP}}(1-p)$, with the expected cost of not calling, $c_{\mathrm{FN}}p$.

	Calls	Caught	Wasted	Missed	Bill at τ^*	Its own best τ
logistic regression	5,259	830	4,429	98	€25,556	0.054 (€25,520)
tree	6,074	864	5,210	64	€25,960	0.052 (€25,692)
random forest	5,061	824	4,237	104	€25,268	0.051 (€25,136)
histogram boosting	5,372	848	4,524	80	€24,496	0.047 (€24,372)

References on the same 8,238 validation contacts: calling everyone costs €29,240; calling nobody costs €74,240. At $2\tau^*$ the bills rise to €28,852, €29,976, €27,376 and €28,604. Cheapest family by threshold: forest at 0.02, boosting at τ^*, forest again at 0.10 and 0.20. A cost ranking is a statement about one operating point, and the counts above assume the claimed probabilities are calibrated, which is what A25 checks.

← **Return to the campaign decision**

Appendix A25: calibration curves and bin uncertainty

Within probability bin B_k, plot

$$\bar{p}_k = \frac{1}{n_k} \sum_{i \in B_k} \widehat{p}_i, \qquad \bar{y}_k = \frac{1}{n_k} \sum_{i \in B_k} y_i.$$

$n_k = |B_k|$ is the number of validation cases in probability bin B_k; bars denote averages within that bin.

Perfect population calibration would satisfy $\mathbb{E}[Y \mid \widehat{p} = p] = p$. A sample curve is noisy because bin membership, sample size, and event count vary. Quantile bins improve count balance but have unequal probability widths; fixed bins aid interpretation but can be sparse. Always show counts or uncertainty, and retain a proper scoring rule: a coarse calibration plot cannot distinguish all probability forecasts. Recalibration fitted after seeing validation outcomes is another modelling step and requires its own untouched evaluation.

← **Return to the calibration curve**

Appendix A26: grouped permutation importance

For raw feature group G, compute held-out loss R_0, jointly permute its represented columns across cases, and recompute $R_G^{(b)}$. Report

$$I_G = \frac{1}{B} \sum_{b=1}^{B} \left(R_G^{(b)} - R_0 \right).$$

B is the number of repeated joint permutations; R_0 is unpermuted held-out loss and $R_G^{(b)}$ is loss after permutation b.

Joint permutation keeps one-hot columns for a raw category together. If evaluation is grouped, permutation should respect the estimand: global permutation asks a different question from within-cluster or cluster-level permutation. Correlated substitutes can mask one another; shuffling one feature can also create implausible combinations. Importance therefore measures reliance of this fitted system under this perturbation and data distribution—not causality or inherent value.

← **Return to the importance plot**

Appendix A27: partial dependence, ICE, and support

For feature $x_j = z$, individual conditional expectation is

$$\mathrm{ICE}_i(z) = \widehat{f}(z, \mathbf{x}_{i,-j}),$$

and partial dependence averages these curves:

$$\mathrm{PD}(z) = \frac{1}{n} \sum_{i=1}^{n} \widehat{f}(z, \mathbf{x}_{i,-j}).$$

$\mathbf{x}_{i,-j}$ is case i's recorded features other than feature j; n is the number of held-out cases averaged.
ICE exposes heterogeneity hidden by the average. Both can evaluate combinations $(z, \mathbf{x}_{i,-j})$ that are rare or impossible when features are correlated. Use a support rug, restrict the grid, inspect conditional variants where helpful, and label the display as a fitted-model probe. Without identification assumptions, neither curve is an intervention effect.

← **Return to the PDP and ICE display**

Appendix A28: versions, seeds, hashes, and provenance

Reproducible builders: `scripts/build_mlr2_day3.py` for the tuning and the family comparison, `scripts/build_mlr2_day3_teaching_figures.py` for the Part 3 to 7 sweeps, and `scripts/build_mlr2_day3_part12_ledger.py` for the Part 1 and 2 numbers. Ledgers: `mlr2_day3_results.json`, `mlr2_day3_teaching.json`, `mlr2_day3_part12.json`, all in `figures/day3/`. The teaching builder reads its model settings from the first ledger and asserts that its refits reproduce it.

Software	Python 3.14 environment; scikit-learn 1.7.2; pandas 2.2.3; NumPy 2.3.5; Matplotlib 3.10.8.
London tuning	five host-grouped development folds; declared grids in the builder; ensemble random state 42.
Bank tuning	five stratified development folds; split seeds 42 and 43; declared grids in the builder.
London SHA-256	`f32fbf1bdd3a4fcdb58ffe901b4ac20d8614f77cd36f1e3bbd18523e6e2acabd`
Bank SHA-256	`74adfc578bf77a7ff4bb1ba4a9f8709d9e3c6907342959c2c8416847e0afb4d8`

Earlier Bank artifacts were not reused because they passed raw sentinel 999 through the numeric pipeline. This deck rebuilds every reported Bank result with the corrected sentinel representation.

← **Return to the family ledger**

Appendix A29: development tuning, validation selection, and final refitting

Let a index model families and λ their settings.

$$\widehat{\lambda}_a = \arg\min_{\lambda} \widehat{R}_{\mathrm{CV,dev}}(a, \lambda), \qquad \widehat{a} = \arg\min_{a} \widehat{R}_{\mathrm{val}}(a, \widehat{\lambda}_a).$$

Freeze the selected recipe $(\widehat{a}, \widehat{\lambda}_{\widehat{a}})$, refit it on $\mathcal{D}_{\mathrm{dev}} \cup \mathcal{D}_{\mathrm{val}}$, and evaluate once:

$$\widehat{R}_{\mathrm{final}} = \frac{1}{|\mathcal{D}_{\mathrm{test}}|} \sum_{i \in \mathcal{D}_{\mathrm{test}}} L\left(y_i, \widehat{f}_{\mathcal{D}_{\mathrm{dev}} \cup \mathcal{D}_{\mathrm{val}}}(\mathbf{x}_i)\right).$$

Final refitting uses more labelled data without changing the frozen design. It does not authorise another round of family or setting selection.

← **Return to the final-audit protocol**