Decision Trees and Ensemble Methods

ML for small and/or tabular data

Jan Meischner

7 January 2026

Agenda

- Motivation: Why decision trees?
- A simple example and terminology
- How do we learn a tree?
- Overfitting and tree size
- Motivation for ensembles
- Random forests
- (Optional) Bagging and boosting
- (Optional) Outlook: gradient-based learning of trees
- Summary

Motivation: Why decision trees?

- **Interpretability:**
  - Every path from root to leaf is a simple if–then rule.
  - Well suited for explaining decisions.

Motivation: Why decision trees?

- **Interpretability:**
  - Every path from root to leaf is a simple if–then rule.
  - Well suited for explaining decisions.
- **Practical for tabular data:**
  - Categorical and numerical features.
  - No feature scaling, little preprocessing required.

Motivation: Why decision trees?

- **Interpretability:**
  - Every path from root to leaf is a simple if–then rule.
  - Well suited for explaining decisions.
- **Practical for tabular data:**
  - Categorical and numerical features.
  - No feature scaling, little preprocessing required.
- **Nonlinearity:**
  - Recursive splits can create complex decision boundaries.

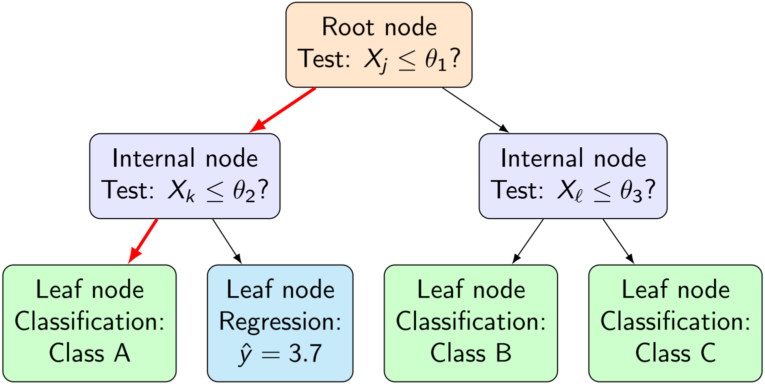
Motivation: Why decision trees?

- **Interpretability:**
  - Every path from root to leaf is a simple if–then rule.
  - Well suited for explaining decisions.
- **Practical for tabular data:**
  - Categorical and numerical features.
  - No feature scaling, little preprocessing required.
- **Nonlinearity:**
  - Recursive splits can create complex decision boundaries.
- **Efficiency:**
  - Relatively fast training on small to medium-sized datasets.
  - Fast prediction (follow the path).

Motivation: Why decision trees?

- **Interpretability:**
  - Every path from root to leaf is a simple if–then rule.
  - Well suited for explaining decisions.
- **Practical for tabular data:**
  - Categorical and numerical features.
  - No feature scaling, little preprocessing required.
- **Nonlinearity:**
  - Recursive splits can create complex decision boundaries.
- **Efficiency:**
  - Relatively fast training on small to medium-sized datasets.
  - Fast prediction (follow the path).
- **Building block for ensembles:**
  - Random forests, gradient boosting, more recent tree ensembles.

Visualisation & terminology



Instance: $x = (x_1, \ldots, x_d)$ travels from the root along a decision path to a leaf.
Features: $X_1, \ldots, X_d$ are the input variables that appear in the tests.

Example: Tennis player

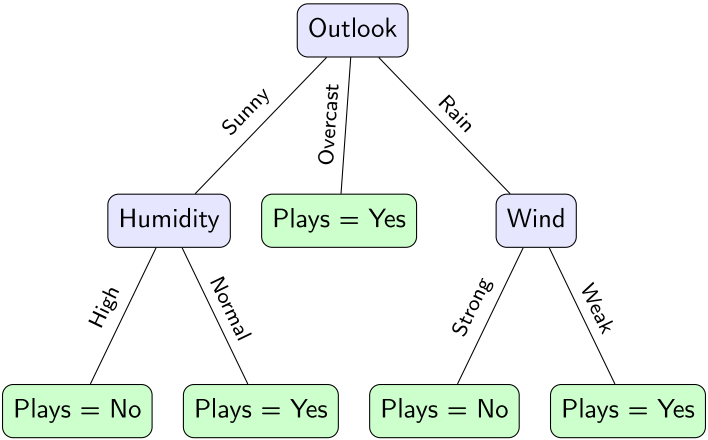
No.	Outlook	Temperature	Humidity	Wind	Plays
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

- **Question:** "We look out of the window at the tennis court across the street and ask: Under which weather conditions does the player come to the court?"

Example: Tennis player

No.	Outlook	Temperature	Humidity	Wind	Plays
Sunny → split by Humidity					
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
11	Sunny	Mild	Normal	Strong	Yes
9	Sunny	Cool	Normal	Weak	Yes
Overcast → Leaf: Plays = Yes					
3	Overcast	Hot	High	Weak	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
Rain → split by Wind					
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Example: Tennis player



Prediction with a decision tree

No.	Outlook	Temperature	Humidity	Wind	Plays
15	Sunny	Hot	Normal	Weak	?

General algorithm:

- Input: new instance $x = (x_1, \ldots, x_d)$.
- Algorithm:
  1. **Start at the root.**
  2. Check the test of the current node.
  3. Follow the corresponding edge.
  4. Repeat until you reach a leaf.
- Output: label or value stored in the leaf.

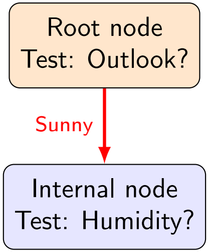
Root node
Test: Outlook?

Prediction with a decision tree

No.	Outlook	Temperature	Humidity	Wind	Plays
15	Sunny	Hot	Normal	Weak	?

General algorithm:

- Input: new instance $x = (x_1, \ldots, x_d)$.
- Algorithm:
  1. Start at the root.
  2. **Check the test of the current node.**
  3. **Follow the corresponding edge.**
  4. Repeat until you reach a leaf.
- Output: label or value stored in the leaf.



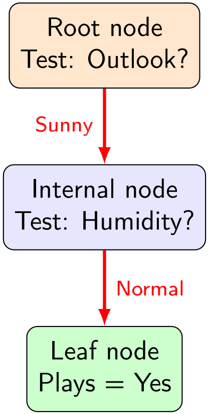
Prediction with a decision tree

No.	Outlook	Temperature	Humidity	Wind	Plays
15	Sunny	Hot	Normal	Weak	?

General algorithm:

- Input: new instance $x = (x_1, \ldots, x_d)$.
- Algorithm:
  1. Start at the root.
  2. Check the test of the current node.
  3. Follow the corresponding edge.
  4. **Repeat until you reach a leaf.**

Output: $\hat{y}(x_{11})$: Plays = Yes.



How do we learn a decision tree?

We need to think about the following questions:

1. Do we allow only binary splits at internal nodes, or also n-ary splits?
2. How do we decide which feature to split on?
3. When do we create a leaf node, and how do we assign the class label?
4. How large do we want the tree to grow?

How do we split decision trees?

In practice, decision trees almost always use binary splits.

Reasons:

- **Efficient optimisation:** Binary splits allow a clean search for the best threshold ($X_j \leq \theta$). Multiple intervals quickly lead to a complex combinatorial optimisation problem.
- **Regularisation:** Splits with more than two regions produce very small subsets and increase overfitting.
- **Interpretability:** Tests such as "$X_j \leq \theta$?" are easy to understand; n-ary nodes become hard to read.
- **Practice:** Common tree algorithms use binary splits for numerical features (CART, ID3, C4.5, Random Forests, XGBoost, LightGBM).

Conclusion: For numerical features, the binary split is the robust, interpretable, and optimisable standard.

How do we learn a decision tree?

- Given: training set $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$.
- Goal: find the tree structure and leaf predictions that predict y from x as well as possible.
- Classical approach: top-down, greedy, recursive.
- Idea:
  - Start with all instances in the root.
  - Choose the **best split** (feature + threshold).
  - Partition into subsets and repeat in each subtree.

CART: basic scheme (informal)

CART = *Classification And Regression Trees* (Breiman et al.)

Recursive algorithm

Given subset S of the training data:

1. If all $y^{(i)}$ in S belong to the same class:
   - Create a leaf with this class.
2. Otherwise:
   - Check stopping criteria (e.g. maximum depth, minimum node size).
   - If stopping: create a leaf with the majority class in S.
   - If not: choose the feature and split that **reduces impurity** the most.
   - Split S into S_{left} and S_{right} (binary split).
   - Call the algorithm recursively on S_{left} and S_{right}.

Impurity: impurity of a node

For classification, let p_k be the proportion of class k in a node.

Gini impurity (CART):

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

- $G(S) = 0$ for a pure class (one $p_k = 1$, all others 0).
- Maximum when classes are equally represented.

Entropy (ID3/C4.5):

$$H(S) = -\sum_k p_k \log_2 p_k.$$

- $H(S) = 0$ for a pure class.
- Maximum impurity when classes are evenly distributed.

Both measures behave similarly; CART uses Gini mainly for efficiency reasons.

Gini impurity as the variance of a class indicator

Idea: For each class we consider an indicator

$$Y_k = \begin{cases} 1 & \text{if the example belongs to class } k \\ 0 & \text{otherwise} \end{cases}$$

with probability $p_k =$ proportion of class k in the node.

The variance of this indicator is:

$$\begin{aligned} \mathrm{Var}(Y) &= \mathbb{E}\left[(Y - \mathbb{E}[Y])^2\right] \\ &= \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2 \\ &= p - p^2 = p(1-p) \end{aligned}$$

The Gini impurity is the sum of these variances over all classes:

$$G = \sum_{k=1}^{K} p_k(1-p_k) \quad = \quad 1 - \sum_{k=1}^{K} p_k^2.$$

Example: computing Gini impurity

Node with 9 instances:

- 6 times class `yes`, 3 times class `no`.

$$
\begin{aligned}
p_{\text{yes}} &= \frac{6}{9}, \quad p_{\text{no}} = \frac{3}{9} \\
G(S) &= 1 - \left(p_{\text{yes}}^2 + p_{\text{no}}^2\right) \\
&= 1 - \left(\frac{4}{9} + \frac{1}{9}\right) \\
&= 1 - \frac{5}{9} = \frac{4}{9} \approx 0.44
\end{aligned}
$$

Interpretation:

- $G = 0$: pure nodes (no dispersion).
- High G: mixed classes $\rightarrow$ high uncertainty.

Split criterion: impurity reduction

Given a node S and a possible split into S_1 and S_2:

$$\Delta G = G(S) - \left(\frac{|S_1|}{|S|} G(S_1) + \frac{|S_2|}{|S|} G(S_2) \right).$$

- We choose the split with the **largest** impurity reduction ΔG.
- Analogously with entropy $H(S)$ instead of $G(S)$.

Gini reduction for feature *Outlook*

Data (full set S):

No.	Outlook	Temperature	Humidity	Wind	Plays
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Gini reduction for feature *Outlook*

Class distribution in S:

$$|S| = 14, \quad 9 \text{ Yes}, \ 5 \text{ No}.$$

Gini at the root node:

$$G(S) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 1 - \frac{81}{196} - \frac{25}{196} = \frac{90}{196} \approx 0.46.$$

Question: How much does a split on *Outlook* reduce this impurity?

Split candidate *Outlook* = *Overcast*

Binary split (CART): $\boxed{Outlook \in \{\text{Overcast}\}\ ?}$

- Left subset $S_L = \{\text{Overcast}\}$: $|S_L| = 4,\ 4\,\text{Yes},\ 0\,\text{No}$

$$G(S_L) = 0.$$

- Right subset $S_R = \{\text{Sunny, Rain}\}$: $|S_R| = 10,\ 5\,\text{Yes},\ 5\,\text{No}$

$$G(S_R) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5.$$

Weighted Gini after the split:

$$G_{\text{after}} = \frac{4}{14} \cdot 0 + \frac{10}{14} \cdot 0.5 = \frac{5}{14} \approx 0.357.$$

Gini reduction:

$$\Delta G = G(S) - G_{\text{after}} = \frac{90}{196} - \frac{5}{14} = \frac{20}{196} \approx 0.102.$$

Split candidate *Outlook* = *Sunny*

Binary split (CART): $\boxed{Outlook \in \{\text{Sunny}\}\ ?}$

- $S_L = \{\text{Sunny}\}$: $|S_L| = 5,\ 2\,\text{Yes},\ 3\,\text{No}$

$$G(S_L) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = \frac{12}{25} = 0.48.$$

- $S_R = \{\text{Overcast, Rain}\}$: $|S_R| = 9,\ 7\,\text{Yes},\ 2\,\text{No}$

$$G(S_R) = 1 - \left(\frac{7}{9}\right)^2 - \left(\frac{2}{9}\right)^2 = \frac{28}{81} \approx 0.346.$$

Weighted Gini after the split:

$$G_{\text{after}} = \frac{5}{14} \cdot \frac{12}{25} + \frac{9}{14} \cdot \frac{28}{81} = \frac{124}{315} \approx 0.394.$$

Gini reduction:

$$\Delta G = \frac{90}{196} - \frac{124}{315} = \frac{289}{4410} \approx 0.066.$$

Split candidate *Outlook* = *Rain*

Binary split (CART): $\boxed{Outlook \in \{\text{Rain}\}\ ?}$

- $S_L = \{\text{Rain}\}$: $|S_L| = 5,\ 3\,\text{Yes},\ 2\,\text{No}$

$$G(S_L) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = \frac{12}{25} = 0.48.$$

- $S_R = \{\text{Sunny, Overcast}\}$: $|S_R| = 9,\ 6\,\text{Yes},\ 3\,\text{No}$

$$G(S_R) = 1 - \left(\frac{6}{9}\right)^2 - \left(\frac{3}{9}\right)^2 = \frac{4}{9} \approx 0.444.$$

Weighted Gini after the split:

$$G_{\text{after}} = \frac{5}{14} \cdot \frac{12}{25} + \frac{9}{14} \cdot \frac{4}{9} = \frac{16}{35} \approx 0.457.$$

Gini reduction:

$$\Delta G = \frac{90}{196} - \frac{16}{35} = \frac{1}{490} \approx 0.002.$$

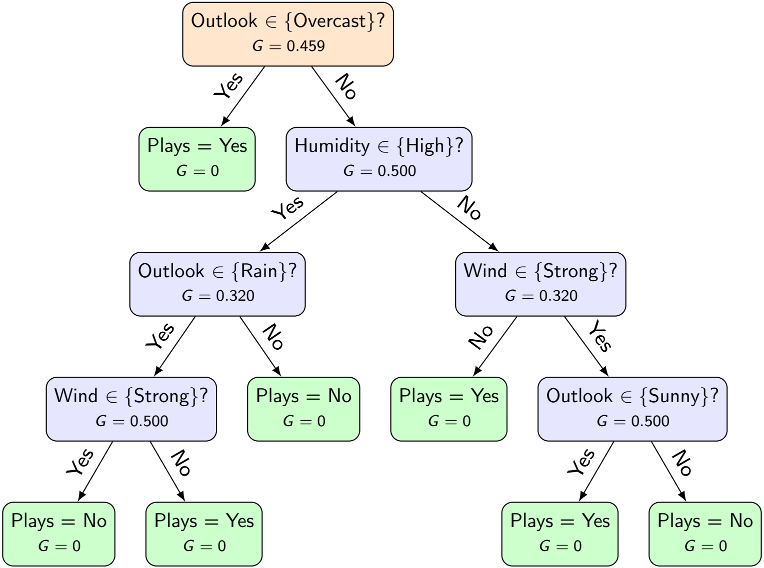
All candidates for the first split

$$G(S) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = \frac{45}{98} \approx 0.459$$

Feature	**Test (value)**	ΔG
Outlook	$\in \{\text{Overcast}\}$?	**0.1020**
Outlook	$\in \{\text{Sunny}\}$?	0.0655
Outlook	$\in \{\text{Rain}\}$?	0.0020
Temperature	$\in \{\text{Hot}\}$?	0.0163
Temperature	$\in \{\text{Cool}\}$?	0.0092
Temperature	$\in \{\text{Mild}\}$?	0.0009
Humidity	$\in \{\text{High}\}$?	0.0918
Humidity	$\in \{\text{Normal}\}$?	0.0918
Wind	$\in \{\text{Weak}\}$?	0.0306
Wind	$\in \{\text{Strong}\}$?	0.0306

Best first split: Outlook $\in \{\text{Overcast}\}$?

Resulting decision tree



Leaf nodes and labels

- **Classification:**
  - Common choice: leaf label = majority class in the node.
  - Option: store full class distribution and use probabilities.

Leaf nodes and labels

- **Classification:**
  - Common choice: leaf label = majority class in the node.
  - Option: store full class distribution and use probabilities.
- **Regression:**
  - Leaf value = mean of the target values in the node.

Leaf nodes and labels

- **Classification:**
  - Common choice: leaf label = majority class in the node.
  - Option: store full class distribution and use probabilities.
- **Regression:**
  - Leaf value = mean of the target values in the node.
- **Stopping criteria (pre-pruning):**
  - Maximum depth reached.
  - Fewer than n_{min} instances in the node.
  - No meaningful impurity reduction anymore.

Overfitting and pruning

- If we let the tree grow without restriction:
  - Nodes can end up with very few instances.
  - The tree fits random quirks of the training set.

Overfitting and pruning

- If we let the tree grow without restriction:
  - Nodes can end up with very few instances.
  - The tree fits random quirks of the training set.
- **Pre-pruning:**
  - Stop growth early.
  - Risk: stopping too early $\Rightarrow$ bias too high.

Overfitting and pruning

- If we let the tree grow without restriction:
  - Nodes can end up with very few instances.
  - The tree fits random quirks of the training set.
- **Pre-pruning:**
  - Stop growth early.
  - Risk: stopping too early $\Rightarrow$ bias too high.
- **Post-pruning:**
  - First grow a large tree.
  - Then cut off subtrees that do not improve validation performance.

Overfitting and pruning

- If we let the tree grow without restriction:
  - Nodes can end up with very few instances.
  - The tree fits random quirks of the training set.
- **Pre-pruning:**
  - Stop growth early.
  - Risk: stopping too early $\Rightarrow$ bias too high.
- **Post-pruning:**
  - First grow a large tree.
  - Then cut off subtrees that do not improve validation performance.
- Decision trees are typically **high-variance** models.
  - Small changes in the data can lead to very different trees.

Interim conclusion: decision trees

Strengths

- Interpretable.
- Flexible for many data types.
- Relatively easy to implement.
- Foundation for many strong ensembles.

Weaknesses

- Tend to overfit.
- High variance.
- A single tree is often not state-of-the-art in accuracy.

Motivation for ensembles

- Idea: instead of training **one** tree, we train **many** trees.
- Analogy: Averaging many different opinions $\Rightarrow$ more robust decision.
- Goal:
  - Reduce variance.
  - Improve generalisation.

Motivation for ensembles

- Idea: instead of training **one** tree, we train **many** trees.
- Analogy: Averaging many different opinions $\Rightarrow$ more robust decision.
- Goal:
  - Reduce variance.
  - Improve generalisation.
- Two important basic ideas:
  1. **Bagging** (bootstrap aggregating)
  2. **Boosting** (sequential error correction)
- Random forests are a special case of bagging with decision trees.

Bagging: bootstrap aggregating

- Generate B bootstrap samples:
  - Draw n instances *with* replacement from the training set.
  - Some instances appear multiple times, others not at all.

Bagging: bootstrap aggregating

- Generate B bootstrap samples:
  - Draw n instances *with* replacement from the training set.
  - Some instances appear multiple times, others not at all.
- Train a separate decision tree on each sample (often with little or no pruning).
- For prediction:
  - Classification: majority vote of the trees.
  - Regression: average of the predictions.

Bagging: bootstrap aggregating

- Generate B bootstrap samples:
  - Draw n instances *with* replacement from the training set.
  - Some instances appear multiple times, others not at all.
- Train a separate decision tree on each sample (often with little or no pruning).
- For prediction:
  - Classification: majority vote of the trees.
  - Regression: average of the predictions.
- Effect:
  - Reduction of variance.
  - Increased robustness to outliers and noise.

Random forest: bagging + feature sampling

- Random forest = bagging with decision trees plus additional randomisation:
  - For each tree: bootstrap sample of the data.
  - At each split: only a random subset of features is considered.

Random forest: bagging + feature sampling

- Random forest = bagging with decision trees plus additional randomisation:
  - For each tree: bootstrap sample of the data.
  - At each split: only a random subset of features is considered.
- Advantage:
  - Trees become more diverse.
  - Lower correlation between trees.
  - Stronger ensemble effect.

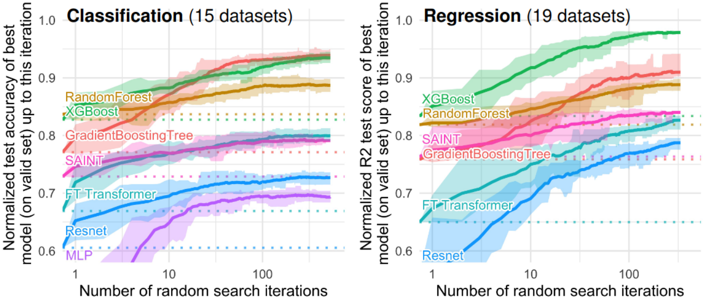
Random forest: bagging + feature sampling

- Random forest = bagging with decision trees plus additional randomisation:
  - For each tree: bootstrap sample of the data.
  - At each split: only a random subset of features is considered.
- Advantage:
  - Trees become more diverse.
  - Lower correlation between trees.
  - Stronger ensemble effect.
- In many practical applications:
  - Very good performance on tabular data.
  - Few hyperparameters, robust.

Random forests

Other models based on similar ideas that often achieve even higher accuracy:

- Gradient-boosted trees (XGBoost)
- CatBoost

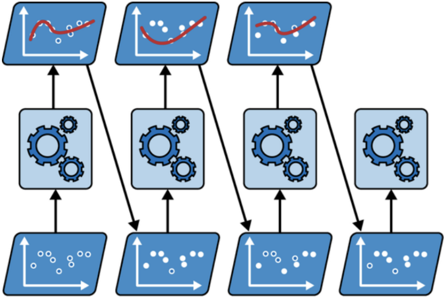


(Optional) Boosting: turning weak learners into strong ones

- Basic idea:
  - Train models sequentially.
  - Each new model focuses on the errors of the previous models.

AdaBoost

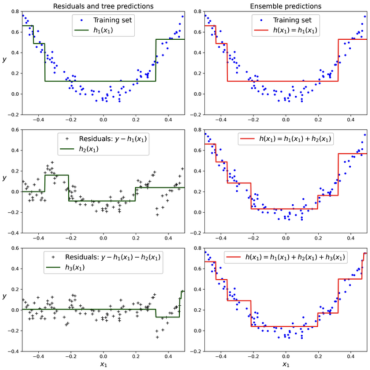
- Start: all instances have equal weight.
- After each tree: increase the weights of misclassified instances.
- Combine the trees with weights.



Source: Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 3rd Edition, O'Reilly Media, 2022.

Gradient boosting (e.g. XGBoost)

- View the errors as residuals.
- Each new tree approximates a step in the direction of the gradient of the loss function.



Source: Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 3rd Edition, O'Reilly Media, 2022.

(Optional) Outlook: gradient-based tree ensembles

- Classical decision trees are non-differentiable:
  - Splits are hard decisions.
  - Training is based on heuristic (greedy) search.

(Optional) Outlook: gradient-based tree ensembles

- Classical decision trees are non-differentiable:
  - Splits are hard decisions.
  - Training is based on heuristic (greedy) search.
- More recent approaches:
  - Relax the hard decisions into "soft" splits.
  - Enable training with gradient-based methods (backpropagation).

(Optional) Outlook: gradient-based tree ensembles

- Classical decision trees are non-differentiable:
  - Splits are hard decisions.
  - Training is based on heuristic (greedy) search.
- More recent approaches:
  - Relax the hard decisions into "soft" splits.
  - Enable training with gradient-based methods (backpropagation).
- Example: GRANDE (Gradient-Based Decision Tree Ensembles for Tabular Data)
  - Trees are formulated as a parameterised, differentiable model.
  - Parameters (e.g. thresholds) are optimised by gradient descent.
  - Goal: combine the strengths of trees (for tabular data) with the optimisability of neural networks.
- Takeaway: research on decision trees is still very active.

Summary

- Decision trees:
  - Intuitive, interpretable, and widely applicable to tabular data.
  - Learning via recursive splits with impurity reduction (e.g. Gini).

Summary

- Decision trees:
  - Intuitive, interpretable, and widely applicable to tabular data.
  - Learning via recursive splits with impurity reduction (e.g. Gini).
- Main issue: overfitting and high variance.
- Remedy: ensembles such as random forests and boosting.
- For many practical tabular problems:
  - Tree ensembles are still very strong baselines.

Summary

- Decision trees:
  - Intuitive, interpretable, and widely applicable to tabular data.
  - Learning via recursive splits with impurity reduction (e.g. Gini).
- Main issue: overfitting and high variance.
- Remedy: ensembles such as random forests and boosting.
- For many practical tabular problems:
  - Tree ensembles are still very strong baselines.
- Active research area: gradient-based tree ensembles, better interpretability, fair and robust models.

Lecture Companion

https://jmeischner.com/writing/decision-trees-and-ensemble-methods