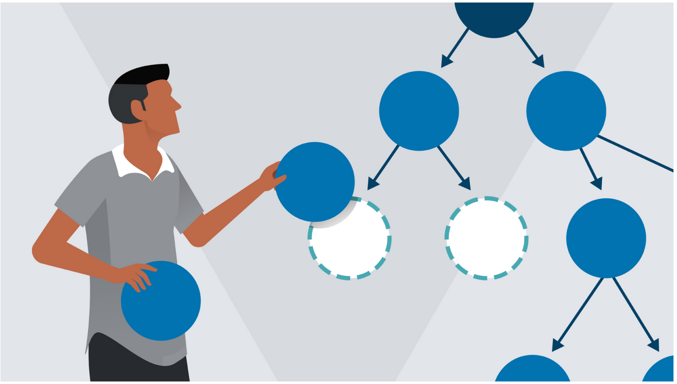
Decision Trees



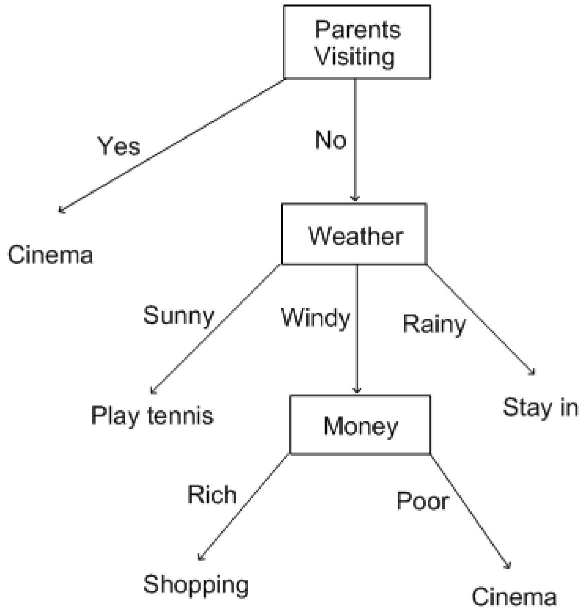
Lluís Talavera, 2022

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

ETSEIB

Introduction

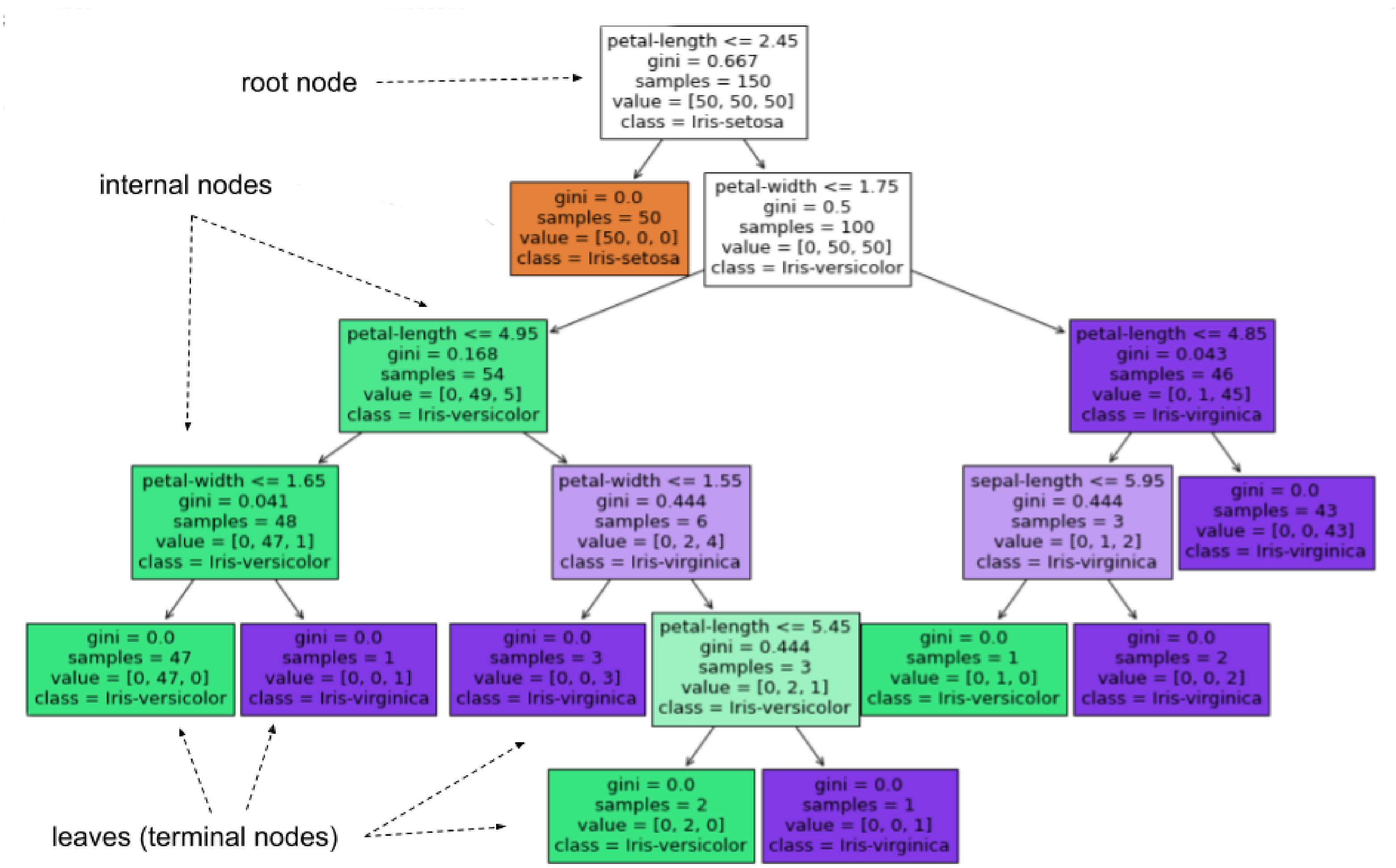
- Mimic human decision making.
- Represents a set of questions in a tree-like structure and makes decisions based on the answer.
- Can be used for classification and regression.
- Can work with categorical and numeric data (only numeric in `sklearn`)



DTs are equivalent to a set of if-then rules:

```
IF parents_visiting=No and weather=Sunny THEN play_tennis
```

A DT for the iris dataset



How to split the nodes?

- Impurity: a measure H of the homogeinity of the labels at a node. It decreases as the classes are more well separated.
- Partition the data so that the impurity of the resulting subsets is *minimized*.
- Gini impurity: A measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

  $H_{gini}(X_m) = \sum_k p_{mk} (1 - p_{mk})$

  where X_m is the data at node m and p_{km} is the proportion of examples from class k in node m.

Other measures can be used, e.g., entropy.

Decision tree algorithm

Training

It is a *recursive* algorithm that applies the same operations each time to smaller amounts of data.

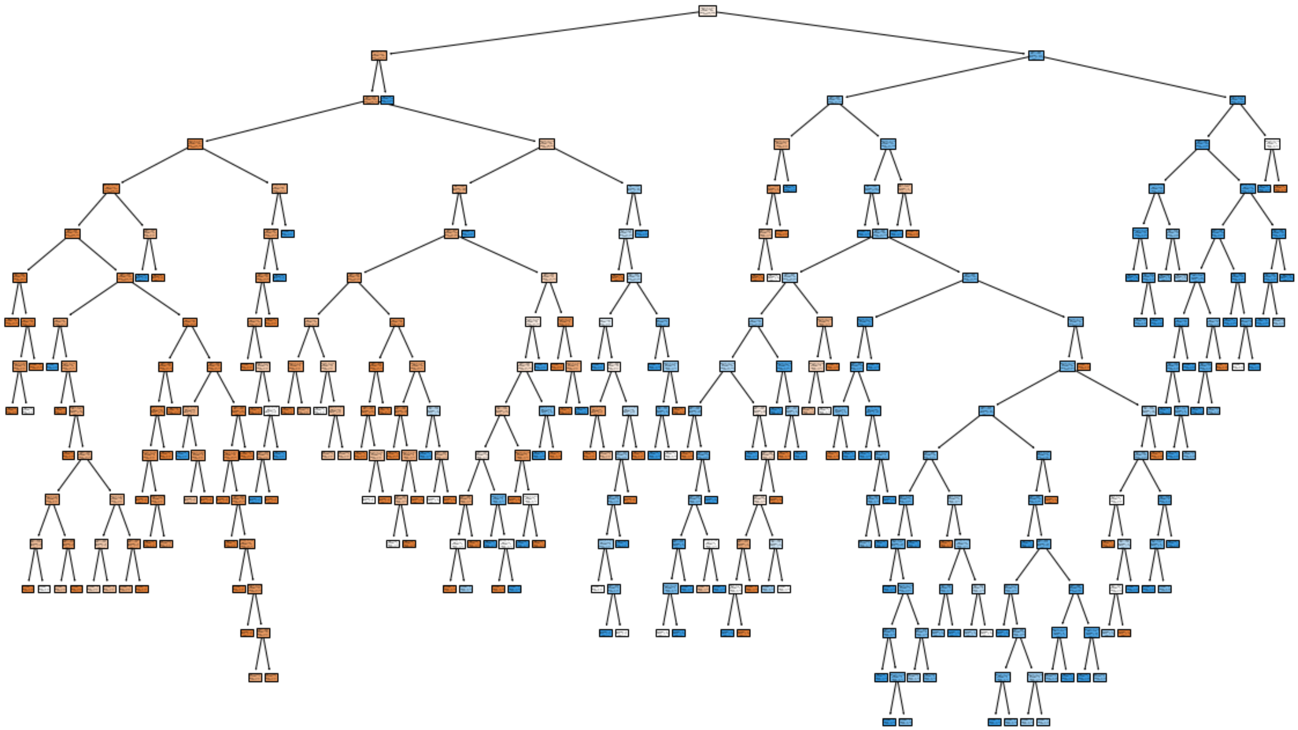
It starts with the root node and then proceeds as follows:

1. If the node is a leaf (all members are from the same class), stop.
2. Test different split values and compute the impurity H of the resulting partitions.
3. Divide the data in the node using the value that minimizes H.
4. Apply the same procedure (steps 1-4) for each of the newly created nodes (subsets of data).

Prediction

Given a new example, starting from the root node, apply the node test and follow the appropriate branch. Repeat the process for the new node until the node is terminal. Return the most frequent value of the class at this node (majority vote).

Overfitting



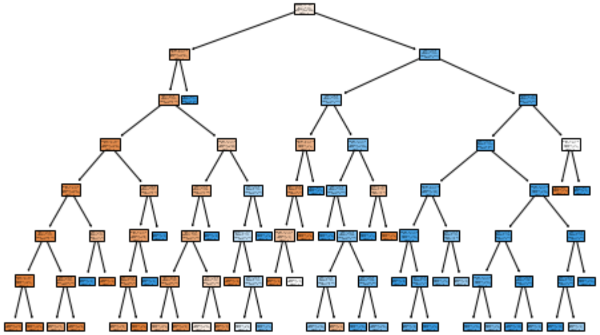
The original algorithm can lead to deep trees that are excessively adjusted to the training data and exhibit poor generalization performance (**overfitting**).

Pruned trees

We can *prune* the trees by relaxing the stopping criterion and generate smaller trees.

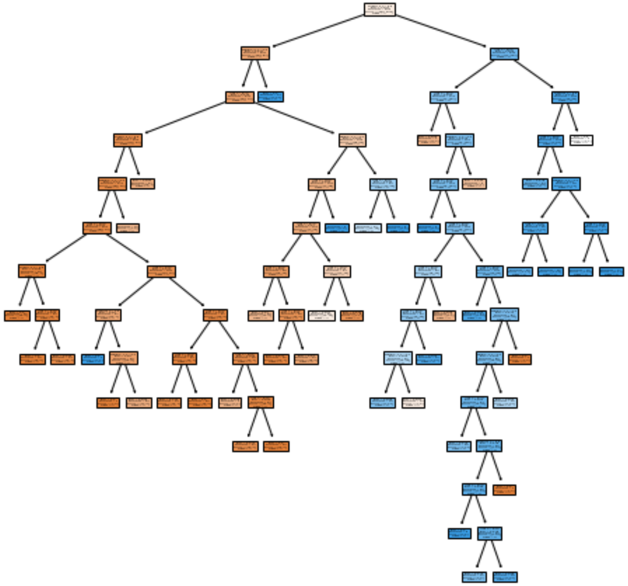
`max_depth = 7`

(limits the maximum depth allowed)

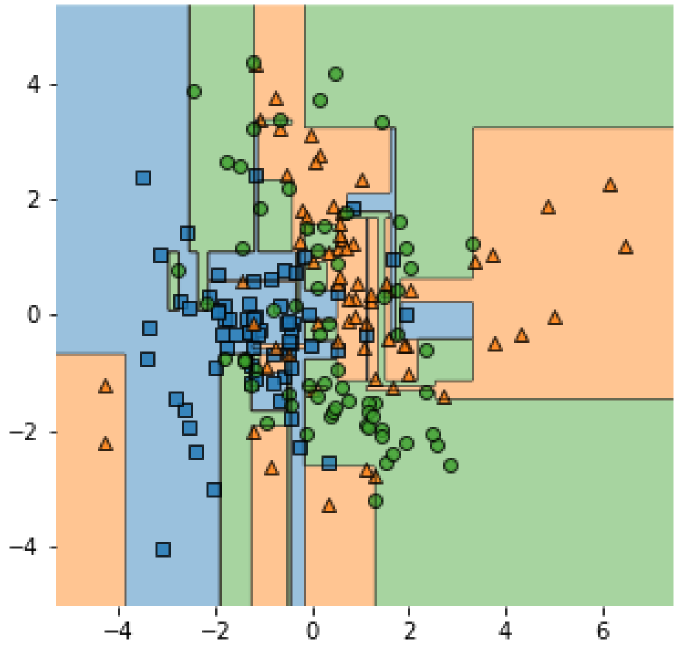


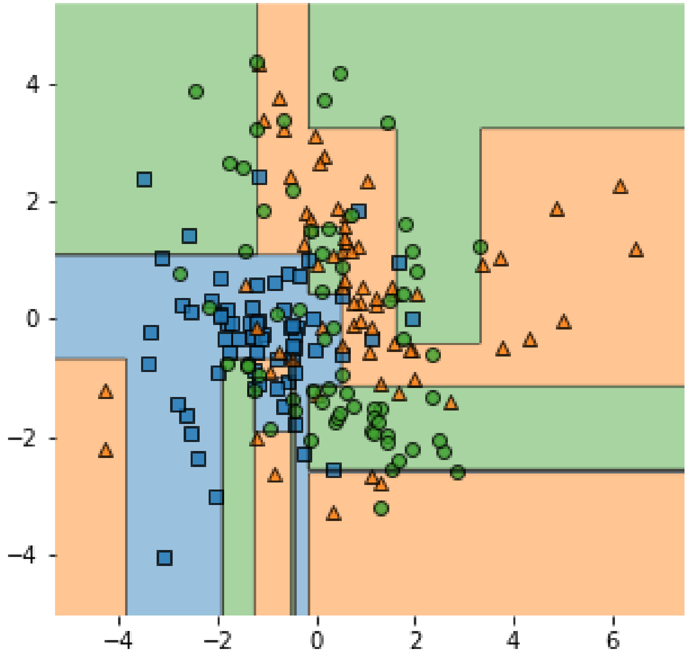
`min_samples_split = 30`

(stops expansion of nodes with less than a certain number of examples)



DT decision boundaries





Regression trees

Training: Apply the same algorithm for training but with a different criterion to measure the quality of the split.

By default, `sklearn` minimizes the Mean Squared Error (MSE):

$$H_{MSE}(X_m) = \Large\frac{1}{N_m} \sum_{y \in X_{m}}(y –
\bar{y}_{m})^{2}$$

where X_m is the data at node m, N_m is the number of examples in node m and $\bar{y}_m$ is the mean value for the target in node m.

Prediction: The mean value for the target at the leaf (terminal node).

Some remarks about DTs

- Non-parametric model.
- Simple to understand and interpret (if-then rules).
- Not sensitive to scaling.
- Can easily overfit without pruning.
- Greedy algorithm: takes the best local decision at each step but does not guarantee learning an optimal DT.
- Small changes in the training data may cause important changes in the resulting tree (unstable).

sklearn imports

Classification

```
from sklearn.tree import DecisionTreeClassifier
```

Regression

```
from sklearn.tree import DecisionTreeRegressor
```