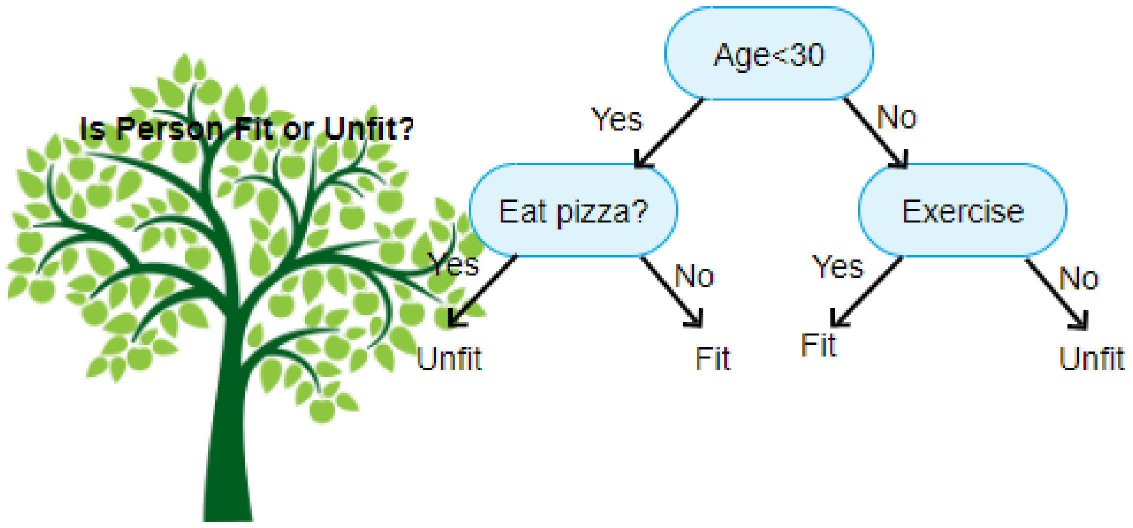
6. Decision Trees



Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks. They are powerful algorithms, capable of fitting complex datasets.

One of the many qualities of Decision Trees is that they require very little data preparation. In fact, they don't require feature scaling or centering at all.

Computational Complexity

Making predictions requires traversing the Decision Tree from the root to a leaf. Decision Trees generally are approximately balanced, so traversing the Decision Tree requires going through roughly $O(\log_2(m))$ nodes , given overall prediction complexity is $O(\log_2(m))$, independent of the number of features.

CART (Classification and Regression Tree) training algorithm

1. **Splitting Process**: CART algorithm splits the training set into two subsets using a feature kk and a threshold tktk (e.g., "petal length ≤ 2.45 cm").
2. **Choosing** kk **and** tktk: It searches for the pair (k,tk)(k,tk) that produces the purest subsets, weighted by their size. The purity is measured using a cost function based on the Gini impurity of the left and right subsets and the number of instances in each subset.
3. **Recursive Splitting**: After the initial split, the algorithm recursively splits the subsets and sub-subsets using the same logic.
4. **Stopping Criteria**: The recursion stops when it reaches the maximum depth set by the **max_depth** hyperparameter or if no split can further reduce impurity. Other hyperparameters like **min_samples_split**, **min_samples_leaf**, **min_weight_fraction_leaf**, and **max_leaf_nodes** also define stopping conditions.
5. **Greedy Nature**: The algorithm is greedy—it searches for an optimal split at each level without considering future splits. This approach generally yields a reasonably good but not necessarily optimal solution.

Gini Impurity & Entropy

Gini Impurity and Entropy are both measures used in decision trees to determine the quality of a split.

Gini Impurity:

- **Concept**: Measures the probability of a randomly chosen element from the set being incorrectly classified.
- **Calculation**: Calculated as 1−∑(pi)^2, where pi is the probability of an element being in class ii.
- **Behavior**: Tends to be quicker to compute as it doesn't involve logarithmic functions.
- **Usage**: Often preferred for its computational efficiency.

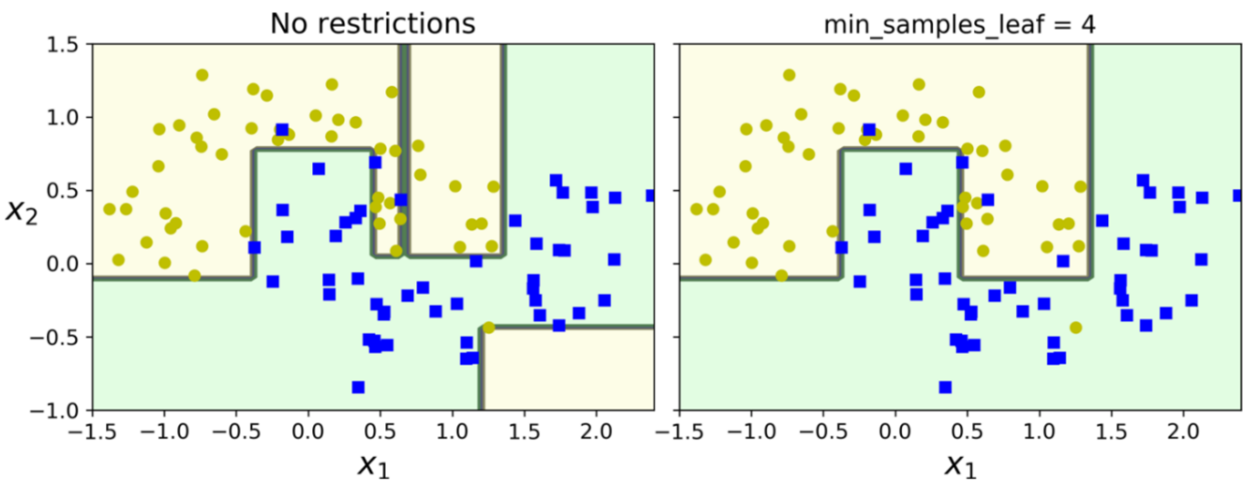
Entropy:

- **Concept**: Measures the amount of randomness or disorder in the information.
- **Calculation**: Calculated as −∑pi log2(pi), where pipi is the probability of an element being in class ii.
- **Behavior**: Involves logarithms, can be more computationally intensive.
- **Usage**: Used when a more detailed level of randomness is required.

Regularization Hyperparameters

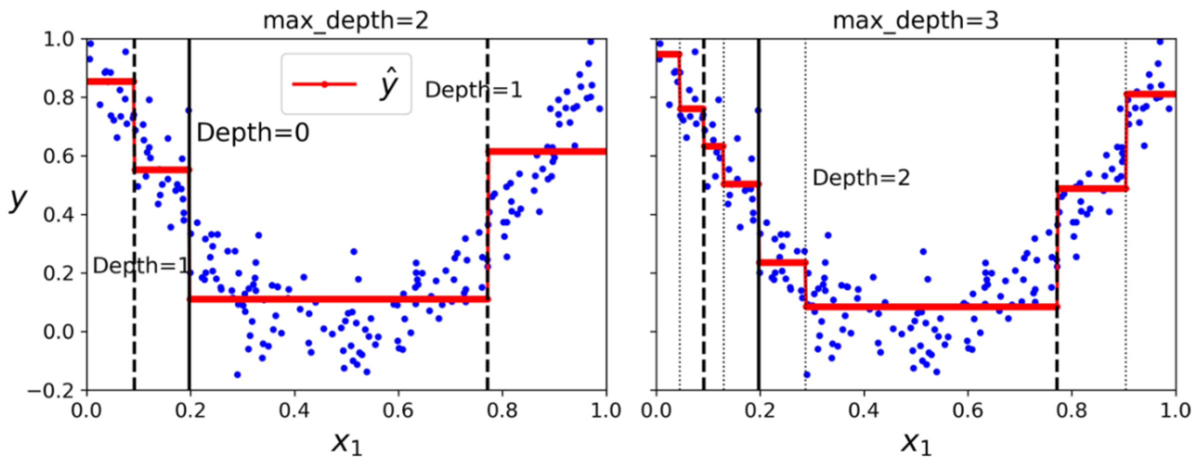
Decision Trees make very few assumptions about the training data (as opposed to linear models, which assume that the data is linear, for example). If left unconstrained, the tree structure will adapt itself to the training data, fitting it very closely—indeed, most likely overfitting it. Such a model is often called a nonparametric model, not because it does not have any parameters (it often has a lot) but because the number of parameters is not determined prior to training, so the model structure is free to stick closely to the data. In contrast, a parametric model, such as a linear model, has a pre- determined number of parameters, so its degree of freedom is limited, reducing the risk of overfitting (but increasing the risk of underfitting).

To avoid overfitting the training data, you need to restrict the Decision Tree's freedom during training. As you know by now, this is called regularization. Most of them use techniques such as max features but also there are others that work by first training the Decision Tree without restrictions, then pruning (deleting) unnecessary nodes.



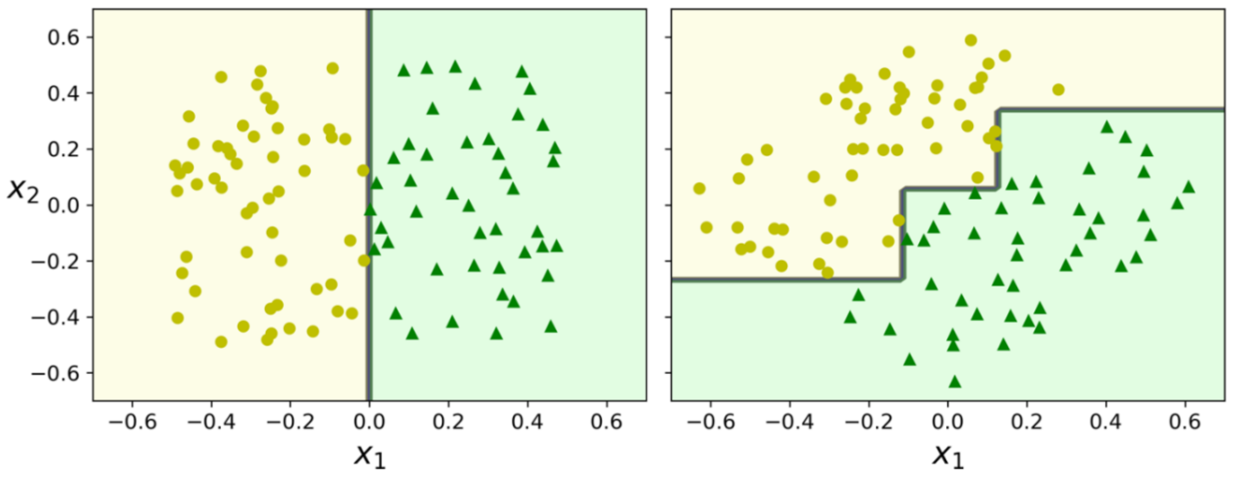
Regression

Decision Trees are also capable of performing regression tasks, the main difference is that instead of predicting a class in each node, it predicts a value. The predicted value for each region is always the average target value of the instances in that region.



Instability

Trees love orthogonal decision boundaries, which makes them sensitive to training set rotation.



More generally, the main issue with Decision Trees is that they are very sensitive to small variations in the training data. Small variations of the data can cause a different tree that the one before, this is due the fact that is stochastic and greedy algorithm. This problem is fixed later on with Random forests.