Decision trees models used for **classification** and **regression** tasks. They learn a hierarchy of if/else questions, leading to a decision.

To make a prediction, traverse the tree **based on the tests in each node** and **find the leaf the new data point falls into**. The output for this data point is the **mean target of the training points in this leaf**.

To prevent overfitting:
- stop the creation of the tree early (**pre-pruning**)
- build the tree but then remove or collapse nodes that contain little information (post-pruning or just pruning).

Possible criteria for pre-pruning:
- **limiting the maximum depth** of the tree
- **limiting the maximum number of leaves**
- **requiring a minimum number of points in a node** to keep splitting it

The reason why the accuracy on the training set is significantly higher is because the leaves are pure, the tree was grown deep enough that it could perfectly memorize all the labels on the training data.

Visualize the tree using the **export_graphviz** function from the tree module.

The most commonly used summary is **.feature_importance_**, which rates how important each feature is for the decision a tree makes (between 0 = ‘’not used’’ and 1 = ’’perfectly predicts the target’’ for each feature). The feature importances always sum to 1.

The **DecisionTreeRegressor** (and all other **tree-based regression** models) is **not able to extrapolate**, or make predictions outside of the range of the training data.

One of the pre-pruning strategies (max_depth, max_leaf_nodes, or min_samples_leaf) is sufficient to prevent overfitting.

Decision tree algorithms are **completely invariant to scaling** of the data. As each feature is processed separately, and the possible splits of the data don’t depend on scaling, **no preprocessing like normalization or standardization of features is needed**.

Decision trees work well when features are on completely different scales, or a mix of binary and continuous features.

Downside - tend to overfit and provide poor generalization performance.

Source:

'Introduction to Machine Learning with Python' (p.70-83) by **Andreas C.Müller** and **Sarah Guido**