DEVYANI BEOHAR

J068

Decision Tree API

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

Code:

class sklearn.tree.DecisionTreeClassifier(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*)

PARAMETERS:

criterion{“gini”, “entropy”}, default=”gini”
The function to measure the quality of a split.

splitter{“best”, “random”}, default=”best”
The strategy used to choose the split at each node.

max_depthint, default=None
The maximum depth of the tree.

min_samples_splitint or float, default=2
The minimum number of samples required to split an internal node:

- If int, then consider min_samples_split as the minimum number.

- If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

min_samples_leafint or float, default=1
The minimum number of samples required to be at a leaf node.

min_weight_fraction_leaffloat, default=0.0
The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

max_featuresint, float or {“auto”, “sqrt”, “log2”}, default=None
The number of features to consider when looking for the best split:

random_stateint, RandomState instance or None, default=None
Controls the randomness of the estimator.

max_leaf_nodesint, default=None
Grow a tree with max_leaf_nodes in best-first fashion.

min_impurity_decreasefloat, default=0.0
A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

class_weightdict, list of dict or “balanced”, default=None
Weights associated with classes in the form {class_label: weight}. If None, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

ccp_alphanon-negative float, default=0.0
Complexity parameter used for Minimal Cost-Complexity Pruning.

ATTRIBUTES:

classes_ndarray of shape (n_classes,) or list of ndarray

The classes labels (single output problem), or a list of arrays of class labels (multi-output problem).

max_features_int

The inferred value of max_features.

n_classes_int or list of int

The number of classes (for single output problems), or a list containing the number of classes for each output (for multi-output problems).

n_features_int

DEPRECATED: The attribute n_features_ is deprecated in 1.0 and will be removed in 1.2.

n_features_in_int

Number of features seen during fit.

feature_names_in_ndarray of shape (n_features_in_,)

Names of features seen during fit. Defined only when X has feature names that are all strings.

n_outputs_int

The number of outputs when fit is performed.

tree_Tree instance

The underlying Tree object.

ADVANTAGES

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.

DISADVANTAGES

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.