Decision Tree API

Aryaman Gautam

J001

A decision tree a tree like structure whereby an internal node represents an attribute, a branch represents a decision rule, and the leaf nodes represent an outcome. This works by splitting the data into separate partitions according to an attribute selection measure.

the Decision Tree Classifier from scikit-learn is that the target variable can be categorical or numerical.

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. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

- splitter{“best”, “random”}, default=”best”

The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

- max_depth int, default=None

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

- min_samples_split int or float, default=2

The minimum number of samples required to split an internal node:

- min_samples_leaf int or float, default=1

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- min_weight_fraction_leaf float, 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_features int, float or {“auto”, “sqrt”, “log2”}, default=None

The number of features to consider when looking for the best split:

the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features.

- random_state int, RandomState instance or None, default=None

Controls the randomness of the estimator. The features are always randomly permuted at each split, even if splitter is set to "best". When max_features < n_features, the algorithm will select max_features at random at each split before finding the best split among them

- max_leaf_nodes int, default=None

Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

- min_impurity_decrease float, 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_weight dict, 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.

For multioutput (including multilabel) weights should be defined for each class of every column in its own dict.

The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))

For multi-output, the weights of each column of y will be multiplied.

- ccp_alpha non-negative float, default=0.0

Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen. By default, no pruning is performed.

Attribute :

- 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).

- feature_importances_ ndarray of shape (n_features,)

Return the feature importances.

- 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

- 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.

The default values for the parameters controlling the size of the trees lead to fully grown and unpruned trees which can potentially be very large on some data sets. To reduce memory consumption, the complexity and size of the trees should be controlled by setting those parameter values.

Methods :

- apply(X[, check_input])
- cost_complexity_pruning_path(X, y[, …])
- decision_path(X[, check_input])
- fit(X, y[, sample_weight, check_input, …])
- get_depth()
- get_n_leaves()
- get_params([deep])
- predict(X[, check_input])
- predict_log_proba(X)
- predict_proba(X[, check_input])
- score(X, y[, sample_weight])
- set_params(**params)