Name: Rakshit

Scikit-learn library

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The library is focused on modeling data.It is not focused on loading, manipulating and summarizing data.

Parameters

**n\_estimators*integer, optional (default=100)***

The number of trees in the forest.

**criterion*string, optional (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.

**max\_depth*integer or None, optional (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, float, optional (default=2)***

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

**min\_samples\_leaf*int, float, optional (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.

**min\_weight\_fraction\_leaf*float, optional (default=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, string or None, optional (default=”auto”)***

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

**max\_leaf\_nodes*int or None, optional (default=None)***

Grow trees 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, optional (default=0.)***

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

**min\_impurity\_split*float, (default=1e-7)***

Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

**bootstrap*boolean, optional (default=True)***

Whether bootstrap samples are used when building trees. If False, the whole datset is used to build each tree.

**oob\_score*bool (default=False)***

Whether to use out-of-bag samples to estimate the generalization accuracy.

**random\_state*int, RandomState instance or None, optional (default=None)***

Controls both the randomness of the bootstrapping of the samples used when building trees (if bootstrap=True) and the sampling of the features to consider when looking for the best split at each node (if max\_features < n\_features).

**verbose*int, optional (default=0)***

Controls the verbosity when fitting and predicting.

**warm\_start*bool, optional (default=False)***

When set to True, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just fit a whole new forest.

**class\_weight*dict, list of dicts, “balanced”, “balanced\_subsample” or None, optional (default=None)***

Weights associated with classes in the form {class\_label: weight}. If not given, 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.

**max\_samples*int or float, default=None***

If bootstrap is True, the number of samples to draw from X to train each base estimator.