sklearn.ensemble.RandomForestRegressor

class sklearn.ensemble.RandomForestRegressor(*n_estimators=100*, ***, *criterion='squared_error'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *ccp_alpha=0.0*, *max_samples=None*

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Parameters:

n_estimators : *int, default=100*

The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"squared_error", "absolute_error", "poisson"}, default="squared_error"*

The function to measure the quality of a split. Supported criteria are "squared_error" for the mean squared error, which is equal to variance reduction as feature selection criterion, "absolute_error" for the mean absolute error, and "poisson" which uses reduction in Poisson deviance to find splits. Training using "absolute_error" is significantly slower than when using "squared_error".

New in version 0.18: Mean Absolute Error (MAE) criterion.

New in version 1.0: Poisson criterion.

Deprecated since version 1.0: Criterion "mse" was deprecated in v1.0 and will be removed in version 1.2. Use `criterion="squared_error"` which is equivalent.

Deprecated since version 1.0: Criterion "mae" was deprecated in v1.0 and will be removed in version 1.2. Use `criterion="absolute_error"` which is equivalent.

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:

- 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.

Changed in version 0.18: Added float values for fractions.

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.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

Changed in version 0.18: Added float values for fractions.

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 : *{"auto", "sqrt", "log2"}, int or float, default="auto"*

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

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `round(max_features * n_features)` features are considered at each split.
- If "auto", then `max_features=n_features`.
- If "sqrt", then `max_features=sqrt(n_features)`.
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Note: 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.

max_leaf_nodes : *int, 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, default=0.0*

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

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.

bootstrap : *bool, default=True*

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

oob_score : *bool, default=False*

Whether to use out-of-bag samples to estimate the generalization score. Only available if bootstrap=True.

n_jobs : *int, default=None*

The number of jobs to run in parallel. `fit`, `predict`, `decision_path` and `apply` are all parallelized over the trees. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

random_state : *int, RandomState instance or None, 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`). See Glossary for details.

verbose : *int, default=0*

Controls the verbosity when fitting and predicting.

warm_start : *bool, 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. See the Glossary.

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. See Minimal Cost-Complexity Pruning for details.

New in version 0.22.

max_samples : *int or float, default=None*

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

- If None (default), then draw `X.shape[0]` samples.
- If int, then draw `max_samples` samples.
- If float, then draw `max_samples * X.shape[0]` samples. Thus, `max_samples` should be in the interval `(0.0, 1.0]`.

New in version 0.22.

Methods

`apply`(X)	Apply trees in the forest to X, return leaf indices.
`decision_path`(X)	Return the decision path in the forest.
`fit`(X, y[, sample_weight])	Build a forest of trees from the training set (X, y).
`get_params`([deep])	Get parameters for this estimator.
`predict`(X)	Predict regression target for X.
`score`(X, y[, sample_weight])	Return the coefficient of determination of the prediction.
`set_params`(**params)	Set the parameters of this estimator.

Attributes:	**base_estimator_ : *DecisionTreeRegressor*** The child estimator template used to create the collection of fitted sub-estimators.
	estimators_ : *list of DecisionTreeRegressor* The collection of fitted sub-estimators.
	`feature_importances_` **: *ndarray of shape (n_features,)*** The impurity-based feature importances.
	`n_features_` **: *int*** DEPRECATED: Attribute `n_features_` was deprecated in version 1.0 and will be removed in 1.2.
	n_features_in_ : *int* Number of features seen during fit. *New in version 0.24.*
	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. *New in version 1.0.*
	n_outputs_ : *int* The number of outputs when `fit` is performed.
	oob_score_ : *float* Score of the training dataset obtained using an out-of-bag estimate. This attribute exists only when `oob_score` is True.
	oob_prediction_ : *ndarray of shape (n_samples,) or (n_samples, n_outputs)* Prediction computed with out-of-bag estimate on the training set. This attribute exists only when `oob_score` is True.