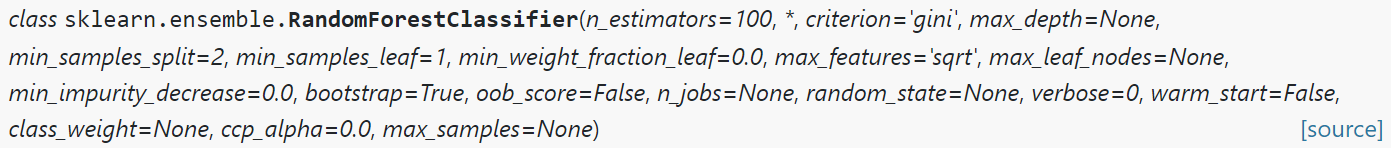
## Random Forest

## Overview of Random Forest algorithm

A random forest is a meta estimator that fits a number of decision tree classifiers 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.

**Code Implementation**

1. **Parameters**



**n\_ Estimators:** the number of trees in the random forest, that is, the number of learners.

**max\_ Depth:** the maximum depth of the tree. If default = none is selected, the tree will expand uniformly until all leaf nodes are of the same type of samples, or the number of minimum sample partitions (min\_samples\_split) is reached.

**min\_ samples\_ Split:** the minimum number of samples divided, that is, if the number of samples is less than or equal to this value, the current node cannot be divided.

**n\_ Jobs:** the number of processes used in parallel. The default is 1. If it is set to - 1, the value is the total number of cores.

**random\_ State:** random state, generated by np.numpy by default.

1. **Examples**

**>>> from** **sklearn.ensemble** **import** RandomForestClassifier

**>>> from** **sklearn.datasets** **import** make\_classification

**>>>** X, y = make\_classification(n\_samples=1000, n\_features=4,

**...**  n\_informative=2, n\_redundant=0,

**...**  random\_state=0, shuffle=**False**)

**>>>** clf = RandomForestClassifier(max\_depth=2, random\_state=0)

**>>>** clf.fit(X, y)

RandomForestClassifier(...)

**>>>** print(clf.predict([[0, 0, 0, 0]]))

**Run the program and screenshot the output result:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **Methods**

文本

描述已自动生成

1. **Practice**

from sklearn.model\_selection import cross\_val\_score  
from sklearn.datasets import make\_blobs  
from sklearn.ensemble import RandomForestClassifier  
  
*##Create 100 classes with 10000 samples and 10 features per sample*X, y = make\_blobs(n\_samples=10000, n\_features=10, centers=100, random\_state=0)  
  
*## Random forest*clf2 = RandomForestClassifier(n\_estimators=10, max\_depth=None, min\_samples\_split=2, random\_state=0)  
scores2 = cross\_val\_score(clf2, X, y)  
print(scores2.mean())

**Run the program and screenshot the output result:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**cross\_ val\_ score(model\_name, X,y， cv=k)**

Function: verify the stability of a model on a training set and output K prediction accuracy.

**K-fold cross validation (k-fold)**

The initial training samples are divided into k pieces, of which (k-1) is used as the training set and the remaining one is used as the evaluation set. In this way, the classifier can be trained K times and K training results can be obtained.