**Random Forest Models: Why Are They Better Than Single Decision Trees?**

# What category of algorithms does it belong to?

Similar to some other algorithms, Random Forest can handle both classification and regression.

A quick recap on the difference between classification and regression :

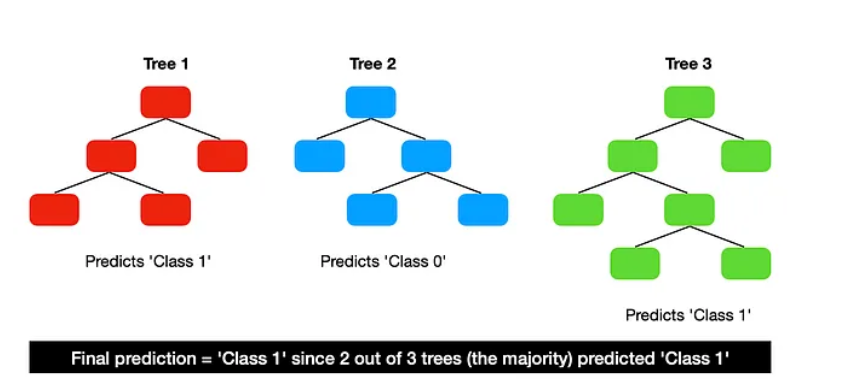
* With **classification**, we attempt to predict a class label. In other words, it is used for problems where the output (target variable) takes a finite set of values, e.g., whether it will rain tomorrow or not. Corresponding algorithm: [sklearn.ensemble.RandomForestClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html" \l "sklearn-tree-decisiontreeclassifier" \t "_blank)
* Meanwhile, **regression** is used to predict a numerical label. This means your output can take an infinite set of values, e.g., a house price. Corresponding algorithm:[sklearn.ensemble.RandomForestRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html)

# How does the Random Forest classification algorithm work?

Under the hood, the random forest is essentially a CART algorithm (Classification and Regression Trees), except it creates an **ensemble** of many trees instead of just one. This provides several advantages such as:

* Improved performance (the wisdom of crowds)
* Improved robustness (less likely to overfit since it relies on many random trees)

Random Forest algorithm uses majority agreement prediction for the class label, which means that each tree predicts whether the observation belongs to ‘Class 0’ or ‘Class 1’. If 55 trees out of a hundred predicted ‘Class 1’ and 45 predicted ‘Class 0’, then the final model prediction would be ‘Class 1’.



The majority prediction from multiple trees is better than an individual tree prediction because the trees protect each other from their individual errors. This is, however, dependant on the trees being relatively uncorrelated with each other.

In case of regression, each tree is created from a different sample of rows and at each node, a different sample of features is selected for splitting. Each of the trees makes its own individual prediction. **These predictions are then averaged to produce a single result.**

## The process of building multiple trees

The question is — how do we build many trees from the same data pool while keeping them relatively uncorrelated? There are two parts to it:

1. Bootstrap aggregation (random sampling with replacement)
2. Feature randomness

Decision trees are so-called **high-variance estimators**, which means that small changes to the sample data can greatly impact the tree structure and its prediction.

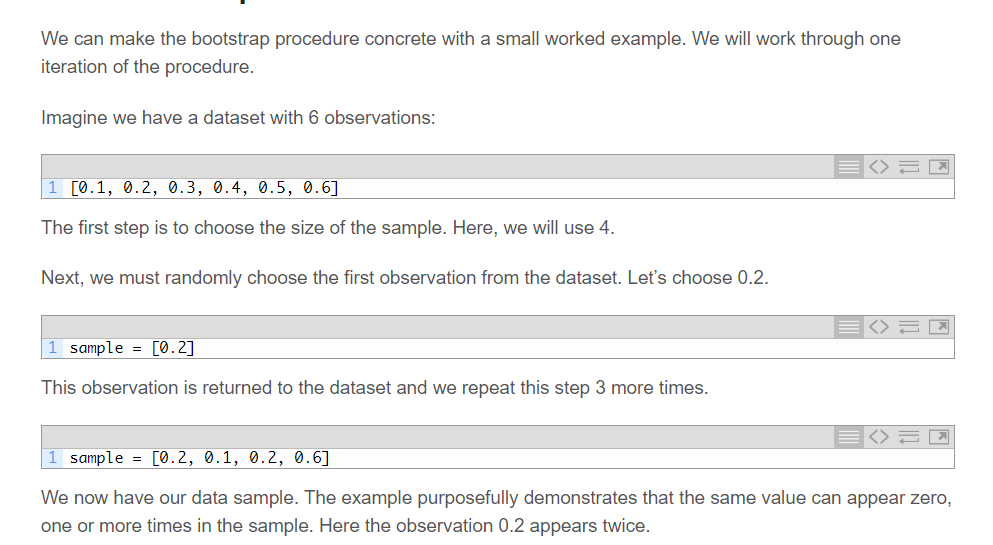
Ideally, you want to turn it into a **low-variance estimator** by creating many trees and using them in aggregation to make the prediction.

**What is bagging?** **Bagging, also known as bootstrap aggregation, is the ensemble learning method** that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once.

#### **How random forest works step by step?**

A. The following steps will tell you how random forest works:

1. **Create Bootstrap Samples**: Construct different samples of the dataset with replacements by randomly selecting the rows and columns from the dataset. These are known as bootstrap samples.



Bootstrapping is a technique where instead of taking the whole training data to build one tree, you use sampling with replacement to create many samples of the same size. Then you can use these samples to build separate trees.

**This method produces many samples with the same observations but different distributions. In the end, the aggregation of these trees leads to a reduction in the variance of your model prediction, turning it into a low-variance estimator**.

## **Feature randomness**

The second part in reducing the correlation between the trees comes from using a random selection of features when looking for the best split for each node.

For example:

Full list of features: [feat1, feat2, ..., feat10]Random selection of features (1): [feat3, feat5, feat8]  
Random selection of features (2): [feat1, feat5, feat10]  
...The split in the first node would use the most predcitive feature from a set of [feat3, feat5, feat8]  
The splint in the second node would use the most predictive feature from a set of [feat1, feat5, feat10]  
...*Note, when building a Random Forest model, you can control the number/proportion of features that are used to find the best split for each node.*

**This process allows the creation of less correlated trees and, combined with bootstrap aggregation, produces a Random Forest.**

#### **How random forest works step by step?**

A. The following steps will tell you how random forest works:

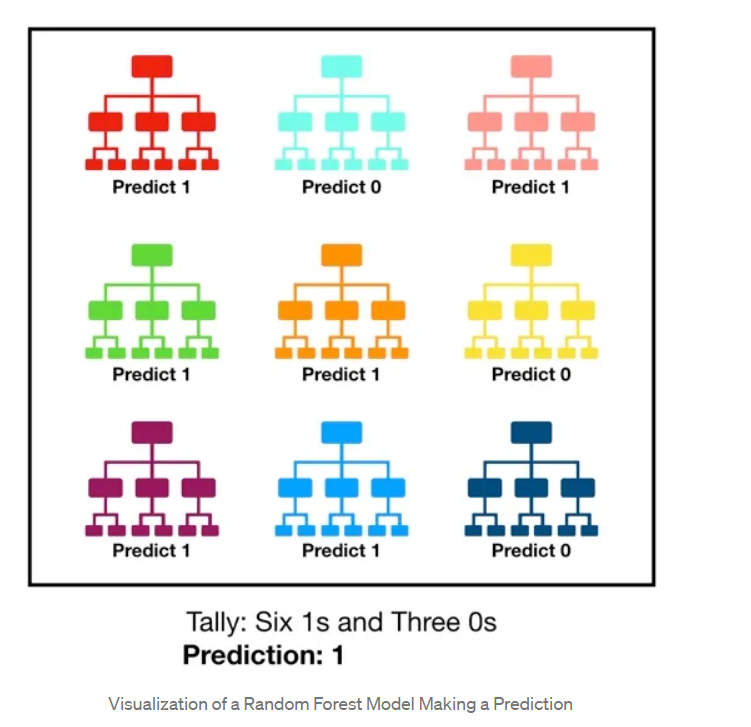
1. **Create Bootstrap Samples**: Construct different samples of the dataset with replacements by randomly selecting the rows and columns from the dataset. These are known as bootstrap samples. Simply put, n random records and m features are taken from the data set having k number of records.
2. **Build Decision Trees**: Construct the decision tree on each bootstrap sample as per the hyperparameters.
3. **Individual Outputs**: Each decision tree will generate an output.
4. **Final Output:** Final output is considered based on **Majority Voting or Averaging**for Classification and regression, respectively.

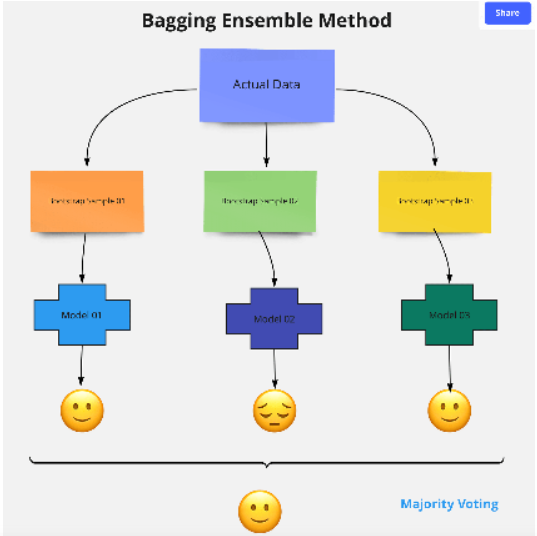
#### **What are the advantages of Random Forest?**

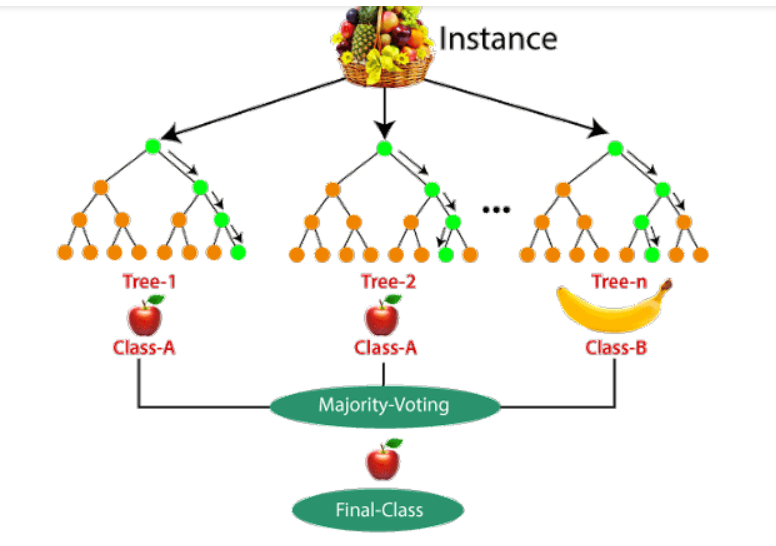
A. Random Forest tends to have a low bias since it works on the concept of bagging. It works well even with a dataset with a large no. of features since it works on a subset of features. Moreover, it is faster to train as the trees are independent of each other, making the training process parallelizable.

**Each tree works on a subset of the full dataset with some rows and columns. (features).**

Random Forest reduces overfitting by averaging multiple decision trees and is less sensitive to noise and outliers in the data. **It provides a measure of feature importance, which can be useful for feature selection and data interpretation.**







***A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.***

**The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don’t constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

## ***Important Features of Random Forest***

* **Diversity:**Not all attributes/variables/features are considered while making an individual tree; each tree is different.
* **Immune to the curse of dimensionality:** Since each tree does not consider all the features, the feature space is reduced.
* **Parallelization:**Each tree is created independently out of different data and attributes. This means we can fully use the CPU to build random forests.
* **Train-Test split:**In a random forest, we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
* **Stability:**Stability arises because the result is based on majority voting/ averaging.

## ***Difference Between Decision Tree and Random Forest***

|  |  |
| --- | --- |
| **Decision trees** | **Random Forest** |
| 1. Decision trees normally suffer from the problem of overfitting if it’s allowed to grow without any control. | 1. Random forests are created from subsets of data, and the final output is based on average or majority ranking; hence the problem of overfitting is taken care of. |
| 2. A single decision tree is faster in computation. | 2. It is comparatively slower. |
| 3. When a data set with features is taken as input by a decision tree, it will formulate some rules to make predictions. | 3. Random forest randomly selects observations, builds a decision tree, and takes the average result. It doesn’t use any set of formulas. |

**Therefore, random forests are much more successful than decision trees only if the trees are diverse and acceptable.**

## ***Important Hyperparameters in Random Forest***

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

***Random forest Classifier:***

[**https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html**](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)

**n\_estimatorsint, default=100:** The number of trees in the forest.

**criterion{“gini”, “entropy”, “log\_loss”}, default=”gini”.** The function to measure the quality of a 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\_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\_features{“sqrt”, “log2”, None}, int or float, default=”sqrt”.** The number of features to consider when looking for the best split:

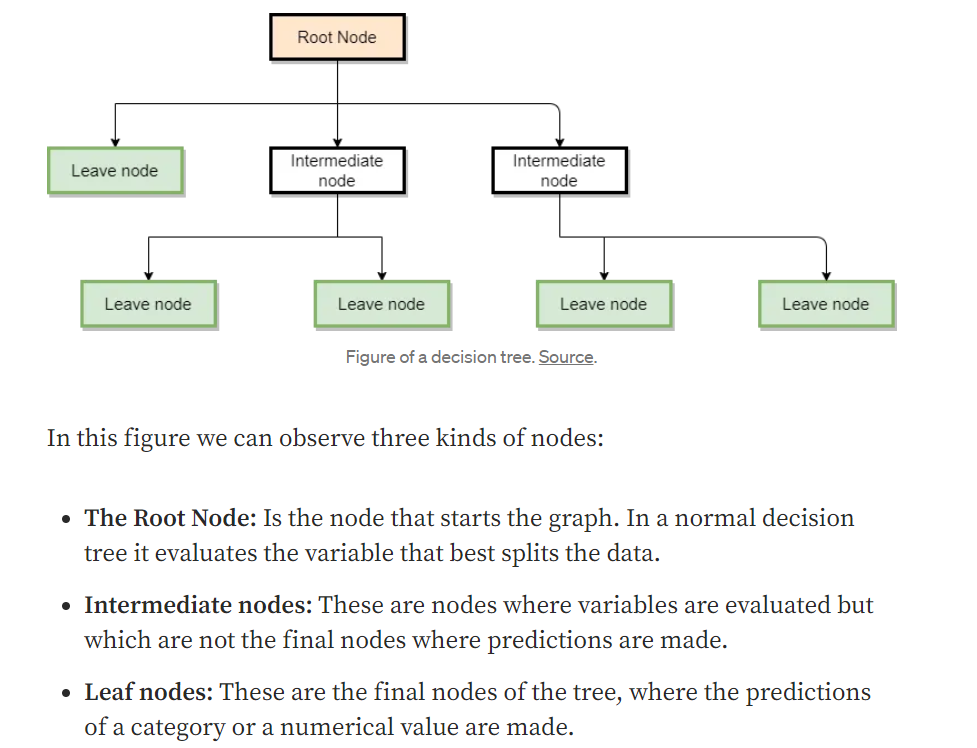
**max\_leaf\_nodes int, default=None**

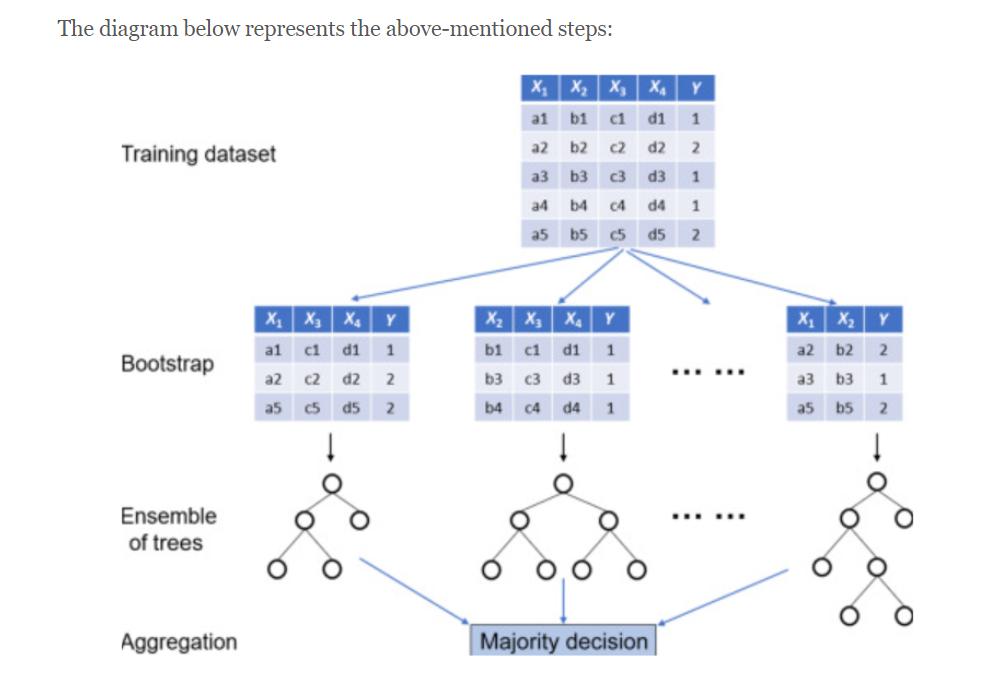
**bootstrapbool, default=True.** Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

**class\_weight{**“balanced”, “balanced\_subsample”}, dict or list of dicts, default=None. Weights associated with classes in the form {class\_label: weight}. If not given, all classes are supposed to have weight one.

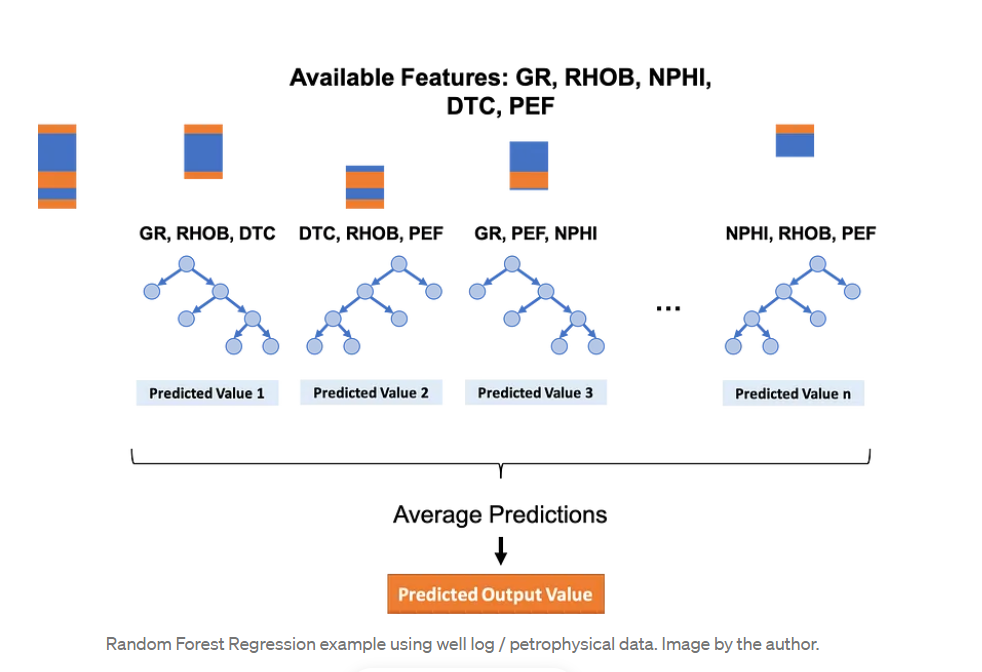
**max\_samples int or float, default=None** If bootstrap is True, the number of samples to draw from X to train each base estimator.

To get a grasp of this piece of documentation I think you should make the distinction between a **leaf** (also called **external node**) and an **internal node**. An internal node will have further splits (also called **children**), while a leaf is by definition a node without any children (without any further splits).

******

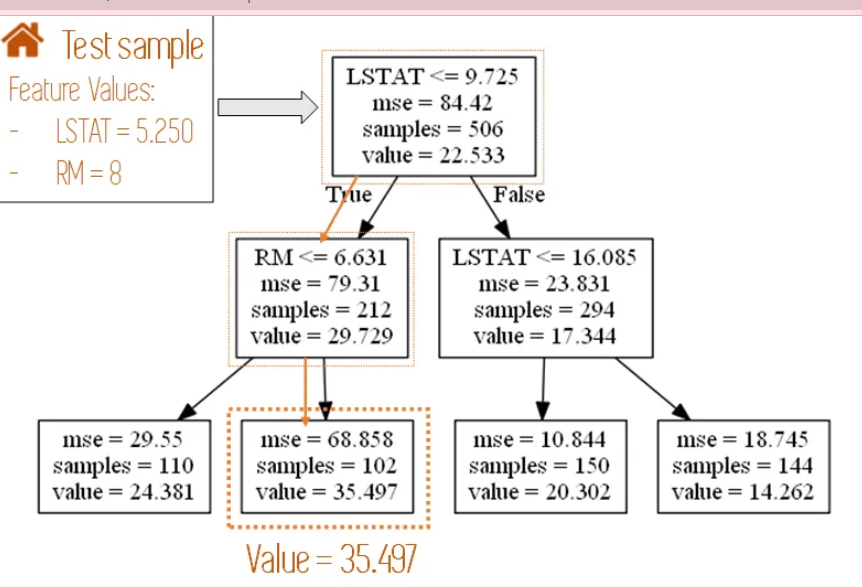


**Random Forest Regression:**



Also, because we are using trees and that the resulting class will be obtained by voting, we aren't inherently comparing between different values, only between the same types of values, so adjusting the features to the same scale isn't necessary in this case. This means that the Random Forest classification model is **scale invariant**, and you don't need to perform feature scaling.

*The more trees in the forest, the more diverse the model can be. There's a point of diminishing returns, though, as with many trees fit on a random subset of features, there will be a fair bit of similar trees that don't offer much diversity in the ensemble, and which will start to have****too much voting power****and skew the ensemble to be overfit on the training dataset, hurting generalization to the validation set.*

******

***In decision tress and random forest, the true goes to the left and false to the right when you split.***

***Parameters tuning in regression:***

[***https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html***](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html)

**n\_estimators *int, default=100***

**criterion*{“squared\_error”, “absolute\_error”, “friedman\_mse”, “poisson”}, default=”squared\_error”***The function to measure the quality of a split.

**max\_depth *int, default=None*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***

**max\_features*{“sqrt”, “log2”, None}, int or float, default=1.0***

**max\_leaf\_nodes *int, default=None***

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

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

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