

## Taken from Müller, Guido: “Introduction to Machine Learning with Python”

**Building random forests.** To build a random forest model, you need to decide on the number of trees to build (the `n_estimators` parameter of `RandomForestRegressor` or `RandomForestClassifier`). Let’s say we want to build 10 trees. These trees will be built completely independently from each other, and the algorithm will make different random choices for each tree to make sure the trees are distinct. To build a tree, we first take what is called a bootstrap sample of our data. That is, from our `n_samples` data points, we repeatedly draw an example randomly with replacement (meaning the same sample can be picked multiple times), `n_samples` times. This will create a dataset that is as big as the original dataset, but some data points will be missing from it (approximately one third), and some will be repeated.

To illustrate, let’s say we want to create a bootstrap sample of the list `['a', 'b', 'c', 'd']`. A possible bootstrap sample would be `['b', 'd', 'd', 'c']`. Another possible sample would be `['d', 'a', 'd', 'a']`.

Next, a decision tree is built based on this newly created dataset. However, the algorithm we described for the decision tree is slightly modified. Instead of looking for the best test for each node, in each node the algorithm randomly selects a subset of the features, and it looks for the best possible test involving one of these features. The number of features that are selected is controlled by the `max_features` parameter. This selection of a subset of features is repeated separately in each node, so that each node in a tree can make a decision using a different subset of the features.

The bootstrap sampling leads to each decision tree in the random forest being built on a slightly different dataset. Because of the selection of features in each node, each split in each tree operates on a different subset of features. Together, these two mechanisms ensure that all the trees in the random forest are different. A critical parameter in this process is `max_features`. If we set `max_features` to `n_features`, that means that each split can look at all features in the dataset, and no randomness will be injected in the feature selection (the randomness due to the bootstrapping remains, though). If we set `max_features` to 1, that means that the splits have no choice at all on which feature to test, and can only search over different thresholds for the feature that was selected randomly. Therefore, a high `max_features` means that the trees in the random forest will be quite similar, and they will be able to fit the data easily, using the most distinctive features. A low `max_features` means that the trees in the random forest will be quite different, and that each tree might need to be very deep in order to fit the data well.

To make a prediction using the random forest, the algorithm first makes a prediction for every tree in the forest. For regression, we can average these results to get our final prediction. For classification, a “soft voting” strategy is used. This means each algorithm makes a “soft” prediction, providing a probability for each possible output label. The probabilities predicted by all the trees are averaged, and the class with the highest probability is predicted.