CHEAT SHEET

Random Forest

Algorithm Name	Random forest
Description	Random forest combines decision trees into an ensemble. The models are trained independently (possibly in parallel) on data bootstraps with one small modification: For each split in each tree, a small subset of k features is randomly sampled and all other features are ignored. This procedure reduces the variance of the final model as training trees on random features results in them being uncorrelated further.
Applicability	Classification and regression problems
Assumptions	Data set is not too high-dimensional; similar inputs have similar labels.
Underlying Mathematical Principles	Full-depth trees have high variance. Random forests combine many high-variance models to create a low-variance ensemble.
Hyperparameters	• Number of trees N to average. • Number of features k to sub-sample. Random forests are famously insensitive to hyperparameters. Default choices: N large enough until the predictions converge (N=100, or N =1000) $k = \sqrt{d}$, where d is the dimensionality of the input data
Setting	Regression or classification. While **regression** trees return continuous values, we can still use them to solve **classification** problems with discrete labels. For example, we can return the sign of the output of the tree.
Out-of-Bag Error	Because each tree is not trained on the full data but only a sub-sample, each tree has its own out-of-bag subset of the training data on which it was not trained. One can estimate the test error of a random forest classifier by computing the classification error of each data point obtained by averaging the predictions of those ensemble members that were **not** trained on this data point.
Ensemble	Combination of many decision trees
Prediction Confidence	Random forests naturally provide prediction confidences. For example, in classification settings, you can output the percentage of trees that predicted the most common label as a statistic of confidence. In regression settings, you can output the variance of the predictions of all ensemble members.

Feature Importance	Random forests can naturally provide a measure of feature importance. For example, for each feature, compute the average (or total) reduction in loss obtained on the training set through splitting on this particular feature.
Strengths	Random forests are particularly well suited for applications with heterogeneous features. The big advantages of random forests are that they tend to not overfit to the training data, are amazingly insensitive to hyperparameters, work naturally for regression and classification settings, provide feature importances, output prediction confidences, and provide an unbiased estimate of the testing error directly from the training set.
Weakness	Random forests tend to excel in lower-dimensional feature spaces (<1000 dimensions) and are often not well suited for very high and sparse feature spaces. For very large data sets (n in the millions), random forests can become slow to evaluate, as the trees become too large.