**Why Random Forest is My Favorite Machine Learning Model**

**Discover the real world advantages and drawbacks of the Random Forest**

[](https://towardsdatascience.com/@julia_kho?source=post_page-----b97651fa3706----------------------)

[Julia Kho](https://towardsdatascience.com/@julia_kho?source=post_page-----b97651fa3706----------------------)

[Oct 19, 2018](https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706?source=post_page-----b97651fa3706----------------------) · 5 min read

“green trees on foggy forest” by [Julien R](https://unsplash.com/@djulien?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

“A model is like a pair of goggles. It puts certain things into focus.” — My Data Science Instructor.

No single algorithm dominates when choosing a machine learning model. Some perform better with large data sets and some perform better with high dimensional data. Thus, it is important to assess a model’s effectiveness for your particular data set. In this article, I will give a high level overview of how random forest works and discuss the real world advantages and drawbacks of this model.

**Essentially, Random Forest is a good model if you want high performance with less need for interpretation.**

Random Forest is always my go to model right after the regression model. Let me tell you why.

**What is Random Forest?**

Random forests are **bagged decision tree** models that split on a **subset of features** on each split. This is a huge mouthful, so let’s break this down by first looking at a single decision tree, then discussing bagged decision trees and finally introduce splitting on arandom subset of features.

**Decision Tree**

Essentially, a decision tree splits the data into smaller data groups based on the features of the data until we have a small enough set of data that only has data points under one label. Let’s look at an example. Below is a decision tree of whether one should play tennis.

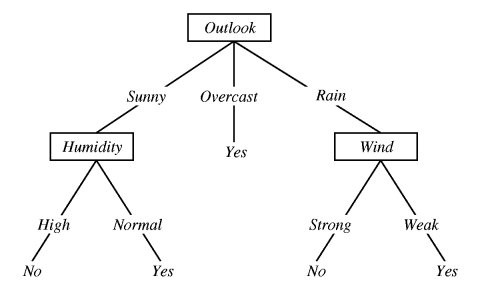


Image courtesy of <http://science.slc.edu/~jmarshall/courses/2005/fall/cs151/lectures/decision-trees/>

In the example above, the decision tree is split on multiple features until we reach a conclusion of “Yes”, we should play tennis, or “No” we should not play tennis. Follow the lines along the tree to determine the decision. For example, if the outlook is overcast, then “Yes” we should play tennis. If the outlook is sunny and humidity is high, then “No” we should not play tennis.

In a decision tree model, these splits are chosen according to a purity measure. That is, at each node, we want information gain to be maximized. For a regression problem, we consider residual sum of square (RSS) and for a classification problem, we consider the Gini index or entropy. I won’t go into too much detail on this, but if you are interested in learning more, check out this [lecture](https://www.slideshare.net/marinasantini1/lecture-4-decision-trees-2-entropy-information-gain-gain-ratio-55241087).

**Bagged Trees**

Now take the decision tree concept and let’s apply the principles of bootstrapping to create bagged trees.

**Bootstrapping** is a sampling technique in which we randomly sample with replacement from the data set.

Side note: When bootstrapping, we use only about 2/3 of the data. The approximately 1/3 of the data (“out-of-bag” data) is not used in the model and can conveniently be used as a test set.

**Bagging**, or bootstrap aggregating, is where we create bagged trees by creating *X* number of decision trees that is trained on *X* bootstrapped training sets. The final predicted value is the average value of all our *X* decision trees. One single decision tree has high variance (tends to overfit), so bybagging or combining many weak learners into strong learners, we are averaging away the variance. It’s a majority vote!

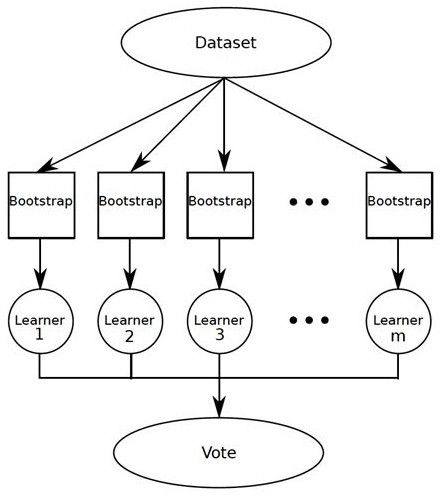


Image courtesy of <https://www.kdnuggets.com/2016/11/data-science-basics-intro-ensemble-learners.html>

**Random Forest**

Random forest improves on bagging because it **decorrelates** the trees with the introduction of splitting on a **random subset of features**. This means that at each split of the tree, the model considers only a small subset of features rather than all of the features of the model. That is, from the set of available features n, a subset of m features (m=square root of n) are selected at random. This is important so that variance can be averaged away. Consider what would happen if the data set contains a few strong predictors. These predictors will consistently be chosen at the top level of the trees, so we will have very similar structured trees. In other words, the trees would be highly correlated.

So in summary of what was stated initially, random forests are bagged decision tree models that split on a subset of features on each split.

**Why is Random Forest So Cool?**

**Impressive in Versatility**

Whether you have a regression or classification task, random forest is an applicable model for your needs. It can handle binary features, categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed.

**Parallelizable**

They are parallelizable, meaning that we can split the process to multiple machines to run. This results in faster computation time. Boosted models are sequential in contrast, and would take longer to compute.

Side note: Specifically, in Python, to run this in multiple machines, provide the parameter “n\_jobs = -1” The -1 is an indication to use all available machines. See [scikit-learn](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) documentation for further details.

**Great with High dimensionality**

Random forests is great with high dimensional data since we are working with subsets of data.

**Quick Prediction/Training Speed**

It is faster to train than decision trees because we are working only on a subset of features in this model, so we can easily work with hundreds of features. Prediction speed is significantly faster than training speed because we can save generated forests for future uses.

**Robust to Outliers and Non-linear Data**

Random forest handles outliers by essentially binning them. It is also indifferent to non-linear features.

**Handles Unbalanced Data**

It has methods for balancing error in class population unbalanced data sets. Random forest tries to minimize the overall error rate, so when we have an unbalance data set, the larger class will get a low error rate while the smaller class will have a larger error rate.

**Low Bias, Moderate Variance**

Each decision tree has a high variance, but low bias. But because we average all the trees in random forest, we are averaging the variance as well so that we have a low bias and moderate variance model.

**Drawbacks**

1. Model interpretability: Random forest models are not all that interpretable; they are like black boxes.
2. For very large data sets, the size of the trees can take up a lot of memory.
3. It can tend to overfit, so you should tune the hyperparameters.