Random Forests

Jan-Philipp Kolb

04 Juni, 2019

Random Forests

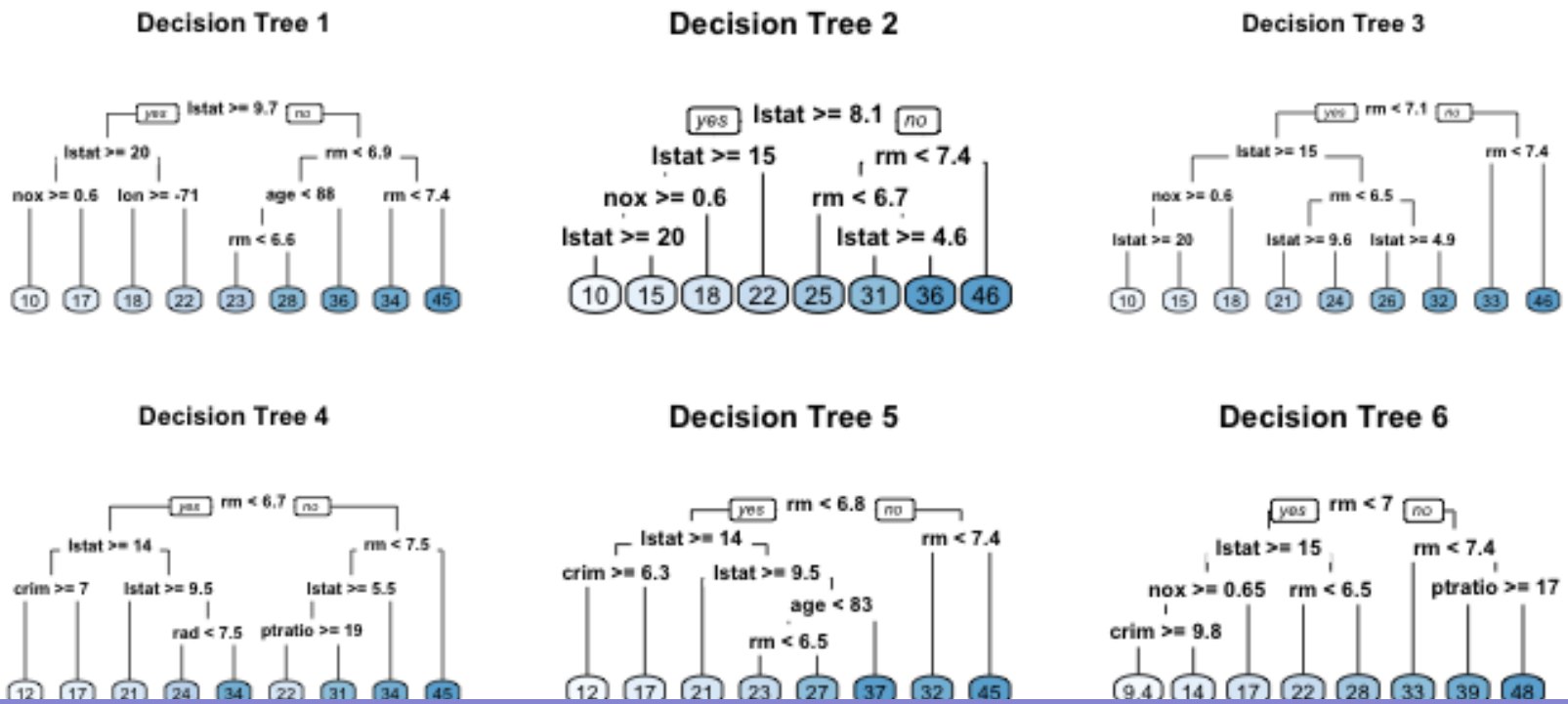
- **Bagging** can turn a single tree model with high variance and poor predictive power into a fairly accurate prediction function.
- But bagging suffers from **tree correlation**, which reduces the overall performance of the model.
- **Random forests** are a modification of bagging that builds a large collection of de-correlated trees
- It is a very popular **out-of-the-box** learning algorithm that enjoys good predictive performance.

Extending the bagging technique

- Bagging introduces a random component in to the tree building process
- The trees in bagging are not completely independent of each other since all the original predictors are considered at every split of every tree.
- Trees from different bootstrap samples have similar structure to each other (especially at the top of the tree) due to underlying relationships.

Similar trees - tree correlation

- If we create six decision trees with different bootstrapped samples of the Boston housing data, the top of the trees all have a very similar structure.
- Although there are 15 predictor variables to split on, all six trees have both `lstat` and `rm` variables driving the first few splits.



Tree correlation

- Tree correlation prevents bagging from optimally reducing variance of the predictive values.
- To reduce variance further, we need to minimize the amount of correlation between the trees.
- This can be achieved by injecting more randomness into the tree-growing process.

Random forests achieve this in two ways:

1) Bootstrap:
   - Similar to bagging, each tree is grown to a bootstrap resampled data set, which makes them different and decorrelates them.
2) Split-variable randomization:
   - For every split, the search for the split variable is limited to a random subset of m of the p variables.
   - For regression trees, typical default values are $m = p/3$ (tuning parameter).
   - When $m = p$, the randomization is limited (only step 1) and is the same as bagging.

Basic algorithm

The basic algorithm for a regression random forest can be generalized:

```
1.  Given training data set
2.  Select number of trees to build (ntrees)
3.  for i = 1 to ntrees do
4.  |  Generate a bootstrap sample of the original data
5.  |  Grow a regression tree to the bootstrapped data
6.  |  for each split do
7.  |  | Select m variables at random from all p variables
8.  |  | Pick the best variable/split-point among the m
9.  |  | Split the node into two child nodes
10. |  end
11. | Use tree model stopping criteria to determine: tree comple
12. end
```

The algorithm randomly selects a bootstrap sample to train and predictors to use at each split.

Summary - random forests

- Random forests provide a very powerful out-of-the-box algorithm that often has great predictive accuracy.
- Because of their more simplistic tuning nature and the fact that they require very little, if any, feature pre-processing they are often one of the first go-to algorithms when facing a predictive modeling problem.

Advantages & Disadvantages

Advantages - random forrests

- Typically have very good performance
- Remarkably good "out-of-the box" - very little tuning required
- Built-in validation set - don't need to sacrifice data for extra validation
- No pre-processing required
- Robust to outliers

Disadvantages - random forrests

- Can become slow on large data sets
- Although accurate, often cannot compete with advanced boosting algorithms
- Less interpretable