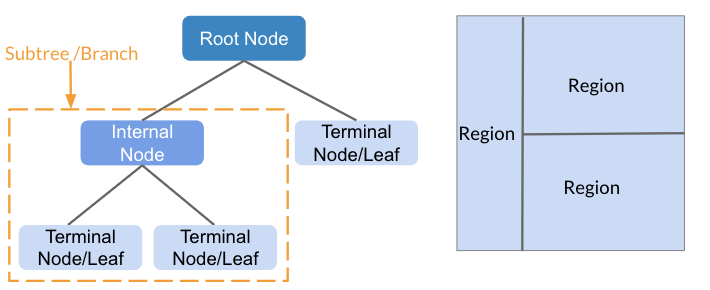
**Tree Based Method: Regression Trees**

* **Definition of Decision Tree:** A decision tree is a [decision support](https://en.wikipedia.org/wiki/Decision_support_system) tool that uses a [tree-like](https://en.wikipedia.org/wiki/Tree_(graph_theory)) [model](https://en.wikipedia.org/wiki/Causal_model) of decisions and their possible consequences.
* **Decision Tree and Tree Map**

****

* **Terminology Related to Decision Tree**

terminal node/leaf: nodes do not split

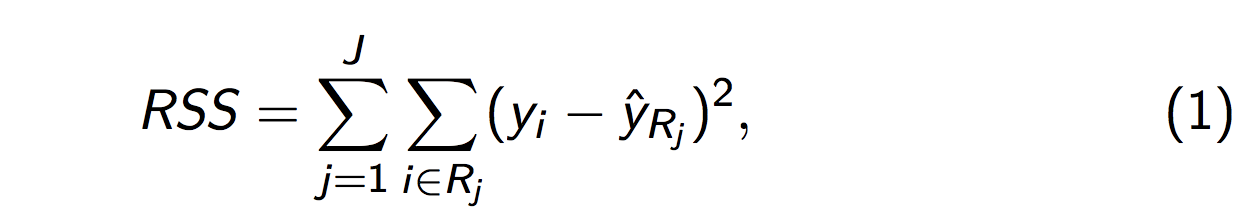
internal node: the point along the tree where the predictor space is split

subtree/branch: the segments of the trees that connect the nodes

* **Main Idea of Regression Tree**

We divide the predictor space—that is, the set of possible values for X1, . . . , Xp—into *J* distinct and non-overlapping regions, R1, . . . , RJ . For every observation that falls into the region RJ , we make the same prediction, which is simply the mean of the response values for the training observations in RJ .

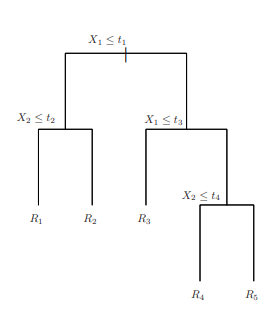
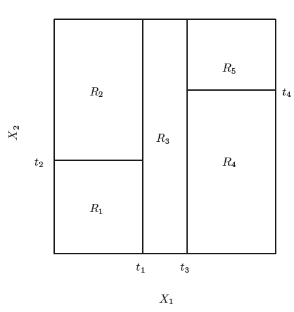
* **RSS Function**

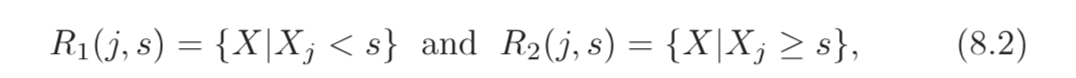
Our goal is to find boxes R1, . . . , RJ that minimize the RSS given by , where is the mean response for the training observations within the jth box.

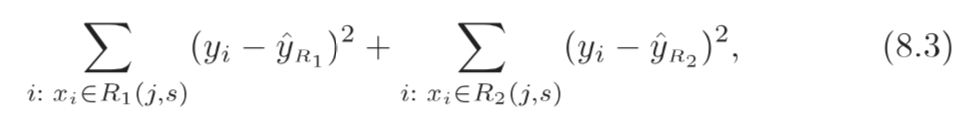
* **Partitioning Method: Recursive binary splitting**

1. It is a top-down, greedy approach.

It’s called top-down since it begins at the top of the tree (all observations below to a single region) and then successively splits the predictor space. Each split is indicated via two new branches further down on the tree. It is greedy since at each step of the tree building process, the best split is made at that particular split (rather than looking ahead and picking a split that will lead to a better tree in a future split).

1. For any j and s, we define the pair of half-planes

And we seek the j and s that minimize the equation

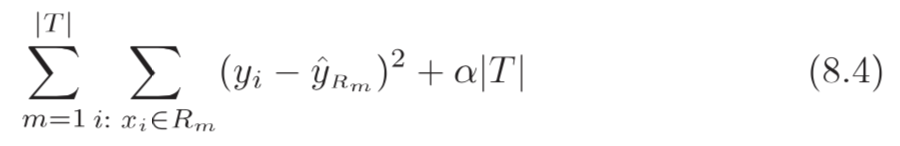
Where is the mean response for the training observations in R1(j,s) and is the mean response for the training observations in R2(j,s).

* **Problem of Overfitting**

The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance. A smaller tree with fewer splits (that is, fewer regions R1, . . . , RJ) might lead to lower variance and better interpretation at the cost of a little bias.

* **Pruning Method: Cost Complexity Pruning**

1. Grow a very large tree T0 and then prune it back in order to obtain a subtree. The suggested way to prune the tree is *Cost Complexity Pruning* - also known as *Weakest Link Pruning*. Rather than looking at all possible subtrees, we consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α, there is a corresponding subtree T ∈ To such that

is as small as possible.

1. When α = 0, then the subtree T will simply equal To, because then it just measures the training error.
2. As α = 0 increases, there is a price to pay for having a tree with many terminal nodes, and so it will be minimized for a smaller sub-tree.
3. If you have seen the lasso, it is similar to it in the sense the ways the lasso controls the complexity of the linear model. As α = 0 increases from 0 in it, branches are pruned from the tree in a nested and predictable way (resulting in the whole sequence of subtrees as a function of α = 0 is easy).

We can select an α using a validation set or using cross-validation. This process is summarized in the algorithm above.

* **Summary of Algorithm for Building a Regression Tree**

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each k = 1, . . . ,K
   1. Repeat Steps 1 and 2 on all but the kth fold of the training data.
   2. Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α. Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.