**Chapter 8 – Tree based methods**

* In this chapter, we describe tree-based methods for regression and classification. These involve stratifying or segmenting the predictor space into a number of simple regions.
* Tree-based methods are simple and useful for interpretation. However, they typically are not competitive with the best supervised learning ap- proaches, such as those seen in Chapters 6 and 7, in terms of prediction accuracy.
* Hence in this chapter we also introduce bagging, random forests, and boosting. Each of these approaches involves producing multiple trees which are then combined to yield a single consensus prediction.

*Decision trees*

* Decision trees can be applied to both regression and classification problems.
* In keeping with the tree analogy, the regions R1, R2, and R3 are known as terminal nodes or leaves of the tree.
* The points along the tree where the predictor space is split are referred to as internal nodes. In Figure 8.1, the two internal nodes are indicated by the text Years<4.5 and Hits<117.5. We refer to the segments of the trees that connect the nodes as branches.
* We now discuss the process of building a regression tree. Roughly speaking, there are two steps.
  + 1. We divide the predictor space—that is, the set of possible values for X1, X2, . . . , Xp—into J distinct and non-overlapping regions, R1,R2,...,RJ.
  + 2. 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.
* Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes. For this reason, we take a top-down, greedy approach that is known as *recursive binary splitting*. The approach is top down because it begins at the top of the tree (at which point all observations belong 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 because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.
* 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. This is because the resulting tree might be too complex. 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.
* Therefore, a better strategy is to grow a very large tree T0, and then prune it back in order to obtain a subtree. How do we determine the best way to prune the tree? Intuitively, our goal is to select a subtree that leads to the lowest test error rate. Given a subtree, we can estimate its test error using cross-validation or the validation set approach.
* However, estimating the cross-validation error for every possible subtree would be too cumbersome, since there is an extremely large number of possible subtrees. Instead, we need a way to select a small set of subtrees for consideration.
* Cost complexity pruning—also known as weakest link pruning—gives us a way to do just this. Rather than considering every possible subtree, we consider a sequence of trees indexed by a nonnegative tuning parameter α.

*Classification trees*

* A classification tree is very similar to a regression tree, except that it is classification used to predict a qualitative response rather than a quantitative one.
* Re- tree call that for a regression tree, the predicted response for an observation is given by the mean response of the training observations that belong to the same terminal node. In contrast, for a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs. In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the class proportions among the training observations that fall into that region.
* However, in the classification setting, RSS cannot be used as a criterion for making the binary splits. A natural alternative to RSS is the classification error rate.

*Trees versus linear models*

* If the relationship between the features and the response is well approximated by a linear model as in (8.8), then an approach such as linear regression will likely work well, and will outperform a method such as a regression tree that does not exploit this linear structure. If instead there is a highly non-linear and complex relationship between the features and the response as indicated by model (8.9), then decision trees may outperform classical approaches.

*Bagging*

* Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the bagging variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
* Recall that given a set of n independent observations Z1,...,Zn, each with variance σ2, the variance of the mean Z ̄ of the observations is given by σ2/n. In other words, averaging a set of observations reduces variance. Hence a natural way to reduce the variance and hence increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.
* Of course, this is not practical because we generally do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training data set
* While bagging can improve predictions for many regression methods, it is particularly useful for decision trees. To apply bagging to regression trees, we simply construct B regression trees using B bootstrapped training sets, and average the resulting predictions. These trees are grown deep, and are not pruned.
* Hence each individual tree has high variance, but low bias. Averaging these B trees reduces the variance.
* How can bagging be extended to a classification problem where Y is qualitative? In that situation, there are a few possible approaches, but the simplest is as follows. For a given test observation, we can record the class predicted by each of the B trees, and take a majority vote: the overall prediction is the most commonly occurring class among the B predictions
* When we bag a large number of trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree, and it is no longer clear which variables are most important to the procedure. Thus, bagging improves prediction accuracy at the expense of interpretability.
* Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees).

*Random forests*

* Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictor
* In other words, in building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors.
* Suppose that there is one very strong predictor in the data set, along with a num- ber of other moderately strong predictors. Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other.
* Random forests overcome this problem by forcing each split to consider only a subset of the predictors. Therefore, on average (p − m)/p of the splits will not even consider the strong predictor, and so other predictors will have more of a chance. We can think of this process as decorrelating the trees, thereby making the average of the resulting trees less variable and hence more reliable.
* The main difference between bagging and random forests is the choice of predictor subset size *m*.

*Boosting*

* Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
* Boosting works in a similar way to bagging, except that the trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.
* Unlike fitting a single large deci- sion tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly. Given the current model, we fit a decision tree to the residuals from the model. That is, we fit a tree using the current residuals, rather than the outcome Y , as the re- sponse. We then add this new decision tree into the fitted function in order to update the residuals
* In general, statistical learning approaches that learn slowly tend to perform well. Note that in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown.