1 Tree Based Models

The idea behind this family of methods is to segment the region into subspaces and the prediction for any subspace is made by taking the *mean* or *mode* of the response in that region.
A simple decision tree may not give performance at par with linear regression or generalized additive models, but when combined with techniques like bagging and boosting, multiple trees can yield significant reduction in bias.
A simple decision tree finds its utility in being relatively simpler in interpretation than it's derived non-linear family.

1.1 Regression Trees

The algorithm behind building trees consists of two basic steps

1. Divide the predictors $X_1, X_2, \ldots, X_n$ into J **distinct** and **non-overlapping** regions $R_1, R_2, \ldots, R_J$.
2. For making a prediction in the region R_j, simply take the **response mean** of all the data points following in that region.

With the above definition, the loss becomes

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_J} (y_i - \bar{y}_{R_J})^2$$

But minimizing this loss is infeasible in practice as the total number of possible partitions are too large !

To overcome this, we build the trees in a greedy top down approach called **recursive binary splitting**. This method aims to find the best possible split at each level in a greedy manner.

1.1.1 The Algorithm

To start off, the alogithm will try to split the data into two regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ using the values of predictor X_j and the search is performed across all the predictors to choose the one which minimizes the RSS. Mathematically

$$R_1(j,s) = \{X|X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \geq s\}$$
$$\underset{j,s}{\text{minimize}} \sum_{i:x_i \in R_1} (y_i - \bar{y}_{R_1})^2 + \sum_{i:x_i \in R_2} (y_i - \bar{y}_{R_2})^2$$

where $\bar{y}_{R_1}$ and $\bar{y}_{R_2}$ are the response mean in the region R_1 and R_2 respectively.

The same algorithm is run recursively. The only change that is made that after the first split, we now wolr independently on two separate regions. For each region, we will not be able to consider all the possible values of s since the region has been restricted. Otherwise, the RSS term will remain the same.

The process is stopped when a pre-determined stopping criteria is reached. For making a prediction, we will first determine the region in which the test observation falls and then make the prediction based on the mean of the training samples in this region.

1.1.2 Tree Pruning

The tree build using above strategy can have good results on the training set, but will yield poor performance on the test set due to the high complexity. A better strategy is to build **short trees which are less complex and produce lower variance at cost of little bias** and are more interpretable.

A simple strategy can be to build only when there is a decrease in RSS above a threshold. This approach can be short-sighted as some future good branches can be missed.

Tree pruning is to **build a large tree and then shrink it back to obtain a subtree**. The RSS of a subtree can be found via cross validation. This approach is expensive for all possible subtrees.

Cost complexity Pruning or **weakest link pruning** is an approach to overcome this problem. For a non-negative parameter α, there exists a subtree T of the large tree T_0 such that the following is minimized

$$RSS = \sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \bar{y}_{R_m})^2 + \alpha |T|$$

where $|T|$ is the number of terminal nodes in the tree. The formulation is similar to lasso and here we are controlling the complexity of the tree via a parameter α. Larger the α, the less number of terminal nodes there would be in the tree.

The algorithm can then be summarized as follows

1. Use *binary recursive splitting* to obtain a large tree on the data set. Stop only when at a terminal node, the number of observations is less than a minimum.
2. Apply *cost complexity pruning* in order to obtain a small tree T as a function of the cost parameter α (fix a set of α and get a set of subtrees).
3. Use K-fold cross validation to choose α. For each of the folds $k = 1, 2, \ldots, K$
   (a) Repeat steps 1 and 2 excluding the data from the kth fold to obtain a family of subtrees as a function of α.
   (b) Evaluate the mean predicted validation error as a function of α.
   Choose the value of α that minimizes the average error across all the folds.
4. Select the substree from step 2 that corresponds to the chosen value of α.

1.2 Classification Trees

Here we predict a qualitative variable and will report the **most occurring class in the region**. Since we are almost always also interested in the probabilistic aspects of the prediction, we also consider **the proportion of the classes occurring in a region**.

1.2.1 The Algorithm

The algorithm is almost similar to the regression trees but instead of RSS, we need to use something that is related to class proportions and frequencies. A very straightforward metric is **classification error rate**

$$E = 1 - \max_k \hat{p}_{mk}$$

where $\hat{p}_{mk}$ is the proportion of the kth class in the mth region. However, in practice, this metric is not sufficient to grow the trees.

1.2.2 Gini Index

Gini Index is defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

for a given region. Clearly, G is close to zero when the node is pure, i.e. most of the observations come from the same class. Gini index is also know as a measure of **purity** since it's low value implies predominance of a single class in the node.

1.2.3 Entropy

Entropy is defined as

$$D = \sum_{k=1}^{K} -\hat{p}_{mk} \log(\hat{p}_{mk})$$

for a region. Entropy takes value close to zero if the class probabilities are close to 0 or 1. Gini index and Entropy are somewhat similar numerically.

1.3 Bagging

Decision tree suffer from a high variance. When the data is divided into smaller parts, it is quite likely to expect variance when the data set itself is changed. Bootstrap Aggregation of Bagging is a method to try to overcome this problem.

Averaging a set of random variables reduces variance. Consider n random variables with the same variance σ^2, then the variance of their average is σ^2/n. Naturally, we can extend this idea to build several prediction models on different training data sets and average out their results to reduce the variance in the response.

We train b different models, make predictions $\hat{f}_i(x)$ using these and then average them out as

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)$$

to get a low variance prediction statistic.

Since we do not have access to different training datasets, we will get the average using bootstraps of the original data set. Note that **trees grown on bootstrapped data sets are deep and not pruned** so that they have low bias. Even though they may have high variance, averaging will reduce it out.

Averaging bootstrapped predictors works for regression while we **take the majority vote in case of classification**. The overall prediction is the most commonly occurring class across the B trees. Note that **using a large value of B will not lead to overfitting**, it should just be sufficiently large to ensure the error has come down compared to a single tree (building too many trees can take up significant time).