6.)

A Detailed Explanation of the Algorithm used to fit a Regression Tree

[redacted]

↳ Based loosely on ISLR Algorithm 8.1.

Step #1: Grow a large tree on the training data.

- We do this by, for each node (including the first node), we check if the stopping criterion has been met. Often this stopping condition [redacted] is to stop if a node has fewer than a given minimum number of observations.
  - ★ If criterion has been met, do not split that terminal node any further.
  - ★ If criterion has NOT yet been met, then make the best binary split at that node. The best split is the one that [redacted] minimizes, [redacted] for 2 regions (of the node in question) that form a partition over the whole node in question space, call those 2 regions R_1 and R_2, the quantity:

[redacted]

$$RSS = \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \quad (\star\star)$$

where: y_i is the value of the response value for obs i,

2 regions of node, regions are defined by j and s

$$\begin{cases} R_1(j,s) = \{x | x_j < s\}; & \text{for a given predictor } X_j \text{ and cutpoint } s \\ R_2(j,s) = \{x | x_j \geq s\}; \end{cases}$$

The above $(\star\star)$ sums over all obs in each of the 2 regions.

$\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are the predictions (often the averages of response var) of the obs in their region.

↓BACK↓

we search over all values of j and s, and b/c decision trees are greedy algorithms, we make the split corresponding to the lowest value of RSS.

Step #2: Apply cost complexity pruning to the above large tree (call it T_0), to obtain a sequence of best trees as a function of α:

$$\{T_0, T_{\alpha_1}, T_{\alpha_2}, \ldots T_{\alpha_Q}\}.$$

we do this by..... We create an increasing ordered sequence of values for α as:

$$\alpha = \{0, \alpha_1, \alpha_2, \ldots \alpha_Q\} \text{ with } 0 < \alpha_1 < \alpha_2 \ldots < \alpha_Q.$$

$\alpha = 0 \Rightarrow$ large tree

For each α (non-negative value of our tuning parm), we compute

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

for each α, there corresponds a subtree $T \subset T_0$ that minimizes RSS_α. We select as T_{α_i} the subtree that minimizes RSS for given val of α

where $\hat{y}_{R_m}$ = prediction in m^{th} region/rectangle

m = indexes the nodes/rectangles.

$|T|$ = number of nodes/rectangles in tree

This looks a bit like LASSO, where α is penalty tuning parm to control complexity of algorithm.

[redacted]

Step #3: Use K-fold cross validation to choose α. Namely, divide obs into K folds, for each $k = 1, 2, \ldots K$, leave fold k out, train on all folds except fold k, evaluate SSE on left-out k^{th} fold, and average over all K vals of SSE to produce a MSE value for each value of α. Select the value of α that minimizes MSE.

Step #4: Return/select the subtree from Step 2 that corresponds to the value of α selected above in Step #3.