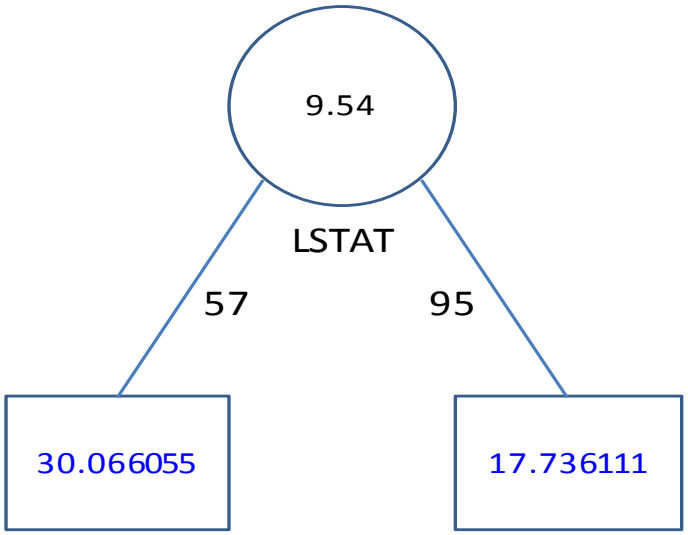
Trees as Indicator Regressions

Prof. Thomas B. Fomby

Southern Methodist University

3-27-14

As it turns out, **regression trees** are nothing more than regression models based on binary indicator variables that turn on and off depending upon whether certain variable "split" conditions are satisfied. Indicator variables and their splits are then determined in such a way to best describe (using mean values) the variation in the target variable. Correspondingly, the binary recursive method divides the input space into rectangular regions, each rectangular region having as its predicted response the mean of the y-points in that region. For example, consider the following regression tree for the "medv" target variable in the Boston Housing Data with the first decision node in place:



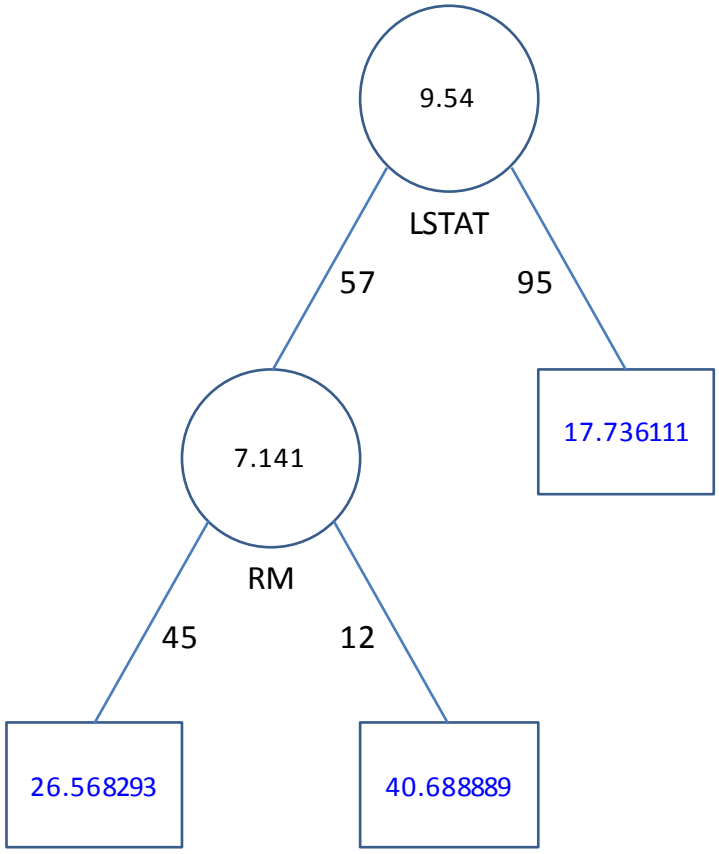
This tree can be written as the following regression model on a binary indicator function:

$$medv = \alpha + \beta I(LSTAT > 9.54) + \varepsilon$$

where $\hat{\alpha} = 30.066055$, $\hat{\alpha} + \hat{\beta} = 17.736111$, and I(.) is an indicator function that takes the value of 1 if the argument is true and 0 otherwise. In region 1 of the LSTAT line, $(LSTAT \leq 9.54)$, 57 y-observations are averaged together to get the mean (fitted) response of 30.066055 for that region while in region 2 of the LSTAT line $(LSTAT > 9.54)$, 95 y-observations are averaged together to get the mean (fitted) response of 17.736111. The choice to split on the variable LSTAT and to do so at the point of 9.54 was made only after trying out all possible splits of all of the input variables being considered (the

input space). Comparing all input variables and splits thereof, it is the case that the best split was at 9.54 on LSTAT because this variable and split gave rise to a tree that produced the largest reduction in the sum-of-squared errors of the fit of the target variable in the training data set. Although this variable split is retained in the next recursion, the search is again exhaustive in the sense that a new split (decision node) is chosen by considering all possible splits of all of the input variables and choosing the second split (node) so as to reduce the sum-of-squared errors fit of the target variable in the training data set.

A second decision node can be implemented in the binary recursion of the Boston Housing data and is of the form



This tree can be written as the following regression model on binary indicator functions:

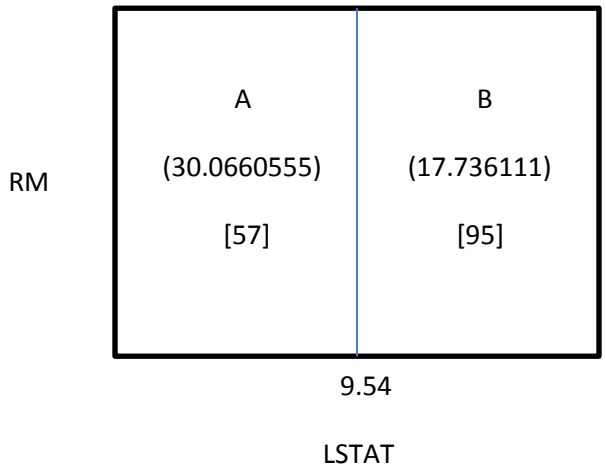
$$medv = \alpha + \beta I(LSTAT > 9.54) + \gamma I(LSTAT \leq 9.54) \cdot I(RM \leq 7.141) +$$

$$\theta I(LSTAT \leq 9.54) \cdot I(RM > 7.141) + \varepsilon$$

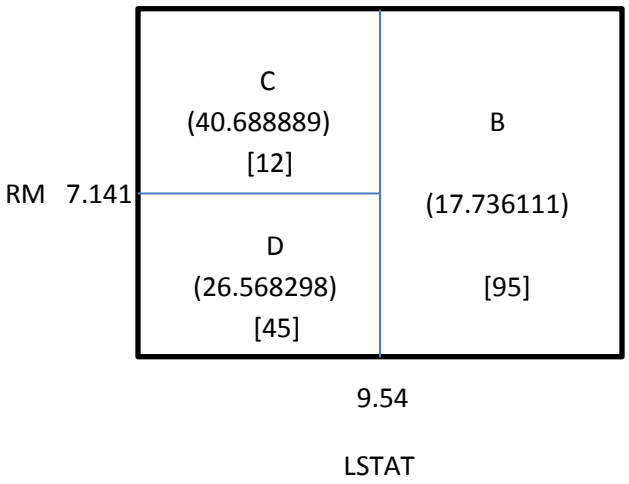
Then the estimated parameters of this regression model satisfy the relationships

$$\hat{\alpha} + \hat{\beta} = 17.736111 \text{ ; } \hat{\alpha} + \hat{\gamma} = 26.568293 \text{ ; and } \hat{\alpha} + \hat{\theta} = 40.688889$$

Notice that the threshold value for the first decision node is still the same in this second tree and the predicted value when LSTAT > 9.54 still remains at 17.736111. However, in a two dimensional representation of the input space we have gone from the first partition based on **one decision node**

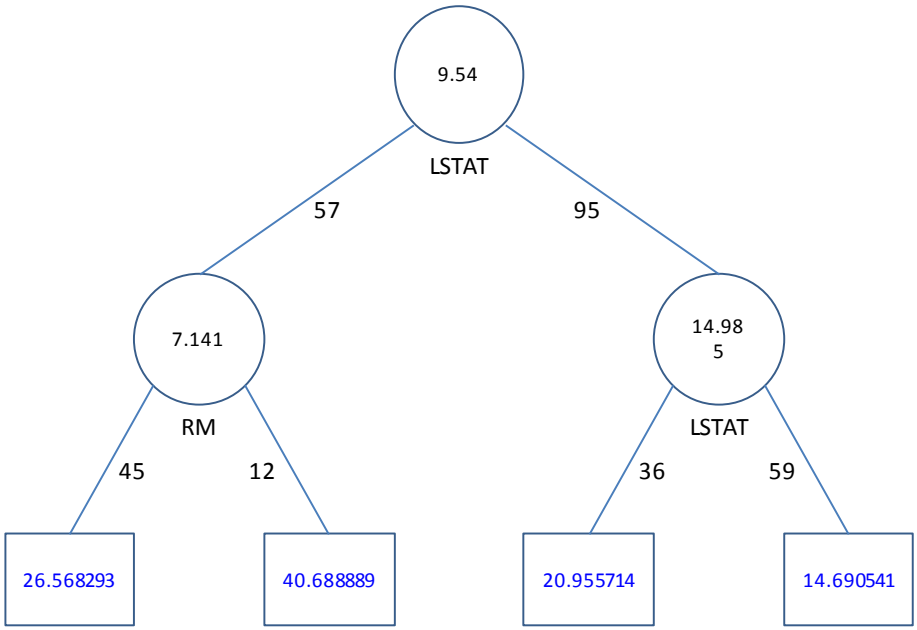


to the second partition based on **two decision nodes**:

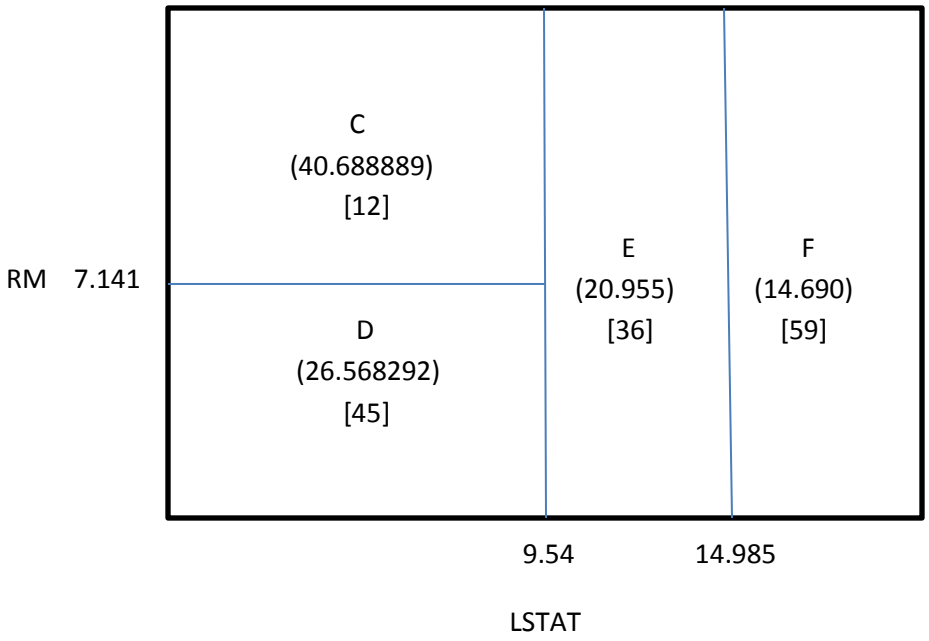


The numbers in the parentheses represent the mean levels of y associated with each region in the input space (the numbers in the square brackets representing the number of cases in the region on which the region's mean is based). Of course, in decision trees using more than 2 input variables the partitions of the input space are rectangular but multidimensional. The optimal split for the second decision node turned out to be using the variable "RM" and split at the point 7.141.

If one more decision node is added to the decision tree we get the following tree:



The partition of the input space can still be represented in two dimensions and it is



Notice, that once split points have been chosen, they do not change, hence the nature of the binary recursive procedure. Binary Recursion builds **unique** trees in the growing process, and, in reverse, the trees can be pruned in a unique order.

Going back to our original discussion of the regression on binary indicators representation of regression trees, one can see that the choice of split points have to be determined endogenously. That is, the computer must try many different split points with all of the input variables and pick the one that reduces the sum-of-squared errors (SSE) as much as possible. Of course, the statistical significance of the reduction of the sum-of-squared errors can be judged by a t-statistic (assuming the errors of the model are independent and homoscedastic) or, equivalently (asymptotically) using a chi-square or F-statistic. If the test statistics have a small enough p-value, say less than 0.05, an additional decision node can be added to the decision tree. On the other hand, if the reduction in the sum-of-squared errors in not statistically significant, the recursive building of the tree ends. Essentially, the "optimal size" tree is built using a stopping rule. This is the essence of the **CHAID method** for building regression trees. All of the building of the tree is done on the training data set. The CHAID method is, in essence, a with-in sample model building procedure.

Alternatively, the binary recursion method can be used to build a "full" tree and the full tree can be "pruned" on the second partition, the validation data set. That is, the full tree, having been uniquely built, it can be uniquely disassembled, one limb at a time. Of course, this gives rise to a set of sequential trees, each having one less limb that its previous (fuller) predecessor. The **CART method** is to score this sequence of pruned trees on the Validation data set and then chooses the tree that has the greatest accuracy measure in the independent Validation data set. In the case of a continuous target variable this is often the RMSE of the Validation errors of the tree.

If the target variable is a categorical variable as in "success" and "failure," then one can use other impurity measures (goodness of fit measures) to gauge which input variables and their split points to use in building a **classification tree** (as compared to the above **regression tree**). One wants to reduce the impurity measures as much as possible at each determination of a decision node for a classification tree. The most popular impurity measures are the following:

- c = number of classes, $p(i|t)$ = probability of i-th class given tree t, and $0log_2(0) = 0$ in the entropy measure. In the binary case, c = 2.
- $Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)log_2\, p(i|t)$
- $Gini(t) = 1 - \sum_{i=0}^{c-1}[p(i|t)]^2$
- $Classification\ error(t) = 1 - max_i[p(i|t)]$

The $Entropy(t)$ measure satisfies the interval restriction, $0 \leq Entropy(t) \leq log_2(c)$. If all of the cases in a terminal node belong to the same class, $Entropy(t) = 0$. If all of the cases in a terminal node are equally distributed to the classes, $Entropy(t) = log_2(c)$. In terms of goodness-of-fit in a terminal node, the closer the entropy of outcomes is to zero, the better the fit of the model. Of course, with many terminal nodes, the over-all entropy is usually calculated as the weighted average of the entropy measures across all of the terminal nodes with the weights being $w_j = n_j/N$ where n_j is the number of cases in the j-th terminal node, and N being the total number of cases in the training data set (or in the case of the CART method, the number of cases in the validation data set). Then the variable and its split that best reduces the overall (weighted) entropy of the final nodes of the tree defines the next decision node of the tree.

Another goodness-of-fit measure is the weighted Gini of the terminal nodes of a tree. The Gini coefficient for a terminal node satisfies the interval restriction, $0 \leq Gini(t) \leq 1$. If all of the cases in a terminal node belong to the same class, $Gini(t) = 0$. If all of the cases are equally distributed to the classes, $Gini(t) = 1$. In terms of goodness-of-fit in a terminal node, the closer the Gini coefficient is to zero, the better the fit of the model. In a similar manner to the entropy case, the goodness-of-fit of a tree is measured by the weighted average of the Gini measures of the terminal nodes of the tree. Then the variable and its split that best reduces the overall (weighted) Gini coefficient of the final nodes of the tree defines the next decision node of the tree.

Finally, another goodness-of-fit measure in classification problems used in building decision trees is the total classification error based on the final nodes of the tree. The smaller the total classification error is, the better the classification performance of the tree. Errors of classification, of course, involve incorrectly classifying a positive (y = 1) as a negative ($\hat{y} = 0$), a Type I Error, or incorrectly classifying a negative (y = 0) as a positive ($\hat{y} = 1$), a Type II Error.

As in the regression tree approach, the **CHAID** method focuses on the statistical significance of an additional node as judged by a p-value of an appropriate statistic. If the p-value falls below a certain pre-set significance level (for, example, p = 0.05) the decision node will be added to the tree. If not, then the building of the decision tree is stopped. In the case of the **CART** validation data set approach, a full tree is built using the training data set, and then a pruned tree is chosen to minimize one of the three overall goodness-of-fit measures described above.