An overview for regression tree

PhD (C.) Adem Meta
University "Ismail Qemali" Vlore, Albania

Abstract

Classification and regression tree is a non-parametric methodology. CART is a methodology that divides populations into meaningful subgroups which will allow the identification of groups of interest. CART is classification method which uses a large data to construct decision trees. Depending on available information about the dataset, classification tree or regression tree can be constructed. The first part of the paper describes fundamental principles of tree construction, different splitting algorithms, and pruning procedures. Second part of the paper answers the questions why should we use or should not use the CART method. Advantages and weaknesses of the method are discussed and tested in detail. In the last part, CART is applied to real data, using the statistical software R. On this paper some graphical and plotting tools are presented. The Regression Tree is a classification tree with a continuous dependent variable in which independent variables receive continuous values or discrete values with an error prediction that is computed with squares of differences between the observed and predicted values. At the beginning of this paper, some basic principles are presented in the construction of a classification / regression tree, while after that a detailed description of the distribution is made, making the theoretical generalization, and giving in detail the algorithms that are used in distribution as well as using a concrete examples, where different algorithms apply to distribute to a database. At the end are detailed analyses for regression trees by making appropriate theoretical generalizations and providing detailed information on how to use different algorithms to prune the overcrowded tree to reach to the final tree.

Keywords: regression tree, overview, classification.

Introduction

The main idea behind tree methods is to recursively partition the data into smaller and smaller strata in order to improve the fit as best as possible. They partition the sample space into a set of rectangles and fit a model in each one. The sample space is originally split into two regions. The optimal split is found over all variables at all possible split points. For each of the two regions created this process is repeated again. The major components of the CART methodology are the selection and stopping rules. The selection rule determines which stratification to perform at every stage and the stopping rule determines the final strata that are formed. Once the strata have been created the impurity of each stratum is measured. The heterogeneity of the outcome categories within a stratum is referred to as "node impurity". Classification trees are employed when the outcome is categorical and regression trees are employed when the outcome is continuous. Classification trees can take most forms of categorical variables including indicator, ordinal and non-ordinal variables and are not limited to the analysis of categorical outcomes with two categories. There are three commonly used measures for node impurity in classification trees. They are misclassification

error, Gini index and cross-entropy or deviance. While the three of them are similar, the latter two are differentiable and easier to optimize numerically. Additionally, the Gini index and cross-entropy are more sensitive to changes in the node probabilities. As a result, they are favorites in computer algorithms designed for classification trees and used more often. The measure of node impurity in regression trees is least squares. CARTs are not as popular compared to traditional statistical methods because of the lack of tests to evaluate the goodness of fit of the tree produced and the relatively short span that they have been around. They are typically model free in their implementation. The main idea of a classification tree is a statistician's version of the popular twenty questions game. Several questions are asked with the aim of answering a particular research question at hand. However, they are advantageous because of their non-parametric and non-linear nature. They do not make any distribution assumptions and treat the data generation process as unknown and do not require a functional form for the predictors. They also do not assume additivity of the predictors which allows them to identify complex interactions. Tree methods are probably one of the most easily interpreted statistical techniques. They can be followed with little or no understanding of Statistics and to a certain extent follow the decision process that humans use to make decisions. In this regard, they are conceptually simple yet present a powerful analysis. A common goal of many clinical research studies is the development of a reliable clinical decision rule, which can be used to classify new patients into clinically-important categories. Examples of such clinical decision rules include triage rules, whether used in the out-of-hospital setting or in the emergency department, and rules used to classify patients into various risk categories so that appropriate decisions can be made regarding treatment or hospitalization.

Traditional statistical methods are hard to use, or of limited utility, in addressing classification problems. There is a number of reasons for these difficulties. First, there are generally many possible "predictor" variables which make the task of variable selection difficult. Traditional statistical methods are poorly suited for this sort of multiple comparisons. Second, the predictor variables are rarely nicely distributed. Many clinical variables are not normally distributed and different groups of patients may have markedly different degrees of variation or variance. Third, complex interactions or patterns may exist in the data. For example, the value of one variable (family history) may substantially affect the importance of another variable. These types of interactions are generally difficult to model and virtually impossible to model when the number of interactions and variables becomes substantial. Fourth, the results of traditional methods may be difficult to use.

Regardless of the statistical methodology being used, the creation of a clinical decision rule requires a relatively large data set. For each patient in the dataset, one variable (the dependent variable), records whether or not that patient had the condition which we hope to predict accurately in future patients. Examples might include significant coronary heart disease related to the age, obesity or family history, another possible predictor would be whether or not the patient has a history of similar problems in the past. In many clinically-important settings, the number of possible predictor variables is quite large. Within the last 10 years, there has been increasing interest

in the use of classification and regression tree (CART) analysis. CART analysis is a tree-building technique which is unlike traditional data analysis methods. It is ideally suited to the generation of clinical decision rules. Because CART analysis is unlike other analysis methods it has been accepted relatively slowly. Other factors which limit CART's general acceptability are the complexity of the analysis and, until recently, the software required to perform CART analysis, but it was difficult to use.
It is now possible to perform a CART analysis without a deep understanding of each of the multiple steps being completed by the software. It is found that CART to be quite effective for creating clinical decision rules which perform as well or better than rules developed using more traditional methods. CART is often able to uncover complex interactions between predictors which may be difficult or impossible to uncover using traditional multivariate techniques. The purpose of this paper is to provide an overview of CART methodology, emphasizing practical use rather than the underlying statistical theory.
CART is a robust data-mining and data-analysis tool that automatically searches for important patterns and relationships and quickly uncovers hidden structure even in highly complex data. This discovered knowledge is then used to generate accurate and reliable predictive models for applications such as profiling customers, targeting direct mailings, detecting telecommunications and credit-card fraud, and managing credit risk.
Trees explain variation of a single response variable by repeatedly splitting the data into more homogeneous groups, using combinations of explanatory variables that may be categorical and/or numeric. Each group is characterized by a typical value of the response variable, the number of observations in the group, and the values of the explanatory variables that define it. CART uses an intuitive Windows-based interface, making it accessible to both technical and nontechnical users. Underlying the “easy” interface, however, is a mature theoretical foundation that distinguishes CART from other methodologies and other decision-tree tools.

2. How to split?

One of the most importance concepts of the CART technics is the Splitting procedure. Before explaining the splitting procedure, we have to give some information and definitions for the nodes and leaves. Node is a connection point, either a redistribution point or an end point for data transmissions. The node has programmed or engineered capability to recognize and process or forward transmissions to other nodes. A tree can be defined recursively as a collection of nodes (starting at a root node), where each node is a data structure consisting of a value, together with a list of nodes (the "children"), with the constraints that no node is duplicated. A tree can be defined abstractly as a whole (globally) as an order tree, with a value assigned to each node. Both these perspectives are useful: while a tree can be analyzed mathematically as a whole, when actually represented as a data structure it is usually represented and worked with separately by node. For example, looking at a tree as a whole, one can talk about "the parent node" of a given node, but in general as a data structure a given node only contains the list of its children, but does not contain a reference to

its parent.
All nodes can be accessed through the tree. Their contents can be modified or deleted, and new elements can be created. The node tree shows the set of nodes, and the connections between them. The tree starts at the root node and branches out to the text nodes at the lowest level of the tree. The nodes in the node tree have a hierarchical relationship to each other. The terms parent, child, and sibling are used to describe the relationships. Parent nodes have children. Children on the same level are called siblings or brothers or sisters.
In a node tree, the top node is called the root. Every node, except the root, has exactly one parent node. A node can have any number of children. A leaf is a node with no children.
Siblings are nodes with the same parent

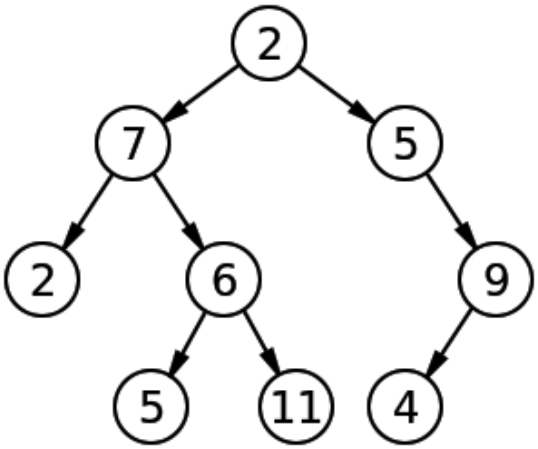

Figure 2. Simple tree.
In this diagram, the node labeled 7 has two children, labeled 2 and 6, and one parent, labeled 2. The root node, at the top, has no parent. To split a node into two child nodes, CART always asks questions that have a "yes" or "no" answer. For example, the question "how is related the coronary heart diseases with family history?"

3. Splitting rule and goodness-of-fit

Construction of a tree involves making choices about three major issues. The first choice is how splits are to be made: which explanatory variables will be used and where the split will be imposed. These are defined by splitting rules. The second choice involves determining appropriate tree size, generally using a pruning process. The third choice is to determine how application-specific costs should be incorporated. This might involve decisions about assigning varying misclassification costs and /or accounting for the cost of model complexity.
Binary recursive partitioning, as described above, applies to the fitting of both classification and regression trees. However, the criteria for minimizing node impurity or maximizing homogeneity, which are different for the two methods.

3.1 Last square deviation

The Deviation of Last Square Distance (LSD) is used as a measure of impurity of a node when the variable is the answer constant and is calculated as::

$$R(t) = \frac{1}{N_w(t)} \sum w_i f_i (y_i - \bar{\bar{y}}(t))^2$$

Where $N_w(t)$ is the number of weight in each case at a given node, which is the weight value of a variable in a case, is the value of a variable with different densities, y is the value of the corresponding variable, and y (t) is the average weight per kilogram.

$$i(p) = \sum_{i \neq j} p_i p_j = 1 - \sum_j p_j^2$$

In this case we have chosen the division where the Gini Index decreases (increases the purity). After this repetition process that generates new divisions from the old divisions that we already have. In this way to do this we need to repeat the same steps when we split the first node. So we need spreadsheets (distribution paper) for each new partition process! This is very difficult to do by hand when we go further and create more new joints.

This is much easier to do using computer software (such as R sofware).

The splitting process takes only one variable at a time and the result of this is the separation of two variables and so on. And so it will look that for (house market) example in the initial tree or Tmax will usually be difficult to read, because it is overloaded by the data. The choice is that we need to compare the initial tree to get a new tree which has a smaller number and is easier to read, and more importantly presents the much-talented data. Gini's testimony is a measure that often randomly selects an element from the group that will be labeled incorrectly if it is randomly labeled in accordance with the distribution of labels in this community. It can be calculated by collecting all the probabilities of each node that is selected and multiplied by the probability errors of these nodes. It reaches its minimum (zero) when all of these nodes tend to a single target node. To calculate the impurity for a given set of values, {1, 2, ..., m}, if p_i = a portion of the value-tagged joints in a set.

$$I(p) = \sum_{i=1}^{m} p_i (1 - p_i) = \sum_{i=1}^{m} (p_i - p_i^2) = \sum_{i=1}^{m} p_i - \sum_{i=1}^{m} p_i^2 = 1 - \sum_{i=1}^{m} p_i^2$$

Gini's purity of a node is: p (1-p)

4. Regression tree

All regression techniques contain a single output (response) variable and one or more input (predictor) variables. The output variable is numerical. The general regression tree building methodology allows input variables to be a mixture of continuous and categorical variables. A decision tree is generated when each decision node in the tree contains a test

on some input variable's value. The terminal nodes of the tree contain the predicted output variable values. A Regression tree may be considered as a variant of decision trees, designed to approximate real-valued functions, instead of being used for classification methods. A regression tree is built through a process known as binary recursive partitioning, which is an iterative process that splits the data into partitions or branches, and then continues splitting each partition into smaller groups as the method moves up each branch. The algorithm then begins allocating the data into the first two partitions or branches, using every possible binary split on every field. The algorithm selects the split that minimizes the sum of the squared deviations from the mean in the two separate partitions. This splitting rule is then applied to each of the new branches. This process continues until each node reaches a user-specified minimum node size and becomes a terminal node. (If the sum of squared deviations from the mean in a node is zero, then that node is considered a terminal node even if it has not reached the minimum size) .

5. Cart advantages

Deals effectively with any data type. CART is easy to understand once you get the ultimate tree. Conditional information can be used effectively in CART. There is no need to transform the data when using CART. CART is robust to outliers. Misclassification rate will be provided in CART.

6. Cart disadvantages

CART doesn't help when use combinations of variables. Tree can be deceptive since a variable may not be included if masked by another. Tree structures may be unstable – a change in the sample may give different trees. Tree is optimal at each split, but it may not be globally optimal. Very complicated to read when variables consist of many categories. There are limited numbers of software programs that can be used for classification and regression trees analysis.

7. Regression tree models

Regression trees are nonlinear predictive models which can incorporate more than one type of variable (continuous, ordered discrete, or categorical). The tree model has two parts: the recursive partition and a simple model for each cell of the partition. Each terminal node (leaf) of the tree represents a partition cell, and each cell has a simple model which only applies to that cell. To find which cell a unit belongs to, start at the root node and ask a sequence of questions. Each interior node is a "yes/no" question and each branch between nodes is the answer to the question. If some data is missing, following the tree may not lead to a leaf, but a prediction can be made by averaging all of the leaves in the "sub-tree" reached. The Boston data frame has 506 observations and 14 variables (described below).

8. Example of regression trees for Boston housing prices

Goal: Predict the price of a house in Boston (dependent variable) with a regression tree

Follow the directions below by entering the red commands into R.

1. Install and load the MASS library and the rpart libraries. Note: rpart uses a measure of statistical inequality called the Gini coefficient.

2. The dataset Boston is included in the rpart package, so there is no need to import data. Look at the variable names in Boston.

3.

```
[1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
[8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

3. Fit a regression tree model to the Boston data and get a summary of the model. In the rpart formula, "anova" indicates a regression tree.

The results below are the complexity table for the tree produced.

Complexity Table

	CP	nsplit	rel error	xerror	xstd
1	0.4527442007	0	1.0000000	1.0008441	0.08279329
2	0.1711724363	1	0.5472558	0.6415611	0.05826556
3	0.0716578409	2	0.3760834	0.4282302	0.04558908
4	0.0361642808	3	0.3044255	0.3524707	0.04251598
5	0.0333692301	4	0.2682612	0.3352451	0.04269041
6	0.0266129999	5	0.2348920	0.3201398	0.04284871
7	0.0158511574	6	0.2082790	0.2833742	0.04035666
8	0.0082454484	7	0.1924279	0.2716571	0.04230165
9	0.0072653855	8	0.1841824	0.2594427	0.03907483
10	0.0069310873	9	0.1769170	0.2556601	0.03828815
11	0.0061263349	10	0.1699859	0.2541351	0.03861434
12	0.0048053197	11	0.1638596	0.2392679	0.03424519
13	0.0045609248	12	0.1590543	0.2332381	0.03402908
14	0.0039410233	13	0.1544934	0.2264541	0.03388449
15	0.0033161209	14	0.1505523	0.2285377	0.03485257
16	0.0031206492	15	0.1472362	0.2311663	0.03650942
17	0.0022459421	16	0.1441156	0.2284981	0.03631381
18	0.0022354038	18	0.1396237	0.2279692	0.03615274
19	0.0021720940	19	0.1373883	0.2279692	0.03615274
20	0.0019335691	20	0.1352162	0.2300202	0.03624752
21	0.0017169079	21	0.1332826	0.2295729	0.03616074
22	0.0014440359	22	0.1315657	0.2281099	0.03614753
23	0.0014098099	23	0.1301217	0.2291503	0.03615862
24	0.0013635419	24	0.1287119	0.2282475	0.03615277
25	0.0012778421	25	0.1273483	0.2279400	0.03613419
26	0.0012473645	26	0.1260705	0.2274864	0.03614715
27	0.0011372540	28	0.1235757	0.2279233	0.03615827
28	0.0009633247	29	0.1224385	0.2272594	0.03615926
29	0.0008486865	30	0.1214752	0.2269299	0.03612045
30	0.0007098930	31	0.1206265	0.2260585	0.03614413
31	0.0005879326	32	0.1199166	0.2274213	0.03612924
32	0.0005115280	33	0.1193287	0.2299455	0.03615310
33	0.0003794039	34	0.1188171	0.2306533	0.03615437
34	0.0003719398	35	0.1184377	0.2303838	0.03613785
35	0.0003438561	36	0.1180658	0.2302506	0.03613847
36	0.0003311450	37	0.1177219	0.2302972	0.03613873
37	0.0002274363	39	0.1170596	0.2300330	0.03614484
38	0.0001955788	40	0.1168322	0.2301039	0.03614407
39	0.0001000000	41	0.1166366	0.2304656	0.03615669

From the output below, we can see that at the root level (Node number 1), before any splits, there are 506 observations. Also, the Mean Square Error

is 84.42 and the overall data mean is 22.53.
The largest tree with 39 terminal nodes has the smallest cross-validated error rate. However, this tree is too large to make predictions. The first primary split is based on the average number of rooms. If a house has less than 7 rooms, the observation goes left, otherwise it goes to the right side of the tree. The second primary split is the lower status of the population. If the number of rooms is unknown, the lower population status could be used with splitting value 9.725.

9. Pruning

Returning to the complexity table above, the lowest error rate is a tree with 27 nodes, but because the tree with 12 terminal nodes is within one standard error of the minimum, the smaller tree with 12 terminal nodes is sufficient. Pruning the tree can be done by selecting a complexity value that is greater than that produced for the optimal tree (tree 12) but less than the complexity value for the tree above it (tree 11). Here, we need a tree with complexity parameter within 0.0048 and 0.0061.
Prune the tree with the code below:
boston.prune=prune(boston.rp,cp=0.005)
plot(boston.prune,main= main="Pruned Model")
text(boston.prune)
Figure 3 shows the primary split at the number of rooms per house (rm < 6.941). The second split on the left lstat seems important in terms of the models ability to partition the data to reduce the residual sums of squares. The expensive houses tend to have a higher average number of rooms. Cheaper houses have less rooms (< 7 on average) and a low status in the population with a high crime rate.

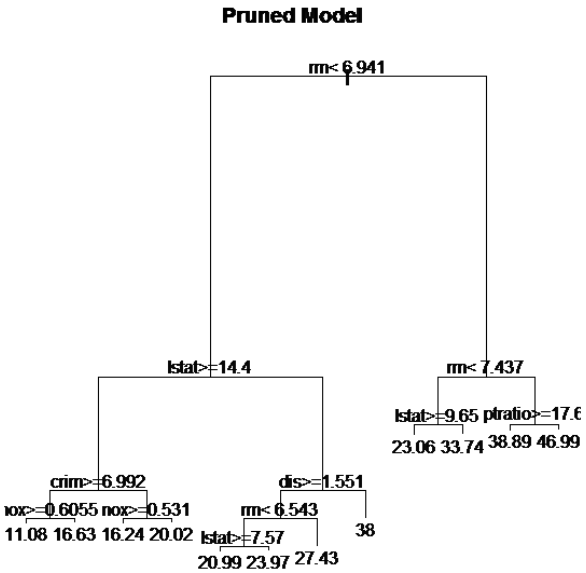

Figure 3: Tree A -- Pruning tree based on the one SE rule

9.1 Interactive Pruning

A complexity plot can aid in determining the size of the pruned tree, here, in relation to the number of terminal nodes.

plotcp(boston.rp, minline=TRUE, lty=3, col=1, upper="size")

Figure 4 shows that cross-validation suggests an optimal tree of size with between seven and fourteen terminal nodes. I chose a tree with nine terminal nodes, so we can fit this model.

A tree can be interactively pruned in several ways. The code below prunes the tree to have only 9 terminal nodes and plots the tree.

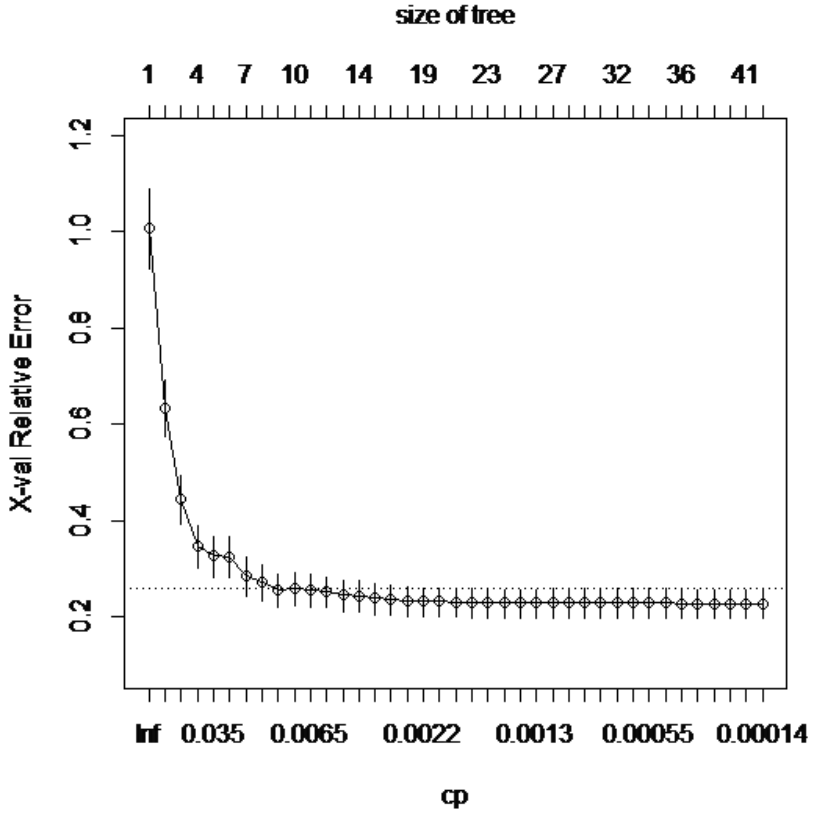

Figure 4: Complexity plot for pruning cross-validation

boston.prune.int=snip.rpart(boston.prune,toss=c(8,9,20))
plot(boston.prune.int,uniform=T,branch=0.1,main= "Interactive Pruning")
text(boston.prune.int,pretty=1,use.n=T)

Notice that the interactive pruned tree below uses the variables rm, lstat, crim, dis, and ptratio for determining the splits.

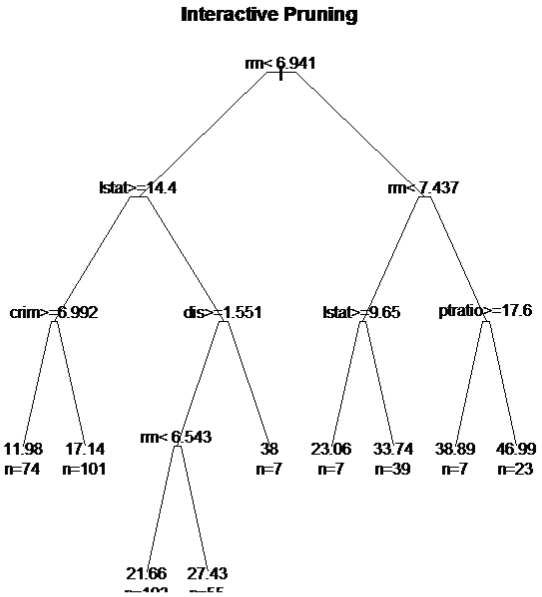

Figure 5: Tree B – Results of Interactive Pruning meanvar(boston.prune.int)

10. Predictions

Examine the predictions from both tree models using the predict function.
Model 1: for Tree A

boston.pred1=predict(boston.prune)
Model 2: for Tree B

boston.pred2=predict(boston.prune.int)
Compute the correlation matrix of predictions with the actual response.
boston.mat.pred=cbind(Boston$medv,boston.pred1,boston.pred2)
boston.mat.pred=data.frame(boston.mat.pred)
names(boston.mat.pred)=c("medv","pred.m1","pred.m2")
cor(boston.mat.pred)

	medv	pred.m1	pred.m2
medv	1.0000000	0.9144071	0.9032262
pred.m1	0.9144071	1.0000000	0.9877725
pred.m2	0.9032262	0.9877725	1.0000000

The correlation matrix above indicates that the predictions between models 1 and 2 are highly correlated with the response. Model 1 predictions are a little better than

model 2 predictions.
The predictions can be plotted using the code below:

```
par(mfrow=c(1,2),pty="s")
with(boston.mat.pred, {
eqscplot(pred.m1, medv, xlim=range(pred.m1,pred.m2),ylab="Observed",
xlab="Predicted", main="Model 1")
abline(0,1,col="blue",lty=5)
eqscplot(pred.m2,medv,xlim=range(pred.m1,pred.m2),ylab= "Observed",
xlab="Predicted", main="Model 2")
abline(0,1,col="blue",lty=5)
```

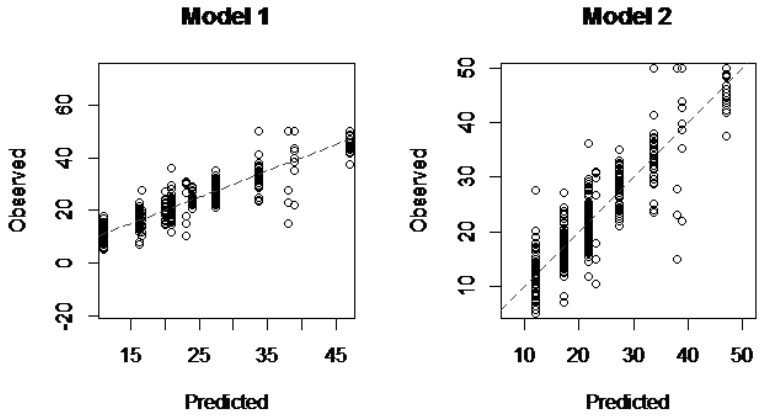

Figure 6: Observed vs. predicted values for model 1 and model 2
The plots in Figure 6 indicate that both models are pretty good for predicting the median value of house prices in Boston. Notice that model 1 is slightly better than model 2.
What if the dataset is missing the rm variable, but we still want to predict the house price? We can create a regression tree with the average number of rooms (rm) variable removed.

```
boston.rp.omitRM=update(boston.rp,~.-rm)
summary(boston.rp.omitRM)
```

…

```
            CP nsplit rel error    xerror       xstd
1  0.4423649998      0 1.0000000 1.0040176 0.08319221
2  0.1528339955      1 0.5576350 0.6009111 0.04900741
3  0.0627501370      2 0.4048010 0.4382420 0.04057083
4  0.0400760532      3 0.3420509 0.4188052 0.04137352
```

Examine the first node.

```
Node number 1: 506 observations, complexity param=0.442365
mean=22.53281, MSE=84.41956
left son=2 (294 obs) right son=3 (212 obs)
Primary splits:
```

lstat < 9.725 to the right, improve=0.4423650, (0 missing)
indus < 6.66 to the right, improve=0.2594613, (0 missing)
ptratio < 19.9 to the right, improve=0.2443727, (0 missing)
nox < 0.6695 to the right, improve=0.2232456, (0 missing)
tax < 416.5 to the right, improve=0.2017517, (0 missing)
Surrogate splits:
indus < 7.625 to the right, agree=0.822, adj=0.575, (0 split)
nox < 0.519 to the right, agree=0.802, adj=0.528, (0 split)

The primary split is now on lstat and the surrogate splits are indus and nox. When rm is omitted, the new model uses the competing split of the original model to make the first split.

11.1 Examining Performance through a Test/Training Set

To get a realistic estimate of model performance, randomly divide the data into a training and test set, and then use the training set to fit the model and the test set to validate the model.

```
set.seed(1234)
n=nrow(Boston)
```

Sample 80% of the data and let that be the training set and the remaining 20% is the test set.

```
boston.samp=sample(n,round(n*.8))
bostonTrain=Boston[boston.samp,]
bostonTest=Boston[-boston.samp,]
```

Below is a function that will produce the MSE for the previous model (re-substitution rate).

```
testPred=function(fit, data=bostonTest){
        #MSE for performance of predictor on test data
        testVals=data[,"medv"]
        predVals=predict(fit,data[,])
        sqrt(sum((testVals - predVals)^2)/nrow(data))}
```

The MSE for the previous (pruned) model is 3.719.

```
testPred(boston.prune, Boston)
```

[1] 3.719268

We compute the MSE for the previous model, where the original Boston dataset was used to fit the model and validate it. The MSE estimate is 3.719268, which is a resubstituting error rate.

Fitting the model again to the training dataset and examining the complexity table reveals that the best model based on the one standard error rule is a tree with seven terminal nodes. The red line across the figure below represents the 1 SE rule.

```
bostonTrain.rp=rpart(medv~.,data=bostonTrain,method="anova",cp=0.0001)
plot(bostonTrain.rp)
abline(v=7,lty=2,col="red")
```

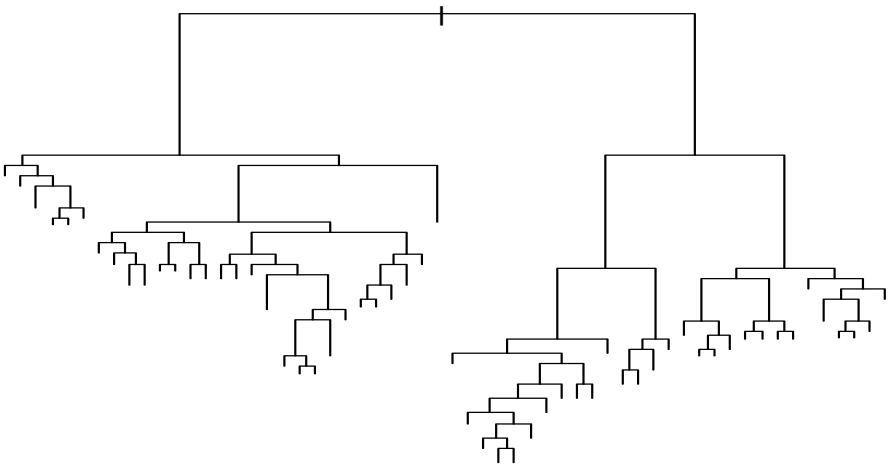

Figure 7: Regression tree with the 1-SE Rule
Now we can prune and plot the training tree.
bostonTrain.prune=prune(bostonTrain.rp, cp=0.01)
plot(bostonTrain.prune, main= "Boston Train Pruning Tree")
text(bostonTrain.prune)

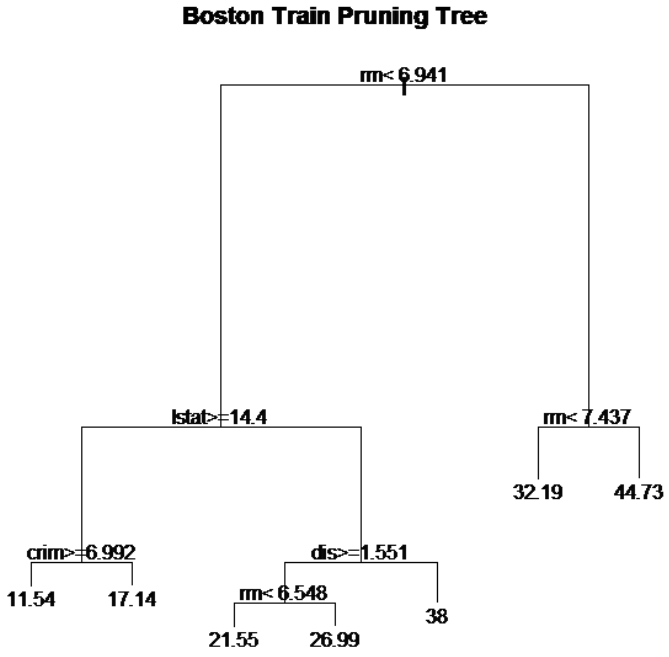

Figure 8: Pruned Regression Tree for the Boston Training set
The computed Mean Square Error rate for the training dataset is 4.06 and the value of the MSE for the test dataset is 4.78. Clearly, these MSE values are similar, but not

equal.

testPred(bostonTrain.prune, bostonTrain)

[1] 4.059407

testPred(bostonTrain.prune, bostonTest)

[1] 4.782395

The prediction performance of the models can be examined through plots of the observed vs. predicted values.

```
bostonTest.pred=predict(bostonTrain.prune, bostonTest)
with(bostonTest,{
cr=range(bostonTest.pred, medv)
eqscplot(bostonTest.pred, medv, xlim=cr, ylim=cr, ylab="Observed",
xlab="Predicted", main="Test Dataset")
abline(0,1,col="blue", lty=5)})
```

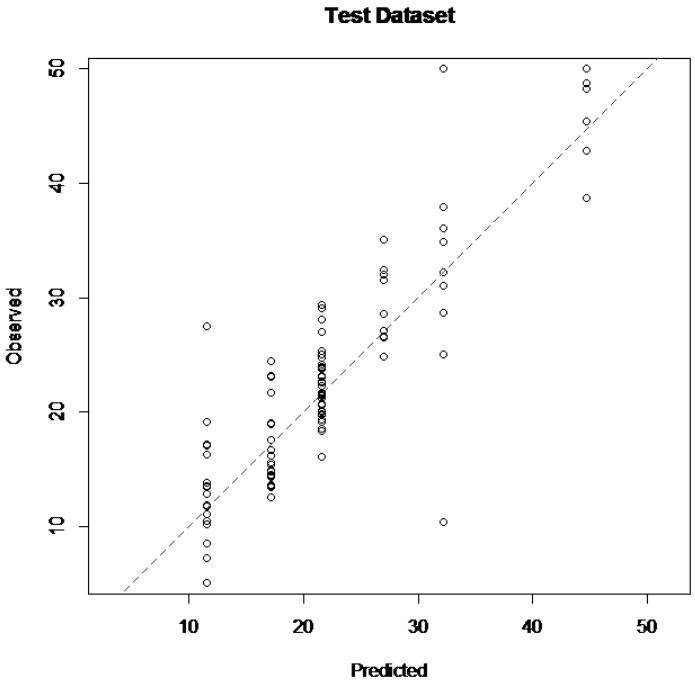

Figure 9: Scatterplot of Observed vs. Predicted prices

Figure 9 above shows that the prediction performance of the models seems to be a good indicator of the price of houses in Boston, since the data points lie close to the line.

References

Breiman, L. (1984). *Classification and regression trees*, Belmont, Calif.: Wadsworth International Group.

Finch, H and Schneider, M. (2007). 'Classification Accuracy of Neural Networks vs. Discriminant Analysis, Logistic Regression, and Classification and Regression Trees', *Methodology*, 3(2), pp.47-57.
Johnson, R & Wichern, D (1982). *Applied multivariate statistical analysis*, Englewood Cliffs, N.J.: Prentice-Hall.
Kuhnert, P & Venables, B (2005). *An Introduction to R: Software for Statistical Modelling & Computing*. Cleveland, Australia: CSIRO Australia.
Lee, S. (2006). 'On Classification and Regression Trees for Multiple Responses and Its Application', *Journal of Classification*, 23(1), pp.123-141.
Li, J. (2016). *Classification/Decision Trees*, viewed 10 July 2016, <http://sites.stat.psu.edu/~jiali/course/stat597e/notes2/trees.pdf>.
Loh, W. (2014). 'Fifty Years of Classification and Regression Trees', *International Statistical Review*, 82(3), pp.329-348.
Stine, R. (2012). *Lecture 8: Classification & Regression Trees*, Department of Statistics, Wharton School, viewed 10 July 2016 <http://www-stat.wharton.upenn.edu/~stine/mich/DM_07.pdf>.
Thernau, T, Atkinson, B & Ripley, B. (2015). *Package 'rpart'*, viewed 10 July 2016, <https://cran.r-project.org/web/packages/rpart/rpart.pdf>.
Yeh, C. (1991). 'Classification and regression trees (CART)', *Chemometrics and Intelligent Laboratory Systems*, 12(1), pp.95-96.