Boston Housing Random Forest and CART

jdt

1/28/2022

Contents

Theory **1**
- Classification and Regression Trees . . . 1
  - SAS - HPSPLIT Procedure . . . 3
- Measures of Model Fit . . . 4
  - Fit for Classification Trees . . . 4
  - Measures of Model Fit for Regression Trees . . . 7
- Random Forest . . . 7
  - SAS - HPFOREST Procedure . . . 7
  - Random Forest - Article . . . 8
  - Algorithm for RF . . . 9
  - Bagging the Data . . . 9

Information about the Boston Housing Data **10**

R **12**

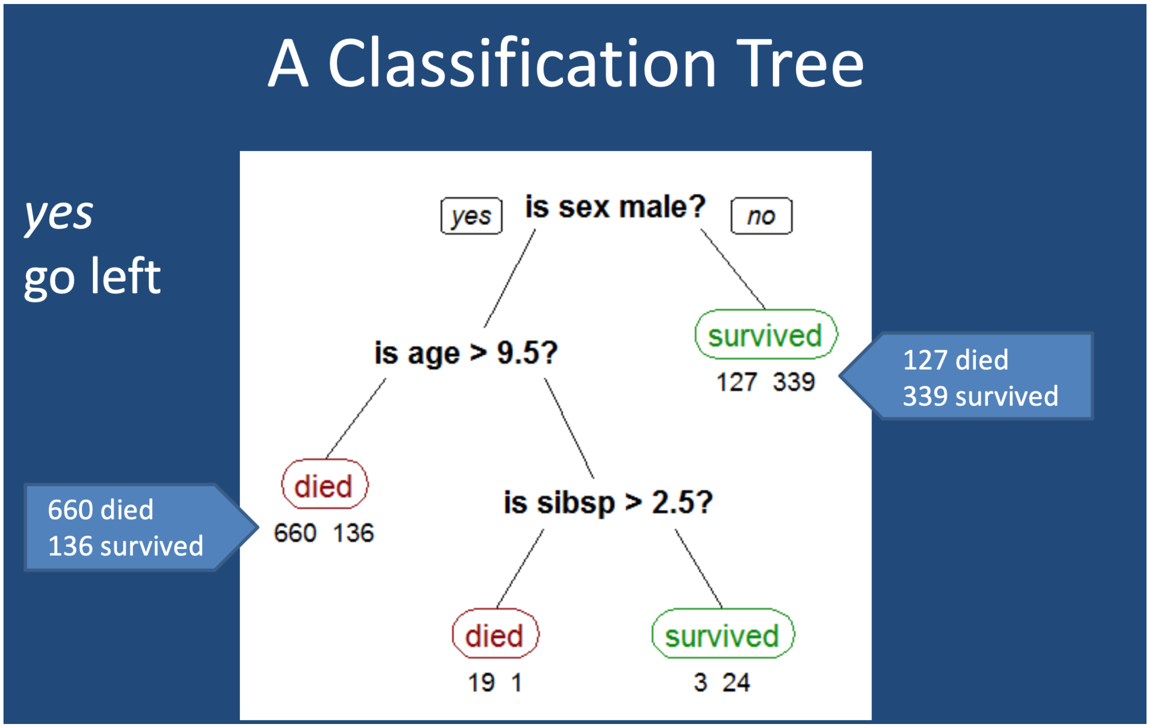
SAS **18**
- Code . . . 18
- Output . . . 20

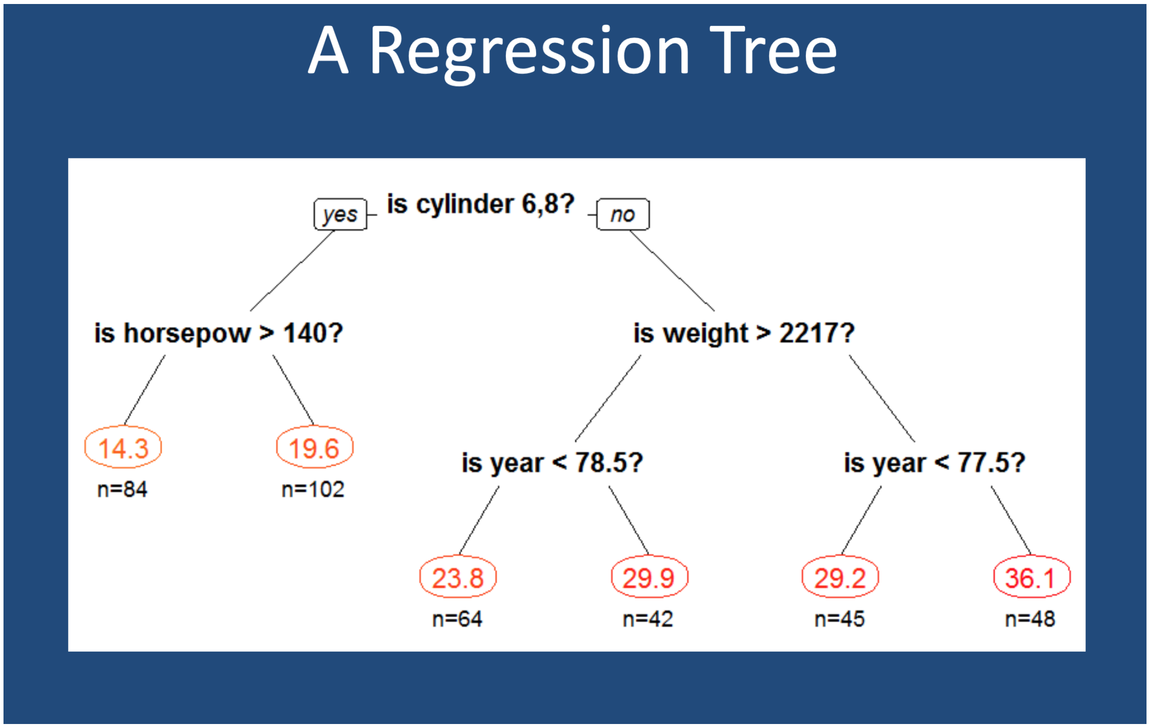
Theory

Classification and Regression Trees

In this section, I have presented some new methods that are being widely used in the era of "Big Data" and "Data Science". Engineers and computer scientists called this area machine learning or "deep learning" for use in problems associated with very large data sets. I have provided several files on the class BOX under a folder, entitled CART_stuff. These provide additional discussion/examples for various aspects of trees and random forest, including topics for bagging/boosting methods. As there are many details associated with these methods, I will demonstrate some of the methods with a commonly used data entitled the Boston Housing Data.

I have included two examples for CART; a classification tree (for use with a binary endpoint) and a regression tree (for use with a continuous endpoint).





SAS - HPSPLIT Procedure

The HPSPLIT procedure is a high-performance procedure that builds tree-based statistical models for classification and regression. The procedure produces classification trees, which model a categorical response, and regression trees, which model a continuous response. Both types of trees are referred to as decision trees because the model is expressed as a series of if-then statements.

The predictor variables for tree models can be categorical or continuous. The model is based on a partition of the predictor space into non-overlapping segments, which correspond to the terminal nodes or leaves of the tree. The partitioning is done recursively, starting with the root node, which contains all the data, and ending with the terminal nodes. At each step of the recursion, the parent node is split into child nodes through selection of a predictor variable and a split value that minimize the variability in the response across the child nodes.

Tree models are built from training data for which the response values are known, and these models are subsequently used to score (classify or predict) response values for new data. For classification trees, the most frequent response level of the training observations in a leaf is used to classify observations in that leaf. For regression trees, the average response of the training observations in a leaf is used to predict the response for observations in that leaf. The splitting rules that define the leaves provide the information that is needed to score new data.

The process of building a decision tree begins with growing a large, full tree. Various measures, such as the Gini index, entropy, and residual sum of squares, are used to assess candidate splits for each node. The full tree can overfit the training data, resulting in a model that does not adequately generalize to new data.

To prevent overfitting, the full tree is pruned back to a smaller subtree that balances the goals of fitting training data and predicting new data. Two commonly applied approaches for finding the best subtree are cost-complexity pruning (Breiman et al. 1984) and C4.5 pruning (Quinlan 1993). For more information, see the section Building a Decision Tree.

SAS/STAT software provides many different methods of regression and classification. Compared with other methods, an advantage of tree models is that they are easy to interpret and visualize, especially when the tree is small. Tree-based methods scale well to large data, and they offer various methods of handling missing values, including surrogate splits.

However, tree models have limitations. Regression tree models fit response surfaces that are constant over rectangular regions of the predictor space, and so they often lack the flexibility needed to capture smooth relationships between the predictor variables and the response. Another limitation of tree models is that small changes in the data can lead to very different splits, and this undermines the interpretability of the model (Hastie, Tibshirani, and Friedman 2009; Kuhn and Johnson 2013).

Measures of Model Fit

Various measures of model fit have been proposed in the data mining literature. The HPSPLIT procedure measures model fit based on a number of metrics for classification trees and regression trees.

If you specify a variable in the WEIGHT statement, then the weight of an observation is the value of the weight variable for that observation. If no WEIGHT statement is specified, then the weight of each observation is equal to one. In this case, the sum of weights of observations is equal to the number of observations.

Fit for Classification Trees

The HPSPLIT procedure measures model fit based on the following metrics for classification tree: entropy, Gini index, misclassification rate (Misc), residual sum of squares (RSS), average square error (ASE, also known as the Brier score), sensitivity, specificity, area under the curve (AUC), and confusion matrix.

Entropy for Classification Trees

Entropy for classifications tree is defined as

$$\text{Entropy} = -\sum_{\lambda} \frac{N_{w\lambda}}{N_{w0}} \sum_{\tau} \frac{N^{\lambda}_{w\tau}}{N_{w\lambda}} \log_2 \left(\frac{N^{\lambda}_{w\tau}}{N_{w\lambda}} \right)$$

where

- λ is a leaf
- $N_{\omega\lambda}$ is the sum of weights of observations on leaf λ
- $N_{\omega 0}$ is the total sum of weights of observations in the entire data set
- τ is a level of the response variable
- $N^{\lambda}_{\omega\tau}$ is the sum of weights of observations on leaf that have response level

Gini Index for Classification Trees

The Gini index for classification trees is defined as

$$\text{Gini} = \sum_{\lambda} \frac{N_{w\lambda}}{N_{w0}} \sum_{\tau} \frac{N_{w\tau}^{\lambda}}{N_{w\lambda}} \left(1 - \frac{N_{w\tau}^{\lambda}}{N_{w\lambda}}\right)$$

Misclassification Rate for Classification Trees

Misclassification (Misc) comes from the number of incorrectly predicted observations. It is defined as

$$\text{Misc} = \frac{1}{N_{w0}} \sum \begin{cases} 0 & \text{if prediction is correct} \\ w_i & \text{otherwise} \end{cases}$$

Residual Sum of Squares for Classification Trees

The residual sum of squares (RSS) for classification trees is defined as

$$\text{RSS} = \sum_{\lambda} \sum_{\Phi} N_{w\Phi}^{\lambda} \left[\sum_{\tau \neq \Phi} \left(P_{w\tau}^{\lambda}\right)^2 + \left(1 - P_{w\Phi}^{\lambda}\right)^2 \right]$$

where

- Φ is the actual response level
- N_{Φ}^{λ} is the number of observations on leaf λ that have response level Φ
- $P_{\omega\tau}^{\lambda}$ is the weighted posterior probability for response level on leaf λ,

$$P_{w\lambda} = \frac{N_{w\tau}^{\lambda}}{N_{w\lambda}}$$

- $P_{\omega\Phi}^{\lambda}$ is the weighted posterior probability for the actual response level Φ on leaf λ,

$$P_{w\Phi} = \frac{N_{w\tau}^{\Phi}}{N_{w\lambda}}$$

Average Square Error for Classification Trees

The average square error (ASE) is also known as the Brier score for classification trees. It is defined as

$$\text{ASE} = \frac{RSS}{N_{w0} N_T}$$

where N_T is the number of levels for the response variable.

Sensitivity for Binary Classification Trees

Sensitivity is the probability of predicting an event for the response variable when the actual state is an event. For example, if the event is “an individual is sick," then sensitivity is the probability of predicting that an individual is sick given that the individual is actually sick. For binary classification trees, it is defined as

$$\text{Sensitivity} = \frac{TP_w}{P_w}$$

where

- TP is the sum of weights of true positives (predicting that an individual is sick)
- P is the sum of weights of positive observations (sick individuals)

Specificity for Binary Classification Trees

Specificity is the probability of predicting a nonevent for the response variable when the actual state is a nonevent. For example, if the event is “an individual is sick," then specificity is the probability of predicting that an individual is not sick given the fact that the individual is actually not sick. For a binary classification tree, specificity is defined as

$$\text{Specificity} = \frac{TN_w}{N_w}$$

where

- TN is the sum of weights of true negatives (predicting that an individual is not sick)
- N is the sum of weights of negative observations (healthy individuals)

Area under the Curve for Binary Classification Trees

Area under the curve (AUC) is defined as the area under the receiver operating characteristic (ROC) curve. PROC HPSPLIT uses sensitivity as the Y axis and 1 – specificity as the X axis to draw the ROC curve. AUC is calculated by trapezoidal rule integration,

$$\text{AUC} = \frac{1}{2}\sum_{\lambda}\left((x_\lambda - x_{\lambda-1})(y_\lambda + y_{\lambda-1})\right)$$

where

- y_λ is the sensitivity value at leaf λ
- x_λ is the 1 – specificity value at leaf λ

Confusion Matrix for Classification Trees

A confusion matrix is also known as a contingency table. It contains information about actual values and predicted values from a classification tree. A confusion matrix has rows and columns, where each row corresponds to the actual response level and each column corresponds to the predicted response level. The values in the matrix represent the number of observations that have the actual response represented in the row and the predicted response represented in the column. The error rate per actual response level is also reported,

$$\mathrm{ErrorRate} = \frac{N_{ww}}{N_{w\Phi}}$$

where

- N_{ww} is the sum of weights of wrong predictions
- $N_{w\Phi}$ is the sum of weights of observations that have response level Φ

Measures of Model Fit for Regression Trees

The HPSPLIT procedure measures model fit for regression trees based on RSS and ASE.

Residual Sum of Squares for Regression Trees

The residual sum of squares (RSS) for regression trees is defined as

$$\mathrm{RSS} = \sum_{\lambda} \sum_{i \in \lambda} w_i \left(y_i - \hat{y}_{\lambda}^{T} \right)^2$$

where

- i is an observation on leaf λ
- y_i is the predicted value of the response variable of observation i
- $\hat{y}_{\lambda}^{T}$ is the actual value of the response variable on leaf λ

Average Square Error for Regression Trees

The average square error (ASE) for regression trees is defined as

$$\mathrm{ASE} = \frac{RSS}{N_{w0}}$$

Random Forest

SAS - HPFOREST Procedure

The HPFOREST procedure is a high-performance procedure that creates a predictive model called a forest that consists of several decision trees. A predictive model defines a relationship between input variables and a target variable. The purpose of a predictive model is to predict a target value from inputs. The HPFOREST procedure trains the model; that is it creates the model using training data in which the target values are known. The model can then be applied to observations in which the target is unknown. If the predictions fit the new data well, the model is said to generalize well. Good generalization is the primary goal for predictive tasks. A predictive model might fit the training data well but generalize poorly.

A decision tree is a type of predictive model that has been developed independently in the statistics and artificial intelligence communities. The HPFOREST procedure creates a tree recursively. An input variable is chosen and used to create a rule to split the data into two segments. The process is then repeated in each segment, and then again in each new segment, and so on until some constraint is met. In the terminology of the tree metaphor, the segments are nodes, the original data set is the root node, and the final unpartitioned segments are leaves or terminal nodes. A node is an internal node if it is not a leaf. The data in a leaf determine the estimates of the value of the target variable. These estimates are subsequently applied to predict the target of a new observation assigned to the leaf.

The HPFOREST procedure creates decision trees that differ from each other in two ways. First, the training data for a tree is a sample, without replacement, from the original training data of the forest. Second, the input variables considered for splitting a node are randomly selected from all available inputs. Among these variables, the HPFOREST procedure considers only a single variable when forming a splitting rule. The chosen variable is the one that is most associated with the target.

PROC HPFOREST runs in either single-machine mode or distributed mode. In distributed mode, PROC HPFOREST trains decision trees in parallel, and accesses all the data for every tree.

Random Forest - Article

The following material is in a R News article by Andy Liaw and Matthew Wiener[1] A portion of the material has been included here.

Recently there has been a lot of interest in “ensemble learning" – methods that generate many classifiers and aggregate their results. Two well-known methods are boosting (see, e.g., Shapire et al., 1998) and bagging Breiman (1996) of classification trees. In boosting, successive trees give extra weight to points incorrectly predicted by earlier predictors. In the end, a weighted vote is taken for prediction. In bagging, successive trees do not depend on earlier trees – each is independently constructed using a bootstrap sample of the data set. In the end, a simple majority vote is taken for prediction.

Breiman (2001) proposed random forests, which add an additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, random forests change how the classification or regression trees are constructed. In standard trees, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node. This somewhat counterintuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis, support vector machines and neural networks, and is robust against overfitting (Breiman, 2001). In addition, it is very user-friendly in the sense that it has only two parameters (the number of variables in the random subset at each node and the number of trees in the forest), and is usually not very sensitive to their values.

The randomForest package provides an R interface to the Fortran programs by Breiman and Cutler (available at http://www.stat.berkeley.edu/users/breiman/). This article provides a brief introduction to the usage and features of the R functions. Suppose that one has a training data set $d = (X, y)$ where X consists of n observations and p dimensions. y is the dependent variable. If y is continuous then the random forest is regression and if y is categorical, the random forest is for classification. Since the random forest is a CART like procedure with bootstrapping, one needs to specify two parameters, the number of bootstrap samples; B = *ntree* and the number variables used at each split for each of the bootstrap samples, $m \leq p$ = *mtry*. Note: $m = \sqrt{p}$ or $p/3$ are common values for m.

[1] included in the course BOX.

Algorithm for RF

1. Draw *ntree* bootstrap samples from the original data.
2. For each of the bootstrap samples, grow an unpruned classification or regression tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample *mtry* of the predictors and choose the best split from among those variables. (Bagging can be thought of as the special case of random forests obtained when *mtry* = p, the number of predictors.)
3. Predict new data by aggregating the predictions of the *ntree* trees (i.e., majority votes for classification, average for regression).

An estimate of the error rate can be obtained, based on the training data, by the following:

1. At each bootstrap iteration, predict the data not in the bootstrap sample (what Breiman calls “out-of-bag", or OOB, data) using the tree grown with the bootstrap sample.
2. Aggregate the OOB predictions. (On the average, each data point would be out-of-bag around 36% of the times, so aggregate these predictions.) Calculate the error rate, and call it the OOB estimate of error rate.

Our experience has been that the OOB estimate of error rate is quite accurate, given that enough trees have been grown (otherwise the OOB estimate can bias upward; see Bylander (2002)).

Bagging the Data

A decision tree in a forest trains on new training data that are derived from the original training data presented to the HPFOREST procedure. Training different trees with different training data reduces the correlation of the predictions of the trees, which in turn should improve the predictions of the forest.

The HPFOREST procedure samples the original data without replacement to create the training data for an individual tree. Most forest algorithms sample with replacement. The convention of sampling with replacement originated with Leo Breiman’s bagging algorithm (Breiman 1996, 2001). The word bagging stems from "bootstrap aggregating," where "bootstrap" refers to a process that uses sampling with replacement. Breiman refers to the observations that are excluded from the sample as out-of-bag (OOB) observations. Therefore, observations in the training sample are called the bagged observations, and the training data for a specific decision tree are called the bagged data. Subsequently, Freedman and Popescu (2003) argued that sampling without replacement can provide more variability between the trees, especially with larger training sets.

The INBAGN= and INBAGFRACTION= options in the PROC HPFOREST statement specify the number of observations to sample without replacement into a bagged data set.

Estimating the goodness-of-fit of the model by using the training data is usually too optimistic; the fit of the model to new data is usually worse than the fit to the training data. Estimating the goodness-of-fit by using the out-of-bag data is usually too pessimistic at first. With enough trees, the out-of-bag estimates are an unbiased estimate of the generalization fit.

The R Perspective

There are many slight variations in CART analysis based on almost any data structure permutation we could imagine. If you decide to apply CART in your own work, Google and R web forums will prove to be very helpful in determining which slight adjustments need to be made now that you are familiar with the basic structure and terminology associated with the procedure. It is appropriate to discuss two commonly used variations: bagging and boosting. Bagging was developed by Breiman and appeared shortly after his seminal work defining the field, and has gained traction following the increased interest in bootstrapping and similar procedures in statistical analysis, and it is useful to think of it as bootstrapping for tree analysis. The name derives from bootstrap aggregating and involves creating multiple similar datasets, re-running the tree analysis, and then collecting the aggregate of the results and re-calculating the tree and associated statistics based on this aggregate. This technique is often used as cross-validation for larger trees a user wishes to prune and where different versions of the same tree have vastly different rates of misclassification. In general, the procedure will improve the results of a highly unstable tree but may decrease the performance of a stable tree. Using the package **ipred** the procedure is easy to implement, and we will briefly present it here using the data from our applied example:

```
mybag =bagging(family medpatient gender+patient age+ patient ethnicity +
patient insurance +...+ comorbidites, data = mydata,nbagg=30,coob=T)
```

where nbagg specifies that the procedure will create 30 full datasets to aggregate and coob specifies the aggregation selection technique, here the averaged model ("out-of-the-bag"). Calling on the command mybag$err will return the new misclassification error, in our case 23.9% – an unimpressive 0.01 decrease in misclassification. Boosting is a technique that seeks to reduce misclassification by a recursive tree model. Here, classifiers are iteratively created by weighted versions of the sample where the weights are adjusted at each iteration based on which cases were misclassified in the previous step – many "mini-trees" which exhibit continuously decreasing misclassification. This technique is often applied to data which has high misclassification because it is largely uninformative, or a "weak learner." In these data sets classification is only slightly better than a random guess (think misclassification only slightly less than 50%), since the data are so loosely (perhaps because of confounders) related [15]. Implementing boosting is either done through a complex series of packages in R or some third-party software specializing in decision tree analysis. Our results indicate that our data are not weak learners, so we will not implement boosting here; in our case, bagging is much more appropriate. Generally, the data structure will indicate whether boosting or bagging is more appropriate. For more information on boosting in R, consult the **adabag** function.

Information about the Boston Housing Data

The Boston housing dataset is small, especially in today's age of big data. But there was a time where neatly collected and labeled data was extremely hard to access, so a publicly available dataset like this was very valuable to researchers. And although we now have things like Kaggle and open government initiatives which give us plenty of datasets to choose from, this one is a staple to machine learning practice as chocolate is to a break-up.

Each of the 506 rows in the dataset describes a Boston suburb or town, and it has 14 columns with information such as average number of rooms per dwelling, pupil-teacher ratio, and per capita crime rate. The last row describes the median price of owner-occupied homes (this leaves out homes that are rented out), and it's usually the row that we are trying to predict when we use it for regression tasks. A description of the variables is included in the following figure.

7.2.1. Boston house prices dataset

Data Set Characteristics:

Number of Instances:	506
Number of Attributes:	13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.
Attribute Information (in order):	• CRIM per capita crime rate by town • ZN proportion of residential land zoned for lots over 25,000 sq.ft. • INDUS proportion of non-retail business acres per town • CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) • NOX nitric oxides concentration (parts per 10 million) • RM average number of rooms per dwelling • AGE proportion of owner-occupied units built prior to 1940 • DIS weighted distances to five Boston employment centres • RAD index of accessibility to radial highways • TAX full-value property-tax rate per $10,000 • PTRATIO pupil-teacher ratio by town • B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town • LSTAT % lower status of the population • MEDV Median value of owner-occupied homes in $1000's
Missing Attribute Values:	None
Creator:	Harrison, D. and Rubinfeld, D.L.

More importantly, Otis W. Gilley also rechecked all of the data against the original census data and found that eight of the median prices in the median value column were plain and simply wrong! This is what he found:

Incorrect data

More importantly, Otis W. Gilley also rechecked all of the data against the original census data and found that eight of the median prices in the median value column were plain and simply wrong!

This is what he found:

Table 1 — Miscoded Dependent Variable Observations

Observation and Tract Number	Median Value	Corrected Median Value	Percentage Error
8-2042	27.1	22.1	22.62%
39-2084	24.7	24.2	2.07%
119-3585	37.0	33.0	12.12%
241-3823	22.0	27.0	-18.42%
438-0905	8.7	8.2	6.1%
443-0911	18.4	14.8	24.32%
455-0923	14.9	14.4	3.47%
506-1805	11.9	19.0	-37.37%

Source: Gilley (1996) On the Harrison and Rubinfeld Data

Gilley proceeded to correct the dataset, run the calculations of the original paper on hedonic pricing and check if the results still held true. Luckily for the history of data science, there were no significant changes.

The goodness-of-fit as measured by R2 rises somewhat when employing the corrected observations. However, the magnitudes of the coefficients did not change much and the qualitative results from the original regression still hold.

R

```r
  # clear the environment and set seed
rm(list = ls())
set.seed(123)
```

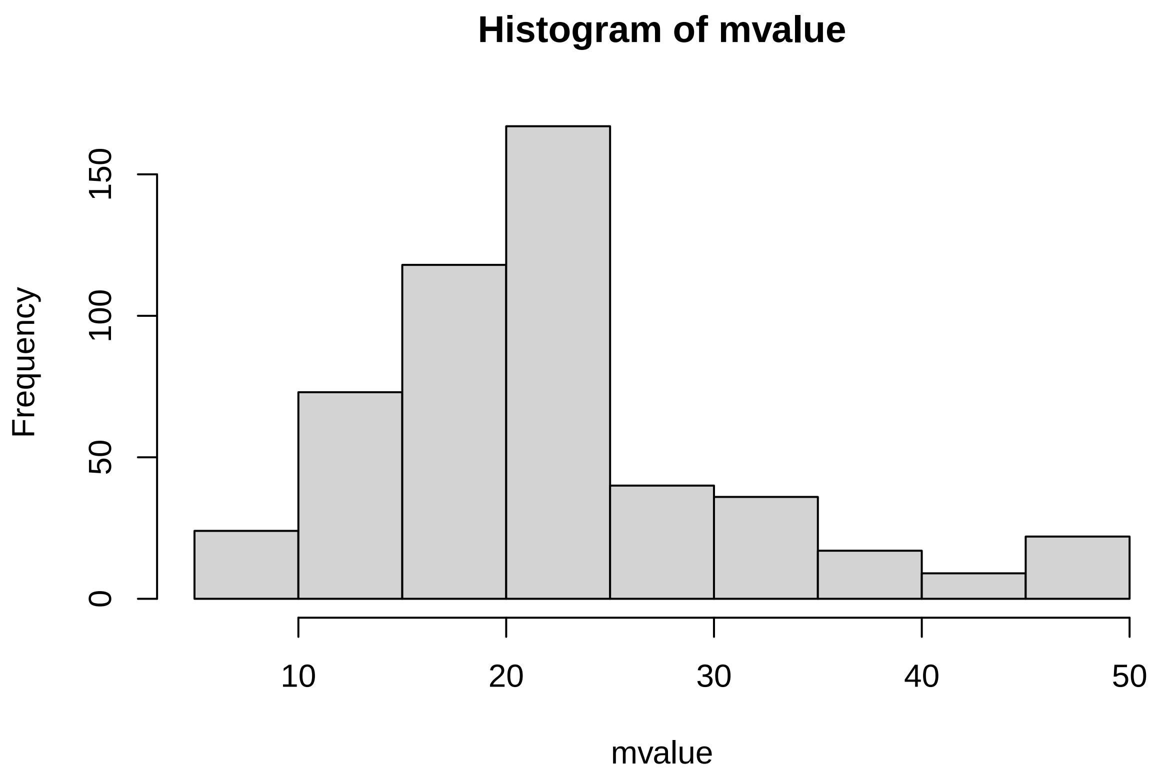
Read Boston Housing Data

```
bhouse = read.csv("bostonhousing.csv")
summary(bhouse)
```

```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox             rooms            age            distance
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##      radial            tax              pt            lstat
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95
##  Median : 5.000   Median :330.0   Median :19.05   Median :11.36
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97
##      mvalue
##  Min.   : 5.00
##  1st Qu.:17.02
##  Median :21.20
##  Mean   :22.53
##  3rd Qu.:25.00
##  Max.   :50.00
```

Describe Dependent Variable

```
mvalue = bhouse$mvalue
hist(mvalue)
```




```
summary(mvalue)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   17.02   21.20   22.53   25.00   50.00
```

```
options("repos" = c(CRAN = "https://cran.rstudio.com"))
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(rpart)
```

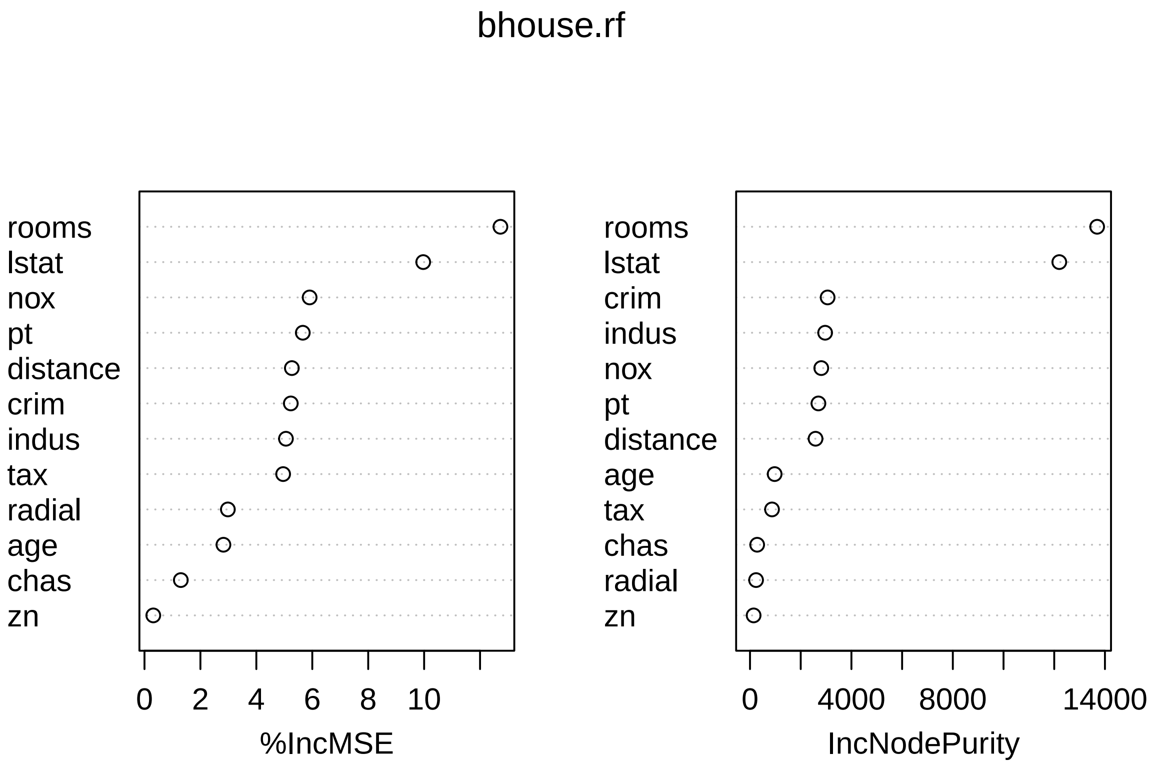
##Perform Regression

```
bhouse.tr = rpart(mvalue ~ . , data=bhouse)
bhouse.rf <- randomForest(mvalue ~ .,data=bhouse,mtry=4,
                  importance=TRUE, ntree=50,
                  na.action=na.omit)

print(bhouse.rf)
```

```
##
## Call:
```

```
##  randomForest(formula = mvalue ~ ., data = bhouse, mtry = 4, importance = TRUE,
ntree = 50, na.action = na.omit)
##                Type of random forest: regression
##                      Number of trees: 50
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 11.40417
##                     % Var explained: 86.49
```

```
varImpPlot(bhouse.rf)
```

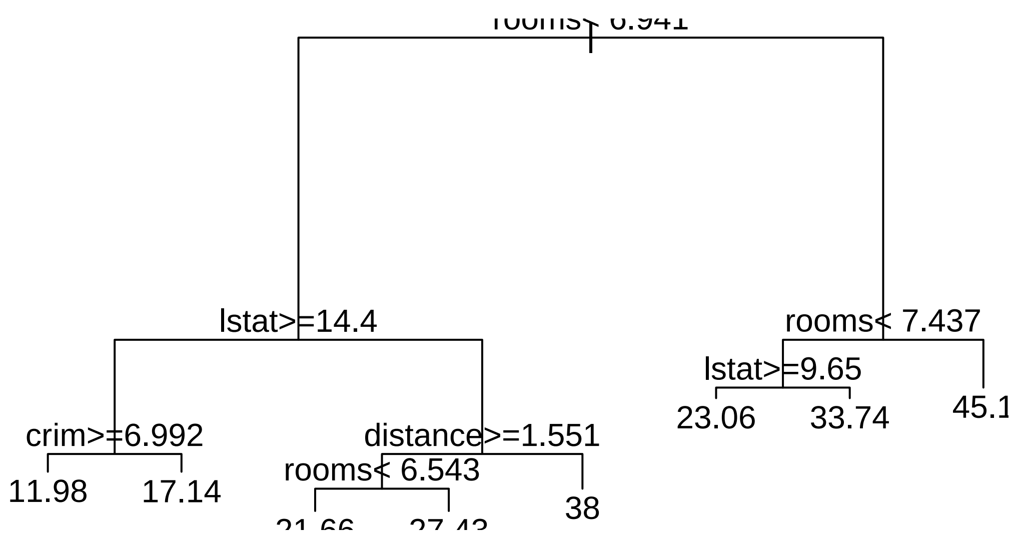



```
# Output for Tree
print(bhouse.tr)
```

```
## n= 506
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 506 42716.3000 22.53281
##    2) rooms< 6.941 430 17317.3200 19.93372
##      4) lstat>=14.4 175  3373.2510 14.95600
##        8) crim>=6.99237 74  1085.9050 11.97838 *
```

```
##        9) crim< 6.99237 101  1150.5370 17.13762 *
##      5) lstat< 14.4 255  6632.2170 23.34980
##       10) distance>=1.5511 248  3658.3930 22.93629
##         20) rooms< 6.543 193  1589.8140 21.65648 *
##         21) rooms>=6.543 55   643.1691 27.42727 *
##       11) distance< 1.5511 7  1429.0200 38.00000 *
##    3) rooms>=6.941 76  6059.4190 37.23816
##      6) rooms< 7.437 46  1899.6120 32.11304
##       12) lstat>=9.65 7   432.9971 23.05714 *
##       13) lstat< 9.65 39   789.5123 33.73846 *
##      7) rooms>=7.437 30  1098.8500 45.09667 *
```

```
plot(bhouse.tr)
text(bhouse.tr)
```




```
#summary(bhouse.tr)
```

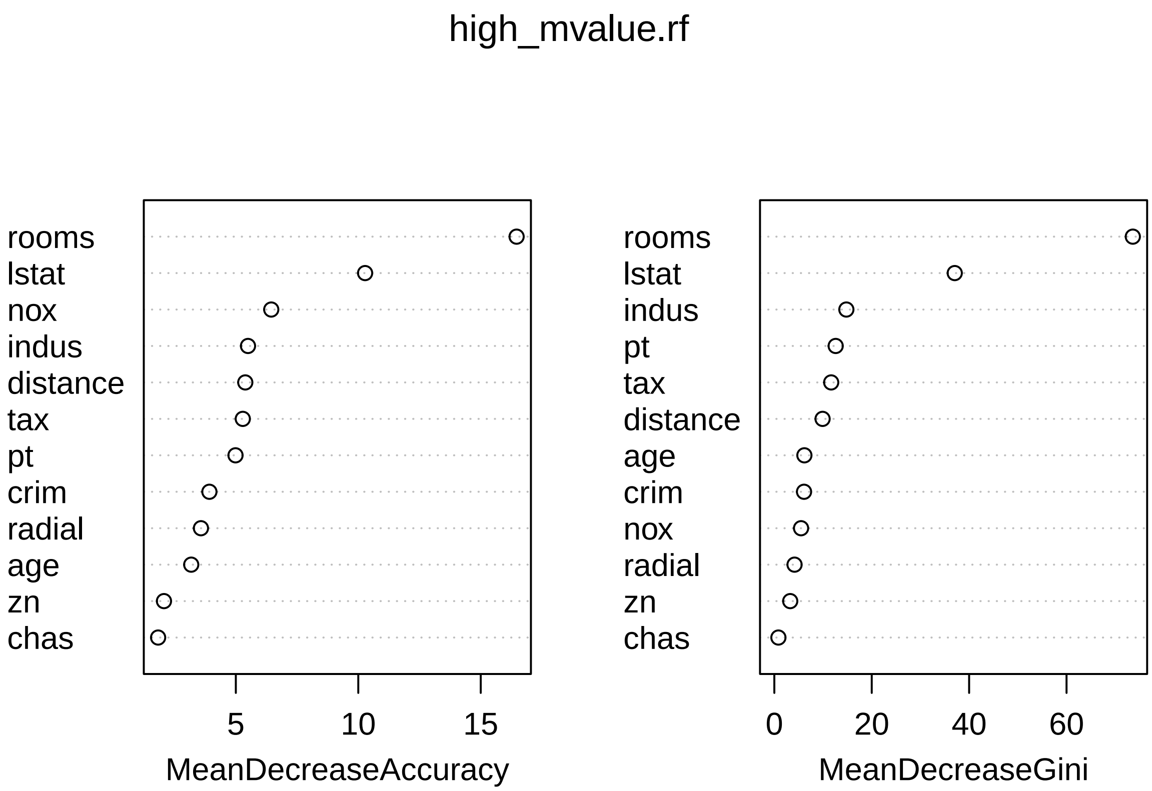
##Perform Classification

```
# Create Binary variable for mvalue
# 25 was my choice
high_mvalue = factor((mvalue > 25))

high_mvalue.tr = rpart(high_mvalue ~ .-mvalue, data=bhouse)
high_mvalue.rf <- randomForest(high_mvalue ~ . - mvalue, data=bhouse,mtry=4,
                   importance=TRUE, ntree=50,
                   proximity=TRUE)

print(high_mvalue.rf)
```

```
##
```

```
## Call:
##  randomForest(formula = high_mvalue ~ . - mvalue, data = bhouse,
mtry = 4, importance = TRUE, ntree = 50, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 5.53%
## Confusion matrix:
##       FALSE TRUE class.error
## FALSE   370   12  0.03141361
## TRUE     16  108  0.12903226
```

```
varImpPlot(high_mvalue.rf)
```

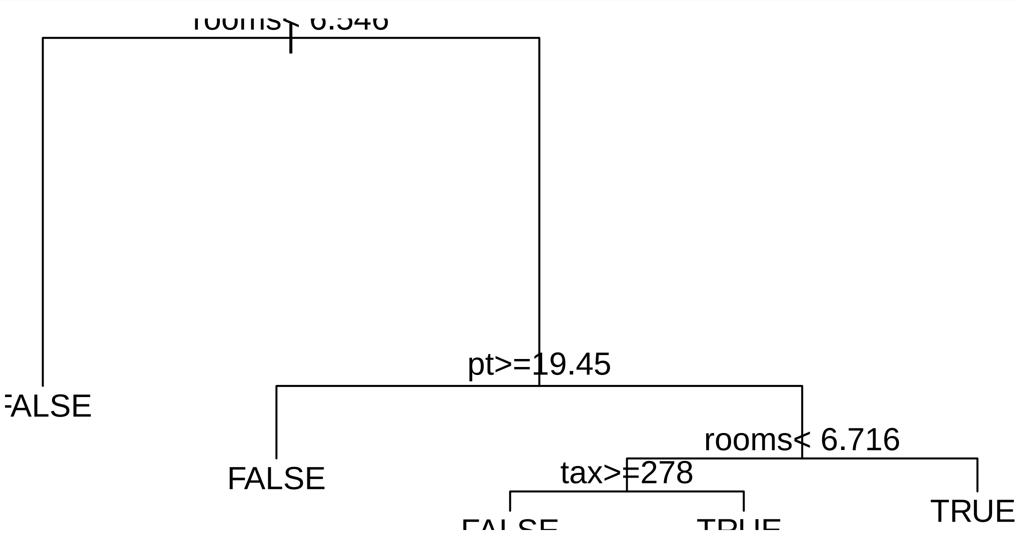



```
# Output for Tree
print(high_mvalue.tr)
```

```
## n= 506
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
```

```
##
##  1) root 506 124 FALSE (0.75494071 0.24505929)
##    2) rooms< 6.5455 362  16 FALSE (0.95580110 0.04419890) *
##    3) rooms>=6.5455 144  36 TRUE (0.25000000 0.75000000)
##      6) pt>=19.45 29   7 FALSE (0.75862069 0.24137931) *
##      7) pt< 19.45 115  14 TRUE (0.12173913 0.87826087)
##       14) rooms< 6.7165 26  11 TRUE (0.42307692 0.57692308)
##         28) tax>=278 18   7 FALSE (0.61111111 0.38888889) *
##         29) tax< 278 8   0 TRUE (0.00000000 1.00000000) *
##       15) rooms>=6.7165 89   3 TRUE (0.03370787 0.96629213) *
```

```
plot(high_mvalue.tr)
text(high_mvalue.tr)
```




```
#summary(high_mvalue.tr)
```

SAS

Code

```
options center nodate pagesize=80 ls=70;
libname ldata '/home/jacktubbs/my_shared_file_links/jacktubbs/LaTeX/Class';

/* Simplified LaTeX output that uses plain LaTeX tables  */
ods latex path='/home/jacktubbs/my_shared_file_links/jacktubbs/LaTeX/clean'
 file='boston_housing_RF.tex' style=journal
stylesheet="sas.sty"(url="sas");

/*
http://support.sas.com/rnd/base/ods/odsmarkup/latex.html
*/
```

```
ods graphics / reset width=5in  outputfmt=png
  antialias=on;

*/;
title "Boston Housing Data";
data bhouse; set ldata.bostonhousing;
run;

data bhouse; set bhouse;
keep age chas crim distance indus lstat
     mvalue nox pt radial rooms tax zn;
run;

proc means data=bhouse q1 median mean q3;
var age  crim distance
indus lstat mvalue nox pt radial rooms tax zn;
run;


data bhouse; set bhouse;
high_mvalue = (mvalue > 25);
run;

title2 'Regression';
proc hpsplit data=bhouse cvmodelfit seed=123;
   class chas;
   model mvalue = age chas crim distance zn
                  indus lstat nox pt radial rooms tax zn;
*   grow entropy;
*   prune costcomplexity;
  output out=hpsplout;
run;


proc hpforest data=bhouse maxtrees=50 inbagfraction=.3;
   input age crim distance indus lstat zn
         nox pt radial rooms tax /level=interval;
   input chas/level=nominal;
   target mvalue/level=interval;
   ods output VariableImportance = variable
           FitStatistics=fitstats(rename=(Ntrees=Trees)) ;
run;

data fitstats;
   set fitstats;
   label Trees = 'Number of Trees';
   label MiscAll = 'Full Data';
   label Miscoob = 'OOB';
run;
```

```
proc sgplot data=fitstats;
   title "OOB vs Training";
   series x=Trees y=predall;
   series x=Trees y=predOob/lineattrs=(pattern=shortdash thickness=2);
   yaxis label='Average Squared Error';
run;

title2 'Classification';
proc hpsplit data=bhouse cvmodelfit seed=123;
   class chas high_mvalue;
   model high_mvalue = age chas crim distance
                       indus lstat nox pt radial rooms tax zn;
   grow entropy;
   prune costcomplexity;
run;


proc hpforest data=bhouse maxtrees=100 inbagfraction=.3;
   input age crim distance indus lstat zn
         nox pt radial rooms tax /level=interval;
   input chas/level=nominal;
   target high_mvalue/level=binary;
   ods output VariableImportance = variable;
run;

ods latex close;
quit;
```

Output

Boston Housing Data

The MEANS Procedure

Variable	Q!	Median	Mean	Q3
age	45.0000000	77.5000000	68.5749012	94.1000000
crim	0.0819900	0.2565100	3.6135236	3.6782200
distance	2.1000000	3.2074500	3.7950427	5.2119000
indus	5.1900000	9.6900000	11.1367787	18.1000000
lstat	6.9300000	11.3600000	12.6530632	16.9600000
mvalue	17.0000000	21.2000000	22.5328063	25.0000000
nox	0.4490000	0.5380000	0.5546951	0.6240000
pt	17.4000000	19.0500000	18.4555336	20.2000000
radial	4.0000000	5.0000000	9.5494071	24.0000000
rooms	5.8850000	6.2085000	6.2846344	6.6250000

Variable	Q!	Median	Mean	Q3
tax	279.0000000	330.0000000	408.2371542	666.0000000
zn	0	0	11.3636364	12.5000000

Boston Housing Data

Regression

The HPSPLIT Procedure

Performance Information	
Execution Mode	Single−Machine
Number of Threads	2

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.BHOUSE	V9	Input	On Client
WORK.HPSPLOUT	V9	Output	On Client

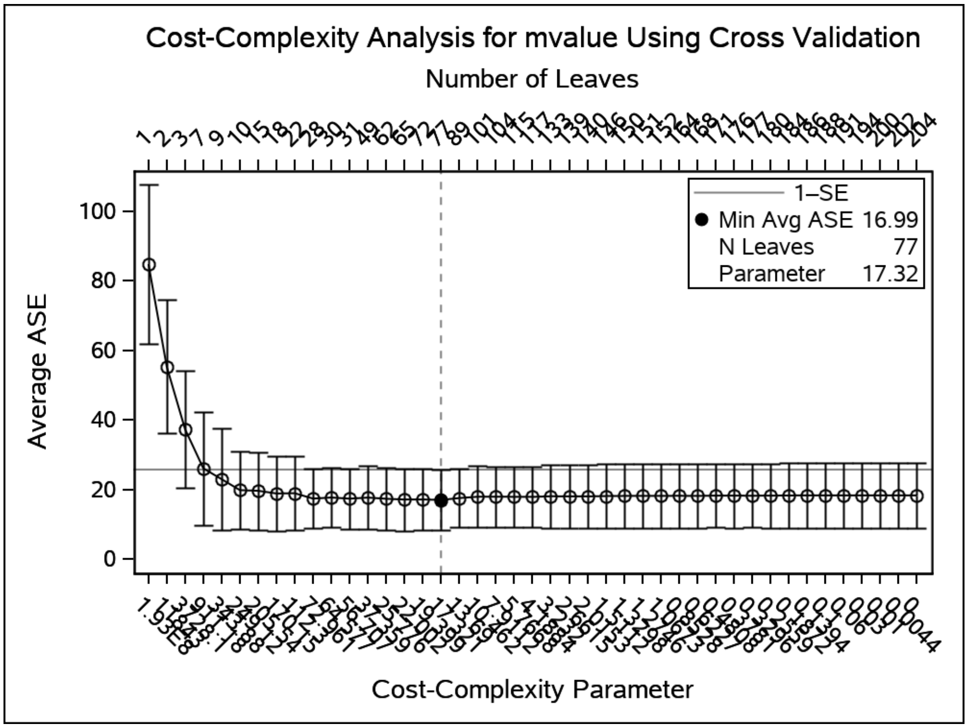
Model Information	
Split Criterion Used	Variance
Pruning Method	Cost−Complexity
Subtree Evaluation Criterion	Cost−Complexity
Number of Branches	2
Maximum Tree Depth Requested	10
Maximum Tree Depth Achieved	10
Tree Depth	10
Number of Leaves Before Pruning	208
Number of Leaves After Pruning	89

Number of Observations Read	506
Number of Observations Used	506

Boston Housing Data

Regression

The HPSPLIT Procedure

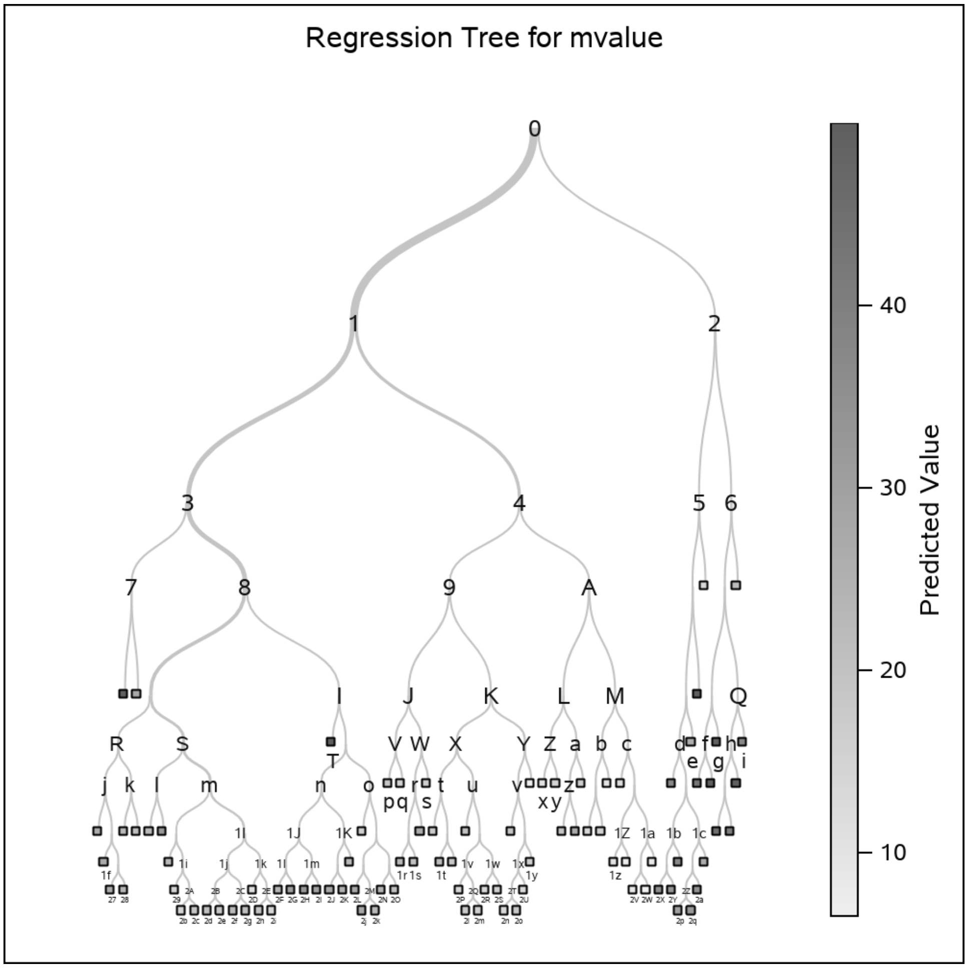


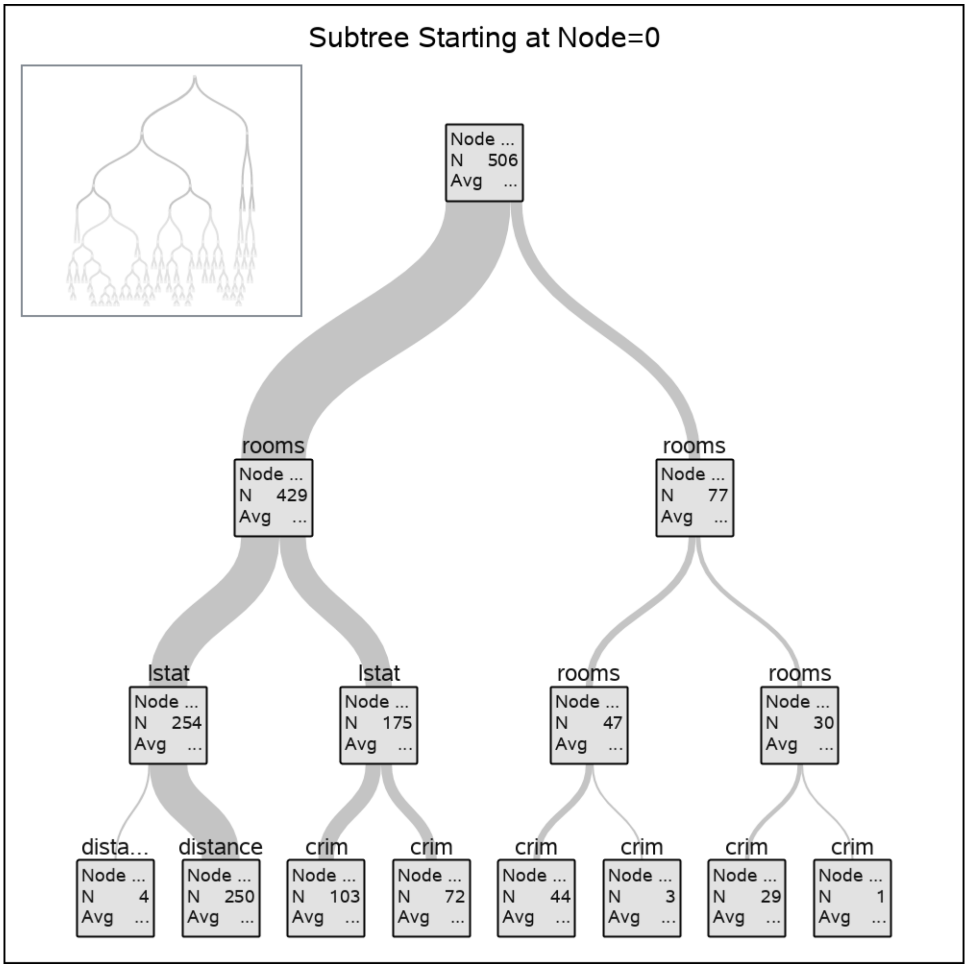
10-Fold Cross Validation Assessment of Model							
N Leaves	Average Square Error				Number of Leaves		
	Min	Avg	Standard Error	Max	Min	Median	Max
76	9.0542	19.1359	8.8368	37.7555	67	76.5	87

Boston Housing Data

Regression

The HPSPLIT Procedure





Subtree Starting at Node=0
Node ...
N 506
Avg ...
rooms
Node ...
N 429
Avg ...
rooms
Node ...
N 77
Avg ...
lstat
Node ...
N 254
Avg ...
lstat
Node ...
N 175
Avg ...
rooms
Node ...
N 47
Avg ...
rooms
Node ...
N 30
Avg ...
dista...
Node ...
N 4
Avg ...
distance
Node ...
N 250
Avg ...
crim
Node ...
N 103
Avg ...
crim
Node ...
N 72
Avg ...
crim
Node ...
N 44
Avg ...
crim
Node ...
N 3
Avg ...
crim
Node ...
N 29
Avg ...
crim
Node ...
N 1
Avg ...

Boston Housing Data

Regression

The HPSPLIT Procedure

Fit Statistics for Selected Tree			
	N Leaves	ASE	RSS
Model Based	89	2.0023	1013.2
Cross Validation	76	19.1359	

Variable Importance			
Variable	Training		Count
	Relative	Importance	
rooms	1.0000	156.2	12
lstat	0.5913	92.3561	15
crim	0.3836	59.9136	10
distance	0.3267	51.0250	10
nox	0.1766	27.5883	7
pt	0.1622	25.3410	8
tax	0.1510	23.5822	8
age	0.1349	21.0741	13
indus	0.0701	10.9450	3
zn	0.0450	7.0291	1
chas	0.0203	3.1754	1

Boston Housing Data

Regression

The HPFOREST Procedure

Performance Information	
Execution Mode	Single–Machine
Number of Threads	2

Data Access Information			
Data	Engine	Role	Path
WORK.BHOUSE	V9	Input	On Client

Model Information		
Parameter	Value	
Variables to Try	3	(Default)
Maximum Trees	50	
Actual Trees	50	
Inbag Fraction	0.3	
Prune Fraction	0	(Default)
Prune Threshold	0.1	(Default)
Leaf Fraction	0.00001	(Default)
Leaf Size Setting	1	(Default)
Leaf Size Used	1	
Category Bins	30	(Default)
Interval Bins	100	
Minimum Category Size	5	(Default)
Node Size	100000	(Default)
Maximum Depth	20	(Default)
Alpha	1	(Default)
Exhaustive	5000	(Default)
Rows of Sequence to Skip	5	(Default)
Split Criterion	.	Variance
Preselection Method	.	BinnedSearch
Missing Value Handling	.	Valid value

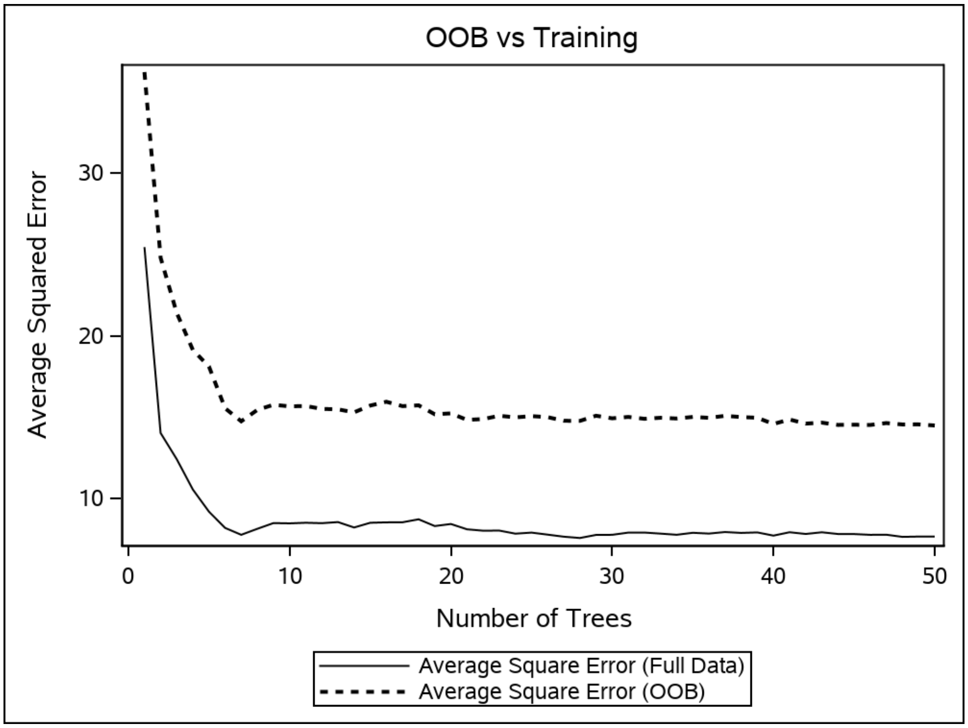
Number of Observations	
Type	**N**
Number of Observations Read	506
Number of Observations Used	506

Baseline Fit Statistics	
Statistic	**Value**
Average Square Error	84.420

Fit Statistics			
# of Trees	**# of Leaves**	**ASE (Train)**	**ASE (OOB)**
1	**137**	25.4592	36.2393
2	**276**	14.0128	24.8102
3	**417**	12.4013	21.4116
4	**556**	10.5306	19.1283
5	**694**	9.1646	18.1032
6	**829**	8.1653	15.5360
7	**969**	7.7406	14.7312
8	**1110**	8.1107	15.4334
9	**1246**	8.4632	15.7497
10	**1373**	8.4460	15.6530
11	**1515**	8.4819	15.6738
12	**1650**	8.4567	15.4963
13	**1769**	8.5239	15.4664
14	**1908**	8.1911	15.2832
15	**2041**	8.4818	15.7009
16	**2177**	8.5074	15.9349
17	**2309**	8.5118	15.6661
18	**2448**	8.6950	15.7148
19	**2576**	8.2802	15.1639
20	**2709**	8.4064	15.2179

Fit Statistics			
# of Trees	# of Leaves	ASE (Train)	ASE (OOB)
21	2844	8.0799	14.8190
22	2983	7.9937	14.8675
23	3105	8.0011	15.0715
24	3248	7.8107	14.9724
25	3388	7.8751	15.0495
26	3526	7.7506	14.9863
27	3671	7.6242	14.7624
28	3800	7.5436	14.7555
29	3943	7.7332	15.0728
30	4085	7.7399	14.9140
31	4211	7.8755	14.9996
32	4352	7.8783	14.8842
33	4476	7.8129	14.9484
34	4605	7.7422	14.9014
35	4729	7.8637	14.9993
36	4869	7.8155	14.9446
37	5006	7.9112	15.0665
38	5143	7.8561	14.9923
39	5283	7.8871	14.9475
40	5421	7.6902	14.5663
41	5539	7.8977	14.8411
42	5682	7.7921	14.5884
43	5798	7.8956	14.6481
44	5939	7.7895	14.5084
45	6069	7.7859	14.5198
46	6210	7.7384	14.5058
47	6353	7.7453	14.6223
48	6491	7.6132	14.5359
49	6627	7.6319	14.5403
50	6765	7.6305	14.4742

Loss Reduction Variable Importance					
Variable	of Rules	MSE	OOB MSE	AError	OOB AError
rooms	1373	25.79040	19.25258	1.701245	1.048281
lstat	1049	20.07961	17.00860	1.478753	0.903041
indus	430	5.93505	3.31363	0.424529	0.157663
tax	745	5.21676	2.92277	0.444815	0.171385
pt	416	3.82947	2.21543	0.322339	0.115898
crim	666	7.59916	1.63267	0.661336	0.180056
nox	621	5.41670	1.44049	0.480321	0.147262
age	439	2.17919	0.04980	0.313483	0.051374
chas	3	0.37087	−0.04171	0.004215	−0.004570
zn	162	0.32661	−0.24555	0.053790	−0.010731
distance	562	3.95016	−0.47153	0.395853	−0.000696
radial	249	0.90687	−0.56833	0.102220	−0.023745


OOB vs Training
Average Squared Error
30
20
10
0
10
20
30
40
50
Number of Trees
Average Square Error (Full Data)
Average Square Error (OOB)

OOB vs Training
Classification
The HPSPLIT Procedure

Performance Information	
Execution Mode	Single−Machine
Number of Threads	2

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.BHOUSE	V9	Input	On Client

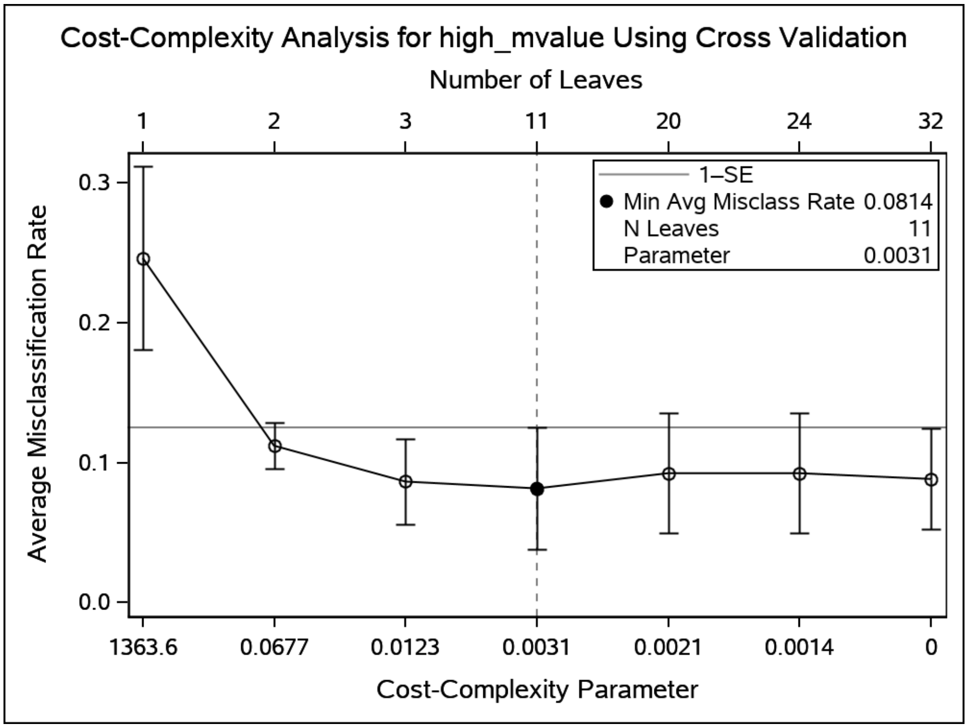
Model Information	
Split Criterion Used	Entropy
Pruning Method	Cost−Complexity
Subtree Evaluation Criterion	Cost−Complexity
Number of Branches	2
Maximum Tree Depth Requested	10
Maximum Tree Depth Achieved	10
Tree Depth	6
Number of Leaves Before Pruning	33
Number of Leaves After Pruning	10
Model Event Level	0

Number of Observations Read	506
Number of Observations Used	506

OOB vs Training

Classification

The HPSPLIT Procedure



10-Fold Cross Validation Assessment of Model

N Leaves	ASE				N Leaves			Misclass Rate			
	Min	Avg	SE	Max	Min	Median	Max	Min	Avg	SE	Max
10	0.0192	0.0763	0.0379	0.1314	5	10.0	15	0.0172	0.0884	0.0462	0.1489

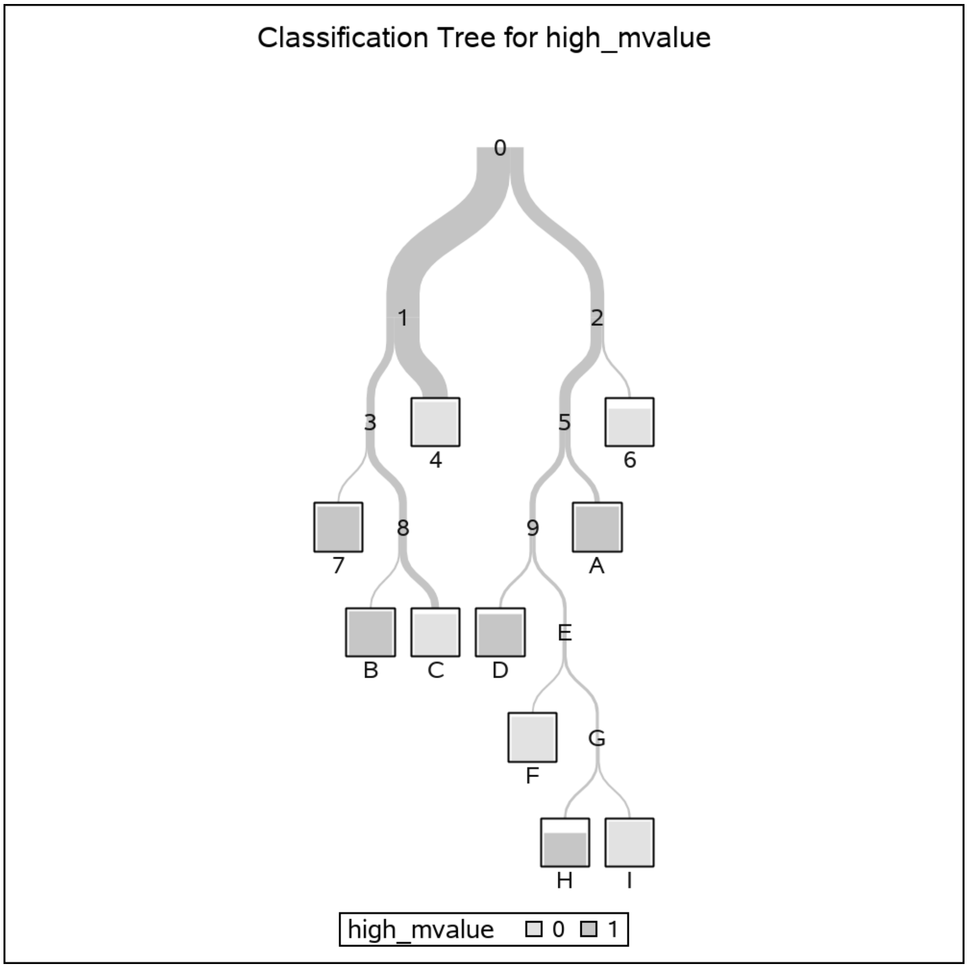
10-Fold Cross Validation Confusion Matrix

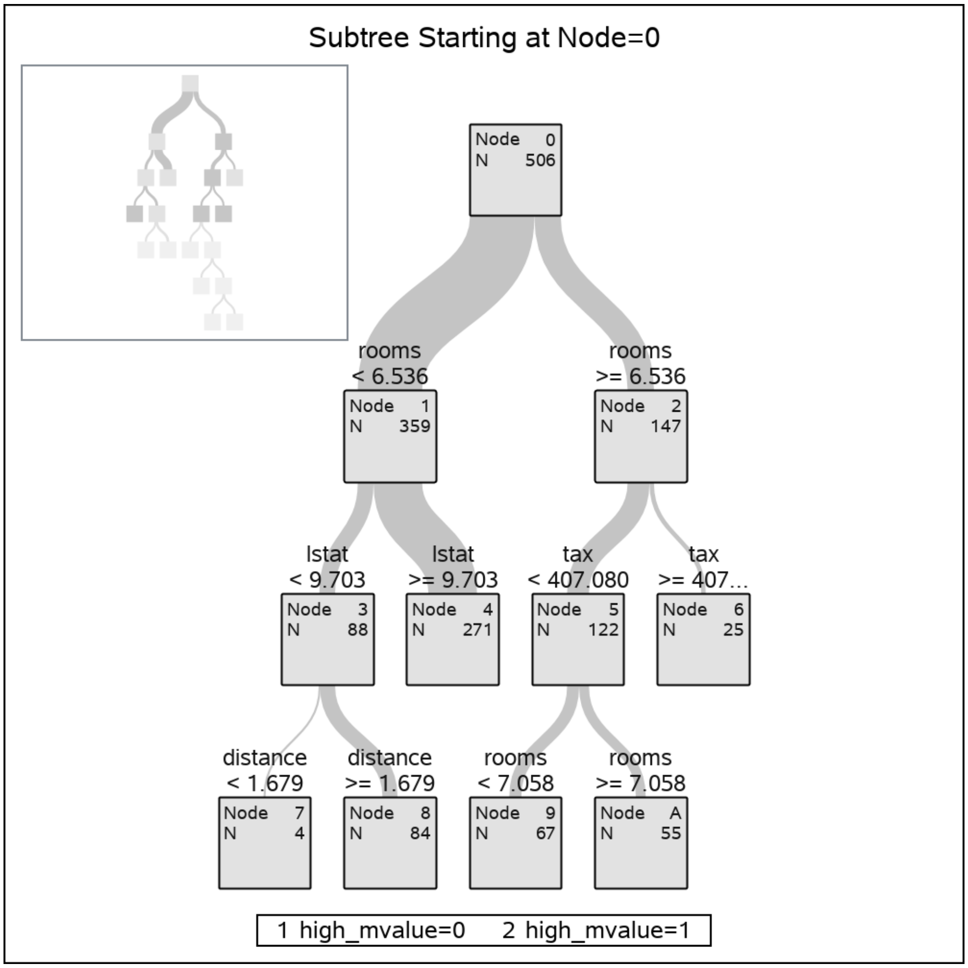
Actual	Predicted		Error Rate
	0	1	
0	361	21	0.0550
1	23	101	0.1855

OOB vs Training

Classification

The HPSPLIT Procedure




Subtree Starting at Node=0
Node 0
N 506
rooms
< 6.536
rooms
>= 6.536
Node 1
N 359
Node 2
N 147
lstat
< 9.703
lstat
>= 9.703
tax
< 407.080
tax
>= 407...
Node 3
N 88
Node 4
N 271
Node 5
N 122
Node 6
N 25
distance
< 1.679
distance
>= 1.679
rooms
< 7.058
rooms
>= 7.058
Node 7
N 4
Node 8
N 84
Node 9
N 67
Node A
N 55
1 high_mvalue=0
2 high_mvalue=1

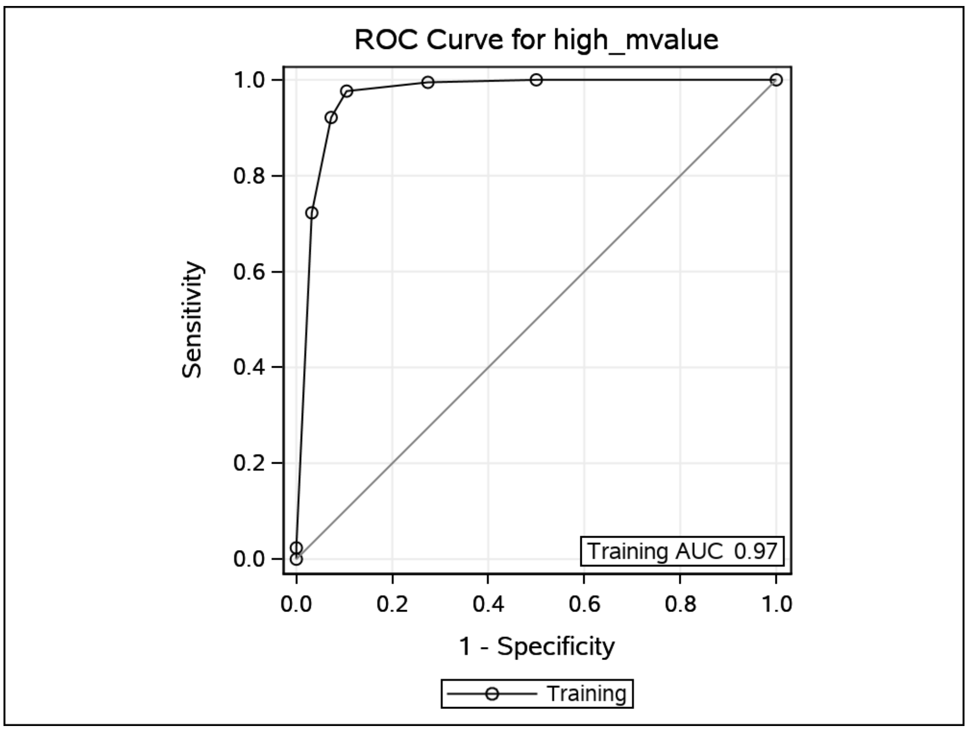
OOB vs Training

Classification

The HPSPLIT Procedure

Confusion Matrices				
	Actual	Predicted		Error Rate
		0	1	
Model Based	0	373	9	0.0236
	1	13	111	0.1048
Cross Validation	0	361	21	0.0550
	1	23	101	0.1855

Fit Statistics for Selected Tree									
	N Leaves	ASE	Misclass	Sens	Spec	Ent	Gini	RSS	AUC
Model Based	10	0.0378	0.0435	0.9764	0.8952	0.2101	0.0755	38.2180	0.9679
Cross Validation	10	0.0763	0.0884	0.9450	0.8145				



Variable Importance			
Variable	Training		Count
	Relative	Importance	
rooms	1.0000	10.1836	2
tax	0.4909	4.9991	2
lstat	0.2484	2.5293	2
distance	0.2455	2.5002	1
pt	0.1949	1.9843	1
nox	0.1895	1.9299	1

OOB vs Training

Classification

The HPFOREST Procedure

Performance Information	
Execution Mode	Single−Machine
Number of Threads	2

Data Access Information			
Data	**Engine**	**Role**	**Path**
WORK.BHOUSE	V9	Input	On Client

Model Information		
Parameter	**Value**	
Variables to Try	3	(Default)
Maximum Trees	100	
Actual Trees	100	
Inbag Fraction	0.3	
Prune Fraction	0	(Default)
Prune Threshold	0.1	(Default)
Leaf Fraction	0.00001	(Default)
Leaf Size Setting	1	(Default)
Leaf Size Used	1	
Category Bins	30	(Default)
Interval Bins	100	
Minimum Category Size	5	(Default)
Node Size	100000	(Default)
Maximum Depth	20	(Default)
Alpha	1	(Default)
Exhaustive	5000	(Default)
Rows of Sequence to Skip	5	(Default)
Split Criterion	.	Gini
Preselection Method	.	BinnedSearch
Missing Value Handling	.	Valid value

Number of Observations	
Type	**N**
Number of Observations Read	506
Number of Observations Used	506

Baseline Fit Statistics	
Statistic	**Value**
Average Square Error	0.185
Misclassification Rate	0.245
Log Loss	0.557

Fit Statistics							
# of Trees	**# of Leaves**	**ASE (Tr)**	**ASE (OB)**	**MisRate (Tr)**	**MisRate (OB)**	**LLoss (Tr)**	**LLoss (OB)**
1	**10**	0.0632	0.0901	0.0632	0.0901	1.456	2.076
2	**29**	0.0535	0.0898	0.0692	0.0978	0.749	1.746
3	**49**	0.0411	0.0755	0.0553	0.0881	0.279	1.208
4	**67**	0.0362	0.0678	0.0514	0.0898	0.197	0.639
5	**82**	0.0356	0.0649	0.0435	0.0772	0.157	0.592
6	**100**	0.0342	0.0622	0.0553	0.0791	0.114	0.505
7	**118**	0.0351	0.0641	0.0375	0.0889	0.119	0.513
8	**135**	0.0321	0.0595	0.0455	0.0672	0.113	0.505
9	**150**	0.0319	0.0594	0.0316	0.0692	0.113	0.506
10	**166**	0.0310	0.0579	0.0375	0.0652	0.114	0.423
11	**186**	0.0306	0.0581	0.0316	0.0652	0.113	0.425
12	**205**	0.0296	0.0566	0.0336	0.0672	0.110	0.381
13	**222**	0.0288	0.0546	0.0336	0.0652	0.107	0.377
14	**240**	0.0283	0.0533	0.0336	0.0652	0.106	0.373
15	**256**	0.0283	0.0521	0.0296	0.0632	0.106	0.372
16	**272**	0.0278	0.0517	0.0316	0.0652	0.104	0.370
17	**291**	0.0273	0.0516	0.0316	0.0613	0.104	0.371
18	**307**	0.0277	0.0521	0.0336	0.0652	0.105	0.373

Fit Statistics							
# of Trees	# of Leaves	ASE (Tr)	ASE (OB)	MisRate (Tr)	MisRate (OB)	LLoss (Tr)	LLoss (OB)
19	328	0.0273	0.0520	0.0316	0.0613	0.106	0.334
20	349	0.0271	0.0517	0.0296	0.0553	0.107	0.334
21	365	0.0273	0.0516	0.0296	0.0593	0.107	0.294
22	385	0.0269	0.0512	0.0296	0.0553	0.107	0.293
23	410	0.0272	0.0519	0.0316	0.0613	0.108	0.296
24	428	0.0270	0.0512	0.0316	0.0534	0.107	0.294
25	443	0.0273	0.0516	0.0316	0.0553	0.108	0.295
26	461	0.0264	0.0509	0.0316	0.0534	0.106	0.293
27	481	0.0259	0.0498	0.0277	0.0553	0.104	0.251
28	500	0.0259	0.0504	0.0296	0.0593	0.104	0.253
29	514	0.0260	0.0501	0.0277	0.0613	0.104	0.252
30	533	0.0258	0.0495	0.0296	0.0593	0.103	0.249
31	553	0.0255	0.0494	0.0277	0.0573	0.103	0.248
32	573	0.0257	0.0498	0.0296	0.0573	0.104	0.249
33	590	0.0259	0.0500	0.0277	0.0553	0.104	0.217
34	612	0.0258	0.0503	0.0257	0.0553	0.104	0.218
35	631	0.0259	0.0503	0.0277	0.0534	0.105	0.219
36	651	0.0259	0.0503	0.0257	0.0553	0.105	0.219
37	668	0.0259	0.0501	0.0296	0.0553	0.104	0.219
38	680	0.0258	0.0496	0.0296	0.0534	0.104	0.218
39	701	0.0256	0.0495	0.0257	0.0573	0.104	0.218
40	717	0.0256	0.0495	0.0277	0.0593	0.104	0.218
41	725	0.0257	0.0494	0.0277	0.0573	0.104	0.217
42	742	0.0257	0.0492	0.0296	0.0573	0.103	0.217
43	762	0.0256	0.0492	0.0277	0.0573	0.103	0.210
44	780	0.0256	0.0493	0.0277	0.0553	0.103	0.211
45	797	0.0257	0.0494	0.0296	0.0553	0.104	0.211
46	820	0.0259	0.0497	0.0316	0.0553	0.104	0.213
47	843	0.0256	0.0494	0.0316	0.0553	0.103	0.212
48	863	0.0255	0.0495	0.0316	0.0553	0.103	0.213
49	884	0.0253	0.0490	0.0296	0.0593	0.103	0.211
50	898	0.0252	0.0488	0.0296	0.0593	0.102	0.210

Fit Statistics							
# of Trees	# of Leaves	ASE (Tr)	ASE (OB)	MisRate (Tr)	MisRate (OB)	LLoss (Tr)	LLoss (OB)
51	915	0.0251	0.0485	0.0296	0.0593	0.102	0.209
52	929	0.0252	0.0487	0.0296	0.0573	0.102	0.209
53	944	0.0251	0.0486	0.0296	0.0593	0.102	0.209
54	954	0.0254	0.0489	0.0316	0.0573	0.103	0.210
55	976	0.0255	0.0491	0.0296	0.0573	0.103	0.210
56	995	0.0255	0.0491	0.0316	0.0593	0.103	0.211
57	1016	0.0255	0.0491	0.0316	0.0593	0.103	0.211
58	1038	0.0256	0.0492	0.0316	0.0573	0.104	0.211
59	1060	0.0257	0.0493	0.0316	0.0573	0.104	0.212
60	1077	0.0258	0.0496	0.0316	0.0573	0.105	0.213
61	1095	0.0259	0.0497	0.0316	0.0593	0.104	0.213
62	1111	0.0258	0.0497	0.0336	0.0613	0.104	0.213
63	1128	0.0258	0.0496	0.0336	0.0593	0.104	0.212
64	1143	0.0259	0.0498	0.0336	0.0632	0.104	0.213
65	1154	0.0261	0.0500	0.0336	0.0632	0.105	0.214
66	1176	0.0260	0.0501	0.0316	0.0652	0.105	0.214
67	1191	0.0260	0.0501	0.0336	0.0652	0.105	0.214
68	1208	0.0260	0.0501	0.0336	0.0672	0.105	0.214
69	1233	0.0261	0.0503	0.0316	0.0692	0.106	0.215
70	1251	0.0261	0.0504	0.0316	0.0652	0.106	0.215
71	1265	0.0263	0.0506	0.0336	0.0672	0.106	0.216
72	1283	0.0264	0.0509	0.0316	0.0652	0.107	0.217
73	1297	0.0264	0.0509	0.0336	0.0632	0.107	0.217
74	1313	0.0265	0.0510	0.0356	0.0632	0.107	0.217
75	1327	0.0265	0.0510	0.0336	0.0632	0.107	0.217
76	1342	0.0266	0.0510	0.0356	0.0652	0.107	0.217
77	1361	0.0266	0.0511	0.0336	0.0652	0.107	0.218
78	1377	0.0267	0.0512	0.0336	0.0652	0.107	0.218
79	1397	0.0267	0.0513	0.0336	0.0652	0.108	0.219
80	1426	0.0265	0.0513	0.0316	0.0652	0.107	0.219
81	1445	0.0266	0.0514	0.0316	0.0652	0.108	0.220
82	1467	0.0266	0.0514	0.0316	0.0652	0.108	0.220

Fit Statistics							
# of Trees	# of Leaves	ASE (Tr)	ASE (OB)	MisRate (Tr)	MisRate (OB)	LLoss (Tr)	LLoss (OB)
83	1487	0.0267	0.0517	0.0316	0.0652	0.109	0.221
84	1507	0.0266	0.0515	0.0316	0.0652	0.108	0.221
85	1525	0.0264	0.0512	0.0316	0.0652	0.108	0.220
86	1542	0.0265	0.0512	0.0316	0.0652	0.108	0.220
87	1557	0.0264	0.0510	0.0316	0.0652	0.108	0.220
88	1576	0.0264	0.0511	0.0316	0.0652	0.108	0.220
89	1591	0.0264	0.0510	0.0316	0.0652	0.108	0.220
90	1610	0.0264	0.0511	0.0316	0.0652	0.108	0.220
91	1634	0.0263	0.0509	0.0316	0.0652	0.107	0.219
92	1658	0.0261	0.0509	0.0316	0.0632	0.107	0.219
93	1682	0.0261	0.0510	0.0316	0.0632	0.108	0.220
94	1699	0.0261	0.0509	0.0316	0.0632	0.108	0.220
95	1716	0.0261	0.0510	0.0316	0.0632	0.108	0.220
96	1734	0.0262	0.0509	0.0316	0.0632	0.108	0.220
97	1751	0.0261	0.0508	0.0296	0.0632	0.108	0.219
98	1767	0.0261	0.0506	0.0296	0.0632	0.107	0.218
99	1789	0.0260	0.0505	0.0296	0.0632	0.107	0.218
100	1807	0.0260	0.0505	0.0316	0.0613	0.107	0.218

Loss Reduction Variable Importance					
Variable	Number of Rules	Gini	OOB Gini	Margin	OOB Margin
rooms	344	0.135221	0.09314	0.270441	0.226259
lstat	243	0.077310	0.03852	0.154621	0.114238
indus	147	0.032876	0.01458	0.065752	0.047864
tax	190	0.030433	0.00689	0.060866	0.038401
pt	114	0.017459	0.00661	0.034918	0.023446
chas	14	0.000905	−0.00012	0.001810	0.001140
radial	87	0.008481	−0.00193	0.016962	0.008502
zn	55	0.005667	−0.00213	0.011334	0.003600
nox	135	0.015034	−0.00389	0.030068	0.012000
crim	127	0.010873	−0.00603	0.021746	0.004207
age	95	0.010292	−0.00635	0.020583	0.003816

Loss Reduction Variable Importance					
Variable	**Number of Rules**	**Gini**	**OOB Gini**	**Margin**	**OOB Margin**
distance	156	0.017765	−0.00790	0.035530	0.010438