Chapter 8 Tree-Based Methods

1. Involve stratifying or segmenting the predictor space in to a number of simple regions🡪
2. In order to make predictions for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs
3. The set of splitting rules used to segment the predictor space can be summarized in a tree
4. Combining a large number of trees can often result in dramatic improvement in prediction accuracy, at the expense of some loss in interpretation

8.1 The Basics of Decision Trees

8.1.1 Regression Trees

Terminal nodes/leaves: Region in which the tree stratifies the sample into

* Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree
* Internal nodes: The points along the tree where the predictor space is split
* Branches: Segments of the trees that connect the nodes
* Regression Tree is easier to interpret, and has a nice graphical representation

Prediction via Stratification of the Feature Space:

1. Divide the predictor space-that is, the set of possible values for into distinct and non-overlapping regions,
   1. Divide the predictor space into high-dimensional rectangles, or boxes
   2. The goal is to find boxes that minimize the RSS

A picture containing clock

Description automatically generated

Where is the mean response for the training observation within the jth box

* 1. Recursive Binary Splitting: Begins at the top of tree, and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
     1. At each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a spilt that will lead to a better tree in some future step
  2. Find the predictor and cut point 

Such that minimize the equation,

A picture containing clock

Description automatically generated

* 1. The process continues until a stopping criterion is reached

1. For every observation that falls into the region , we make the same prediction, which is simply the mean of the response values for the training observations in .

Tree Pruning

Grow a very large tree , and then prune it back in order to obtain a subtree,

Cost Complexity Pruning(Weakest Link Pruning):

1. Consider a sequence of trees indexed by a nonnegative tunning parameter
   1. For each value of there corresponds a subtree such that

A close up of a logo

Description automatically generated

is as small as possible

is the number of terminal nodes, and is the rectangle corresponding to the mth terminal node, is the predicted response associated with

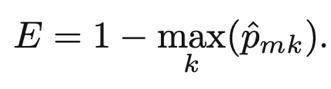
* 1. controls the trade off between the bias-variance trade off🡪Lower is a more flexible fit
  2. We can select using cross validation and applying the lowest test error to the whole data set

A screenshot of a cell phone

Description automatically generated

8.1.2 Classification Trees

1. Each observation belongs to the most commonly occurring class of training observations in the region to which it belongs
2. We often interested in both the prediction corresponding to a particular terminal node region, but also in the class proportions among the training observations that fall into that region.
3. Classification error rate🡪The fraction of the training observations in that region that do not belong to the most common class



Where represents the proportion of training observations in the mth region that are from the kth class

1. Gini Index: A measure of total variance across K classes

A picture containing object, clock

Description automatically generated

A measure of node purity—A small value indicates that a node contains predominantly observations from a single class

1. Entropy: Similar to Gini index

A picture containing object, clock

Description automatically generated

Entropy will take on a small value if the mth node is pure

1. Both Gini and Entropy are more sensitive to node purity than the classification error rate, but classification error rate is preferable if the prediction accuracy of the final pruned tree is the goal.
2. Some of the splits yield two terminal nodes that have the same predicted values
   1. The split is performed because it leads to increased node purity.

8.1.3 Trees Versus Linear Models

The relative performances of tree-based and classical approaches can be assessed by estimating the test error, using either cross-validation or the validation set approach

8.1.4 Advantages and Disadvantages of Trees

Advantages:

1. Trees are easy to explain to people
2. Decision trees more closely mirror human decision-making than do regression and classification approaches
3. Trees can be displayed graphically, and are easily interpreted
4. Trees can easily handle qualitative predictors without the need to create dummy variables

Disadvantages:

1. Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches
2. Trees can be very non-robust

8.2 Bagging, Random Forests, Boosting

8.2.1 Bagging

Averaging a set of observations reduces variance🡺

1. Take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions

A picture containing clock

Description automatically generated

1. Bagging: Taking repeated samples from the (single) training data set🡪Generate B different bootstrapped training sets🡪 Train our method on the bth bootstrapped training set to obtain the model🡪Average all predictions

A picture containing clock

Description automatically generated

Bagging in Continuous Setting

1. Construct B regression trees using B bootsrapped training sets, and average the resulting predictions
2. Each tree has high variance, but low bias
3. Average reduces the variance

Bagging in Classification Setting

1. For a given test observation, we can record the class predicted by each of the B trees, and take a majority vote:
   1. the overall prediction is the most commonly occurring class among the B predictions

The number of trees B is not a critical parameter with bagging; using a very large value B will not lead to overfitting.

Out-of-Bag Error Estimation

Each bagged tree make use of around two-thirds of the observations.

Out-of-bag observations: The remaining one-third of the observations not used to fit a given bagged tree

1. An OOB prediction can be obtained for each of the n observations, for which the overall OOB MSE or classification error can be computed
2. The resulting OOB error is a valid estimate o the test error.

Variable Importance Measures

Bagging improves predictions accuracy at the expense of interpretability

1. We can obtain an overall summary of the importance of each predictor using RSS or the Gini index
   1. RSS: Record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees
   2. Gini index: Add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees

8.2.2 Random Forests

1. At each split, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

2. The split is allowed to use only one of those m predictors.

\* The number of predictors considered at each split is approximately equal to the square root of the total number of predictors🡪The algorithm is not even allowed to consider a majority of the available predictors

\*Decorrelating the tree: Making the average of the resulting trees less variable and hence more reliable

3. The main difference between bagging and random forests is the choice of predictor subset size m.

4. Using a small value of m in building a random forest will typically be helpful when we have a large number of predictors.

\*Random forests will not overfit if we increase B.

8.2.3 Boosting

Works similar to bagging, except that trees are grown sequentially:

1. Each tree is grown using information from previously grown trees.
2. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set.

For a large number of decision trees,

A screenshot of a cell phone

Description automatically generated

Idea behind boosting:  
1. The boosting approach learns slowly🡪 We fit a decision tree to the residuals from the model, rather than the outcome Y, as the response.

1. Each tree can be quiet small, with just a few terminal nodes, determined by the parameter d (splits)
2. The shrinkage parameter slows the process down even further, allowing more and different shaped trees to attack the residuals
3. In boosting, the construction of each tree depends strongly on the trees that have already been grown.

Boosting Parameters:

1. Number of trees B Can result in overfitting if B is too large, although this overfitting tends to occur slowly🡪Use cross-validation to select B
2. Shrinkage parameter, (between 0.01 or 0.001)🡪Controls the rate at which boosting learns, very small can require using a very large B in order to achieve good performance
3. The number of splits, d, in each tree: Controls the complexity of the boosted ensemble.
   1. D=1, each tree is a stump, consisting of a single split🡪The boosted ensemble is fitting an additive model, since each term involves only a single variable
   2. D is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables
4. Boosting vs. Random Forests:
   1. In boosting, because the growth of a particular tree takes into account the other trees that have already been grown, smaller trees are typically sufficient🡪 Aid the interpretability

8.3 Lab: Decision Trees

8.3.1 Fitting Classification Trees

Tree Package  
Create a new variable, called High, which takes on a value of Yes if the Sales variable exceeds 8. And takes on a value of No otherwise

A picture containing table

Description automatically generated

Ifelse(): ternary operator

Data.frame(): merge High with the rest of Carseats data



Tree(): syntax similar lm function,



A screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated

Deviance:

A picture containing object, clock, drawing

Description automatically generated

1. A small deviance indicates a tree that provides a good fit to the training data
2. Residual Mean Deviance: Deviance dived by

A close up of a logo

Description automatically generated

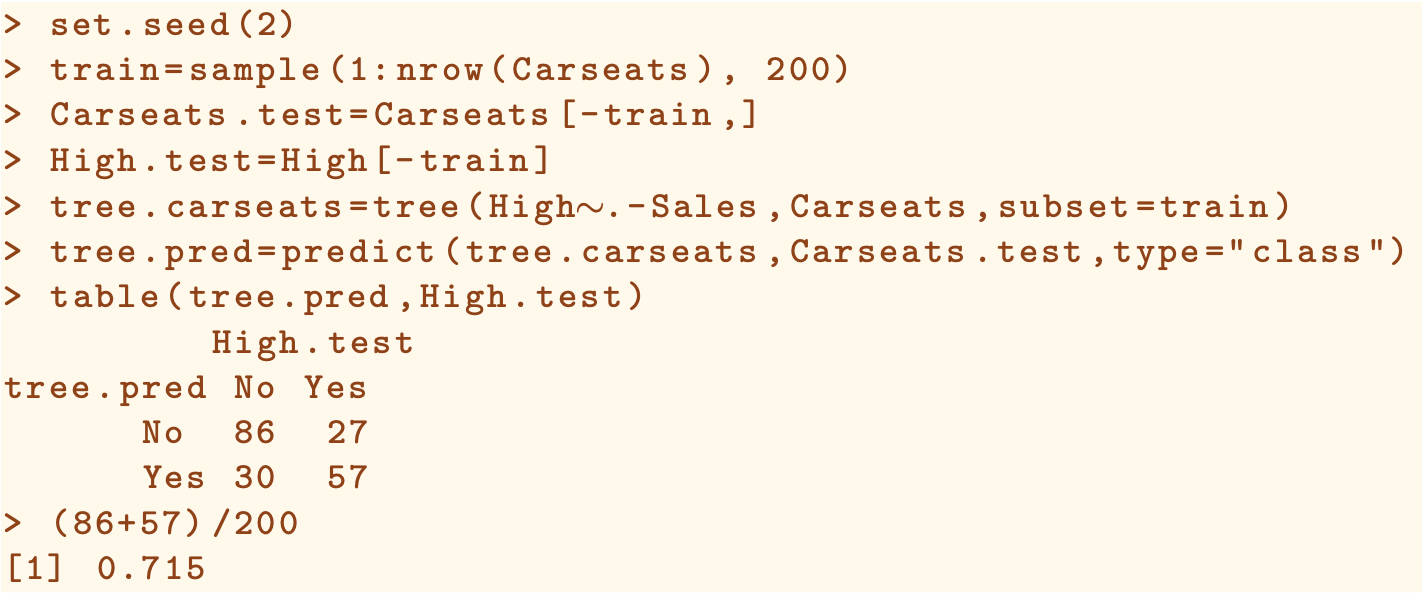
Plot() displays the tree structure, text() function to display the node labels, and pretty=0 instructs R to include the category names for any qualitative predictors

A screenshot of a cell phone

Description automatically generated

1. Prints output corresponding to each branch of the tree
2. Displays the split the criterion, the number of observations in that branch, the deviance, the overall prediction for the branch, and the fraction of observation in that branch
3. Branches that lead to terminal nodes are indicated using asterisks

Predict(): to find the test error



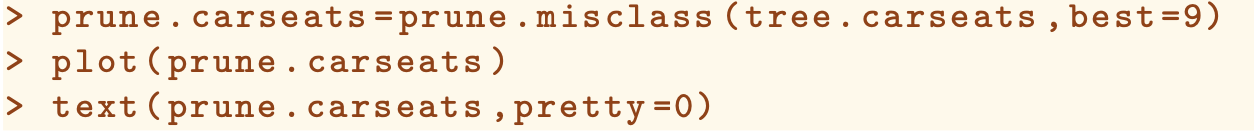
cv.tree(): Perform cross-validation in order to determine the optimal level of tree complexity; cost complexity pruning is used in order to select a sequence of trees for considerations.

1. Fun=prune.misclass: Indicate that we want the classification error rate to guide the cross-validation and pruning process
2. Cv.tree() function reports the number of terminal nodes of each tree considered(size) as well as the corresponding error rate and the value of the cost-complexity parameter used(k). dev is the cross-validation error of the instance

A screenshot of a social media post

Description automatically generated

Prune the tree



8.3.2 Fitting Regression Trees

Rpart() function: Syntax similar to lm(), from rpart package

A screenshot of a cell phone

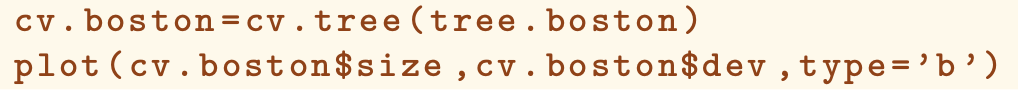
Description automatically generated

A screenshot of a cell phone

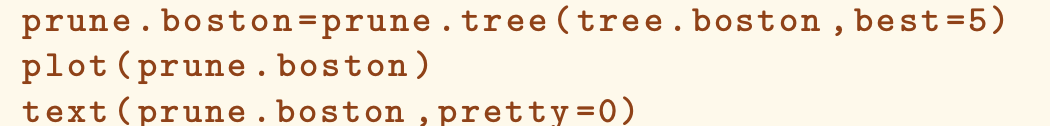
Description automatically generated

Deviance in the regression tree context is the sum of squared errors

cv.tree() function can be used to get the cross validation error



Prune.tree() function can be used to prune the tree



8.3.3 Bagging and Random Forests

Bagging is simply a special case of a random forest with m=p🡪 RandomForest() function can be used to perform both random forests and bagging

Bagging:

A screenshot of a social media post

Description automatically generated

1. Mtry=13, all predictors are considered🡪Bagging
2. We chould change the number of trees grown by randomForest() using the ntree argument

A picture containing knife, table

Description automatically generated

Random ForestL

Use a smaller value in the mtry argument. By default, randomForest() uses variables when building a random forest od regression trees, and variables when building a random forest of classification trees

A picture containing bird

Description automatically generated

Importance(): view the importance of each variable

A close up of text on a white background

Description automatically generated

1. Mean decrease of accuracy in predictions on the out of the bag samples when a given variable is excluded from the model
2. Total decrease in node impurity that results from the splits over that variable🡪
   1. Regression: node impurity is measured by the training RSS
   2. Classification: by the deviance

varImpPlot(): plot the importance measures

A close up of text on a white background

Description automatically generated

8.3.4 Boosting

Gbm package:

Gbm() function:

1. Distribution=”gaussion” for regression and distribution=”bernoulli” for binary classification
2. Syntax similar to random forest
3. N.trees indicates the total number of tree
4. Interaction.depth limits the depth of each tree

A picture containing knife

Description automatically generated

Summary() function produces a relative influence plot and also outputs the relative influence statistics

A screenshot of text

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Partial Dependene plots():

Illustrate the marginal effect of the selected variables on the response after integrating out the other variables

A picture containing table

Description automatically generatedA close up of a map

Description automatically generated

Predict(): use the fitted boost model on the test set.

A picture containing knife, table

Description automatically generated

To adjust shrinkage parameter:

Shrinkage=…L argument

A screenshot of a cell phone

Description automatically generated