Prof. Dr. Steffen Wagner
Angewandte Statistik
Fachbereich II
Berliner Hochschule für Technik

Exercise 09: Regression Trees

Machine Learning I – SoSe 24

This workshop covers hierarchical clustering and soft clustering. At the end of the worksheet there are a couple of written exercises for you to do at home, which should be good practice for the exam.

1 Preparations

1.1 RStudio Project

1. Open your Machine Learning 1 RStudio Project
2. Create an R Script file to perform this exercise

1.2 Required Packages

For this exercise you require the following additional R packages. Please make sure that you have installed them on your computer before coming to the workshop session for the case that Eduroam is not working.

```
# check if packages can be loaded, i.e. they are already installed
library(ISLR2)        # contains data sets we work with
library(rpart)        # package for tree fitting
library(tree)         # package for tree fitting used by James et all
library(partykit)     # visualisation of tree structures
library(rpart.plot)   # visualisation of tree structures
```

If you get an error at this stage, you need to install the packages.

2 Regression trees with R

2.1 Boston data set

You will fit a regression tree to the Boston Data, a classic data set in the field of statistical learning. The data are part of the ISLR2 package from James et al. To access the data set, first load the package and read about the data set using

```
library(ISLR2)
?Boston
```

2.1.1 Explorative data analysis

As usual it is a good idea to understand the basics of the data using descriptive statistics before applying statistical learning methods.

The following questions are based on those asked in James *et al.*~Page 57. Use R to answer them.

1. How many rows are in this data set? Each row represents a different suburb in Boston, Massachusetts.
2. How many columns are there? Use the function `names()` to output the the variable names.
3. The variable `medv` is our outcome variable used in the regression tree. This is the median value of owner occupied homes in each Boston suburb in $1000s. Obtain the `summary` Statistics and a boxplot for `medv`.
4. Obtain a scatter plot for `medv` against each of the other (predictor) variables in the data set. Are there any which have a clear association with `medv`.
5. How many of the suburbs in this data set border the Charles river? Obtain a boxplot of `medv` split by this variable?
6. What is the median pupil-teacher ratio among the suburbs in this data set?
7. Which suburb of Boston has lowest median value of owner occupied homes (`medv`)? What are the values of the other predictors for that suburb, and how do those values compare to the overall ranges for those predictors?
8. In this data set, how many of the suburbs average more than seven rooms per dwelling? More than eight rooms per dwelling?

```
data("Boston")

# Q1:  ----
nrow(Boston)

# Q2:  ----
ncol(Boston)
names(Boston)

# Q3:  ----
summary(Boston$medv)
boxplot(Boston$medv, horizontal = TRUE)

# Q4:  ----
plot(medv ~ ., data = Boston)
```

```r
# Q5:  ----
table(Boston$chas)
boxplot(medv ~ chas, data = Boston, horizontal = TRUE, varwidth = TRUE)

# Q6:  ----
summary(Boston$ptratio)

# Q7:  ----
Boston[which.min(Boston$medv), ]
round(colMeans(Boston), 3)

# Q8:  ----
table(Boston$rm > 7)
table(Boston$rm > 8)
```

2.1.2 Regression Tree Model

```
library(rpart)
library(rpart.plot)

set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/2)

tree.boston <- rpart(medv~., data = Boston, subset = train)
print(tree.boston)
```

You will now fit a regression tree to the Boston data using `medv` as the outcome variable. Remember that a regression trees is used when the outcome variable is continuous. The majority of this exercise follows *Lab 8.3.2* in James et al. (pp. 356) but using the `rpart` package instead of the `tree` package.

- Start by loading the `rpart` libraries, creating a training set, and fitting the default tree to the training data.

  ```
  library(rpart)
  library(rpart.plot)

  set.seed(1)
  train <- sample(1:nrow(Boston), nrow(Boston)/2)

  tree.boston <- rpart(medv~., data = Boston, subset = train)
  print(tree.boston)
  ```

  Notice that the output indicates that only four of the variables have been used in constructing the tree. In the context of a regression tree, the deviance is simply the sum of squared errors for that node.

- Now plot the tree using `rpart.plot(tree.boston)`. Interpret the results. Do they look reasonable?

- What is the median house price for a flat located in a suburb with `rm = 6` and `lstat = 12`?

- We will now consider pruning the tree. The best approach is to fit the full tree with the argument `cp = 0`, and then look at the cross validation assessment of the complexity parameter using `printcp()` and `plotcp()`.

  ```
  set.seed(1)
  tree.boston.full <- rpart(medv ~ . , data = Boston, subset = train, cp = 0)
  print(tree.boston.full)
  rpart.plot(tree.boston.full)
  ```

  ```
  # Table with cp values from cross validation
  cptable <- tree.boston.full$cptable
  plotcp(tree.boston.full)
  ```

  The full tree is quite a bit more complex than the default tree. The pruning rule suggested by the authors of the `rpart` package is to choose the smallest number of nodes

(largest cp value) which lies within 1 standard deviation of the cross-validation smallest deviance, i.e. lies below the dotted line, so compare `xerror` with `xerror+xstd` for the last row.

```
minDeviance <- which.min(cptable[ , "xerror"])
dotted <- cptable[minDeviance, 4] + cptable[minDeviance, 5]
abline(h = dotted, col = "red", lty = 2)
# which is the first row less than this value
cpPruning <- cptable[cptable[, "xerror"] < ???, ][1, ]
cpPruning
```

- So this method suggests a tree with 5 splits/6 nodes. Note that the cp values in the matrix do not exactly match those in the plot. A true graph of the xerror is a step function and the values in the matrix indicate the step points. The cp values in the plot are the mid-point of each step. I recommend reading the relevant cp-value from the `cptable`, which avoids a possible error due to rounding.

```
prune.boston <- prune(tree.boston, cp = ???)
prune.boston
rpart.plot(prune.boston)
```

- Compute and compare the mean square error (MSE) for the full, default and pruned tree.

```
# on training data
# full tree
pred.train.full<-predict(tree.boston.full, newdata = Boston[train,])
mean((Boston$medv[train]-pred.train.full)^2)

## default tree
pred.train.default<-predict(tree.boston, newdata = Boston[train,])
mean((Boston$medv[train]-pred.train.default)^2)

## pruned tree
pred.train.pruned<-predict(prune.boston, newdata = Boston[train,])
mean((Boston$medv[train]-pred.train.pruned)^2)
```

- Obtain the predictions for the full tree applied to the test data and for the the pruned tree applied to the test data. Calculate the MSE in both cases.

```
### on the test data
# full tree
pred.test.full <- predict(tree.boston.full, newdata = Boston[-train, ])
mean((Boston$medv[-train] - pred.test.full)^2)

# default tree
pred.test.default <- predict(tree.boston,newdata=Boston[-train,])
mean((Boston$medv[-train]-pred.test.default)^2)

# pruned tree
pred.test.pruned<-predict(prune.boston,newdata=Boston[-train,])
```

```
mean((Boston$medv[-train]-pred.test.pruned)^2)
```

The test set MSE associated with the pruned regression tree is 35.2. The square root of the MSE is therefore a bit less than 6, indicating that this model leads to test predictions that are within around $5,900 of the true median home value for each suburb.

- Plot the observed median values `medv` against the pruned tree predictions (test data).

```
boston.test <- Boston[-train, "medv"]
plot(pred.test.pruned, boston.test)
abline(c(0,1), col = "red", lty = 2)
```

2.2 Car seat data

The `ISLR2` package also includes a data set on the sales of child car seats called `Carseats`. Use what you learnt dealing with the Boston data set to find a regression tree for car seat sales.

1. Load this package and investigate the `Carseats` data in a similar way to Exercise 2.1.1.
2. Split the data into a 50-50 split for training and testing. Use `set.seed(1)` to work with a reproducible train-test-split.
3. Obtain the `rpart` default regression tree for the outcome variable `sales`, and plot it.
4. Find an acceptable pruned tree and plot it.
5. The first row in the dataset has the following values:

```
Carseats[1,]
```

```
  Sales CompPrice Income Advertising Population Price ShelveLoc Age Education Urban  US
1   9.5       138     73          11        276   120       Bad  42        17   Yes Yes
```

   Using the tree diagram for the pruned tree find the predicted value for Sales. Obtain the predicted values for the training data and check your answer with the first predicted value.
6. Which leaf node has the most elements? Summarise the decision rules which lead to this leaf node and give it's predicted value.
7. Repeat the analysis with a differest random seed, e.g. `set.seed(1234567890)`. What do you observe?

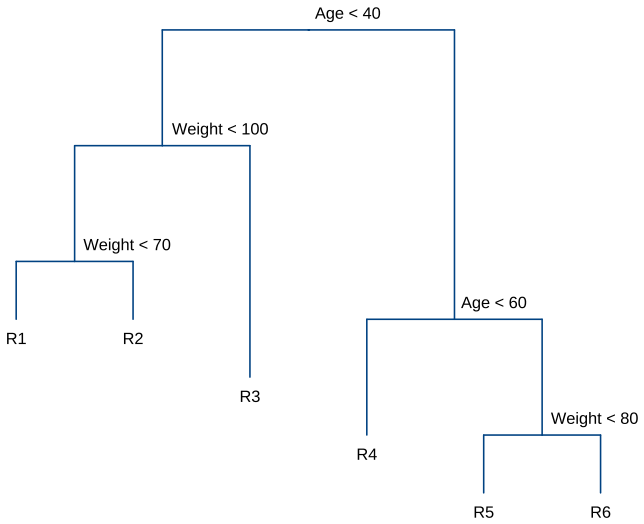
3 Written Exercise

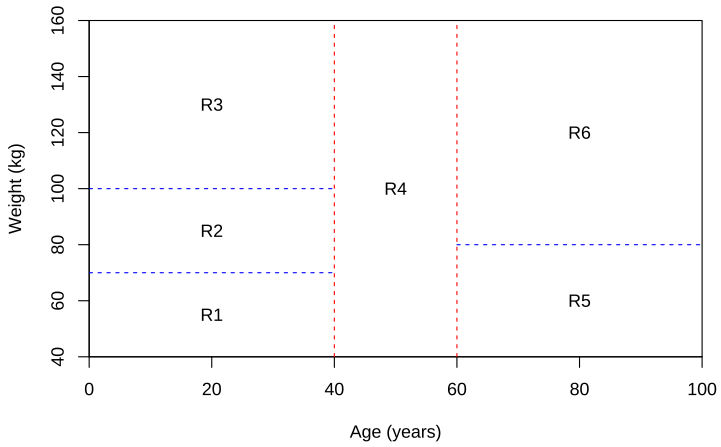
3.1 Imagine a tree

Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions $R_1, R_2, \ldots, R_6, \ldots$ and the cutpoints $t_1, t_2, \ldots$ and so forth.

Your result should look something like Figures on slides 12 and 14 in the slide deck covering regression trees.

Solution: a possible example





3.2 From Feature Space to Tree and vice versa

Work through Exercise 4 in James et al. on page 362.

Solution: