Machine Learning 2
Data Science
Winter Semester 2018/19

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Prof. Tim Downie

Workshop 1 — 8th October 2018
Tree Models: Revision

At the end of last semester you learnt about regression trees and classification trees. In the Workshop today you will work through a couple more exercises on tree models. The motivation is partly for revision and partly as preparation for the lecture on *ensemble methods* this afternoon.

Exercise 1 Regression tree

In James, Witten, Hastie and Tibshirani (2013) *An Introduction to Statistical Learning with Applications in R.* `http://www-bcf.usc.edu/~gareth/ISL/index.html` The *Boston* data set contains data on *housing values and other information about Boston suburbs*.

► Start R or R-Studio open a new script file and set your working directory.

► The data are part of the MASS package[1]. To access the data set, first load the package and read about the data set using

```
library(MASS)
?Boston
```

You have not yet come across these data and it is always a good idea to understand the type of data before applying any kind of statistical learning method. Use R to answer the following questions.

(a) How many rows are in this data set? Each row represents a different suburb in Boston, Massachusetts.
How many columns are there? Enter the A command whih outputs the the variable names.

(b) Make some pairwise scatterplots of the predictors (columns) in this data set. Describe your findings.

(c) Are any of the predictors associated with per capita crime rate?

(d) Do any of the suburbs of Boston appear to have particularly high crime rates? Tax rates? Pupil-teacher ratios?

(e) How many of the suburbs in this data set bound the Charles river?

[1] Modern applied statistics with S, Venables and Ripley

(f) What is the median pupil-teacher ratio among the towns in this data set?

(g) Which suburb of Boston has lowest median value of owner occupied homes? What are the values of the other predictors for that suburb, and how do those values compare to the overall ranges for those predictors ?

(h) In this data set, how many of the suburbs average more than seven rooms per dwelling? More than eight rooms per dwelling? Comment on the suburbs that average more than eight rooms per dwelling.

You will now fit a regression tree to the Boston data using `medv` (median value of owner-occupied homes in $1000s) 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. but using the `rpart` package instead of the `tree` package.

► Start by loading the `rpart` libraries, creating a training set, and fitting the 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~.,Boston,subset=train)
print(tree.boston)
```

Notice that the output indicates that only three 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 the tree.

► Now plot the tree.

```
rpart.plot(tree.boston)
```

The variable `lstat` measures the percentage of individuals with lower socioeconomic status. The tree indicates that lower values of `lstat` correspond to more expensive houses. The tree predicts a median house price of $46 400 for larger homes in suburbs in which residents have high socioeconomic status (`rm`>=7.437 and `lstat`<9.715).

► Use the `printcp()` and `plotcp()` function to see whether pruning the tree will improve performance.

```
printcp(tree.boston)
plotcp(tree.boston)
```

The 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 smallest deviance, i.e. lies below the dotted line.

```
prune.boston=prune(tree.boston,cp=0.016)
prune.boston
```

```
rpart.plot(prune.boston)
```

► Compare the mean square error (MSE) for the unpruned and pruned tree.

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

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

Notice that for the test data the MSE for the pruned tree is only a little larger than for the unpruned tree, but we have gained a slightly simpler model.

Comment on the MSE. The term *mean square error* (MSE) is used for slightly different things, and here we are using it i) for the MSE used in the fitting algorithm (training data) and ii) as an estimate of the true MSE by using the test data. A more detailed explanation will be given in the Lecture.

The test set MSE associated with the regression tree is 25.82. The square root of the MSE is therefore around 5.08, indicating that this model leads to test predictions that are within around \$5 080 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,boston.test)
abline(c(0,1))
```

Exercise 2 Classification tree: prostate cancer

Classification models are used when the output variable is takes a small number of distinct levels known as classes.

A data set called `stagec` contains information about patients with "stage C" (advanced) prostate cancer is provided with the `rpart` package.

► Read the help page for this data set. How many patients are in this data set?

The main clinical endpoint of interest is whether the disease recurs after initial surgical removal of the prostate. The variable `pgstat` (for progression status) takes the value 1 if the disease has progressed and 0 if not.

```
table(stagec$pgstat)
```

► Lets code this variable sa a factor variable, so that *no progression* has the label "No" and `progression` has the label "Prog".

```
stagec$progstat <- factor(stagec$pgstat, levels = 0:1, labels = c("No", "Prog"))
```

This makes reading the output easier and `rpart` will recognise that the outcome variable is a factor variable and so will use the Gini coefficient to calculate the loss statistic, in order to determine each splits. As `pgstat` is numeric `rpart` would assume that a regression tree is wanted, and will use mean square error for the loss function.

Note that `stagec` has a variable `pgtime` which encodes the time at which *progression* was first observed. These data could be modelled using *survival analysis* methods. Survival analysis is an area of statistics used to model the "time to a specific event". Many studies take death as the event and the probability of being event free after a specific time is called the survival function, which is why the field is called survival analysis regardless of what the event actually is.

► Plot `progstat` against the other variables (ignoring `pgtime`). If the other variable is continuous then produce a boxplot, if the other variable is a factor or is discrete numeric with only a few levels then obtain a contingency table and a bar-chart. Example commands are given below.

```
plot(xxx~progstat,data=stagec )
barplot(table(stagec$xxx,stagec$progstat),beside=TRUE,legend.text=TRUE )
```

► As this data set has too few observations for a sensible test data set, we will use the whole data for the training set.

► Fit the full tree and output it as text and a diagram.

```
c.tree  <- rpart(progstat ~  age + eet + g2 + grade + gleason + ploidy,
                 data = stagec)
rpart.plot(c.tree)
print(c.tree)
```

► As with the regression tree we should look to see if pruning the tree is better.

```
printcp(c.tree)
plotcp(c.tree)
c.pruned<-prune(c.tree,cp=???)
print(c.pruned)
rpart.plot(c.pruned)
```

► We will use a classifier with $\alpha = 0.5$, i.e. the mostlikely of the two outcomes is predicted

```
stagec$predict<-(predict(c.pruned)[,2]>??)
table(stagec$progstat,stagec$predict)
tt<-table(stagec$progstat,stagec$predict)
sens<-tt[2,2]/sum(tt[2,]);sens
spec<-tt[1,1]/sum(tt[1,]);spec
```

► Comment on the value of the specificity.

► The following code uses the `ROCR` package, to produce the ROC diagram and the AUC.

```
library(ROCR)
p <- predict(c.pruned)[,2]
#rpart function to get the prediction for Yes
pr <- prediction(p, stagec$progstat)  #convert the predictions into ROCR format
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
#ROCR function calculates everything for the ROC curve
plot(prf) #plot the ROC curve
abline(c(0,1))
AUC<-performance(pr, measure ="auc")@y.values[[1]];AUC
#RORC function calculates the AUC
```

► Use your values of the specificity and sensitivity in the previous part to find where on the ROC curve this classifier sits.