Introduction to classification and regression trees, random forests and model-based recursive partitioning in R

Day 1: Single trees

Exercise 0: Illustrating variable selection bias

Using function `rnorm()`, generate three standard normally distributed (that is, $\mu = 0$ and $\sigma = 1$) variables `x1`, `x2` and `y`, each with 100 observations. Round the values of `x2` to 0 decimal places. Note that `x1` now has much more levels than `x2`. Fit two trees that aim to predict `y` from `x1` and `x2`: one using the `tree()` function and one using the `ctree()` function.

Did the algorithms create spurious splits? If so, were `x1` and `x2` equally often selected for splitting, or was there a selection bias towards `x1`? Repeat the sampling and fitting process ten times. Do you seem the same pattern every time?

Exercise 1: Impurity measures

We have a root node, in which a proportion of .7999 observations belong to class 1. One of the splitting candidates is a categorical variable with two levels. If this variable would be used to split the observations, a proportion of .565 would go left, and a proportion of .435 would go right. The proportion of class 1 observations in the left node would be 1.0. The proportion of class 1 observations in the right node would be .54.

Would this split improve purity according to the Gini index? According to the Shannon entropy? According to the classification error?

Exercise 2: Predicting carseat sales

The Carseats dataset from the package ISLR contains data of child car seats sales in 400 different stores:

```
data("Carseats", package = "ISLR")
summary(Carseats)
?Carseats
```

Our aim is to predict car seat sales using the other ten predictor variables in the dataset. Note that we do not need to specify all ten predictor variables, `Sales ~ .` would instruct R to regress Sales on all the remaining variables in the dataset.

a) Randomly separate the dataset in 300 training and 100 test observations (don't forget to set the random seed!)

b) Fit and plot a CART tree using the `tree()` function.

c) Obtain the optimal complexity parameter value using the `cv.tree()` function and prune the tree using the `prune.tree()` function.

d) Fit and plot a tree using the `ctree()` function.

e) Do the two trees look alike? What seems to be the most important predictor of carseat sales? Do the two trees agree on this?

f) Using the `predict()` function, generate predictions for the test observations. For the ctree, the pruned CART tree and the unpruned CART tree, calculate the correlation with the true car seat sales in the test data. Did pruning the CART tree indeed improve predictive accuracy?

Exercise 3: Logistic Regression Based Recursive Partitioning

For this exercise, we'll use the Pima Indians Diabetes dataset. We will partition the dataset based on a logistic regression model, predicting the probability of having diabetes based on the outcome of a glucose tolerance test.

```
data("PimaIndiansDiabetes", package = "mlbench")
?PimaIndiansDiabetes
```

Use the function `glmtree()` from package `partykit`. Regress the diabetes variable on the glucose variable and specify all other variables as potential partitioning variables. Add the argument `family = binomial` to specify a logistic regression model. Plot the tree and interpret the results.

Exercise 4: Predicting Glaucoma

Load the `Glaucoma' dataset from the package`TH.data‘:

```
data("GlaucomaM", package="TH.data")
?GlaucomaM
```

The data set contains 62 continuous predictor variables. The response is the binary factor `GlaucomaM$Class`.

a) Fit a classification tree with the `ctree` function.

b) Plot the tree. Does it indicate main and/or interaction effects? Of which variables?

Day 2: Stability and tree ensembles

Exercise 5: Glaucoma data revisited

a) Use the `stabletree()` function from the `stablelearner` package to assess stability of the tree you fitted yesterday. Do not forget to set the random seed, as the `stabletree()` function uses bootstrap samples to assess stability. Use (at least) the `summary()` function for inspecting the result.

b) Is the first variable in the tree grown in Exercise 4 often selected? And how about the variables that appear further down the tree?

c) Check whether there is multicollinearity in the data: Could that explain that the variables appearing in the tree are not selected in each of the bootstrap samples?

d) Fit a random forest using the `cforest` function from package `party`*, with either bootsrap sampling with size N or subsampling with size $.632 \cdot N$.

e) Using the `predict()` function, compute the percentage of correctly classified observations — based on all trees and based only on those trees for which the observations are out-of-bag (by adding `OOB = FALSE` or `OOB = TRUE` to the function call) — and compare your results. Determine the predicted response class of each case and create a confusion matrix.

f) Compute and plot the standard permutation importance for the predictor variables.

Set random seeds everywhere in the code where random sampling is involved.

* You will compute variable importances in the next part, which is only supported in package `party`, not in package `partykit`. So you will first have to detach the latter package and all packages that depend on it and afterwards load package `party`. This may seem like a detour, but there is currently no other way to get the variable importances:

```
detach("package:stablelearner", unload = TRUE)
detach("package:glmertree", unload = TRUE)
detach("package:partykit", unload = TRUE)
```

Exercise 6: Carseat sales revisited

Using the `cforest()` function, fit a random forest and a bagged ensemble to the 300 training observations. Using the `predict()` function, calculate the mean squared error (MSE) for:

a) The training observations
b) The OOB predictions for training observations (add `OOB = TRUE` to the function call)
c) The test observations (specify `newdata` argument)

$$MSE = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$$

d) Are the OOB training error estimates indeed less optimistic / more realistic estimates of the test error?

Exercise 7: Boston housing data

Fit a random forest and bagged ensemble of CART trees to the Boston housing data. Use the `randomForest()` function from the `randomForest` package. You could create a random forest using a formula statement, but if you supply arguments `x`, `y`, `xtest` and `ytest`, you can assess the test error and OOB error for random forests consisting of 1 through the specified number of trees to grow. Therefore, first select the training and test data:

```
library("randomForest")
?randomForest
data("Boston", package = "MASS")
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/4)
x <- Boston[train, -14]
y <- Boston[train, 14]
xtest <- Boston[-train, -14]
ytest <- Boston[-train, 14]
```

Set the `ntree` argument to 750, and create two ensembles: a bagged ensemble, using all 13 predictor variables for selecting each split, and a random forest using `sqrt(13)` for selecting each split.

Inspect the test error (it can be found in the `$test$mse` slot of the randomForest object) and the OOB error (it can be found in the `$mse` slot of the randomForest object) for both ensembles. Using the `plot()` and `lines()` functions, try to create a plot where the number of trees in the ensemble is on the x-axis, and the MSE is on the y-axis. Different lines should be drawn for the OOB and test error of the bagged ensemble, as well as the OOB and test error of the random forest.

After what tree size does the MSE start to stabilize?

Which ensemble is most accurate on the test data? Does the OOB error give a better estimate of test error for the random forest, or for the bagged ensemble? Can you explain this?