Solution Lesson05: Decision trees

Machine Learning using R

Exercise 1: Diabetes in Pima Indian women

This data set includes the test-results of women who are of Pima Indian heritage and were tested for diabetes according to Wold Health Organization criteria.

a) The `Pima.tr` data set is available in the `MASS` package. Load the package to be able to access the data. Get an overview of the data and use the command `?Pima.tr` to see the individual variables' meaning.

```
library(MASS)
head(Pima.tr)
```

```
##   npreg glu bp skin  bmi   ped age type
## 1     5  86 68   28 30.2 0.364  24   No
## 2     7 195 70   33 25.1 0.163  55  Yes
## 3     5  77 82   41 35.8 0.156  35   No
## 4     0 165 76   43 47.9 0.259  26   No
## 5     0 107 60   25 26.4 0.133  23   No
## 6     5  97 76   27 35.6 0.378  52  Yes
```

```
str(Pima.tr)
```

```
## 'data.frame':    200 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

```
dim(Pima.tr)
```

```
## [1] 200   8
```

```
summary(Pima.tr)
```

```
##      npreg            glu              bp              skin      
##  Min.   : 0.00   Min.   : 56.0   Min.   : 38.00   Min.   : 7.00  
##  1st Qu.: 1.00   1st Qu.:100.0   1st Qu.: 64.00   1st Qu.:20.75  
##  Median : 2.00   Median :120.5   Median : 70.00   Median :29.00  
##  Mean   : 3.57   Mean   :124.0   Mean   : 71.26   Mean   :29.21  
##  3rd Qu.: 6.00   3rd Qu.:144.0   3rd Qu.: 78.00   3rd Qu.:36.00  
##  Max.   :14.00   Max.   :199.0   Max.   :110.00   Max.   :99.00  
##       bmi             ped              age         type    
##  Min.   :18.20   Min.   :0.0850   Min.   :21.00   No :132  
##  1st Qu.:27.57   1st Qu.:0.2535   1st Qu.:23.00   Yes: 68
```

```
##  Median :32.80   Median :0.3725   Median :28.00
##  Mean   :32.31   Mean   :0.4608   Mean   :32.11
##  3rd Qu.:36.50   3rd Qu.:0.6160   3rd Qu.:39.25
##  Max.   :47.90   Max.   :2.2880   Max.   :63.00
```

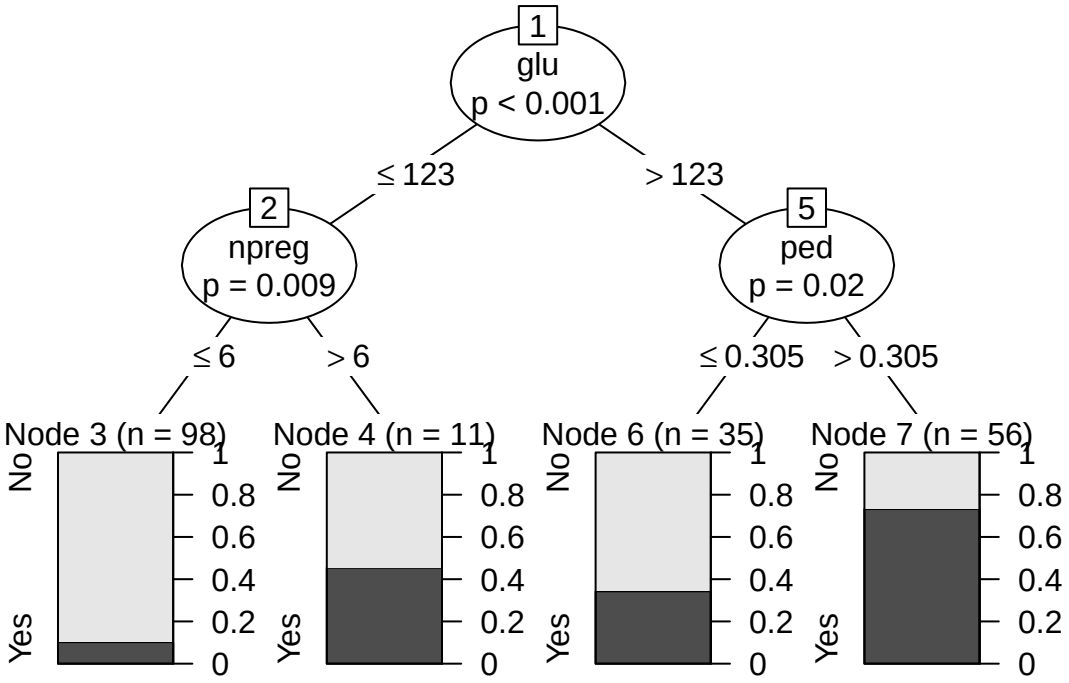
b) We want to model the variable `type` (does the woman have diabetes: Yes/No) using a decision tree. Fit a decision tree to the data using the `ctree()` function (**Hint**: `library(party)`)

```
suppressMessages(library(party))
tr <- ctree(type ~ ., data = Pima.tr)
tr
```

```
##
##   Conditional inference tree with 4 terminal nodes
##
## Response:  type
## Inputs:  npreg, glu, bp, skin, bmi, ped, age
## Number of observations:  200
##
## 1) glu <= 123; criterion = 1, statistic = 45.693
##   2) npreg <= 6; criterion = 0.991, statistic = 10.321
##     3)*  weights = 98
##   2) npreg > 6
##     4)*  weights = 11
## 1) glu > 123
##   5) ped <= 0.305; criterion = 0.98, statistic = 8.855
##     6)*  weights = 35
##   5) ped > 0.305
##     7)*  weights = 56
```

c) Plot the tree structure. Which variables were used to split the data? What is the meaning of the p-values printed below the splitting variables?

```
plot(tr)
```



The decision tree created by `ctree()` used the variables `glu`, `npreg` and `ped` for splitting. Since `ctree()` uses a significance test based method for variable selection, it compares the p-values (resulting from testing the relation between each predictor and the target variable statistically) between all predictors. The variable with the lowest p-value is selected for splitting. The p-value of each splitting variable is printed below the variable name.

d) Calculate the training error and show the corresponding confusion matrix (**Hint**: `predict(model, newdata=..., method='response')`)

```
pred.tr <- predict(tr, newdata=Pima.tr, method='response')
# Confusion matrix:
confT <- table(pred.tr, Pima.tr$type)
confT
```

```
##
## pred.tr  No Yes
##     No  117  27
##     Yes  15  41
```

```
# Training error:
missMat <- confT
diag(missMat) <- 0
missCount <- sum(missMat)
(test.err <- missCount/sum(confT))
```

```
## [1] 0.21
```

e) Yesterday we looked at a cross-validation function for the KNN classifier. Make the appropriate changes in the function so that it can be used for decision trees (generated with `ctree()`). Evaluate the performance of a `ctree`-generated decision tree (using default options) on the `Pima.tr` data set using

cross-validation. What can you say about the predictive performance of the decision tree?

```
ctree_crossVal <- function(data, label, k_fold=10){
  stopifnot(nrow(data)==length(label), is.factor(label),
            (1<k_fold & k_fold<=nrow(data)))
  # Create k sub-selections
  n <- nrow(data)
  ind_s <- sample(1:n)
  ind.L <- list()
  j1 <- 1
  for (i in 1:k_fold){
    j2 <- (i*n) %/% k_fold
    ind.L[[i]] <- ind_s[j1:j2]
    j1 <- j2+1
  }
  # Now run ctree on each selection (and collect results):
  confMat <- matrix(0,nrow=nlevels(label), ncol=nlevels(label))
  for(fold in 1:k_fold){
    ind_fold <- ind.L[[fold]]
    testDat <- data[ind_fold,]
    trainDat <- data[-ind_fold,]
    test.solu <- label[ind_fold]
    train.solu <- label[-ind_fold]
    trainDat.tr <- cbind(trainDat, y=train.solu) # needed for ctree function

    tr <- ctree(y~., data = trainDat.tr)
    tr.pred <- predict(tr, newdata=testDat, type='response')
    confMat.fold <-  table(tr.pred, test.solu)
    confMat <- confMat + confMat.fold
  }
  # Calculate estimated test-error
  missMat <- confMat
  diag(missMat) <- 0
  missCount <- sum(missMat)
  test.err <- missCount/n
  L.res <- list(k_fold=k_fold, Indices=ind.L,
                confMatrix=confMat, errorRate=test.err)
  return(L.res)
}

set.seed(4646)
ctree_crossVal(data = Pima.tr[,-8], label = Pima.tr$type, k_fold = 10)
```

```
## $k_fold
## [1] 10
##
## $Indices
## $Indices[[1]]
##  [1] 179 146  73 120 193 103 118 152 139  59  42  51  34 167  53 175 107  72  19
## [20] 127
##
## $Indices[[2]]
##  [1]  17  65 198 168  86 141 173  32   8 125  64  54 113  89  55  49  26 177 106
```

```
## [20]   4
##
## $Indices[[3]]
##  [1] 166  96 109  67 129 182 165 143 160 184 115  39 164 183 136 108  36  91 178
## [20]  70
##
## $Indices[[4]]
##  [1]  48  40 189  82 128  60 104 117 191 188  77 170  74 196 133 123 159 147  21
## [20]  79
##
## $Indices[[5]]
##  [1] 156   7 102  56  23  61 172  87 171  58  78 148 124 140 131 145  95 181 200
## [20]  38
##
## $Indices[[6]]
##  [1] 187  12   3 174 135 194  75 199 121  10  22 126  52 111 197 157  90  68  27
## [20] 176
##
## $Indices[[7]]
##  [1] 180  14  88 161 155  41 105  85  13 116  47  97 132 158 101   6 112  92  62
## [20]  63
##
## $Indices[[8]]
##  [1] 195  24 119  28  81  76   9 149   2 162 154 151 110 100  99 130   5  57  71
## [20] 186
##
## $Indices[[9]]
##  [1] 185  66 150  18  11 144 134  37  31  44 122 163  29  15  25   1  35  20  69
## [20]  46
##
## $Indices[[10]]
##  [1]  84  80 192 190  98 153  33  94 142  30  83  93 138 114  45 137  43 169  50
## [20]  16
##
##
## $confMatrix
##        test.solu
## tr.pred  No Yes
##     No  109  35
##     Yes  23  33
##
## $errorRate
## [1] 0.29
```

It seems that while the level "No" is predicted relatively well, the prediction of the level "Yes" is not better than random guessing.