Exercise:
Decision Trees

Machine Learning and Prediction Modelling

SOLUTION

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.

```r
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
```

```r
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 ...
```

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(partykit)`). Look at the output of the created object.

```r
suppressMessages(library(partykit))   # Dont show messages when loading package
set.seed(3487)
tr <- ctree(type ~ ., data = Pima.tr)
tr
```

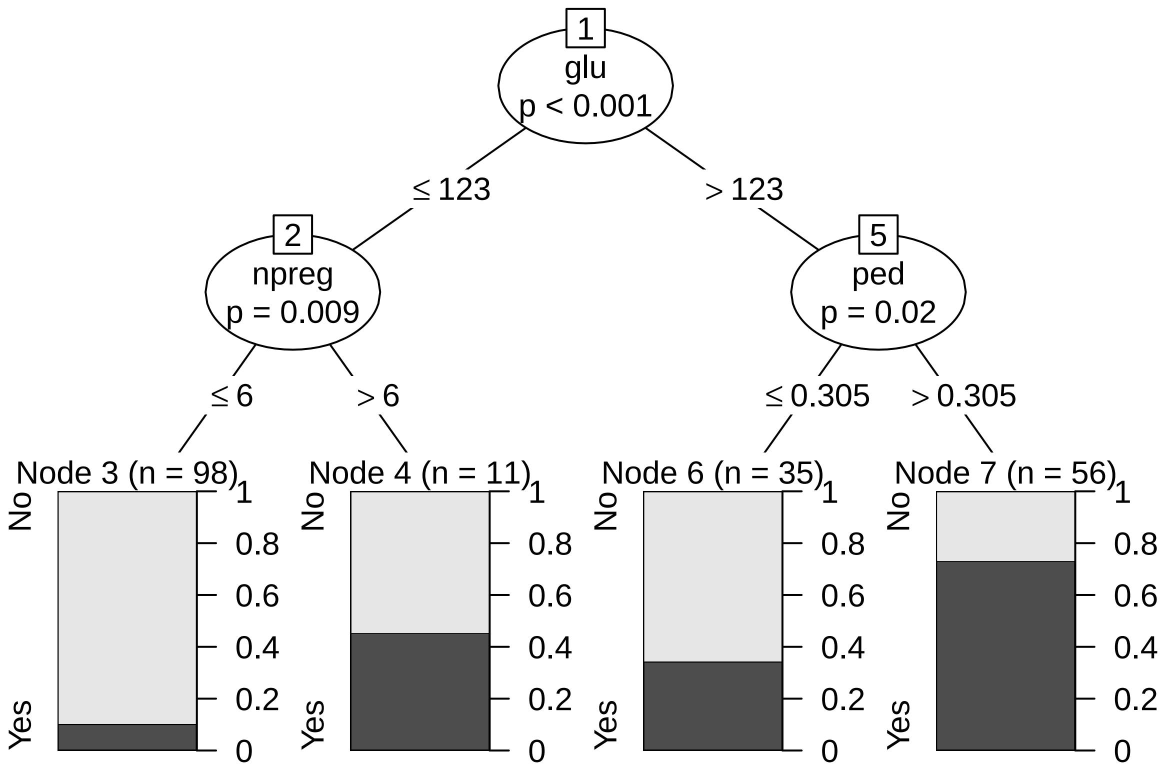
```
##
```

```
## Model formula:
## type ~ npreg + glu + bp + skin + bmi + ped + age
##
## Fitted party:
## [1] root
## |   [2] glu <= 123
## |   |   [3] npreg <= 6: No (n = 98, err = 10.2%)
## |   |   [4] npreg > 6: No (n = 11, err = 45.5%)
## |   [5] glu > 123
## |   |   [6] ped <= 0.305: No (n = 35, err = 34.3%)
## |   |   [7] ped > 0.305: Yes (n = 56, err = 26.8%)
##
## Number of inner nodes:    3
## Number of terminal nodes: 4
```

The generated object prints information about the used formula and the structure of the fitted tree.

c) Plot the tree structure. Look at all the information printed on the generated figure. Which variables were used to split the data? What is the meaning of the p-values printed below the splitting variables?

```r
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) The MASS package also contains a Pima.te data frame which is supposed to serve as a test data set for the `Pima.tr` data. Calculate the test error (misclassification rate) of our fitted tree using `Pima.te` as the test data. How does the tree perform?

```r
str(Pima.te)
```

```
## 'data.frame':    332 obs. of  8 variables:
##  $ npreg: int  6 1 1 3 2 5 0 1 3 9 ...
##  $ glu  : int  148 85 89 78 197 166 118 103 126 119 ...
##  $ bp   : int  72 66 66 50 70 72 84 30 88 80 ...
##  $ skin : int  35 29 23 32 45 19 47 38 41 35 ...
##  $ bmi  : num  33.6 26.6 28.1 31 30.5 25.8 45.8 43.3 39.3 29 ...
##  $ ped  : num  0.627 0.351 0.167 0.248 0.158 0.587 0.551 0.183 0.704 0.263 ...
##  $ age  : int  50 31 21 26 53 51 31 33 27 29 ...
##  $ type : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 2 1 1 2 ...
```

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

```
##
## pred_tre  No Yes
##      No  184  51
##      Yes  39  58
```

```r
# Test error:
missMat <- confT
diag(missMat) <- 0
test_err <- sum(missMat)/sum(confT)
test_err
```

```
## [1] 0.2710843
```

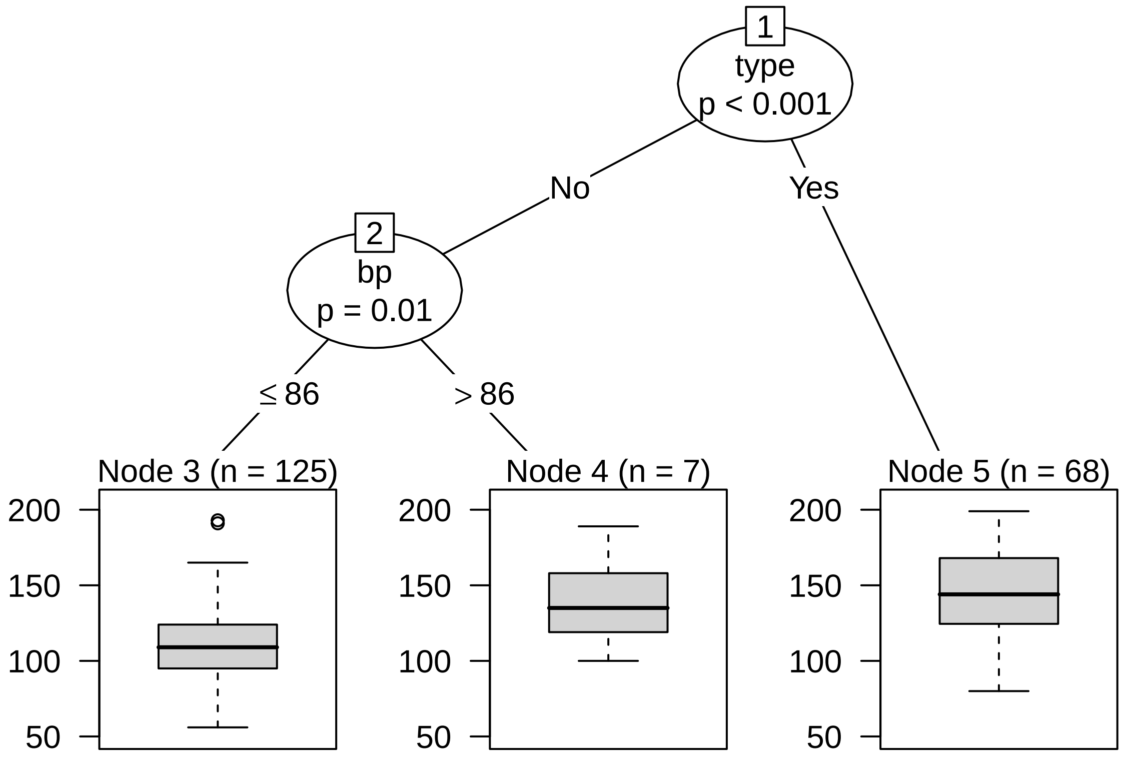
The decision tree reaches an error rate of 27.1% which is not that impressive. From the confusion matrix we can see that while No cases are mostly predicted correctly, the model has problems to predict the Yes cases. However, whether this performance is due to the model or whether the predictors simply don't carry that much information about the target variable is difficult to assess at this point.

Exercise 2: Plasma glucose in Pima Indian women

In this exercise we work again with the `Pima.tr` and `Pima.te` data from the `MASS` package. This time, we want to predict the numeric variable `glu` which indicates the plasma glucose concentration in an oral glucose tolerance test. Therefore, we are performing a regression task and not a classification task like in the previous exercise.

a) Fit a decision tree with the `ctree` function to the `Pima.tr` data. Plot the created tree and look at its structure. Which variables were used for splitting? What do the boxplots in the end nodes show?

```r
set.seed(2948294)
tr_glu <- ctree(glu ~ ., data = Pima.tr)
plot(tr_glu)
```



Only the variables type and bp were used as splitting variables. The boxplots in the endnodes show the distribution of glu values in the respective partitions of the training data.

b) Using the fitted decision tree, predict the `glu` values in the Pima.te data set. Since we are not doing classification we cannot produce a confusion matrix or calculate a misclassification rate. To compare the predicted `glu` values with the true `glu` values in `Pima.te` we instead calculate the mean squared error (MSE) of the predictions. The mean squared error is simply the mean of the squared differences between the predicted and the true `glu` values (formula below). Therefore, the smaller the MSE the closer are the predictions to the true values.

$$MSE = \frac{1}{n}\sum_{i}^{n}(\widehat{glu_i} - glu_i)^2$$

```
pred_glu <- predict(tr_glu, newdata = Pima.te)
### MSE:
mse_tree <- mean((pred_glu - Pima.te$glu)^2)
mse_tree
```

```
## [1] 730.7441
```

c) We want to compare the performance of the decision tree with the performance of a simple linear model. Fit a linear regression model to the `Pima.tr` data with `glu` as the target variable. (**Hint**: `lm()`)

```
lm_glu <- lm(glu ~ ., data = Pima.tr)
summary(lm_glu)
```

```
##
## Call:
## lm(formula = glu ~ ., data = Pima.tr)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.595 -17.396  -1.641  12.952  89.977
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 68.96159   15.62700   4.413 1.70e-05 ***
## npreg       -0.84245    0.72494  -1.162   0.2466
## bp           0.31786    0.18792   1.691   0.0924 .
## skin         0.07046    0.22559   0.312   0.7551
## bmi          0.23301    0.43405   0.537   0.5920
## ped         -2.36903    6.64389  -0.357   0.7218
## age          0.55832    0.24166   2.310   0.0219 *
## typeYes     26.29928    4.64856   5.658 5.51e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.26 on 192 degrees of freedom
## Multiple R-squared:  0.2853, Adjusted R-squared:  0.2592
## F-statistic: 10.95 on 7 and 192 DF,  p-value: 1.312e-11
```

d) Using the fitted linear model, predict the `glu` values in the `Pima.te` data and calculate the corresponding MSE. Looking at the MSE, how did the linear model perform compared to the decision tree? (**Hint**: `predict()`)

```r
predlm_glu <- predict(lm_glu, newdata = Pima.te)
mse_lm <- mean((predlm_glu - Pima.te$glu)^2)
### Compare MSE:
c('MSE_tree'=mse_tree, 'MSE_lm'=mse_lm)
```

```
## MSE_tree   MSE_lm
## 730.7441 678.5494
```

The linear model achieved a lower MSE on the test data and, therefore, performed slightly better than the decision tree.

Exercise 3: Crossvalidation with decision tree

In the first exercise, we have predicted the binary variable `type` and have estimated the test error by calculating the missclassification rate on the test data (`Pima.te`). In this exercise, we want to estimate the test error using cross validation. Try to adapt the cross validation function which we wrote for the k-nearest-neighbor classifier in the previous session, so that it can be used for a decision tree. Use the adpated function to estimate the test error for the prediction of `type`. For the cross validation, you can combine `Pima.tr` and `Pima.te` into one data frame.

```r
### Adapted cross validation function (for classiciation only):
classTree_crossVal <- function(d, yname, k_fold=10){
  ### Check input:
  stopifnot(is.factor(d[, yname]),
            1<k_fold, k_fold<=nrow(d), yname %in% colnames(d))
  ### Create k sub-selections:
  n <- nrow(d)
  inds <- sample(1:n)
  indL <- suppressWarnings(split(inds, f = 1:k_fold))
  ### Fit decision tree to each selection:
  preds <- list()   # Create empty list
  for(i in 1:k_fold){
    ### Extract fold data:
    ind_fold <- indL[[i]]
    testDat <- d[ind_fold,]
    trainDat <- d[-ind_fold,]
    ### Fit tree:
    trform <- as.formula(paste(yname, '~.'))   # Prepare formula
    treefit <- partykit::ctree(formula = trform, data=trainDat)
    ### Make prediction:
    trpreds <- predict(treefit, newdata = testDat, type='response')
    ### Combine true solution with predictions:
    preds[[i]] <- data.frame("TrueValue"=testDat[, yname], "Prediction"=trpreds)
  }
  ### Merge into one data frame:
  ### In one command:
  # predmrg <- do.call(rbind, preds)
  ### Using a for loop instead:
  predmrg <- preds[[1]]
  for(i in 2:length(preds)){
    predmrg <- rbind(predmrg, preds[[i]])
  }
  ### Confusion matrix:
  confm <- table(predmrg)
  ### Error rate:
  missMat <- confm
  diag(missMat) <- 0
  missCount <- sum(missMat)
  testErr <- missCount/sum(confm)
  ### Return value:
  rval <- list("ConfusionMatrix"=confm, "MissclassRate"=testErr)
  return(rval)
}


### Apply function to the data:
### Prepare data:
dcv <- rbind(Pima.tr, Pima.te)
### Apply function:
set.seed(2987482)
cvres <- classTree_crossVal(d = dcv, yname = 'type')
cvres
```

```
## $ConfusionMatrix
##          Prediction
## TrueValue  No Yes
##       No  281  74
##       Yes  61 116
##
## $MissclassRate
## [1] 0.2537594
```

Using crossvalidation, the missclassification rate is estimated at 25.4%.