Answers to exercises Session 5

Marjolein Fokkema

Exercise 1

Read in data:

```
load("MASQ.Rda")
set.seed(1)
train <- sample(1:nrow(MASQ), size = nrow(MASQ)*.8)
summary(MASQ)
```

```
##  D_DEPDYS       AD               AA              GDD             GDA
##  0:1927   Min.   : 26.00   Min.   :17.00   Min.   :12.00   Min.   :11.0
##  1:1670   1st Qu.: 64.00   1st Qu.:22.00   1st Qu.:20.00   1st Qu.:19.0
##           Median : 77.00   Median :28.00   Median :29.00   Median :24.0
##           Mean   : 75.05   Mean   :32.01   Mean   :30.64   Mean   :25.4
##           3rd Qu.: 88.00   3rd Qu.:39.00   3rd Qu.:40.00   3rd Qu.:31.0
##           Max.   :110.00   Max.   :83.00   Max.   :60.00   Max.   :54.0
##       GDM          leeftijd    geslacht
##  Min.   :15.0   Min.   :17.0   m:1317
##  1st Qu.:31.0   1st Qu.:28.0   v:2280
##  Median :40.0   Median :38.0
##  Mean   :40.6   Mean   :38.8
##  3rd Qu.:50.0   3rd Qu.:48.0
##  Max.   :75.0   Max.   :91.0
```

```
round(cor(MASQ[train, sapply(MASQ, is.numeric)]), digits = 2)
```

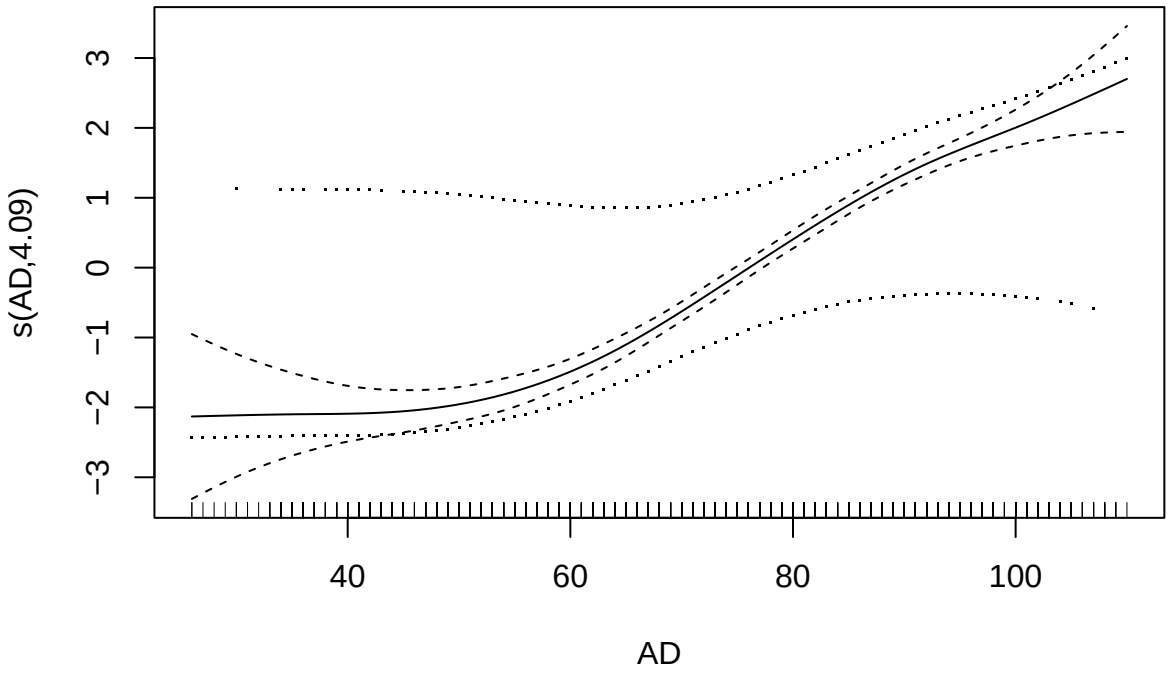
```
##              AD   AA   GDD   GDA   GDM leeftijd
## AD         1.00 0.51  0.79  0.61  0.74     0.01
## AA         0.51 1.00  0.58  0.79  0.70     0.01
## GDD        0.79 0.58  1.00  0.72  0.81    -0.07
## GDA        0.61 0.79  0.72  1.00  0.80    -0.05
## GDM        0.74 0.70  0.81  0.80  1.00    -0.04
## leeftijd 0.01 0.01 -0.07 -0.05 -0.04     1.00
```

Fit a smoothing spline of the `AD` variable to predict `D_DEPDYS`:

```
library("mgcv")
GAM <- gam(D_DEPDYS ~ s(AD, bs = "cr"),
           data = MASQ[train, ], method = "REML", family = "binomial")
summary(GAM)
```

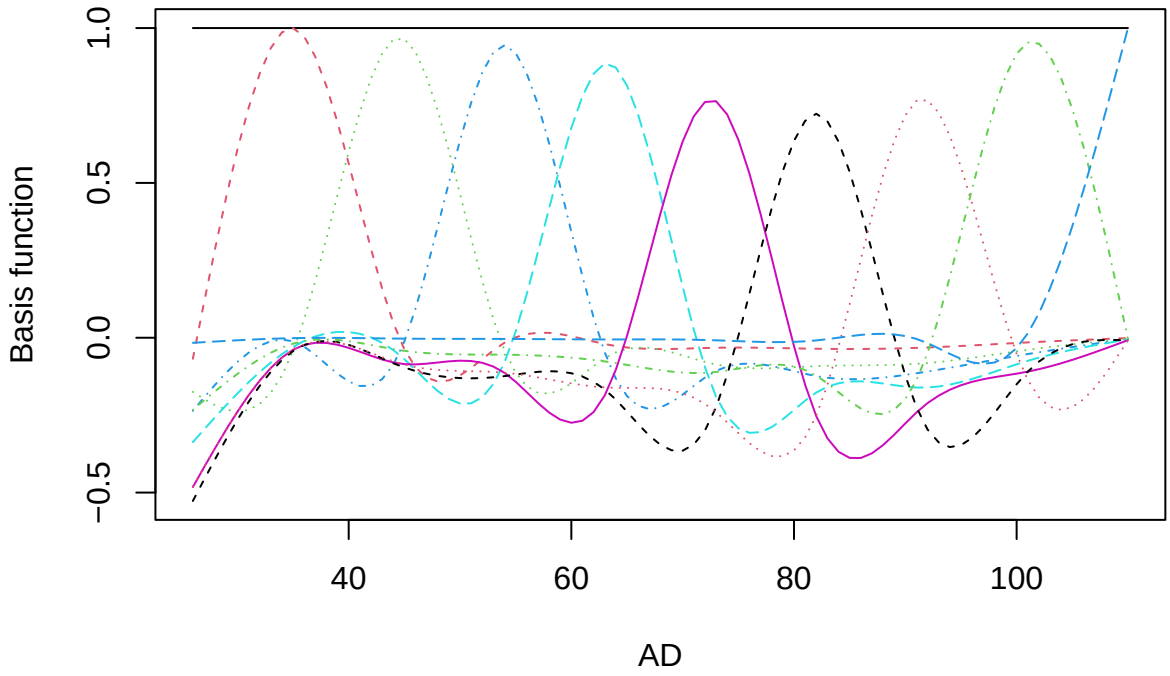
```
##
## Family: binomial
## Link function: logit
##
## Formula:
## D_DEPDYS ~ s(AD, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.24089    0.04751   -5.07 3.98e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df Chi.sq p-value
## s(AD) 4.09  5.041  672.4  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.297   Deviance explained = 23.7%
## -REML = 1522.2  Scale est. = 1         n = 2877
```

```
plot(GAM, residuals = TRUE)
```



Inspect the basis functions that were created for `AD`:

```r
mod_mat <- model.matrix(GAM)
matplot(sort(MASQ$AD[train]), mod_mat[order(MASQ$AD[train]), ], type = "l",
        xlab = "AD", ylab = "Basis function")
```



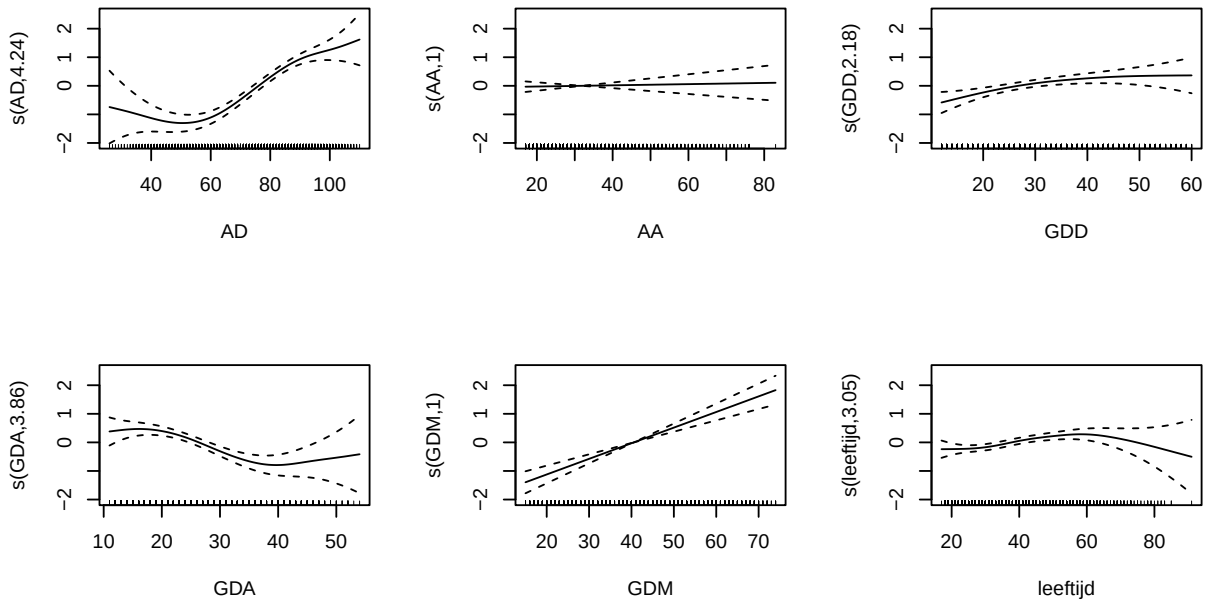
Exercise 2: Multiple predictors

```r
library("mgcv")
GAM <- gam(D_DEPDYS ~ s(AD) + s(AA) + s(GDD) + s(GDA) + s(GDM) + s(leeftijd) + geslacht,
           data = MASQ[train, ], method = "REML", family = "binomial")
summary(GAM)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## D_DEPDYS ~ s(AD) + s(AA) + s(GDD) + s(GDA) + s(GDM) + s(leeftijd) +
##     geslacht
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.33822    0.07755  -4.361 1.29e-05 ***
```

```
## geslachtv    0.13314    0.09517   1.399    0.162
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df  Chi.sq  p-value
## s(AD)        4.243  5.238 148.910  < 2e-16 ***
## s(AA)        1.001  1.002   0.115  0.73591
## s(GDD)       2.185  2.793  11.185  0.00971 **
## s(GDA)       3.855  4.806  33.279 4.62e-06 ***
## s(GDM)       1.001  1.001  52.723  < 2e-16 ***
## s(leeftijd) 3.050  3.817  17.530  0.00137 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.331   Deviance explained =   27%
## -REML = 1478.1  Scale est. = 1         n = 2877
```

```r
par(mfrow = c(2, 3))
plot(GAM)
```



We compute the mean squared error and misclassification rate using predicted probabilities, for both training and test observations:

```r
y_train <- as.numeric(MASQ[train, "D_DEPDYS"]) - 1
y_test <- as.numeric(MASQ[-train, "D_DEPDYS"]) - 1

## Training data
GAM_preds_train <- predict(GAM, newdata = MASQ[train, ], type = "response")
mean((y_train - GAM_preds_train)^2) ## Brier score
```

```
## [1] 0.1653101
```

```
tab_train <- prop.table(table(MASQ[train, "D_DEPDYS"], GAM_preds_train > .5)) ## confusion matrix
tab_train
```

```
##
##         FALSE      TRUE
##   0 0.4174487 0.1223497
##   1 0.1160932 0.3441084
```

```
1 - sum(diag(tab_train)) ## MCR
```

```
## [1] 0.2384428
```

```
## Test data
GAM_preds_test <- predict(GAM, newdata = MASQ[-train, ], type = "response")
mean((y_test - GAM_preds_test)^2) ## Brier score
```

```
## [1] 0.1666467
```

```
tab_test <- prop.table(table(MASQ[-train, "D_DEPDYS"], GAM_preds_test > .5)) ## confusion matrix
tab_test
```

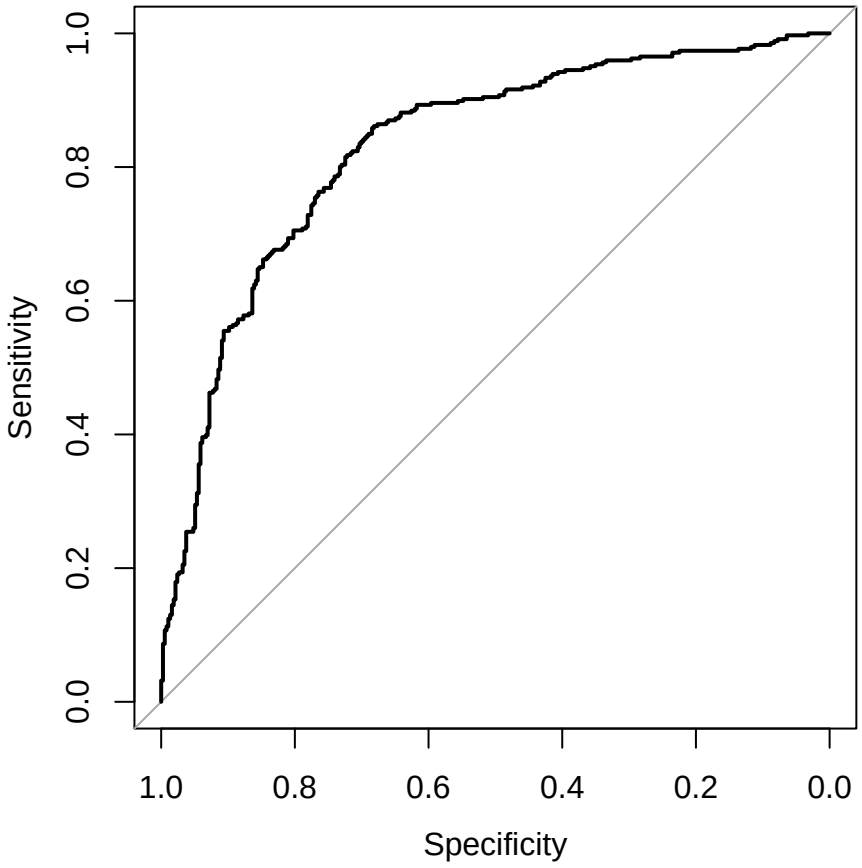
```
##
##         FALSE      TRUE
##   0 0.4027778 0.1166667
##   1 0.1236111 0.3569444
```

```
1 - sum(diag(tab_test)) ## MCR
```

```
## [1] 0.2402778
```

The Brier score and confusion matrices are quite similar between training and test data, indicating little overfitting.

```
## Or, compute ROC curve
library("pROC")
plot(roc(resp = y_test, pred = GAM_preds_test))
```

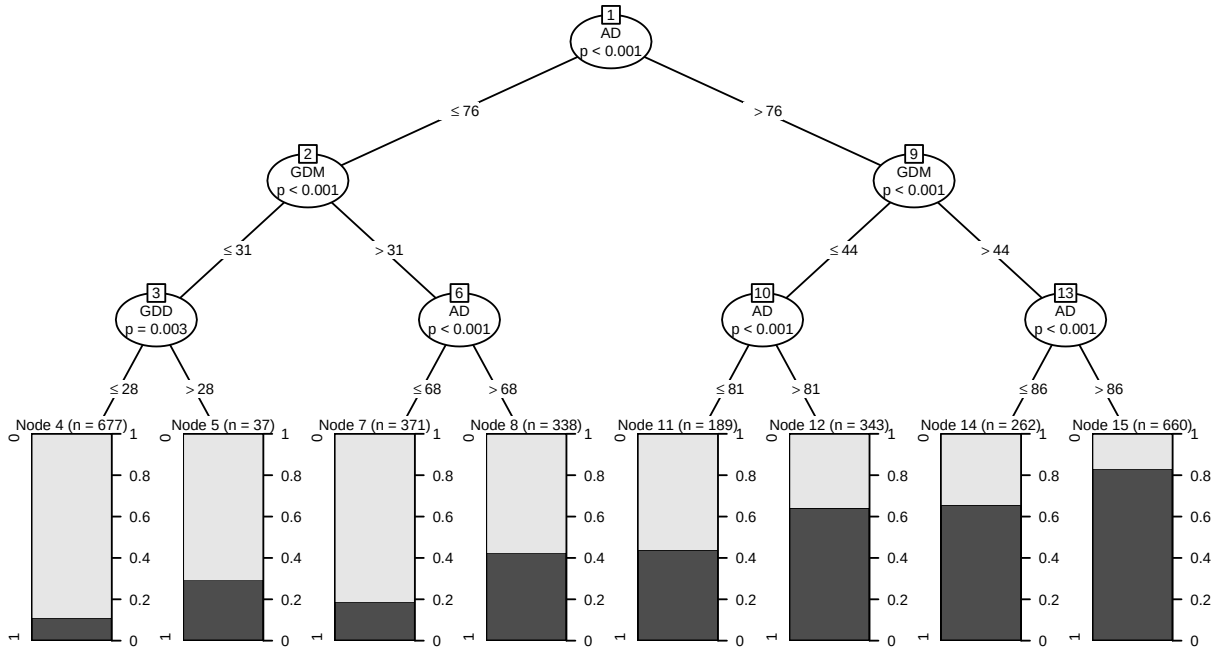



```
auc(resp = y_test, pred = GAM_preds_test)
```

```
## Area under the curve: 0.8295
```

Exercise 4: Fit a conditional inference tree

```
library("partykit")
ct <- ctree(D_DEPDYS ~ . , data = MASQ[train, ])
plot(ct, gp = gpar(cex = .5))
```



The conditional inference tree indicates a positive effect of the AD, GDM and GDD subscales on the probability of having a depressive / dysthymic disorder.

```
## Training data
ct_preds_train <- predict(ct, newdata = MASQ[train, ], type = "prob")[ , 2]
mean((y_train - ct_preds_train)^2) ## Brier score
```

```
## [1] 0.1705674
```

```
tab_train <- prop.table(table(MASQ[train, "D_DEPDYS"], ct_preds_train > .5)) ## confusion matrix
1 - sum(diag(tab_train)) ## MCR
```

```
## [1] 0.2457421
```

```
## Test data
y_test <- as.numeric(MASQ[-train, "D_DEPDYS"]) - 1
ct_preds_test <- predict(ct, newdata = MASQ[-train, ], type = "prob")[ , 2]
mean((y_test - ct_preds_test)^2) ## Brier score
```

```
## [1] 0.1738697
```

```r
tab_test <- prop.table(table(MASQ[-train, "D_DEPDYS"], ct_preds_test > .5)) ## confusion matrix
1 - sum(diag(tab_test)) ## MCR
```

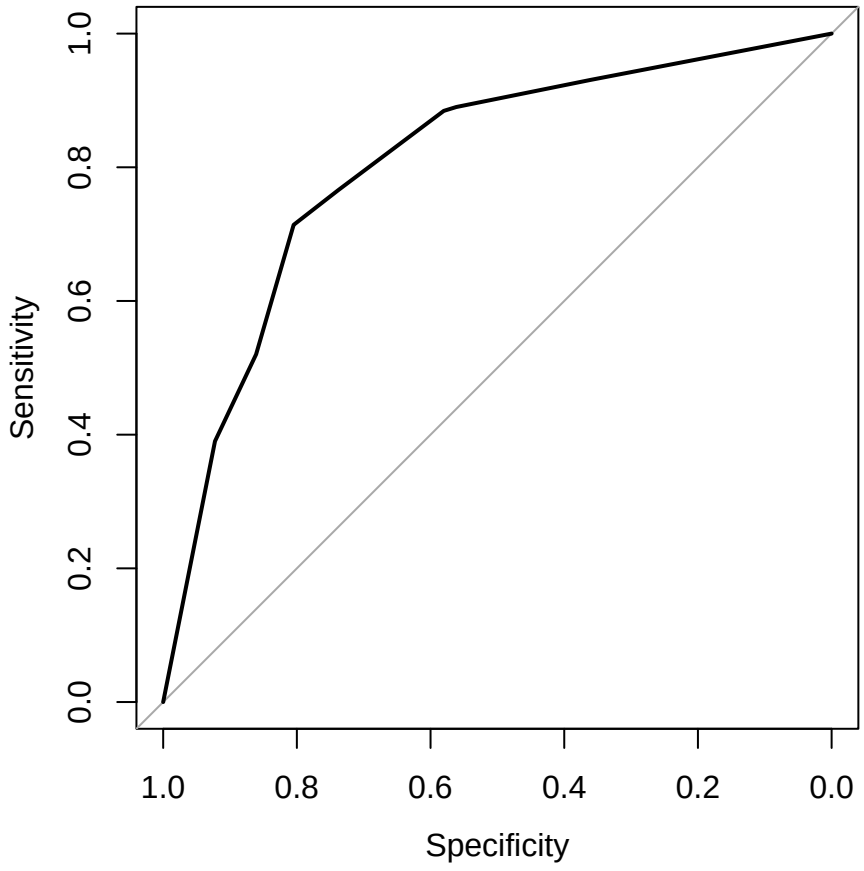
```
## [1] 0.2388889
```

The conditional inference tree provided best predictive accuracy of the single trees.

```r
## AUC on test observations
plot(roc(resp = y_test, pred = ct_preds_test))
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```




```r
auc(resp = y_test, pred = ct_preds_test)
```

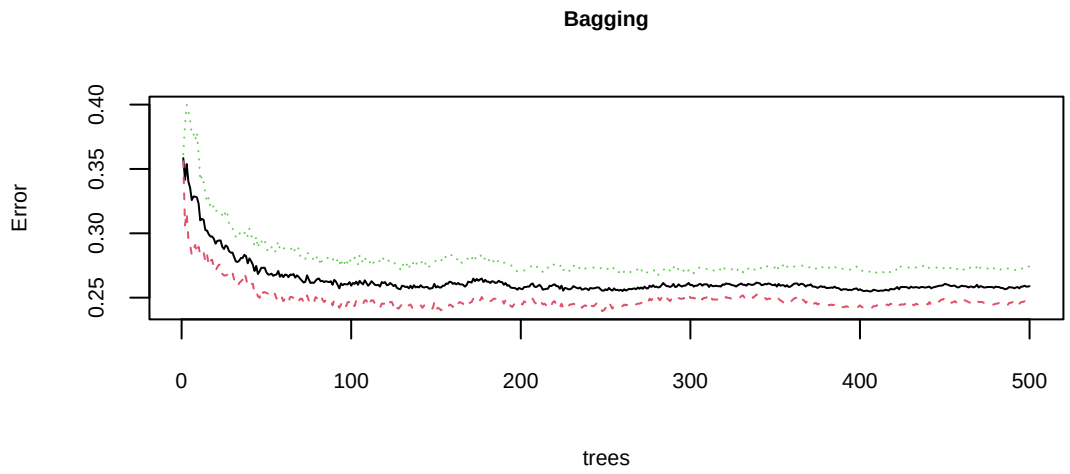
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

## Area under the curve: 0.805
```

Exercise 5: Fit a bagged ensemble and random forest

Fit the ensembles:

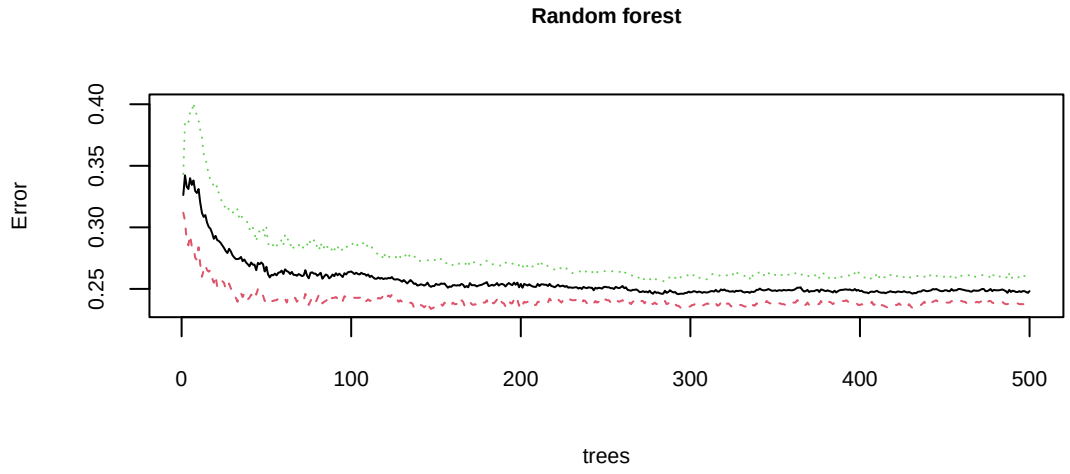
```
library("randomForest")
set.seed(1)
bag.ens <- randomForest(D_DEPDYS ~ AD + AA + GDD + GDA + GDM + leeftijd + geslacht,
                        data = MASQ[train,], importance = TRUE, mtry = 7)
set.seed(1)
rf.ens <- randomForest(D_DEPDYS ~ AD + AA + GDD + GDA + GDM + leeftijd + geslacht,
                       data = MASQ[train,], importance = TRUE, mtry = sqrt(7))
plot(bag.ens, cex.lab = .7, cex.axis = .7, cex.main = .7, main = "Bagging")
```



For these data, the out-of-bag (OOB) error decreases fast with the first 100 trees. After 200-300 trees, the OOB error stabilizes.

Note that for binary classification, three curves are provided: The black curve shows the misclassification error, the green and red curves show the classification error in each of the classes (comparable to sensitivity and specificity).

```
plot(rf.ens, cex.lab = .7, cex.axis = .7, cex.main = .7, main = "Random forest")
```



The OOB error plotted against the number of trees shows a very similar pattern as with the bagged ensemble.

Compute train MCR:

```
tab <- prop.table(table(MASQ[train, "D_DEPDYS"],
                        predict(bag.ens, newdata = MASQ[train,])))
1 - sum(diag(tab))  ## misclassification rate for bagging
```

```
## [1] 0
```

```
tab <- prop.table(table(MASQ[train, "D_DEPDYS"],
                        predict(rf.ens, newdata = MASQ[train,])))
1 - sum(diag(tab)) ## misclassification rate for RF
```

```
## [1] 0
```

Compute test MCR:

```
tab <- prop.table(table(MASQ[-train, "D_DEPDYS"],
predict(bag.ens, newdata = MASQ[-train,])))
1 - sum(diag(tab))  ## misclassification rate for bagging
```

```
## [1] 0.2388889
```

```
tab <- prop.table(table(MASQ[-train, "D_DEPDYS"],
predict(rf.ens, newdata = MASQ[-train,])))
1 - sum(diag(tab)) ## misclassification rate for RF
```

```
## [1] 0.2388889
```

We compute squared error on predicted probabilitites (Brier score) for the training data:

```
predict(bag.ens, newdata = MASQ[train,], type = "prob")[1:10, ]
```

```
##          0     1
## 1017 0.982 0.018
## 679  0.816 0.184
## 2177 0.960 0.040
## 930  0.058 0.942
## 1533 0.788 0.212
## 471  0.998 0.002
## 2347 0.812 0.188
## 270  0.966 0.034
## 1211 0.922 0.078
## 3379 0.878 0.122
```

Note that the predict method returns predicted probabilities for both classes, for objects of class `randomForest`.

Therefore, we selected the second column of the returned probabilities (`[ , 2]`):

```
mean((y_train -
        predict(bag.ens, newdata = MASQ[train,], type = "prob")[ , 2])^2)
```

```
## [1] 0.02466598
```

```
mean((y_train -
        predict(rf.ens, newdata = MASQ[train,], type = "prob")[ , 2])^2)
```

```
## [1] 0.02431729
```

And for the test data:

```
mean((y_test -
        predict(bag.ens, newdata = MASQ[-train,], type = "prob")[ , 2])^2)
```

```
## [1] 0.1765018
```

```
mean((y_test -
        predict(rf.ens, newdata = MASQ[-train,], type = "prob")[ , 2])^2)
```

```
## [1] 0.1723298
```

Test MCRs are identical for the bagged ensemble and random forest. Test SEL is lower for the random forest.

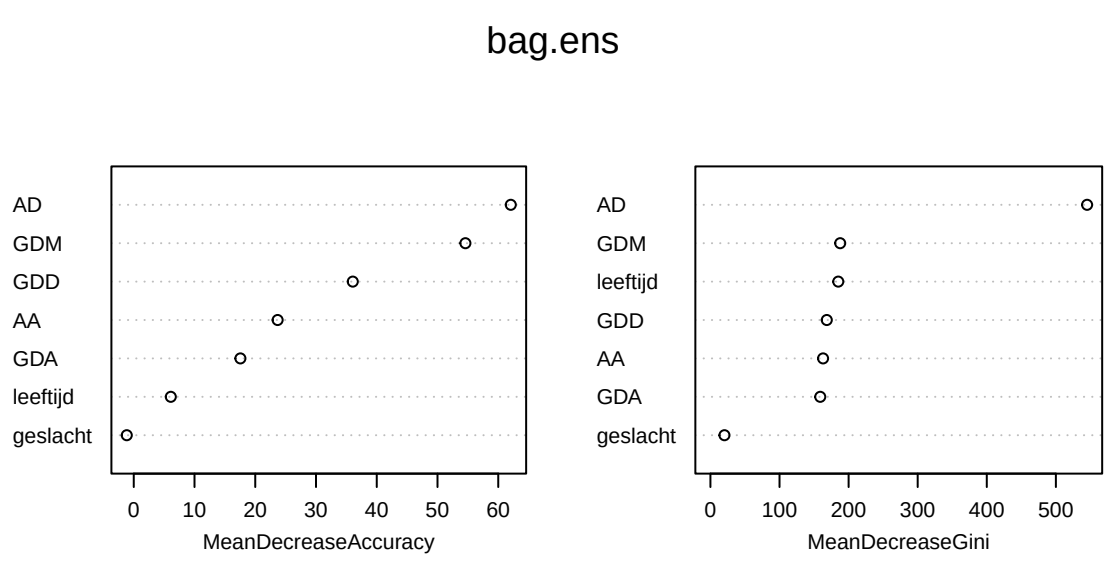
Interpretation

We inspect variable importances:

```r
importance(bag.ens)
```

```
##                  0          1 MeanDecreaseAccuracy MeanDecreaseGini
## AD       26.008074 46.6261372            62.061330        545.20309
## AA       11.210061 19.5952066            23.683674        163.04057
## GDD      24.875687 18.2097877            36.050773        168.41636
## GDA      18.096720  3.1288186            17.552676        158.92755
## GDM      24.393229 39.5161589            54.590675        187.79771
## leeftijd  2.752565  5.6442541             6.098998        185.04514
## geslacht -1.634730 -0.1247933            -1.150591         20.31331
```

```r
varImpPlot(bag.ens, cex = .7, cex.main = .7)
```



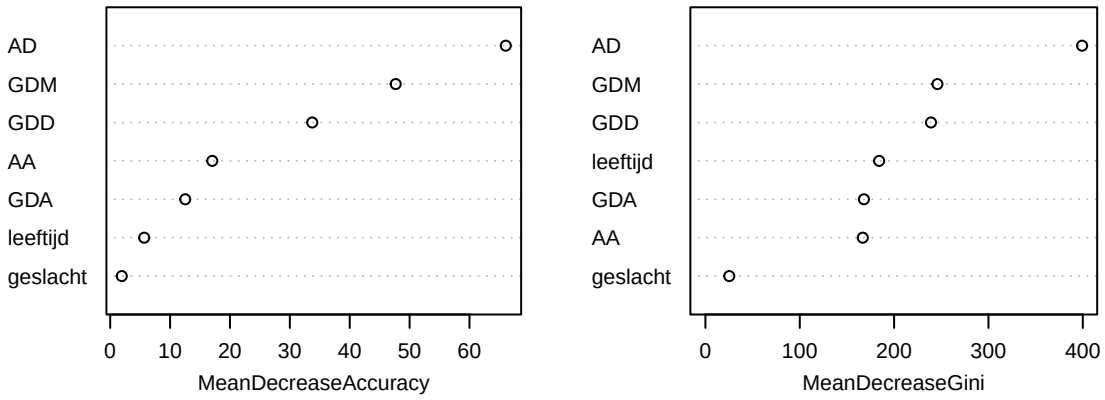
According to the reduction in MSE for the out-of-bag observations (left panel) if the values of each predictor variable are permuted, the AD (anhedonic depression), GDM (general distress mixed), and GDD (general distress depression) are the most important predictors of a depressive disorder diagnosis.

According to the improvement in node purity (i.e., training error; right panel), AD, leeftijd (age) and GDM are the most important predictors of of a depressive disorder diagnosis.

```r
importance(rf.ens)
```

```
##                  0          1 MeanDecreaseAccuracy MeanDecreaseGini
## AD       33.630440 48.266024            66.064052        399.27306
## AA        6.599139 15.579649            17.047234        166.79307
## GDD      22.394085 16.774428            33.745150        239.10990
## GDA      13.584839  1.184124            12.532481        168.00515
## GDM      21.001515 36.662315            47.677333        245.91414
## leeftijd  1.773007  5.834744             5.682049        184.13887
## geslacht  1.344184  1.423987             1.931168         25.26403
```

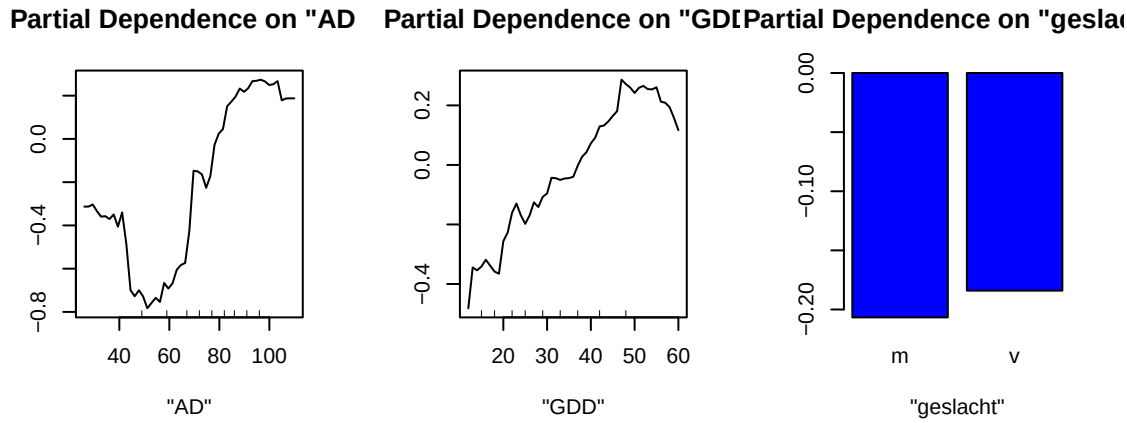
```r
varImpPlot(rf.ens, cex = .7, cex.main = .7)
```



The AD, GDM and GDD scales appear most important in the random forest.

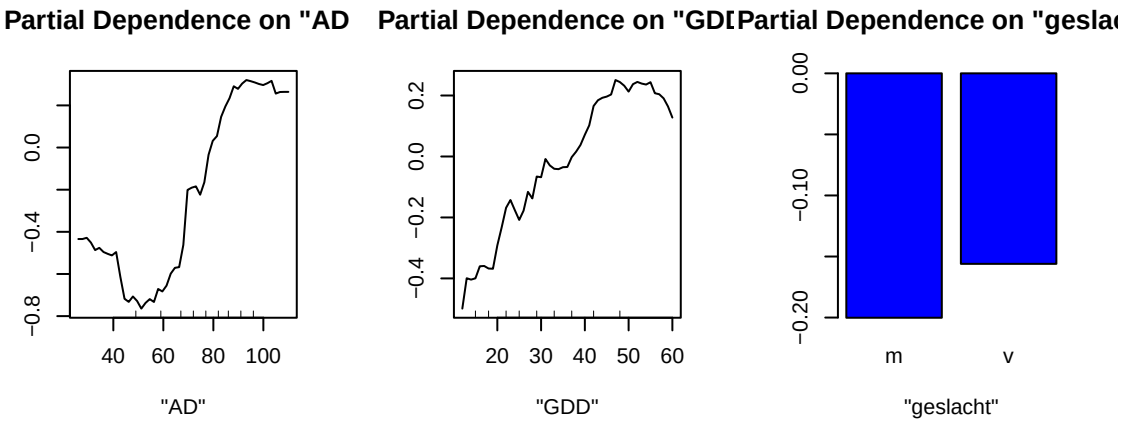
We request partial dependence plots for the bagged ensemble:

```
par(mfrow = c(1, 3))
partialPlot(bag.ens, x.var = "AD", pred.data = MASQ[train,], which.class = "1")
partialPlot(bag.ens, x.var = "GDD", pred.data = MASQ[train,], which.class = "1")
partialPlot(bag.ens, x.var = "geslacht", pred.data = MASQ[train,], which.class = "1")
```



Note that we have to specify the appropriate class label for these plots if we perform classification, otherwise we get partial dependence plots for the effect on the probability of belonging to the first ("0", non-depressed) class.

```
par(mfrow = c(1, 3))
partialPlot(rf.ens, x.var = "AD", pred.data = MASQ[train,], which.class = "1")
partialPlot(rf.ens, x.var = "GDD", pred.data = MASQ[train,], which.class = "1")
partialPlot(rf.ens, x.var = "geslacht", pred.data = MASQ[train,], which.class = "1")
```


Partial Dependence on "AD
Partial Dependence on "GDD
Partial Dependence on "geslac
0.0
−0.4
−0.8
40
60
80
100
"AD"
0.2
0.0
−0.2
−0.4
20
30
40
50
60
"GDD"
0.00
−0.10
−0.20
m
v
"geslacht"