Exercise 9

Tobias Raidl, 11717659

2023-01-06

Load the data Diabetes from the package ROCit. Delete observations with missings using na.omit(). Our goal is to find a classification model for diabetes, based on the variable dtest – see help file. For this task we shall use Generalized Additive Models, implemented in the function gam() of the package library(mgcv), using the argument family="binomial". Select randomly a training set of about 2/3 of the observations, build the classification model, predict the group membership for the (remaining) test data and compute the misclassification rate.

```
library(ROCit)
```

```
## Warning: package 'ROCit' was built under R version 4.3.2
```

```
library(mgcv)
```

```
## Warning: package 'mgcv' was built under R version 4.3.2
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:nlme':
##
##     collapse
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(mltools)
```

```
## Warning: package 'mltools' was built under R version 4.3.2
```

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.3.2
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
data(Diabetes)
df = na.omit(Diabetes)
df = df %>%
    select(-c(id, glyhb))  #%>%
# mutate(dtest=if_else(dtest=='+', TRUE, FALSE))

# df = one_hot(as.data.table(df))

set.seed(123)
sample <- sample(c(TRUE, FALSE), nrow(df), replace = TRUE, prob = c(2/3, 1/3))
train <- df[sample, ]
test <- df[!sample, ]
```

a)

Which of the remaining variables should be considered in the model? Argue why it could make sense to exclude predictor variables

exclude id because there is no relation between it and dtest exclude glyhb because dtest (our response variable) indicates glyhb exclude ratio because its just the chol/hdl ratio -> already got those exclude bmi because its purely based on height and weight -> already got those exclude whr because its just the waist/hip ratio -> already got those

b)

The smooth functions in GAMs can be defined for every variable by s(variable), see also course notes. It might not make sense to use smooth functions for all variables, for sure not for factor variables. Now compute the GAM based on your chosen "formula". ### c) You might experience difficulties with estimating too many parameters. This could be solved by using the parameter k within s(), which allows to set an upper bound for the effective degrees of freedom.

```
# model.gam = gam(as.factor(dtest) ~ s(chol, k=3) + s(hdl, k=3) + s(age, k=3) +
# s(bmi, k=3) + s(bp.1s, k=3) + s(bp.1d, k=3) + s(waist, k=3) + s(hip, k=3) +
# s(time.ppn, k=3) + s(whr, k=3) + gender + frame, family='binomial',
# data=train)

model.gam = gam(as.factor(dtest) ~ s(chol, k = 3) + s(hdl, k = 3) + s(age, k = 3) +
    s(height) + s(weight) + s(bp.1s, k = 3) + s(bp.1d, k = 3) + s(bp.2s, k = 3) +
    s(bp.2d, k = 3) + s(waist, k = 3) + s(hip, k = 3) + s(time.ppn, k = 3) + gender +
    frame + location, family = "binomial", data = train)

summary(model.gam)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## as.factor(dtest) ~ s(chol, k = 3) + s(hdl, k = 3) + s(age, k = 3) +
##     s(height) + s(weight) + s(bp.1s, k = 3) + s(bp.1d, k = 3) +
##     s(bp.2s, k = 3) + s(bp.2d, k = 3) + s(waist, k = 3) + s(hip,
##     k = 3) + s(time.ppn, k = 3) + gender + frame + location
```

```
##
## Parametric coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -0.3407     2.7330  -0.125   0.9008
## gendermale      -3.6540     2.1070  -1.734   0.0829 .
## framemedium     -2.0778     1.4715  -1.412   0.1579
## framesmall      -4.6203     2.0014  -2.309   0.0210 *
## locationLouisa   0.2700     1.1107   0.243   0.8080
## ---
## 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(chol)     1.928  1.992  5.575 0.065157 .
## s(hdl)      1.000  1.000  0.002 0.960687
## s(age)      1.000  1.000 12.046 0.000519 ***
## s(height)   1.000  1.000  6.266 0.012315 *
## s(weight)   7.110  7.842  9.529 0.266793
## s(bp.1s)    1.000  1.000  0.043 0.836330
## s(bp.1d)    1.000  1.000  1.549 0.213257
## s(bp.2s)    1.000  1.000  0.392 0.531131
## s(bp.2d)    1.000  1.000  0.275 0.600140
## s(waist)    1.000  1.000  0.323 0.569780
## s(hip)      1.000  1.000  0.014 0.906475
## s(time.ppn) 1.126  1.231  0.665 0.409737
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.541   Deviance explained = 59.6%
## UBRE = 0.031156  Scale est. = 1         n = 88
```

Interpretation of summary: - edf=effective degrees of freedom (complexity of smooth function; 1=linear, 2=quadratic) - a significant smooth term is one where you cannot draw a horizontal line through the 95% conf. interval

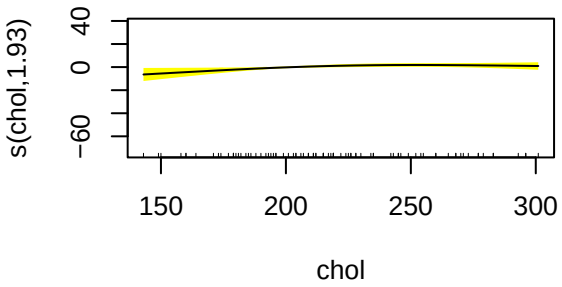
d)

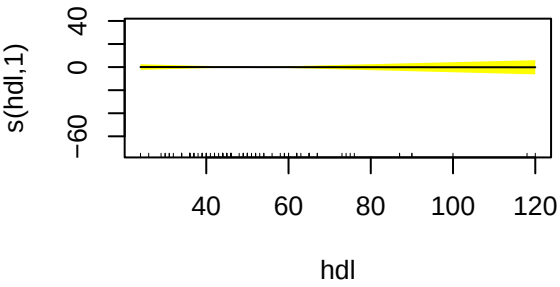
Which variables are significant in the model? How complex are the smooth functions? Say critical value $\alpha = 0.05$ significant variables are age and height. The complexity is the first column (edf) of the smooth terms summary. weight has a high complexity. If edf=1 -> linear, 2 -> quadratic

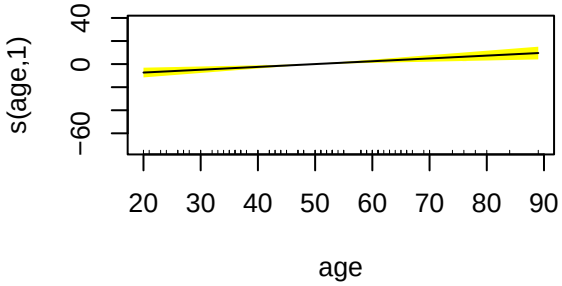
e)

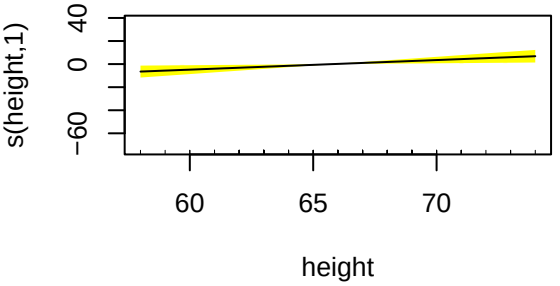
Plot the explanatory variables against their smoothed values as they are used in the model. You can simply use: plot(gam.object,page=1,shade=TRUE,shade.col="yellow") How can you interpret these plots?

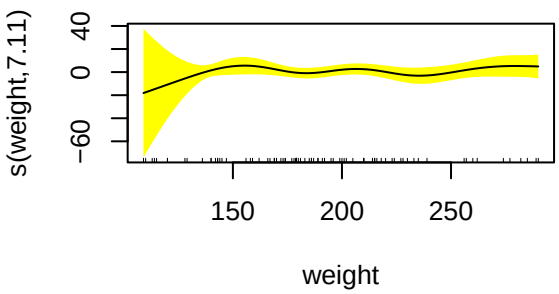
```
plot(model.gam, page = 5, shade = TRUE, shade.col = "yellow")
```

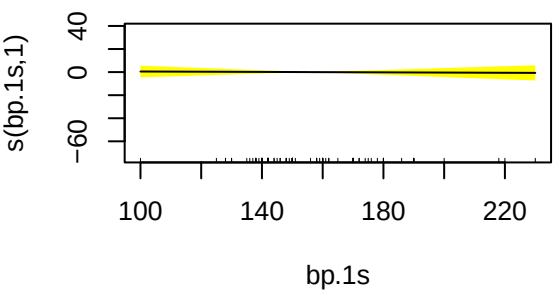

s(chol,1.93)
−60
0
40
150
200
250
300
chol

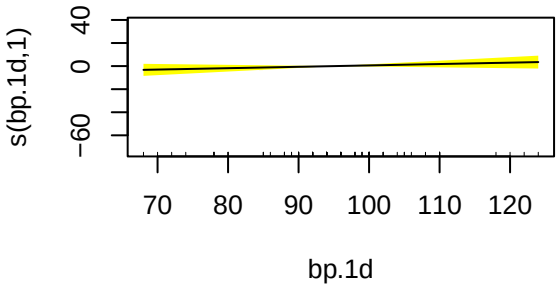

s(hdl,1)
−60
0
40
40
60
80
100
120
hdl

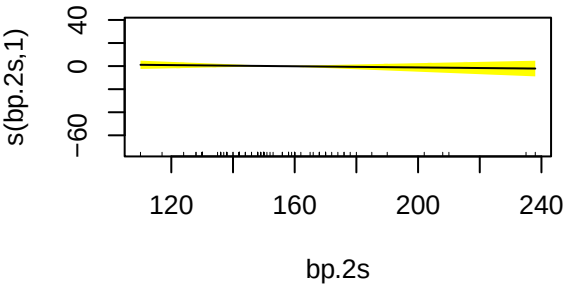

s(age,1)
−60
0
40
20
30
40
50
60
70
80
90
age

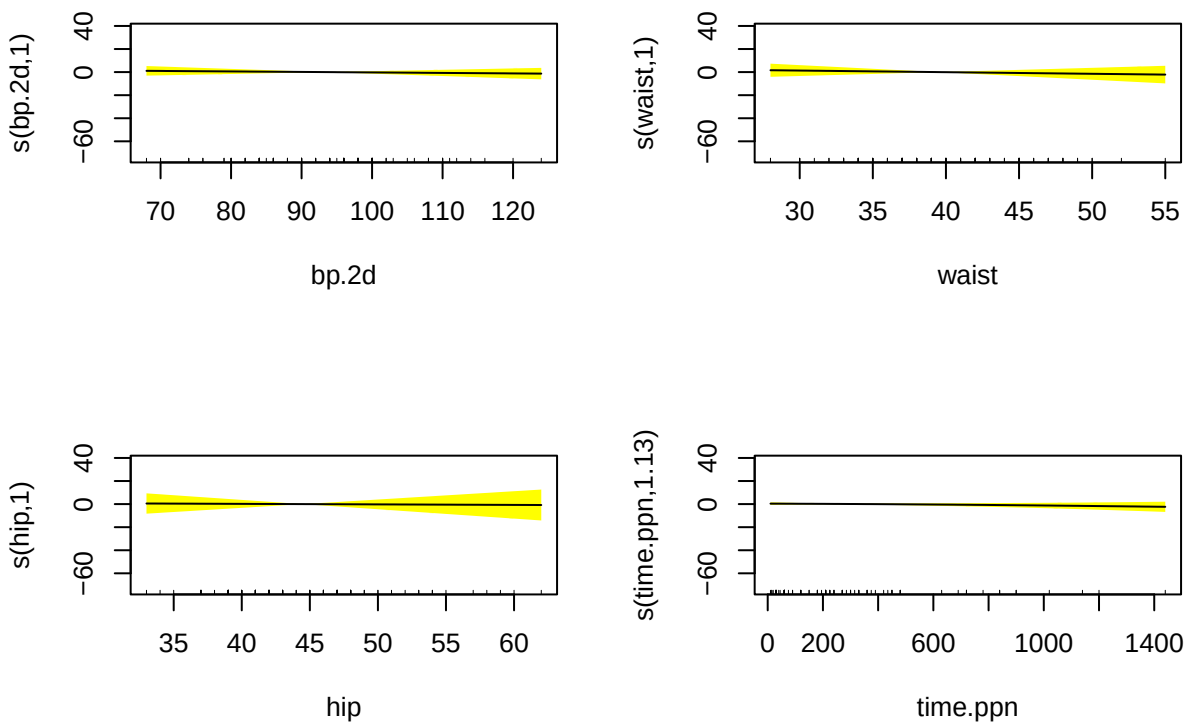

s(height,1)
−60
0
40
60
65
70
height


s(weight,7.11)
−60
0
40
150
200
250
weight


s(bp.1s,1)
−60
0
40
100
140
180
220
bp.1s


s(bp.1d,1)
−60
0
40
70
80
90
100
110
120
bp.1d


s(bp.2s,1)
−60
0
40
120
160
200
240
bp.2s



The yellow areas are the confidence intervals. You can visually see the complexity of the smooth functions

f)

Report the misclassification error for the test set.

```
get_misclass_rate = function(model, data) {
    y_pred = predict.gam(model, newdata = data)
    y_pred[y_pred < 0.5] = "-"
    y_pred[y_pred >= 0.5] = "+"

    conf_mat = table(y_pred, data$dtest, dnn = c("pred", "gt"))
    misclass_rate = (conf_mat[1, 2] + conf_mat[2, 1])/sum(conf_mat)
    return(misclass_rate = misclass_rate)
}

cat(paste("gam misclassification rate: ", get_misclass_rate(model.gam, test)))
```

```
## gam misclassification rate:  0.285714285714286
```

g)

We can try to improve the classifier by variable selection. A natural option would be stepwise variable selection. A look into the help file of step.gam says that There is no step.gam in package mgcv. Nice. However, you can find some hints to still improve the model. Try out one of these ideas. Which variables are not used in the model? Compare with the misclassification error from (f).

h)

Fit again a GAM, but only with the explanatory variables selected in (g). Inspect and interpret the outputs of summary() and plot(), and report the misclassification error for the test set.