Hochschule Kaiserslautern
University of Applied Sciences

Informatik und Mikrosystemtechnik
Zweibrücken

Studiengang

Digital Engineering

PO Version 2019

Projektarbeit

Datenanalyse für Kardiovaskuläre Erkrankung

Explorative Analyse (mit statistischen Tests)
Prediction model zur Vorhersage
Feature Importance/ Feature Selection Analyse
Cluster Analyse
Shiny App für die Analyse des Datensatzes

vorgelegt von

Ali Abdorrahimi

Datum

29.04.2022

Betreuung: Prof. Dr. Bastian Beggel
Zweitkorrektor: Prof.Dr.Manfred Brill

Data Exploration - Cardiovascular Study Dataset

Projektbericht

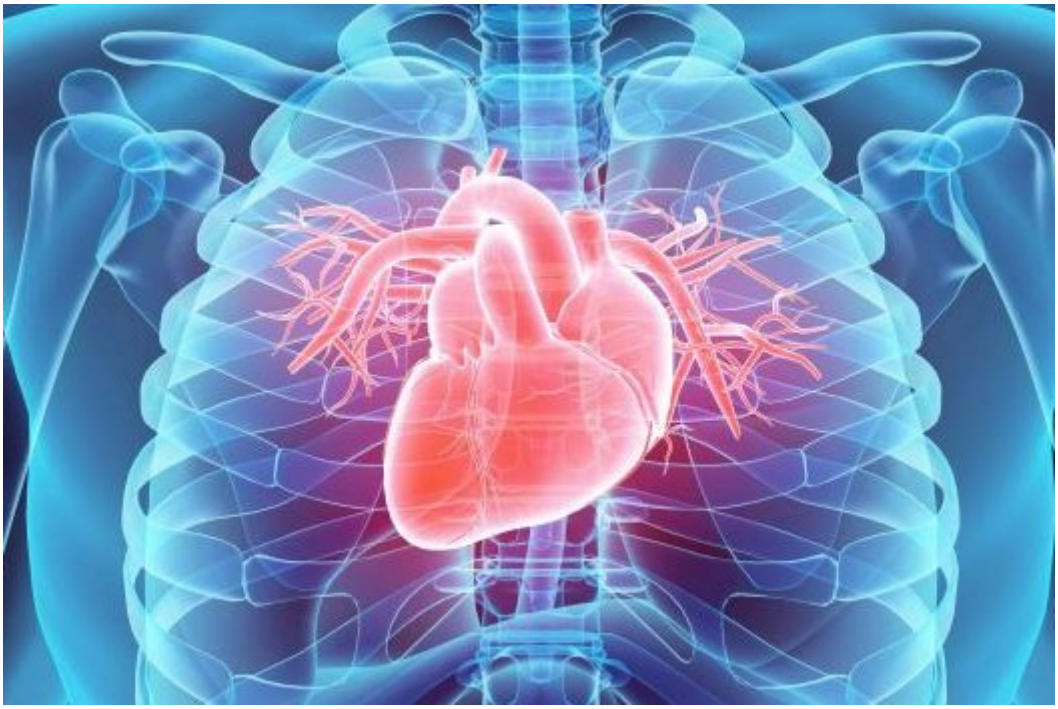


Figure 1: Cardio

Das Ziel ist es, vorherzusagen, ob ein Patient ein 10-Jahres-Risiko für eine koronare Herzkrankheit (KHK) hat oder nicht. Eine Datenvisualisierung zu den Daten zu erstellen, um verwertbare Erkenntnisse über das Thema zu gewinnen. welche Faktoren die größte und welche die geringste Auswirkung hatten, analysieren Sie die Faktoren mit dem p-Value,

Die WHO schätzt, dass jedes Jahr weltweit 12 Millionen Todesfälle auf Herzkrankheiten zurückzuführen sind. Die Hälfte der Todesfälle in den USA und anderen entwickelten Ländern ist auf cardio vascular Krankheiten zurückzuführen. Die frühzeitige Prognose von Cardio-Erkrankungen kann dazu beitragen, dass bei Risikopatienten Entscheidungen zur Änderung der Lebensweise getroffen werden, was wiederum zu einer Verringerung der Komplikationen führt. Ziel dieser Untersuchung ist es, die wichtigsten Risikofaktoren für Herzkrankheiten zu ermitteln und das Gesamtrisiko mithilfe Prediction model vorherzusagen.

Load Libraries

```
library(magrittr)
library(plyr)
library(dplyr)
library(ggplot2)
library(grid)
library(gridExtra)
library(stringr)
library(here)
library(VIM)
library(ggrepel)
library(patchwork)
library("RColorBrewer")
```

Load Data

```
cardio_raw= read.csv(here::here ("D:\\DataMining\\train.csv"))
```

Datensatzbeschreibung

Der Datensatz ist auf der Kaggle-Website öffentlich verfügbar und stammt aus einer laufenden kardiovaskulären Studie über Einwohner der Stadt Framingham, Massachusetts. Ziel der Klassifizierung ist die Vorhersage, ob ein Patient ein 10-Jahres-Risiko für eine künftige koronare Herzkrankheit (CHD) hat. Der Datensatz enthält Informationen über die Patienten. Er umfasst über 4.000 Datensätze und 15 Attribute.

Variablen

Jedes Attribut ist ein potenzieller Risikofaktor. Es gibt sowohl demografische, verhaltensbezogene als auch medizinische Risikofaktoren.

Demographic:

- Sex: male or female(“M” or “F”)
- Age: Age of the patient;(Continuous - Although the recorded ages have been truncated to whole numbers, the concept of age is continuous) Behavioral
- is_smoking: whether or not the patient is a current smoker (“YES” or “NO”)
- Cigs Per Day: the number of cigarettes that the person smoked on average in one day.(can be considered continuous as one can have any number of cigarettes, even half a cigarette.)

Medical(history)

- BP Meds: whether or not the patient was on blood pressure medication (Nominal)
- Prevalent Stroke: whether or not the patient had previously had a stroke (Nominal)
- Prevalent Hyp: whether or not the patient was hypertensive (Nominal)
- Diabetes: whether or not the patient had diabetes (Nominal)

Medical(current) • Tot Chol: total cholesterol level (Continuous)

- Sys BP: systolic blood pressure (Continuous)
- Dia BP: diastolic blood pressure (Continuous)

• BMI: Body Mass Index (Continuous)

• Heart Rate: heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of large number of possible values.)

• Glucose: glucose level (Continuous) Predict variable (desired target) • 10 year risk of coronary heart disease CHD(binary: “1”, means “Yes”, “0” means “No”)

Descriptive Statistics

R’s str function gibt uns einen Blick auf die Datentypen im Datensatz , die head function druckt die ersten 5 Zeilen. Mit der summary-function können wir grundlegende Zusammenfassungsstatistiken für jede Spalte anzeigen.

Die ersten 5 Zeilen anzeigen.

```
head(cardio_raw)
```

```
##   id age education sex is_smoking cigsPerDay BPMeds prevalentStroke
## 1  0  64         2   F        YES          3      0               0
## 2  1  36         4   M         NO          0      0               0
## 3  2  46         1   F        YES         10      0               0
## 4  3  50         1   M        YES         20      0               0
## 5  4  64         1   F        YES         30      0               0
## 6  5  61         3   F         NO          0      0               0
##   prevalentHyp diabetes totChol sysBP diaBP   BMI heartRate glucose TenYearCHD
## 1            0        0     221 148.0    85    NA        90      80          1
## 2            1        0     212 168.0    98 29.77        72      75          0
## 3            0        0     250 116.0    71 20.35        88      94          0
## 4            1        0     233 158.0    88 28.26        68      94          1
## 5            0        0     241 136.5    85 26.42        70      77          0
## 6            1        0     272 182.0   121 32.80        85      65          1
```

Zeigt Strukturinformationen über den Datenrahmen an.

```
str(cardio_raw)
```

```
## 'data.frame':    3390 obs. of  17 variables:
##  $ id             : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ age            : int  64 36 46 50 64 61 61 36 41 55 ...
##  $ education      : num  2 4 1 1 1 3 1 4 2 2 ...
##  $ sex            : chr  "F" "M" "F" "M" ...
##  $ is_smoking     : chr  "YES" "NO" "YES" "YES" ...
##  $ cigsPerDay     : num  3 0 10 20 30 0 0 35 20 0 ...
##  $ BPMeds         : num  0 0 0 0 0 0 0 0 NA 0 ...
##  $ prevalentStroke: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ prevalentHyp   : int  0 1 0 1 0 1 1 0 0 1 ...
##  $ diabetes       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ totChol        : num  221 212 250 233 241 272 238 295 220 326 ...
##  $ sysBP          : num  148 168 116 158 136 ...
##  $ diaBP          : num  85 98 71 88 85 121 136 68 78 81 ...
##  $ BMI            : num  NA 29.8 20.4 28.3 26.4 ...
##  $ heartRate      : num  90 72 88 68 70 85 75 60 86 85 ...
##  $ glucose        : num  80 75 94 94 77 65 79 63 79 NA ...
##  $ TenYearCHD     : int  1 0 0 1 0 1 0 0 0 0 ...
```

Cleaning and Preparing the Data

Aufgrund der Ergebnisse der obigen Funktion wurden mehrere Probleme mit dem Import der Daten durch die Funktion read.csv festgestellt, die vor einer tiefer gehenden Analyse behoben werden müssen:

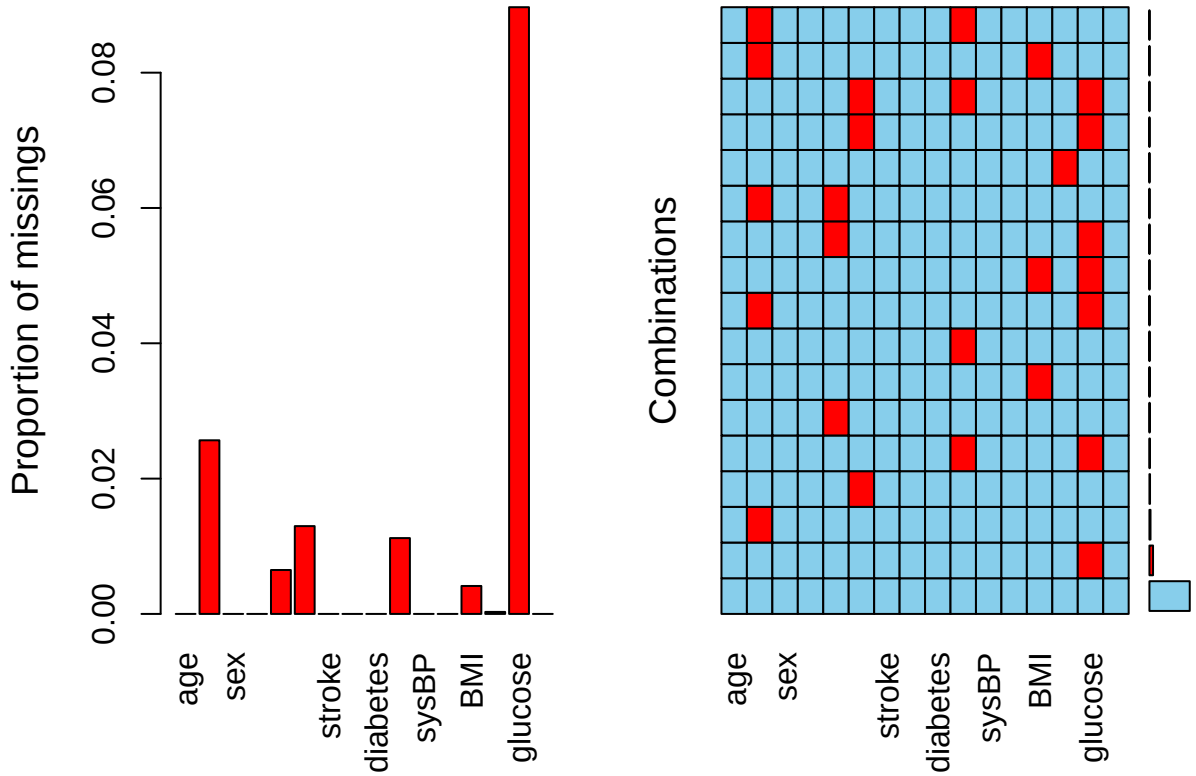
```
workPath = "D:\\DataMining\\"

cardio <- cardio_raw[,-1]

cardio$education <- factor(cardio$education)

cardio$sex <- ifelse (cardio$sex == "F", "female", "male")

cardio$is_smoking <-
  ifelse (cardio$is_smoking == "YES", "smoking", "not smoking")

colnames(cardio)[4] <- "smoking"

colnames(cardio)[6] <- "BloodPresMed"

cardio$BloodPresMed <-
  ifelse (cardio$BloodPresMed == 0, "no", "yes")

cardio$prevalentStroke <-
  ifelse (cardio$prevalentStroke == 0, "no stroke", "stroke")
colnames(cardio)[7] <- "stroke"

cardio$TenYearCHD <-
  ifelse (cardio$TenYearCHD == 0, "healthy", "CHD")

colnames(cardio)[8] <- "hypertensive"
cardio$hypertensive <-
  ifelse (cardio$hypertensive == 0, "no hypertensive", "hypertensive")

cardio$diabetes <-
  ifelse (cardio$diabetes == 0, "no diabetes", "diabetes")

colnames(cardio)[ncol(cardio)] <- "target"

## hilfsdatensätze
cardio_chd = subset(cardio, target == "CHD")
cardio_healthy = subset(cardio, target == "healthy")

saveRDS(cardio, file  = paste (workPath , "cardio.rds", sep = ""))
```

Missing Data

Glukose hat die meisten fehlenden Werte, etwas mehr als 8 % des gesamten Datensatzes. Es gibt auch einige Beobachtungen, bei denen wichtige Variablen wie BloodPresMed und Education fehlen.

Die gute Nachricht ist, dass die Variablen age, sex , stroke ,smoking ,hypertensive, diabetes, sysBP und diaBP sind und keine fehlenden Werte aufweisen.

```
aggr(cardio)
```



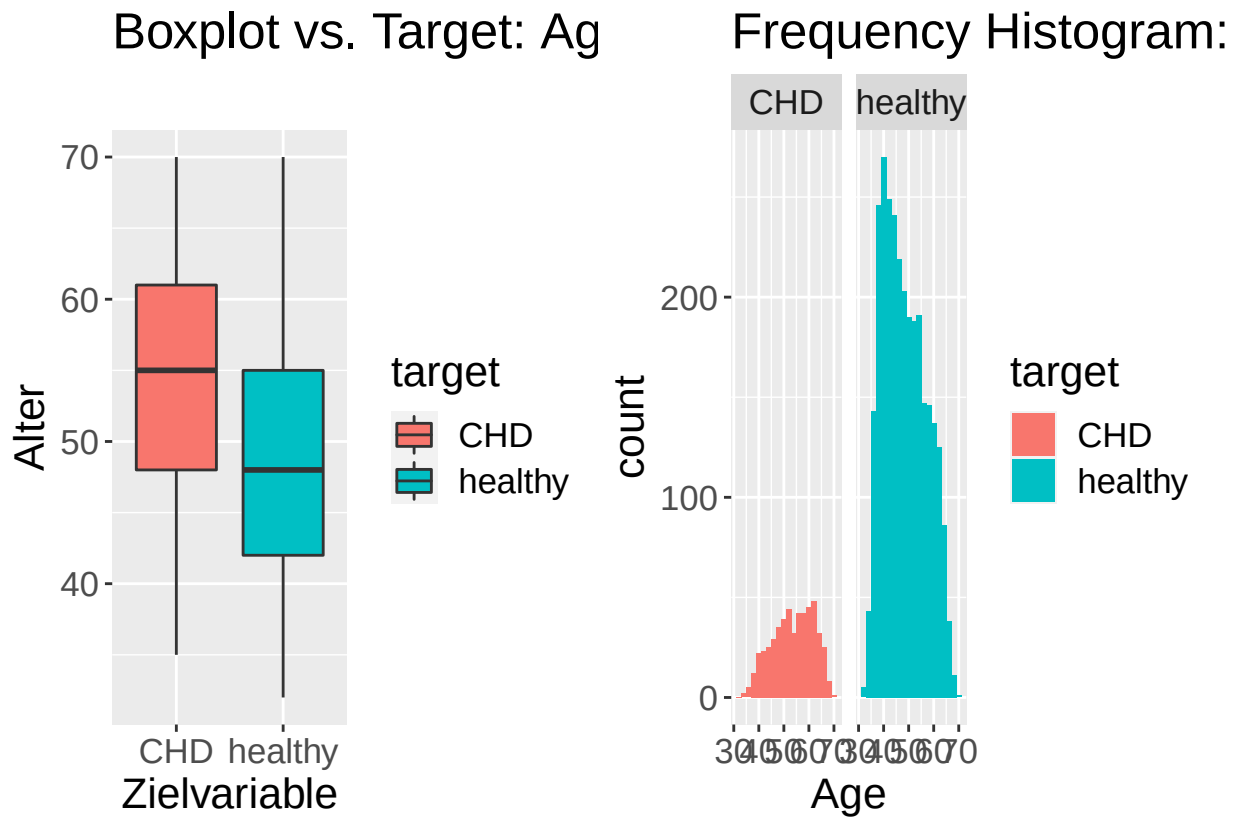
Univariate Plots

In diesem Abschnitt werde ich einen Blick auf die Verteilung der Werte für jede Variable im Datensatz werfen, indem ich Histogramme mit der ggplot-Funktion von ggplot2 erstelle. Ich versuche herauszufinden, ob es mehr Daten gibt, die bereinigt werden müssen, einschließlich Ausreißer oder fremde Werte. Dies könnte mir auch helfen, Beziehungen zwischen Variablen zu erkennen, die es wert sind, weiter untersucht zu werden.

Alter

```
#Frequency Histogram
Histogram <- cardio %>%
  ggplot(aes(x = age)) +
  geom_histogram(color = "black", fill = "#41AB5D") +
  theme(text = element_text(size = 14)) +
  labs (title = "Frequency Histogram: Age")  +
  xlab ("Age") +
  ylab  ("count")

#By Age and Target
```

```
Histogram_target <- cardio %>%
  ggplot(aes(x = age, fill = target)) +
  geom_histogram(binwidth = 2) +
  facet_wrap( ~ target) +
  theme(text = element_text(size = 16)) +
  labs (title = "Frequency Histogram: Age vs Target") +
  xlab ("Age") +
  ylab ("count")
#Boxplot
boxplot <- cardio %>%
  ggplot(aes(x = target, y = age, fill = target)) +
  geom_boxplot() +
  theme(text = element_text(size = 16)) +
  labs (title = "Boxplot vs. Target: Age") +
  xlab ("Zielvariable") +
  ylab ("Alter")
#Diagram
boxplot + Histogram_target
```




```
#pvalue
age_pvalue = t.test(cardio_chd$age,cardio_healthy$age)$p.value
```

Interpretation: Der Großteil der Patienten ist zw. 40 und 60 Jahren. Nur sehr wenige sind unter 35 bzw. über 65.

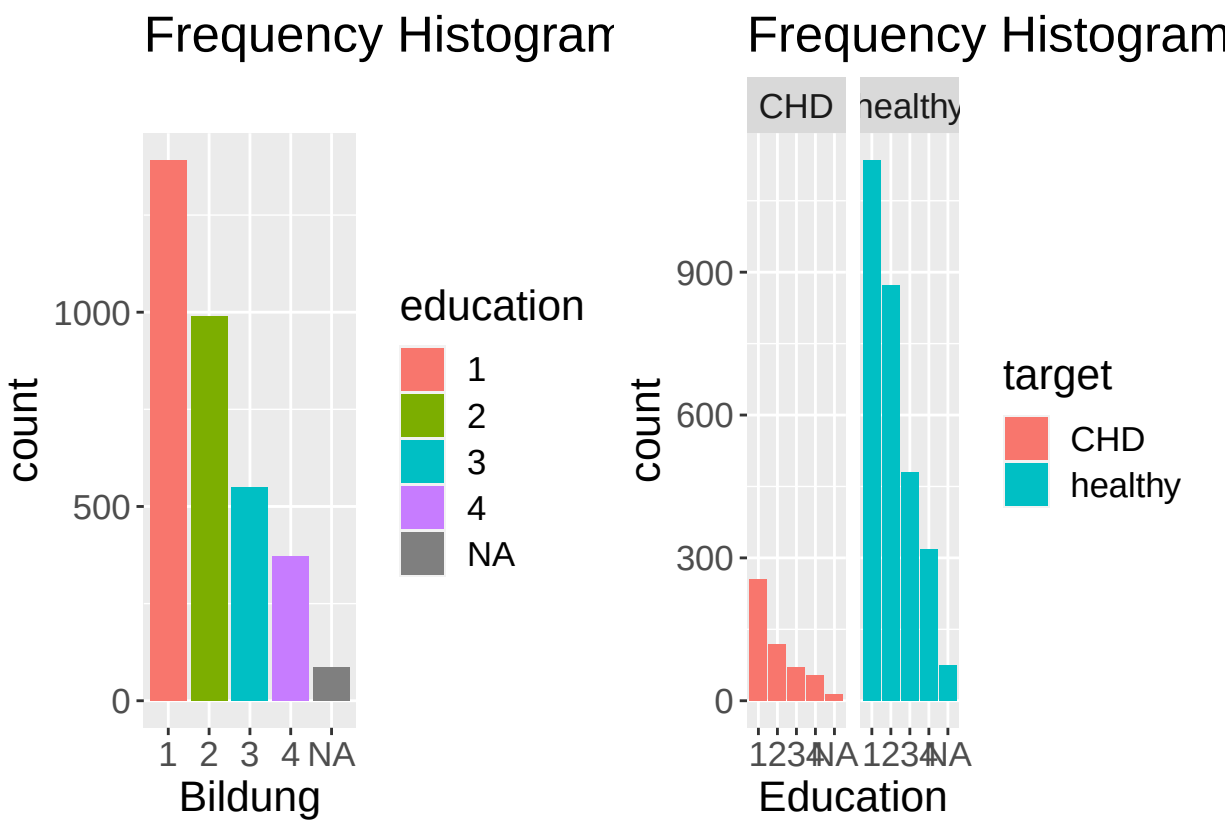
Das Alter scheint einen Einfluss auf den Gesundheitszustand zu haben. Ältere Patienten (ca. 50 bis 60 Jahre) sind im Vergleich zu jüngeren Patienten (< 50 Jahre) häufiger betroffen.

Das Alter hat jedoch nur eine begrenzte Aussagekraft, da sich die beiden Verteilungen im Bereich zwischen 45 und 55 Jahren für Gesunde und Kranke eindeutig überlappen.

Das Alter ist statistisch stark signifikant. Der p-value liegt bei 1.84555411177536e-38.

Bildungsgrad

```
# By education
Histogram<-cardio %>%
  ggplot( aes(x = education ,fill=education)) +
    geom_bar()+
  theme(text = element_text(size=16)) +
    labs ( title = "Frequency Histogram: Bildung " )  +
          xlab ("Bildung") +
          ylab  ("count")
education_level1 =round(table(cardio$education)["1"]/nrow(cardio) *100,1)
education_level2 =round(table(cardio$education)["2"]/nrow(cardio) *100,1)
education_level4 =round(table(cardio$education)["4"]/nrow(cardio) *100,1)
# By Education  and Target
target<-cardio %>%
  ggplot( aes(x = education, fill = target)) +
    geom_bar()+
    facet_wrap(~ target) +
    theme(text = element_text(size=16)) +
    labs ( title = "Frequency Histogram: Education vs Target" ) +
          xlab ("Education") +
          ylab ("count")
#Diagram
Histogram+target
```




```
#p_value
education_p_value <- fisher.test(table(cardio$target,cardio$education))$p.value
```

Interpretation:

Wie die Tabelle zeigt, 41% der Teilnehmer haben Level 1 und 29.2% Level 2.

Level 4 zeigt die Personen mit höherer Bildung und der Anteil beträgt 11%.

Interpretation:

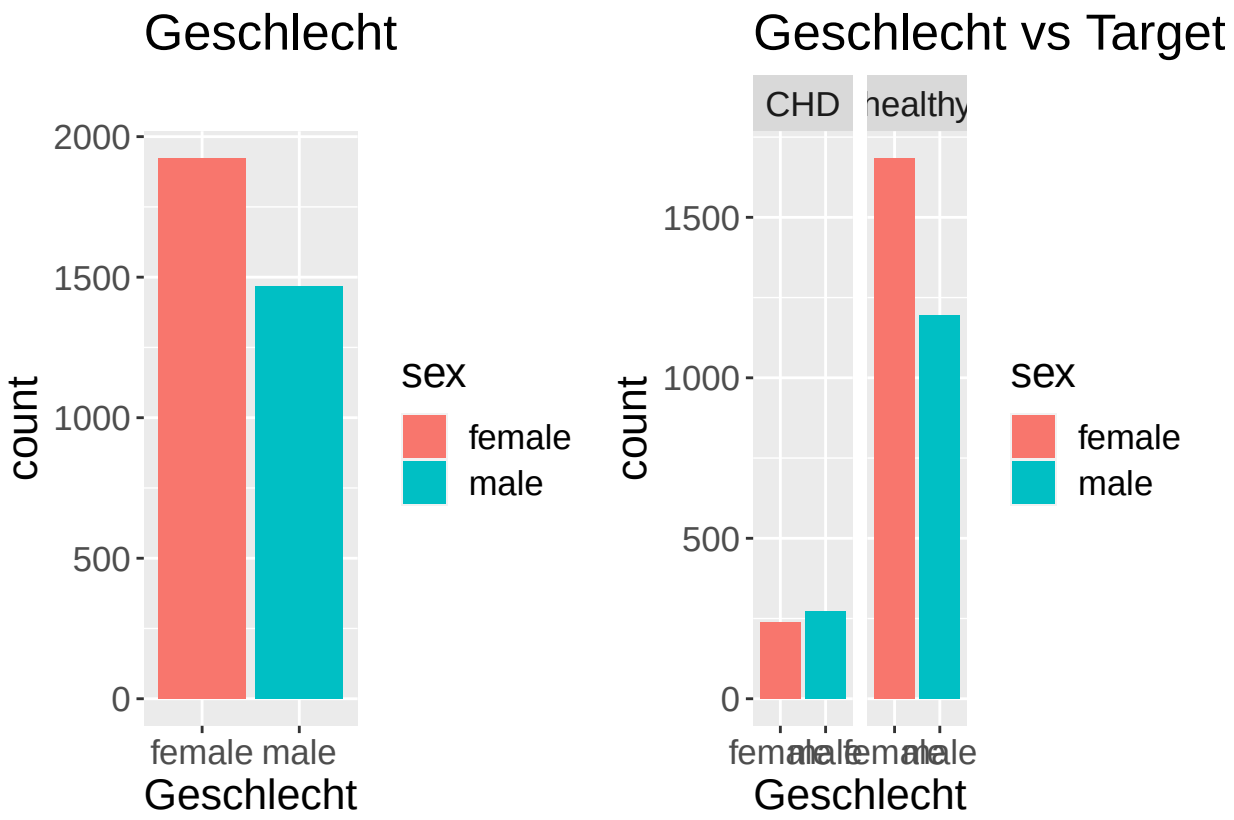
```
table(cardio$target, cardio$education)
```

```
##
##                1    2    3    4
##    CHD       256  118   70   54
##    healthy  1135  872  479  319
```

18,40 % der Telnehmer in Level 1 , 11,91 % der Telnehmer in Level 2, 12,75 % der Telnehmer in Level 3 und 14,47 % der Telnehmer in Level 4 sind an Chd erkrankt . Das heißt die Telnehmer in Level 4 , die Personen mit höherer Bildung ,haben mehr Chance um diese Krankheit zu bekommen.

Geschlecht

```r
# By sex
bar <- cardio %>%
  ggplot(aes(x = sex, fill = sex)) +
  geom_bar() +
  theme(text = element_text(size = 16)) +
  labs (title = "Geschlecht")  +
  xlab ("Geschlecht") +
  ylab  ("count")

t_femail_count = round(table(cardio$sex)["female"] / nrow(cardio) * 100, 1)
# By sex and target
target <- cardio %>%
  ggplot(aes(x = sex, fill = sex)) +
  geom_bar() +
  facet_wrap( ~ target) +
  theme(text = element_text(size = 16)) +
  labs (title = "Geschlecht vs Target")  +
  xlab ("Geschlecht") +
  ylab  ("count")
t_femail_chd_count = round(table(cardio_chd$sex)["female"] / nrow(cardio_chd) *
                             100, 1)
t_femail_healthy_count = round(table(cardio_healthy$sex)["female"] / nrow(cardio_healthy) *
                                 100, 1)
#Diagram
bar + target
```



```
sex_pvalue = fisher.test(table(cardio$target,cardio$sex))$p.value
```

Interpretation: Es sind 56.7 % der Patienten weiblich.

Interpretation:

Männer haben ein erhöhtes Risiko die Erkrankung zu entwickeln. Obwohl Frauen in dem Datensatz mit 56.7 % vorkommen liegt der Anteil der Frauen bei den erkrankten bei nur 46.8 % und bei den gesunden bei 58.5 %.

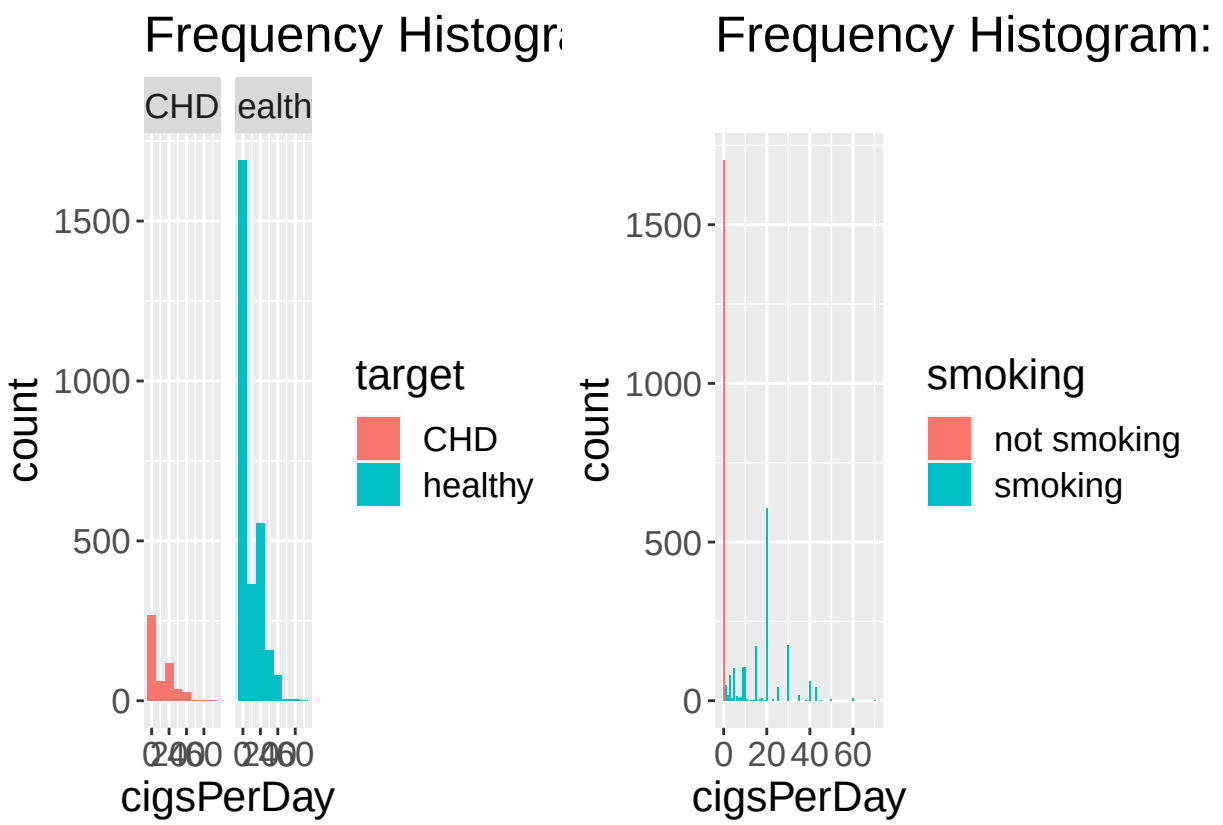
Der Effekt des Geschlechtes ist statistisch signifikant. Der P-Value liegt bei 9.50443889414983e-07.

Cigs Per Day

```
# By Cigs Per Day
Frequency<-cardio %>%
  ggplot( aes(x =cigsPerDay ,fill=smoking))  +
    geom_bar()+

  theme(text = element_text(size=16)) +
    labs ( title = "Frequency Histogram: cigsPerDay" )  +
          xlab ("cigsPerDay") +
          ylab  ("count")
histogram<-cardio %>%
  ggplot( aes(x = cigsPerDay, fill = target)) +
    geom_histogram(binwidth =   10)+
    facet_wrap(~ target) +
    theme(text = element_text(size=16)) +
    labs ( title = "Frequency Histogram: CigsPerDay vs Target" ) +
          xlab ("cigsPerDay") +
          ylab ("count")
histogram+Frequency
```

```
## Warning: Removed 22 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 22 rows containing non-finite values (stat_count).
```



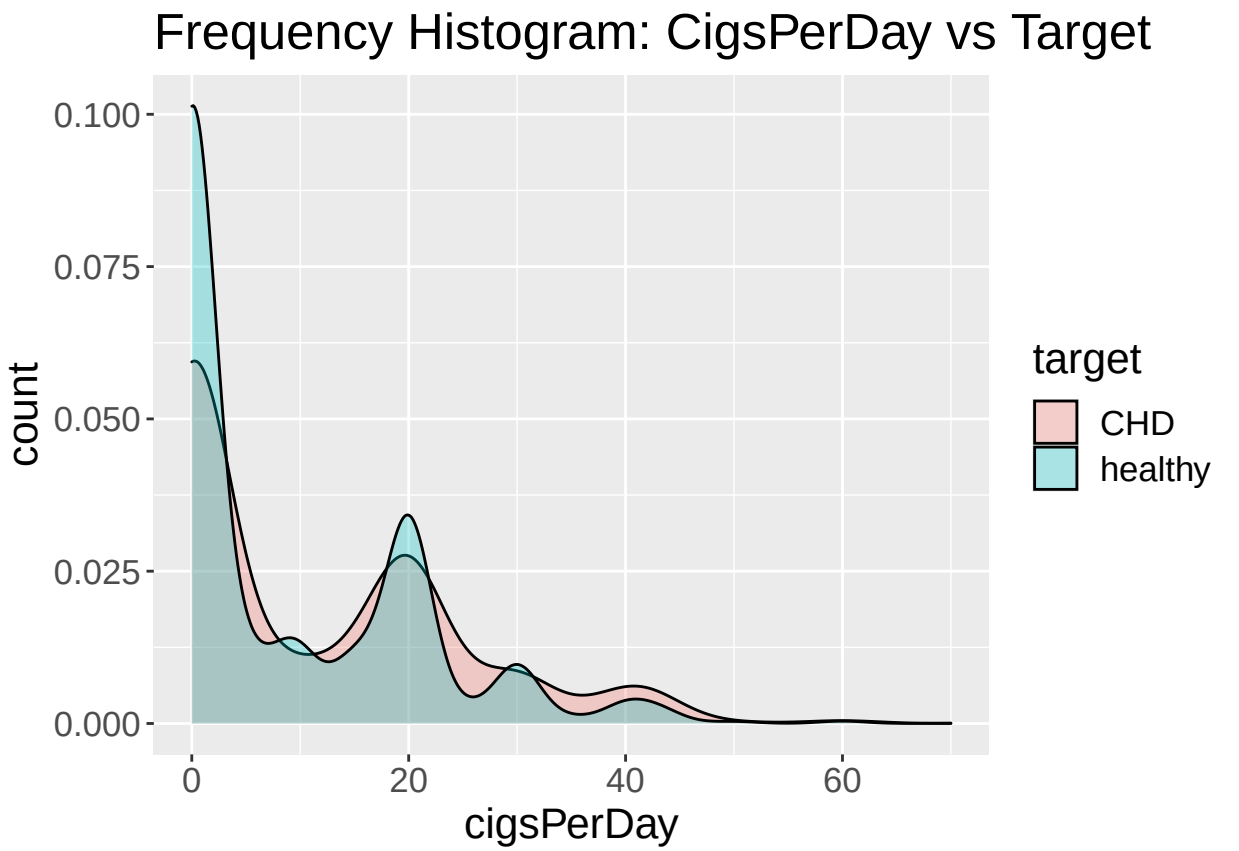

```
smoking_pvalue=fisher.test(table(cardio$target, cardio$smoking))$p.value
```

Interpretation:

" Cigs Per Day" zeigt : Anzahl der Zigaretten, die die Person im Durchschnitt an einem Tag geraucht hat. Die Mehrheit der Raucher raucht 20 Zigaretten pro Tag.

```
cardio %>%
  ggplot( aes(x = cigsPerDay, fill = target)) +
    geom_density(alpha = 0.3)+
    #facet_wrap(~ target) +
    theme(text = element_text(size=16)) +
    labs ( title = "Frequency Histogram: CigsPerDay vs Target" ) +
          xlab ("cigsPerDay") +
          ylab ("count")
```

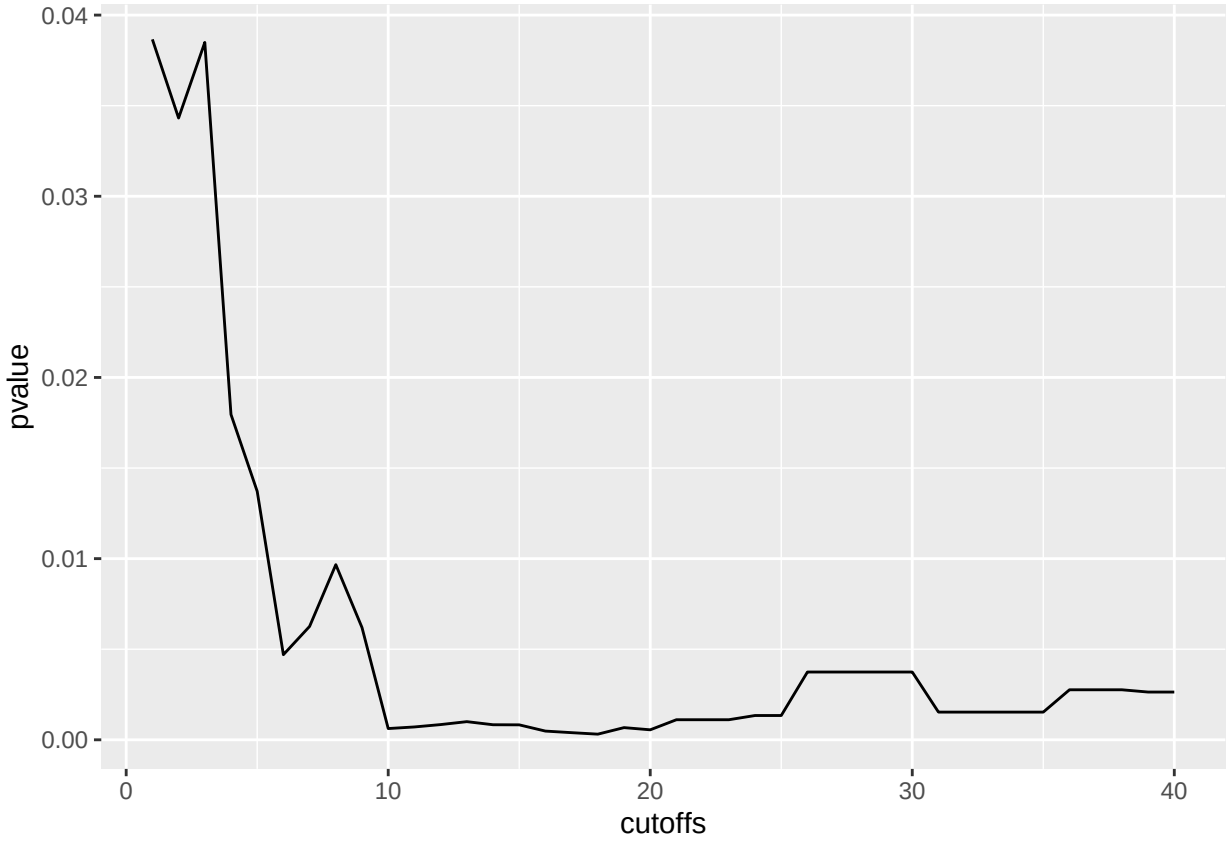
```
## Warning: Removed 22 rows containing non-finite values (stat_density).
```




```
cigsPerDay_pvalue = t.test(cardio_chd$cigsPerDay, cardio_healthy$cigsPerDay)$p.value
```

Die Anzahl an Zigaretten pro Tag hat einen Effekt auf das Risiko zu erkranken. Es wird die Anzahl an Zigaretten gesucht, welche den Effekt auf das Krankheitsrisiko maximiert.

```
#fisher.test(table(cardio$target, cardio$cigsPerDay >= 10 ))$p.value

cutoffs <- seq(1,40)
cigsPerDay_pvalues = rep (NA, length(cutoffs))
for (i in 1:length(cutoffs)) {
   cigsPerDay_pvalues[i]= fisher.test(table(cardio$target, cardio$cigsPerDay >= cutoffs[i] ))$p.value
}
data.frame(cutoffs = cutoffs, pvalue = cigsPerDay_pvalues ) %>%
  ggplot(aes (cutoffs,pvalue )) +
  geom_line()
```




```
min_idx <- which.min (cigsPerDay_pvalues)
print (paste("Bei einem Cutoff von <= " ,cutoffs[min_idx] , " ergibt sich der särkste statistische Effe
```

```
## [1] "Bei einem Cutoff von <=  18  ergibt sich der särkste statistische Effekt. Hier ergibt sich ein
```

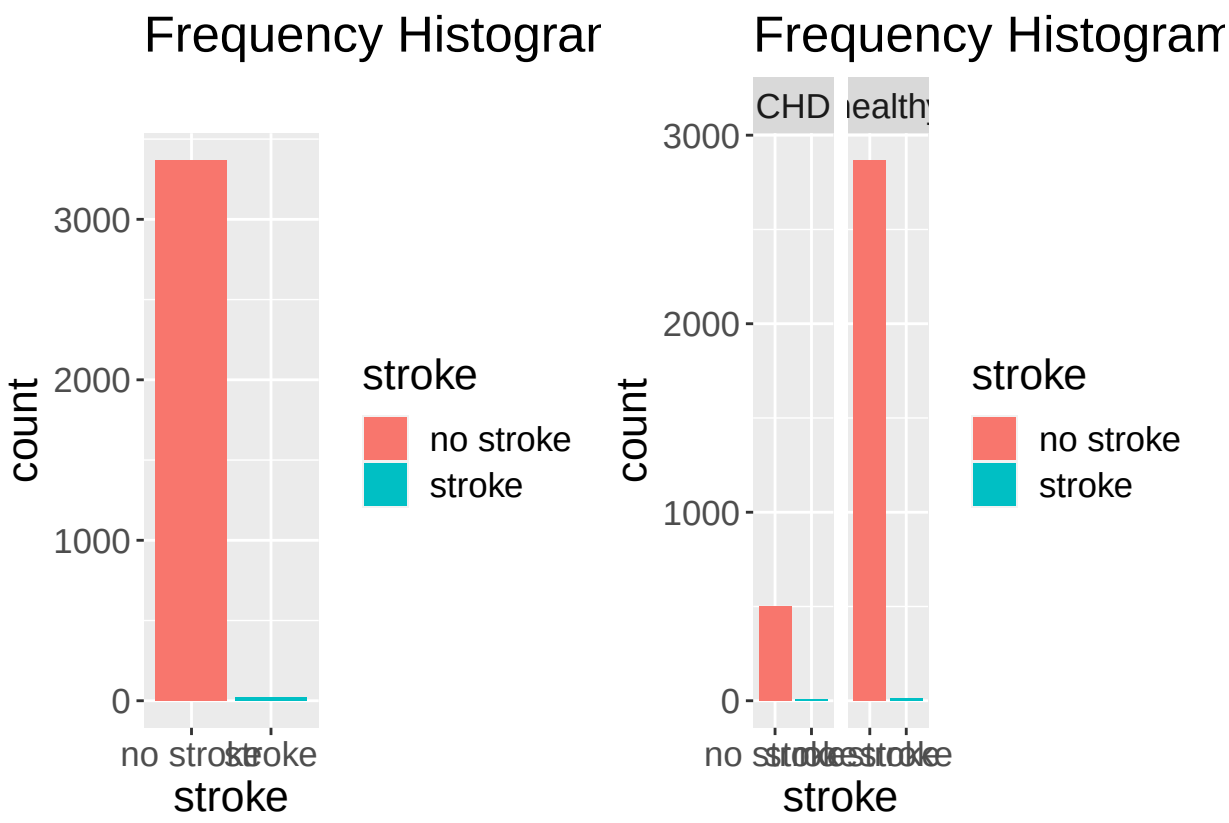
```
cardio[1:10, c("target","smoking","cigsPerDay")]
```

```
##     target     smoking cigsPerDay
## 1      CHD     smoking          3
## 2  healthy not smoking          0
## 3  healthy     smoking         10
## 4      CHD     smoking         20
## 5  healthy     smoking         30
## 6      CHD not smoking          0
## 7  healthy not smoking          0
## 8  healthy     smoking         35
## 9  healthy     smoking         20
## 10 healthy not smoking          0
```

Interpretation: Die Personen, die weniger als 18 Zigaretten pro Tag rauchen, haben ein geringes Risiko, an CHD zu erkranken.

stroke

```
#stroke
geom_bar<-cardio %>%
  ggplot( aes(x = stroke,fill=stroke)) +
    geom_bar()+

  theme(text = element_text(size=16)) +
    labs ( title = "Frequency Histogram: stroke" )  +
          xlab ("stroke") +
          ylab  ("count")


t_stroke_count =round(table(cardio$stroke)["stroke"]/nrow(cardio) *100,1)

#Frequency Histogram
 Histogram<-cardio %>%
  ggplot( aes(x = stroke,fill=stroke)) +
    geom_bar()+
  facet_wrap(~target) +
  theme(text = element_text(size=16)) +
    labs ( title = "Frequency Histogram: stroke" )  +
          xlab ("stroke") +
          ylab  ("count")

t_stroke_chd_count =round(table(cardio_chd$stroke )["stroke"]/nrow(cardio_chd) *100,1)

t_stroke_healthy_count =round(table(cardio_healthy$stroke )["stroke"]/nrow(cardio_healthy) *100,1)

geom_bar+Histogram
```




```
stroke_pvalue =fisher.test(table(cardio$target, cardio$stroke))$p.value
```

Interpretation:

" stroke" zeigt : ob der Teilnehmer zuvor einen Schlaganfall hatte oder nicht . 0.6 % der Teilnehmer hatten zuvor einen Schlaganfall gehabt.

Der Anteil von der Teilnehmer, die zuvor einen Schlaganfall hatten, ist bei den Gesunden 0.4% und bei den Kranken 0.4%

Der Anteil von der Teilnehmer, die zuvor einen Schlaganfall hatten, ist bei den Gesunden und bei den Kranken ist fast gleich.

Ein vorheriger Schlaganfall hat einen statistisch signifikaten Effekt. Der P-value liegt bei 0.000647033881655412.

pvalue

```
glucose_pvalue = t.test (cardio_chd$glucose, cardio_healthy$glucose)$p.value

BMI_pvalue = t.test (cardio_healthy$BMI, cardio_chd$BMI)$p.value
t_sysBP_count =round(table(cardio$sysBP)["sysBP"]/nrow(cardio) *100,1)

diaBP_pvalue = t.test (cardio_healthy$diaBP, cardio_chd$diaBP)$p.value

sysBP_pvalue = t.test (cardio_healthy$sysBP, cardio_chd$sysBP)$p.value
```

```
totChol_pvalue = t.test (cardio_healthy$totChol, cardio_chd$totChol)$p.value

stroke_pvalue =fisher.test(table(cardio$target, cardio$stroke))$p.value


hypertensive_pvalue=fisher.test(table(cardio$target, cardio$hypertensive))$p.value

BloodPresMed_pvalue=fisher.test(table(cardio$target, cardio$BloodPresMed))$p.value

heartRate_pvalue = t.test (cardio_healthy$heartRat, cardio_chd$heartRat)$p.value
```

Zusammenfassung

Tabelle mit jedem Feature ist eine Zeile.

Featurename, cardinal, ordinal oder nominal, Effekt, p-value, Anzahl an missing values

```
feature_summary = data.frame(feature = colnames(cardio)[1:ncol(cardio)-1])
feature_summary$art = c("nominal","ordinal","cardinal","cardinal","nominal","cardinal","cardinal","cardi

feature_summary$effekt =c("höheres Alter hat höheres Risiko"
                          , "education_inter"
                          , "Männer erkranken häufiger als Frauen  "
                          ,"der Effekt in beiden fast gleich"
                          , "weniger als 20 Zigaretten,geringes Risiko",
                          "meisten nehmen keine BPMeds ein",
                          "die Wirkung ist nicht effektiv",
                          "mit hypertensive ,höheres Risiko  ",
                         "Diabetes-Patienten haben ein höheres Risiko",
                         "der Effekt in beiden fast gleich",
                         "mit sysBP,höheres Risiko",
                         "höherer diaBP,höheres Risiko",
                         "höherer BMI hat höheres Risiko",
                         "der Effekt in beiden fast gleich",
                         "Glucose-Patienten haben ein höheres Risiko ")

feature_summary$p_value = c(toString(age_pvalue),toString(education_p_value), sex_pvalue,smoking_pvalue
feature_summary$nbr_missing = ""


for (ir in 1:nrow(feature_summary)) {
    tmp_col <- cardio[,feature_summary$feature[ir] ]
    feature_summary$nbr_missing[ir] <- nrow(cardio) -  table(is.na(tmp_col))["FALSE"]
}

knitr::kable(feature_summary)
```

feature	art	effekt	p_value	nbr_missing
age	nominal	höheres Alter hat höheres Risiko	1.84555411177536e-38	0

feature	art	effekt	p_value	nbr_missing
education	ordinal	education_inter	6.68203706919536e-05	87
sex	cardinal	Männer erkranken häufiger als Frauen	9.50443889414983e-07	0
smoking	cardinal	der Effekt in beiden fast gleich	0.0490648844448193	0
cigsPerDay	nominal	weniger als 20 Zigaretten,geringes Risiko	0.000373330337109478	22
BloodPresMed	cardinal	meisten nehmen keine BPMeds ein	5.23410965432645e-06	44
stroke	cardinal	die Wirkung ist nicht effektiv	0.000647033881655412	0
hypertensive	cardinal	mit hypertensive ,höheres Risiko	4.91775745429724e-21	0
diabetes	cardinal	Diabetes-Patienten haben ein höheres Risiko	8.89527945660407e-12	0
totChol	nominal	der Effekt in beiden fast gleich	5.18268946012133e-07	38
sysBP	nominal	mit sysBP,höheres Risiko	5.51533444666561e-24	0
diaBP	nominal	höherer diaBP,höheres Risiko	8.89527945660407e-12	0
BMI	nominal	höherer BMI hat höheres Risiko	0.000414155694522985	14
heartRate	nominal	der Effekt in beiden fast gleich	0.245536931489089	1
glucose	nominal	Glucose-Patienten haben ein höheres Risiko	3.66688738012339e-06	304

Prediction

Autor: “Ali Abdorrahimi”
Studiengang: “Digital Engineering”
Fach: “Data Mining”
Semester : “Wintersemester 2022”

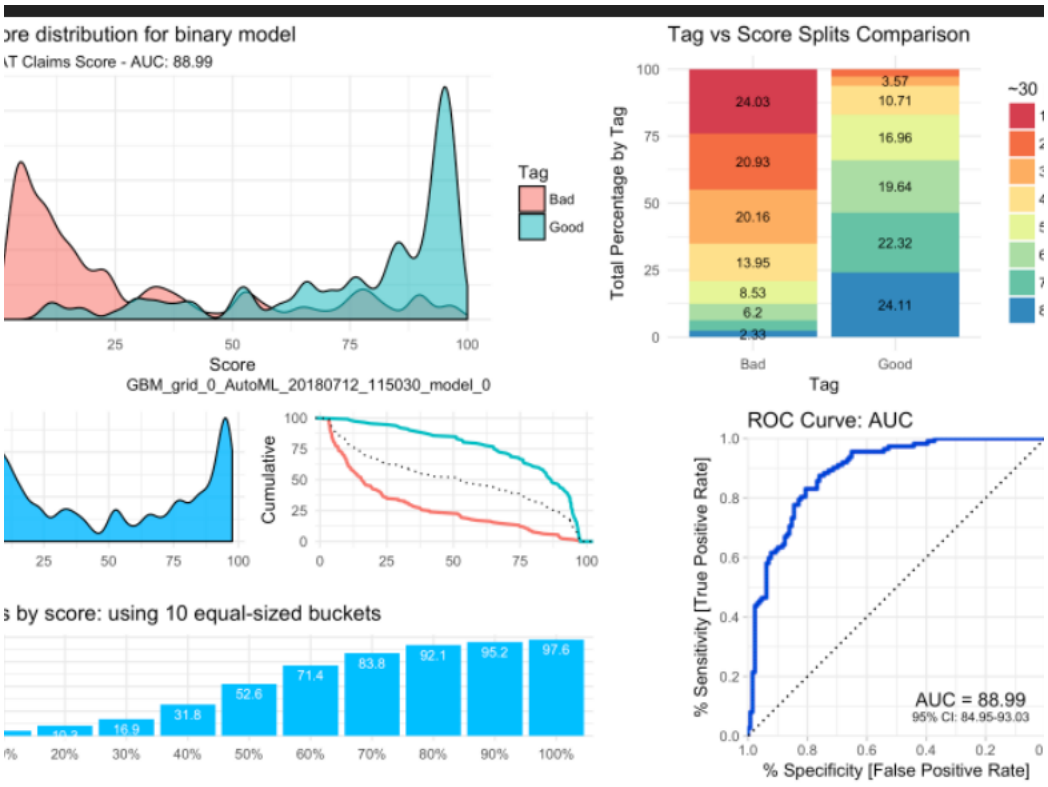


Figure 1: prediction

Introduction

In der R-Programmierung sind Prognosemodelle äußerst nützlich für die Vorhersage zukünftiger Ergebnisse .

Mit Hilfe von Modellen können Sie zukünftiges Verhalten auf der Grundlage von vergangenem Verhalten vorhersagen. Nachdem Sie ein Modell aufgebaut haben, verwenden Sie es, um neue Daten zu bewerten, d. h. um Vorhersagen zu treffen.

Mit R können Sie viele Arten von Modellen erstellen.

these R models:

glm : Generalized linear models

lda : Linear Discriminant Analysis

knn : K-nearest Neighbors Algorithm

rpart: Recursive Partitioning and Regression Trees

kmeans: k-Means clustering

rf : Random Forest

Load Libraries

```
library(magrittr)
library(plyr)
library(dplyr)
library(ggplot2)
library(grid)
library(gridExtra)
library(stringr)
library(caret)
library(mvtnorm)
library(ggplot2)
library(randomForest)
library(caret)
library(pROC)
library(klaR)
library(psych)
library(MASS)
library(devtools)
library(ROCR)
```

Load Data

```
workPath ="D:\\DataMining\\"
cardio <- readRDS(paste (workPath , "cardio.rds", sep =""))
```

Train test split

Wenn wir Modelle für maschinelles Lernen erstellen, müssen wir unser Modell auf einer Teilmenge der verfügbaren Daten trainieren und die Genauigkeit des Modells auf einer Teilmenge der Daten testen.

verwenden wir die Funktion complete.cases() in R, um (Missing values) in einem Vektor, einer Matrix oder einem Datenrahmen zu entfernen.

```
cardio_complete <- cardio[(complete.cases(cardio)),]
nrow(cardio_complete)
```

```
## [1] 2927
```

Nach der Entfernung der Missing Valuses haben wir 2927 Datensätze, von denen wir 1500 Datensätze für den Test verwenden werden.

```
set.seed(1001)

n_test <- 1500

idx_test <- sample(1:nrow(cardio_complete), n_test)

cardio_test <- cardio_complete[ idx_test,]
cardio_train <- cardio_complete[ -idx_test,]
```

Train first models

Accuarcy ist ein Maßstab für die Bewertung von Klassifizierungsmodellen und

Formal ist die accuracy wie folgt definiert:

$$Accuracy = \frac{NumberofCorrectPrediction}{TotalNumberofPredicted}$$

```
control <- trainControl(method="cv", number=5)
metric <- "Accuracy"
```

The train function can be used to:

- evaluate, using resampling, the effect of model tuning parameters on performance
- choose the "optimal" model across these parameters
- estimate model performance from a training set

```
mod.glm <- train(target~., data=cardio_train, method = "glm",family = "binomial",metric=metric,trContro
```

The Kappa statistic (or value) is a metric that compares an Observed Accuracy with an Expected Accuracy (random chance). The kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves. In addition, it takes into account random chance (agreement with a random classifier), which generally means it is less misleading than simply using accuracy as a metric (an Observed Accuracy of 80% is a lot less impressive with an Expected Accuracy of 75% versus an Expected Accuracy of 50%). Computation of Observed Accuracy and Expected Accuracy is integral to comprehension of the kappa statistic, and is most easily illustrated through use of a confusion matrix

```
mod.lda <- train(target~., data=cardio_train, method="lda",metric=metric,trControl=control)
mod.lda$results
```

```
##   parameter  Accuracy    Kappa AccuracySD  KappaSD
## 1      none 0.8507324 0.128926 0.01813854 0.107302
```

```
mod.cart <- train(target~., data=cardio_train, method="rpart",metric=metric,trControl=control)
mod.cart$results
```

```
##            cp  Accuracy      Kappa  AccuracySD    KappaSD
## 1 0.007898894 0.8437370 0.15153304 0.007838914 0.08565146
## 2 0.009478673 0.8437370 0.11206497 0.008588368 0.06216736
## 3 0.026066351 0.8528401 0.01988439 0.002304735 0.04446285
```

```
mod.knn <- train(target~., data=cardio_train, method="knn", metric=metric, trControl=control)
mod.knn$results
```

```
##   k  Accuracy      Kappa  AccuracySD    KappaSD
## 1 5 0.8374310 0.09635543 0.008192540 0.03566687
## 2 7 0.8437419 0.07216125 0.010423923 0.05015822
## 3 9 0.8437370 0.02081593 0.007015914 0.04194563
```

```
mod.svm.radial <- train(target~., data=cardio_train, method="svmRadial", metric=metric, trControl=contr

mod.svm.radial$results
```

```
##       sigma    C  Accuracy      Kappa  AccuracySD   KappaSD
## 1 0.0461617 0.25 0.8521401 0.00000000 0.001098966 0.0000000
## 2 0.0461617 0.50 0.8521401 0.00000000 0.001098966 0.0000000
## 3 0.0461617 1.00 0.8521352 0.01290007 0.001741712 0.0211263
```

```
mod.svm.linear <- train(target~., data=cardio_train, method="svmLinear", metric=metric, trControl=contr
mod.svm.linear$results
```

```
##   C  Accuracy        Kappa  AccuracySD    KappaSD
## 1 1 0.8507349 -0.002760661 0.001986759 0.00378019
```

Support Vector Machine

```
mod.rf <- train(target~., data=cardio_train, method="rf", metric=metric, trControl=control)
mod.rf$results
```

```
##   mtry  Accuracy      Kappa  AccuracySD    KappaSD
## 1    2 0.8514366 0.01155700 0.001892435 0.01602853
## 2    9 0.8479328 0.07532032 0.003120357 0.04284008
## 3   17 0.8444265 0.10546684 0.016218403 0.08464020
```

The SVM is a state-of-the-art maximum margin classification algorithm rooted in statistical learning theory. SVM is method for classification of both linear and non-linear data. It uses a non-linear mapping to transform the original training data into a higher dimension. Within this new dimension it searches for linear optimal separating hyperplane. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM find this hyperplane using support vectors and margins [13]. SVM performs classification tasks by maximizing the margin separating both classes while minimizing the classification errors

summarize all models

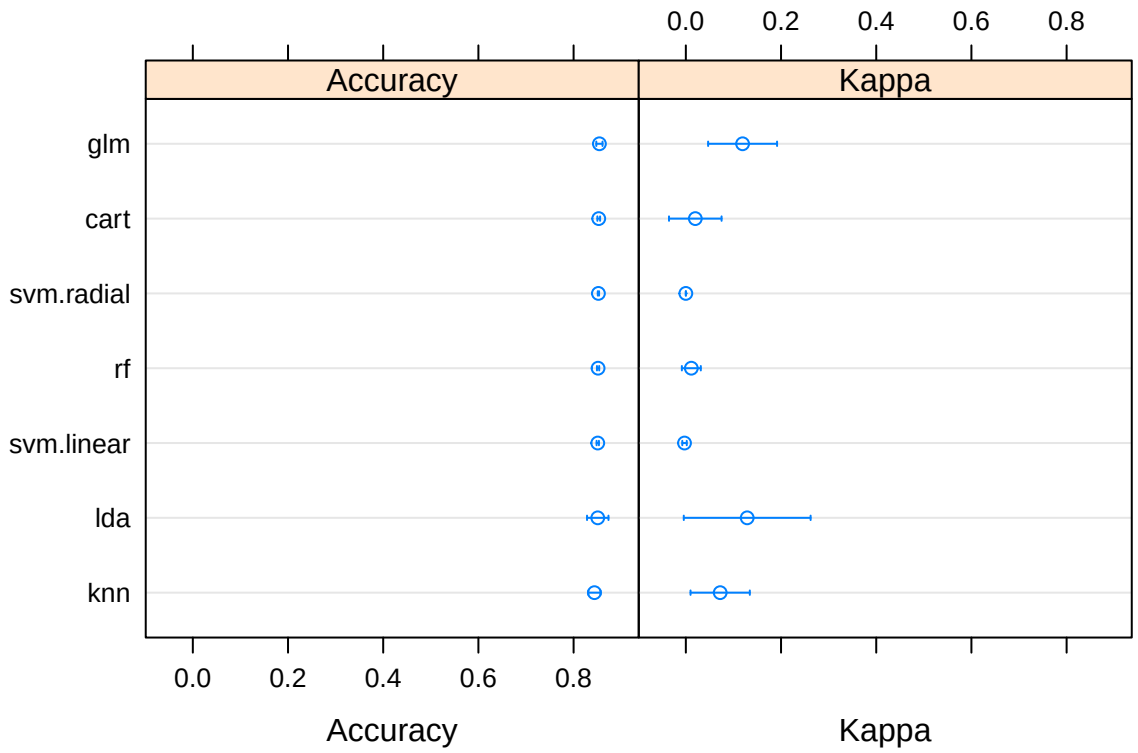
```
results <- resamples(list(glm = mod.glm , lda= mod.lda, cart=mod.cart, knn=mod.knn, svm.radial=mod.svm.
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: glm, lda, cart, knn, svm.radial, svm.linear, rf
## Number of resamples: 5
##
## Accuracy
##                 Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm        0.8491228 0.8526316 0.8526316 0.8542405 0.8536585 0.8631579    0
## lda        0.8286713 0.8421053 0.8491228 0.8507324 0.8561404 0.8776224    0
## cart       0.8496503 0.8526316 0.8526316 0.8528401 0.8531469 0.8561404    0
## knn        0.8321678 0.8356643 0.8421053 0.8437419 0.8526316 0.8561404    0
## svm.radial 0.8501742 0.8526316 0.8526316 0.8521401 0.8526316 0.8526316    0
## svm.linear 0.8491228 0.8491228 0.8496503 0.8507349 0.8526316 0.8531469    0
## rf         0.8491228 0.8496503 0.8526316 0.8514366 0.8526316 0.8531469    0
##
## Kappa
##                     Min.      1st Qu.     Median         Mean    3rd Qu.
## glm         0.062940348  0.083875308 0.11596278  0.119247051 0.11842687
## lda         0.019862918  0.055491329 0.12055133  0.128926043 0.15197039
## cart        0.000000000  0.000000000 0.00000000  0.019884393 0.00000000
## knn         0.026244857  0.028054953 0.06294035  0.072161253 0.09942197
## svm.radial  0.000000000  0.000000000 0.00000000  0.000000000 0.00000000
## svm.linear -0.006901651 -0.006901651 0.00000000 -0.002760661 0.00000000
## rf          0.000000000  0.000000000 0.00000000  0.011557004 0.02529229
##                  Max. NA's
## glm        0.21502996    0
## lda        0.29675425    0
## cart       0.09942197    0
## knn        0.14414414    0
## svm.radial 0.00000000    0
## svm.linear 0.00000000    0
## rf         0.03249273    0
```

```
saveRDS(results,file  =paste (workPath , "results.rds", sep =""))
```

plot accuracies

```
dotplot(results)
```



Confidence Level: 0.95

Analyse der ersten Modelle

AUC (Area Under The Curve) Die ROC-Kurve (Receiver Operating Characteristics) ist eines der wichtigsten Bewertungsmaße für die Beurteilung der Leistung von binären Klassifikationsproblemen. ROAC ist eine Wahrscheinlichkeitskurve, die die TPR (True Positive Rate) gegen die FPR (False Positive Rate) aufträgt. AUC ist das Maß für die Trennbarkeit und zeigt an, wie gut unser Modell in der Lage ist, zwischen den Klassen zu unterscheiden. Die AUC gibt an, wie gut das Modell zwischen positiven und negativen Klassen unterscheidet. Je größer der AUC, desto besser.

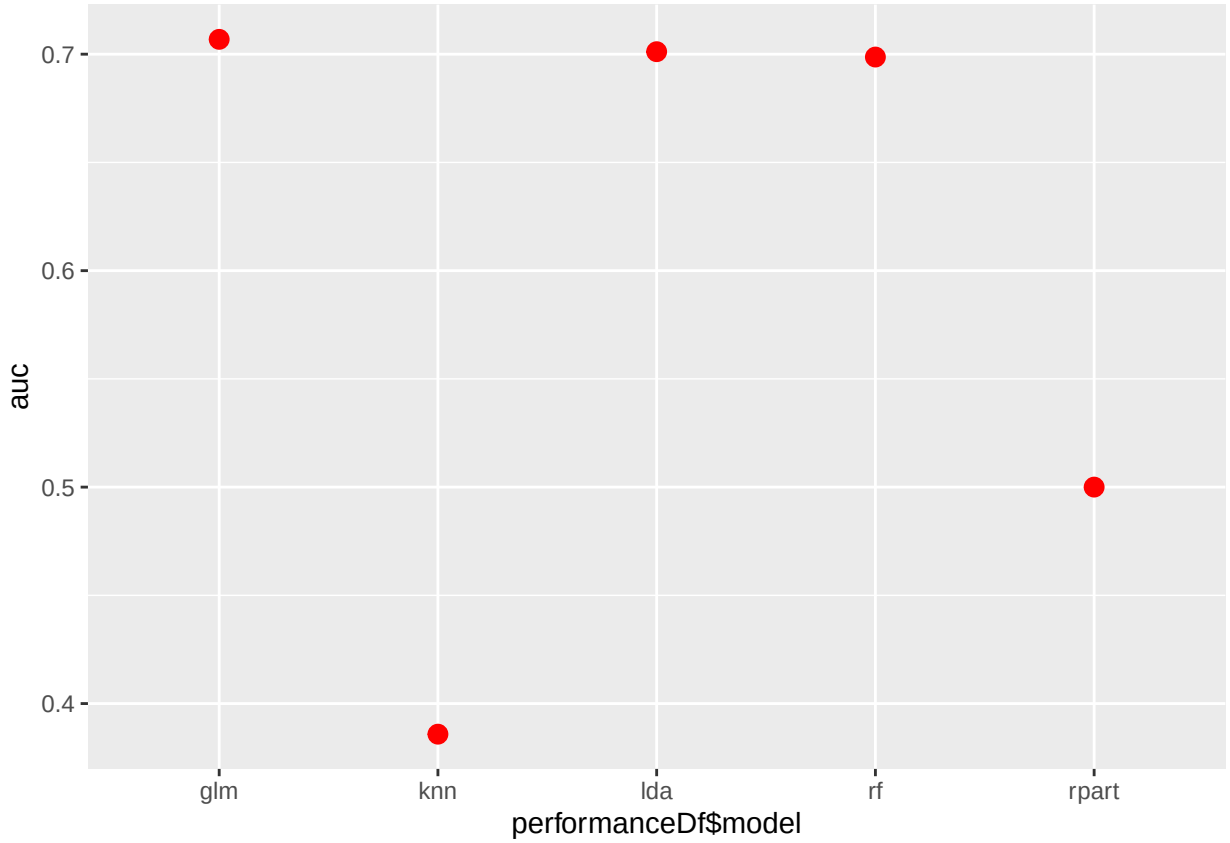
```
computeData <- F
models <- c("rf", "lda", "rpart", "knn", "glm")
workPath = "D:\\DataMining\\Seafile\\01_cardio\\Projekt\\"

if (computeData) {
  performanceDf <- data.frame(model = models)
  performanceDf$accuracy <- NA
  performanceDf$sensitivity <- NA
  performanceDf$specificity <- NA
  performanceDf$auc <- NA
  for (ir in 1:nrow (performanceDf)) {
    mod <-
      train(
        target ~ .,
        data = cardio_train,
        method = performanceDf$model[ir]
```

```
        ,
        metric = metric,
        trControl = control
      )

    y_pred_prob <- predict(mod ,  cardio_test, type = "prob")

    y_pred <- predict(mod ,  cardio_test)

    table(y_pred, cardio_test$target)

    accuracy = table(y_pred == cardio_test$target)["TRUE"] / length (y_pred)

    performanceDf$accuracy[ir] = accuracy

    sensitivity   = sensitivity(factor(y_pred), factor(cardio_test$target))

    performanceDf$sensitivity[ir] = sensitivity

    specificity = specificity(factor(y_pred), factor(cardio_test$target))
    performanceDf$specificity[ir] = specificity

    auc <- roc(cardio_test$target , y_pred_prob$CHD)$auc
    performanceDf$auc[ir] = auc

  }
  saveRDS(performanceDf, file  = paste (workPath , "performanceDf.rds", sep =
                                          ""))
}
performanceDf <-
  readRDS(paste (workPath , "performanceDf.rds", sep = ""))
performanceDf
```

```
##   model  accuracy sensitivity specificity       auc
## 1    rf 0.8440000  0.01287554   0.9968429 0.6986715
## 2   lda 0.8380000  0.10729614   0.9723757 0.7011527
## 3 rpart 0.8446667  0.00000000   1.0000000 0.5000000
## 4   knn 0.8420000  0.04291845   0.9889503 0.3858732
## 5   glm 0.8453333  0.09871245   0.9826361 0.7068537
```

```
ggplot(performanceDf, aes(x =performanceDf$model , y = auc )) +
   geom_point(colour = "red",size = 3)
```

```
## Warning: Use of ‘performanceDf$model‘ is discouraged. Use ‘model‘ instead.
```



Die Modelle Linear Discriminant Analysis (lda) und generalization of ordinary linear regression (glm)zeigten das beste Ergebnis mit einem auc von 0.7011527 und 0.7068537.

as Modell k nearest neighbor hat die schlechteste Performance mit einem auc von 0,3858732.

Featurte importance / Feature selection

2022-04-26

In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for several reasons:

simplification of models to make them easier to interpret by researchers/users shorter training times *to avoid the curse of dimensionality* improve data's compatibility with a learning model class *encode inherent symmetries present in the input space.

we know from the prediction that the rf and lda and glm models have the best performance compared to the other models, so we will only apply the Feaure selection algorithm for these 3 models.

Load Data

```
workPath ="D:\\DataMining\\"
cardio <- readRDS(paste (workPath , "cardio.rds", sep =""))
```

Train test split

```
set.seed(1001)

cardio_complete <- cardio[(complete.cases(cardio)),]

nrow(cardio_complete)

## [1] 2927

n_test <- 1500

idx_test <- sample(1:nrow(cardio_complete), n_test)

cardio_test <- cardio_complete[ idx_test,]
cardio_train <- cardio_complete[ -idx_test,]

control <- trainControl(method="cv", number=5)
metric <- "Accuracy"
```

Greedy Forward Selection

Forward stepwise selection is a variable selection method which:

Begins with a model that contains no variables (called the Null Model) Then starts adding the most significant variables one after the other Until a pre-specified stopping rule is reached or until all the variables under consideration are included in the mode.

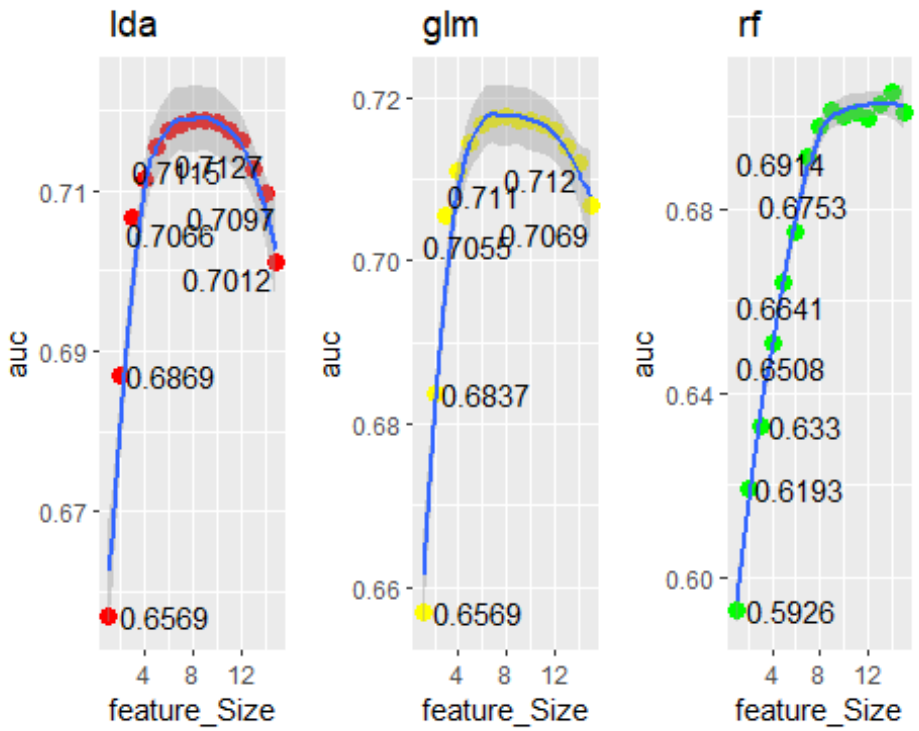
```
computeData <- F
workPath = "D:\\DataMining\\Seafile\\01_cardio\\Projekt\\"

modell <- c("glm", "lda", "rf")

if (computeData) {

sff_performanceDf_Forward = expand.grid(feature_Size = seq(1, 15), model =
modell)

sff_performanceDf_Forward$auc <- NA

sff_performanceDf_Forward$festure <- ""

sff_performanceDf_Forward$Bestfesture <- NA

cardio_train_rf <- cardio_train

nbr_features <- 15
aucperfor <- NA



for (model in  c("glm", "lda", "rf"))
{
  sff_featureset <- c()

  while (length(sff_featureset) < nbr_features) {
    featutes_to_test <- setdiff(rel_features, sff_featureset)

    aucperfor <- rep(NA, length(featutes_to_test))

    for (i_featutes_to_test  in 1:length(featutes_to_test)) {
      tmp_feat_set <-
        c(sff_featureset, featutes_to_test[i_featutes_to_test])

      mod <-
```

```
      train(
        target ~ .,
        data = cardio_train_rf[, c(tmp_feat_set , "target")],
        method = model ,
        metric = metric,
        trControl = control
      )

    y_pred_prob <- predict(mod ,  cardio_test, type = "prob")
    auc <-   roc(cardio_test$target , y_pred_prob[, 1])
    aucperfor[i_featutes_to_test] <- auc$auc

  }

  best_idx <-   which.max(aucperfor)

  print (paste(
    "new Feature:" ,
    featutes_to_test[best_idx],
    " with AUC:",
    max(aucperfor)
  ))
  sff_featureset <- c(sff_featureset, featutes_to_test[best_idx])

  print (paste("FeatureSet Size:" , length(sff_featureset)))

  print (paste("FeatureSet :" , paste(sff_featureset, collapse = ",")))

  print(paste("model :", model))


  if (model == "glm") {
    sff_performanceDf_Forward$Bestfesture[length(sff_featureset)] <-
      featutes_to_test[best_idx]
    sff_performanceDf_Forward$auc[length(sff_featureset)] <-
      max(aucperfor)
    sff_performanceDf_Forward$festure[length(sff_featureset)] <-
      paste(sff_featureset, sep = ",", collapse = ",")
  }


  if (model == "lda") {
    sff_performanceDf_Forward$Bestfesture[15 + length(sff_featureset)] <-
      featutes_to_test[best_idx]
    sff_performanceDf_Forward$auc[15 + length(sff_featureset)] <-
      max(aucperfor)
    sff_performanceDf_Forward$festure[15 + length(sff_featureset)] <-
      paste(sff_featureset, sep = ",", collapse = ",")
  }
```

```
    if (model == "rf") {
      sff_performanceDf_Forward$Bestfesture[30 + length(sff_featureset)] <-
        featutes_to_test[best_idx]
      sff_performanceDf_Forward$auc[30 + length(sff_featureset)] <-
        max(aucperfor)
      sff_performanceDf_Forward$festure[30 + length(sff_featureset)] <-
        paste(sff_featureset, sep = ",", collapse = ",")
    }

  }
}
saveRDS(sff_performanceDf_Forward, file  = paste (workPath ,
"sff_performanceDf_Forward.rds", sep = ""))

}
sff_performanceDf_Forward <-
  readRDS(paste (workPath , "sff_performanceDf_Forward.rds", sep = ""))


#Diagramm lda
lda_forward <-
  sff_performanceDf_Forward[sff_performanceDf_Forward$model   == "lda",]
gg_lda_forward <-
  ggplot(lda_forward , aes(x = feature_Size , y = auc)) +
  geom_point(colour = "red", size = 3) +
  geom_smooth() + ggtitle(" lda") +
  geom_text_repel(label = round(lda_forward$auc, 4))

#Diagramm glm
glm_forward <-
  sff_performanceDf_Forward[sff_performanceDf_Forward$model   == "glm",]
gg_glm_forward <-
  ggplot(glm_forward, aes(x = feature_Size , y = auc)) +
  geom_point(colour = "yellow", size = 3) +
  geom_smooth() + ggtitle(" glm") +
  geom_text_repel(label = round(glm_forward$auc, 4))


#Diagramm rf
rf_forward <-
  sff_performanceDf_Forward[sff_performanceDf_Forward$model   == "rf",]
gg_rf_forward <-
  ggplot(rf_forward, aes(x = feature_Size , y = auc)) +
  geom_point(colour = "green", size = 3) +
  geom_smooth() + ggtitle(" rf") +
  geom_text_repel(label = round(rf_forward$auc, 4))
```

```
#Diagramm
gg_lda_forward + gg_glm_forward + gg_rf_forward
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
#performanceModell

glm
```

	feature_Size	model	auc	festure	Bestfesture
1	1	glm	0.6568708	age	age
2	2	glm	0.6837093	age,sysBP	sysBP
3	3	glm	0.7055242	age,sysBP,cigsPerDay	cigsPerDay
4	4	glm	0.7110169	age,sysBP,cigsPerDay,sex	sex
5	5	glm	0.7143704	age,sysBP,cigsPerDay,sex,diabetes	diabetes
6	6	glm	0.7166704	age,sysBP,cigsPerDay,sex,diabetes,totChol	totChol
7	7	glm	0.7174970	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking	smoking
8	8	glm	0.7176460	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI	BMI
9	9	glm	0.7173344	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI,BloodPresMed	BloodPresMed
10	10	glm	0.7171176	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI,BloodPresMed,stroke	stroke
11	11	glm	0.7167721	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI,BloodPresMed,stroke,heartRate	heartRate
12	12	glm	0.7159659	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI,BloodPresMed,stroke,heartRate,education	education
13	13	glm	0.7139063	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI,BloodPresMed,stroke,heartRate,education,diaBP	diaBP
14	14	glm	0.7120128	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI,BloodPresMed,stroke,heartRate,education,diaBP,hypertensive	hypertensive
15	15	glm	0.7068537	age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI,BloodPresMed,stroke,heartRate,education,diaBP,hypertensive,gluc...	glucose

Interpretation:

Das Modell glm hat die beste leistung mit 8 feature, ,also "age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI" und die Auc beträgt 0,7176460. Das Modell lda hat die beste leistung mit 8 feature, ,also age,sysBP,cigsPerDay,diabetes,sex,totChol,smoking,BMI und die Auc beträgt 0.7187706. **Der Unterschied besteht darin, dass im Modell glm das sex 4-stellig ist und im Modell lda 5-stellig und das Model lda hat bessere Leistung. Das Modell rf hat die beste leistung mit 12 feature, ,also 8"sysBP,sex,diabetes,cigsPerDay,stroke,age,BMI,BloodPresMed,smoking,totChol,diaBP,hypertensive" und die uc beträgt 0.7013221.

Greedy Backward selection

Backward stepwise selection (or backward elimination) is a variable selection method which:

Begins with a model that contains all variables under consideration (called the Full Model)

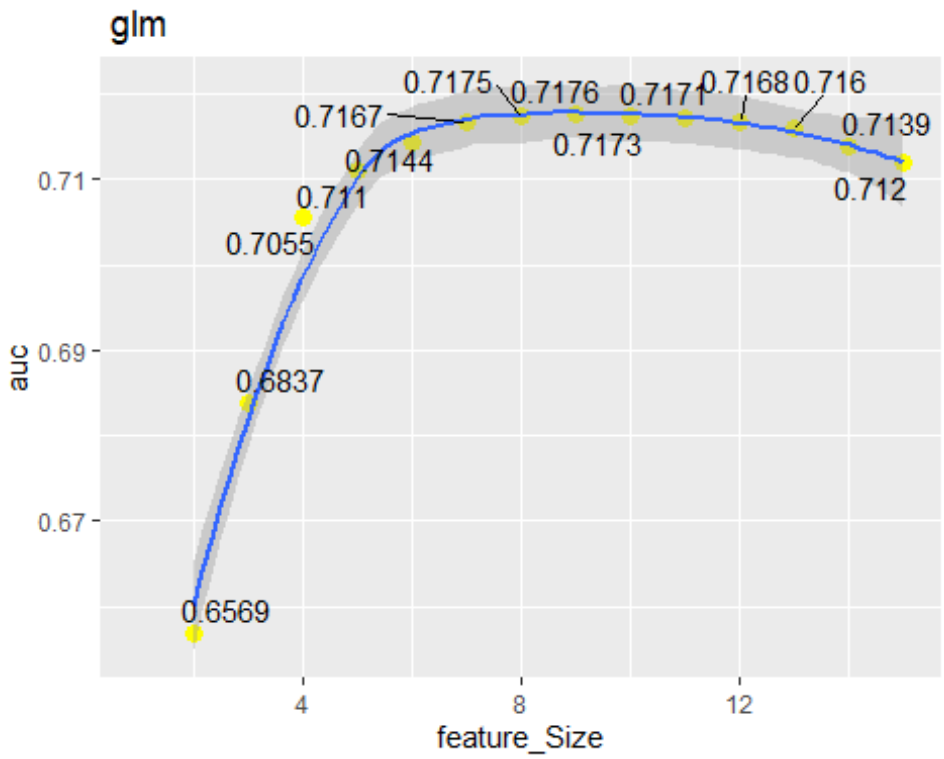
Then starts removing the least significant variables one after the other

Until a pre-specified stopping rule is reached or until no variable is left in the model

```
computeData <- F
modell <- c("glm", "lda", "rf")

if (computeData) {

sff_performanceDf_Backward = expand.grid(feature_Size = seq(15, 1), model =
modell)

sff_performanceDf_Backward$auc <- NA
sff_performanceDf_Backward$worstfeature <- NA

sff_performanceDf_Backward$festure <- ""

cardio_train_rf <- cardio_train

nbr_features <- 15
aucperfor <- NA


for (model in  c("glm", "lda", "rf")) {
```

```
  sff_featureset <- c() #  feature die rausgeschmissen werde

  rel_features <- colnames(cardio_train_rf[, -16])


  while (length(sff_featureset) < nbr_features) {
    featutes_to_test <- setdiff(rel_features, sff_featureset)
    aucperfor <- rep(NA, length(featutes_to_test))

    for (i_featutes_to_test  in 1:length(featutes_to_test)) {
      tmp_feat_set <-
        c(sff_featureset, featutes_to_test[i_featutes_to_test])

      mod <-
        caret::train(
          target ~ .,
          data = cardio_train_rf[, c("target", setdiff(rel_features,
tmp_feat_set))],
          method = model ,
          metric = metric,
          trControl = control
        )

      y_pred_prob <- predict(mod ,  cardio_test, type = "prob")
      auc <-   roc(cardio_test$target , y_pred_prob[, 1])
      aucperfor[i_featutes_to_test] <- auc$auc

    }# for feature to test

    best_idx <- which.max(aucperfor)
    sff_featureset <- c(sff_featureset, featutes_to_test[best_idx])

    print (
      paste (
        "model ",
        model  ,
        ";feautur size "  ,
        length(rel_features)  - length(sff_featureset) ,
        "  ; AUC:",
        aucperfor[best_idx] ,
        featutes_to_test[best_idx]
      )
    )

    perf_df_idx = sff_performanceDf_Backward$model == model &
      (sff_performanceDf_Backward$feature_Size == (length(rel_features) -
length(sff_featureset)+1))

    sff_performanceDf_Backward$auc[perf_df_idx] <- aucperfor[best_idx]
```

```
    sff_performanceDf_Backward$worstfeature[perf_df_idx] <-
      featutes_to_test[best_idx]

    sff_performanceDf_Backward$festure[perf_df_idx] <-
      paste(setdiff(rel_features, sff_featureset),
            sep = ",",
            collapse = ",")

  }  # for feature set size
} # for model
saveRDS(sff_performanceDf_Backward, file  = paste (workPath ,
"sff_performanceDf_Backward.rds", sep = ""))
}

sff_performanceDf_Backward <-
  readRDS(paste (workPath , "sff_performanceDf_Backward.rds", sep = ""))


glm_aucc <-
  sff_performanceDf_Backward[sff_performanceDf_Backward$model   == "glm", ]

gg_glm_aucc <- ggplot(glm_aucc  , aes(x = feature_Size , y = auc)) +
  geom_point(colour = "yellow", size = 3) +
  geom_smooth() + ggtitle(" glm") +
  geom_text_repel(label = round(glm_aucc$auc, 4))


#Diagramm
gg_glm_aucc

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning: Removed 1 rows containing non-finite values (stat_smooth).

## Warning: Removed 1 rows containing missing values (geom_point).

## Warning: Removed 1 rows containing missing values (geom_text_repel).
```



##Interpretation:

Das Model glm hat die beste leistung ,mit 8 feature, ,also

```
"age,sysBP,cigsPerDay,sex,diabetes,totChol,smoking,BMI"
```

und die Auc beträgt 0,7176460.

Das Model lda hat die beste leistung mit 8 feature, ,also

age,sysBP,cigsPerDay,diabetes,sex,totChol,smoking,BMI

und die Auc beträgt 0.7187706.

**Der Unterschied besteht darin, dass im Modell glm das sex 4-stellig ist und im Modell lda 5-stellig und das Model lda hat bessere Leistung.

Das Model rf hat die beste leistung mit 12 feature, ,also

```
"sysBP,sex,diabetes,cigsPerDay,stroke,age,BMI,BloodPresMed,smoking,totChol,diaBP,hypertensive"
```

und die Auc beträgt 0.7013221.

Featurte importance / Feature selection

```
computeData <- F


if (computeData) {
cardio_train_rf <- cardio_train
cardio_train_rf$target = factor(cardio_train_rf$target)

feature_list <- colnames(cardio_train_rf[, -16])

rfePerformace <- data.frame (id = seq(15))
rfePerformace$auc <- NA
rfePerformace$feature_list <- NA

while (length(feature_list) > 0) {
  mod <-
    randomForest(target ~ ., data = cardio_train_rf[, c(feature_list,
"target")]
                 , importance = T)

  feat_importance <- data.frame(mod$importance)
  feat_importance <-
    feat_importance[order(feat_importance$MeanDecreaseGini, decreasing = T),
]

  y_pred_prob <- predict(mod ,  cardio_test, type = "prob")

  auc <-   roc(cardio_test$target , y_pred_prob[, 1])
  rfePerformace$auc[16 - length(feature_list)]  <- auc$auc
  rfePerformace$feature_list[16 - length(feature_list)] <-
    paste(feature_list, sep = ",", collapse = ",")

  feature_list <-
    row.names(feat_importance)[1:nrow(feat_importance) - 1]

}
feature_list <- row.names(varImp(mod)$importance)
feature_list <- feature_list[1:length(feature_list)]
}
```

learning curve

2022-05-01

Load Libraries

```
library(magrittr)
library(plyr)
library(dplyr)
library(ggplot2)
library(grid)
library(gridExtra)
library(stringr)
library(here)
library(caret)
library(mvtnorm)
library(ggplot2)
library(randomForest)
library(e1071)
library(caret)
library(pROC)
library(xgboost)
```

*wie ist die Performance (AUC) bei 100,200, 300, 400, 500, ... 1427 Trainingsdaten (ca 14 verschiedene Größen von Trainingsdaten, jedes mal 100 mal. $=>$ 1400 Modelle trainiren). Kurve wird wackelig, weil es stark vom Zufall abhänge, welche 100 Datenpunkte man nimmt. Damit es nicht so wackelig ist müssen wir mehrfach das experimen durchführen. Es muss mehrfach 100 Datenauswählen, das Modell trainieren und die Performance evaluieren (bootstraping).

Load Data

```
workPath ="D:\\DataMining\\"
cardio <- readRDS(paste (workPath , "cardio.rds", sep =""))
```

Train test split

```
set.seed(1001)

cardio_complete <- cardio[(complete.cases(cardio)),]

nrow(cardio_complete)
```

```
## [1] 2927
```

```
n_test <- 1500

idx_test <- sample(1:nrow(cardio_complete), n_test)

cardio_test <- cardio_complete[ idx_test,]

cardio_train <- cardio_complete[ -idx_test,]
```

```
computeData <- F

if (computeData) {
  control <- trainControl(method = "cv",
                          number = 5,
                          classProbs =  TRUE)
  metric <- "Accuracy"

  models <- c("glm", "lda", "rf", "knn", "svmLinear", "svmRadial", "rpart")


  learningCurveData = expand.grid(trainingSize = seq(200, 1400, by = 100),
                                  model = models)
  learningCurveData$auc <- NA

  ExPerformace <- data.frame (trainingSize = seq(200, 1400, by = 100))

  ExPerformace$auc <- NA

  Model <- data.frame (models)

  performance <- data.frame (models = models)
  performance$mean_unter500 <- NA
  performance$mean_ueber500  <- NA
  performance$variance  <- NA
  performance$max_Auc <- NA
  performance$min_Auc <- NA
  performance$running_time  <-
    c(
      "1.21 mins",
      " 1.03 mins" ,
      "16.45 mins" ,
      "1.18 mins" ,
      "3.55 mins" ,
      "6.18 mins",
      "1.39 mins"
    )

  evalModel <- function(model, trainingSize, nReplicate) {
    print (paste(
      "model :",
      model,
      "trainigSize:" ,
      trainingSize ,
```

```
    " repli:" ,
    nReplicate
  ))

  ## create trainingSet based on trainigSize and nReplica
  set.seed(trainingSize * nReplicate + nReplicate)

  idx_test <- sample(1:nrow(cardio_train), trainingSize)

  cardio_Example <- cardio_train[idx_test, ]

  if (length(unique(cardio_Example$stroke)) <= 1) {
    cardio_Example <-
      cardio_Example[, colnames (cardio_Example) != "stroke"]
  }

  ## compute model
  if (model == "glm") {
    mod <-
      train(
        target ~ .,
        data = cardio_Example,
        method = "glm",
        family = "binomial",
        trControl = control
      )
  } else {
    mod <-
      train(
        target ~ .,
        data = cardio_Example,
        method = model ,
        metric = metric,
        trControl = control
      )
  }

  ## compute AUC
  y_pred_prob <- predict(mod , cardio_test, type = "prob")
  y_pred <- predict(mod ,  cardio_test)
  return (roc(cardio_test$target , y_pred_prob$CHD)$auc)
}

nRep = 10

performanceMatt <- matrix (NA, nrow(learningCurveData), ncol = nRep)

for (i in 1:nrow(learningCurveData)) {
  tmp <-
    lapply(seq(1, nRep) , function(zz)
      evalModel (
        learningCurveData$model [i],
        learningCurveData$trainingSize[i],
```

```
          zz
        ))
    performanceMatt[i,] <- unlist(tmp)
  }
  learningCurveData$auc <- apply(performanceMatt, 1, mean)

  saveRDS(learningCurveData,
          file  = paste (workPath , "learningCurveData.rds", sep = ""))
}

learningCurveData <-
  readRDS(paste (workPath , "learningCurveData.rds", sep = ""))
```

Zusammenfassung

```
computeData <- F

if (computeData) {
  for (i in 1:nrow(Model)) {
    #Mean mehe als 500 Trainingsdaten
    performance[performance$models == models[i], 2] <-
      mean(learningCurveData[learningCurveData$model == models[i],]$auc[0:5])

    #Mean weniger als 500 Trainingsdaten
    performance[performance$models == models[i], 3] <-
      mean(learningCurveData[learningCurveData$model == models[i],]$auc[4:nrow(ExPerformace)])

    # Variance  #Max_Auc#Min_Auc
    performance[performance$models == models[i], 4] <-
      var(learningCurveData[learningCurveData$model == models[i],]$auc[0:nrow(ExPerformace)])

    #Max_Auc
    performance[performance$models == models[i], 5] <-
      max(learningCurveData[learningCurveData$model == models[i],]$auc[0:nrow(ExPerformace)])

    #Min_Auc
    performance[performance$models == models[i], 6] <-
      min(learningCurveData[learningCurveData$model == models[i],]$auc[0:nrow(ExPerformace)])

  }
  saveRDS(performance, file  = paste (workPath , "learningCurveDf.rds", sep =
                                        ""))
}

performance <-
  readRDS(paste (workPath , "learningCurveDf.rds", sep = ""))
```

Diagramm

```
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.1.3
```

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.1.3
```

```
#Generalized linear model
model_glm <- learningCurveData[learningCurveData$model == "glm", ]

g_glm <- ggplot(model_glm, aes(x = trainingSize , y = auc)) +
  geom_point(colour = "red", size = 3) + geom_smooth() +
  ggtitle("Generalized linear model") +
  geom_text_repel(label = round(model_glm$auc, 4))

# Linear discriminant analysis]
model_Lda <- learningCurveData[learningCurveData$model == "lda",]
g_Lda <-
  ggplot(model_Lda, aes(x = trainingSize , y = auc)) + geom_point(colour = "red", size = 3) +  geom_smo
  ggtitle("Linear discriminant analysis") + geom_text_repel(label = round(model_Lda$auc, 4))

# K Nearest Neighbor
model_knn <- learningCurveData[learningCurveData$model == "knn", ]
g_Knn <- ggplot(model_knn, aes(x = trainingSize , y = auc)) +
  geom_point(colour = "red", size = 3) +
  geom_smooth() + ggtitle("K Nearest Neighbor") +
  geom_text_repel(label = round(model_knn$auc, 4))


# Random forest
model_rf <- learningCurveData[learningCurveData$model == "rf", ]
g_rf <- ggplot(model_rf, aes(x = trainingSize , y = auc)) +
  geom_point(colour = "red", size = 3) +
  geom_smooth() + ggtitle(" Random forestr") +
  geom_text_repel(label = round(model_rf$auc, 4))

#Support Vector Machine linear
model_svmLinear <-
  learningCurveData[learningCurveData$model == "svmLinear",]
g_svmLinear <-
  ggplot(model_svmLinear, aes(x = trainingSize , y = auc)) +
  geom_point(colour = "red", size = 3) +
  geom_smooth() + ggtitle(" svmLinear") +
  geom_text_repel(label = round(model_svmLinear$auc, 4))

#Support Vector Machine Radial
model_svmRadial <-
  learningCurveData[learningCurveData$model == "svmRadial",]
```

```
g_svmRadial <-
  ggplot(model_svmRadial, aes(x = trainingSize , y = auc)) +
  geom_point(colour = "red", size = 3) + geom_smooth() +
  ggtitle("svmRadial") +
  geom_text_repel(label = round(model_svmRadial$auc, 4))

#"rpart"

model_rpart <-
  learningCurveData[learningCurveData$model == "rpart",]

g_rpart <- ggplot(model_rpart, aes(x = trainingSize , y = auc)) +
  geom_point(colour = "red", size = 3) +
  geom_smooth() + ggtitle("rpart") +
  geom_text_repel(label = round(model_rpart$auc, 4))
```

diagram von rf ,lda and glm

```
g_glm+g_Lda+g_rf
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

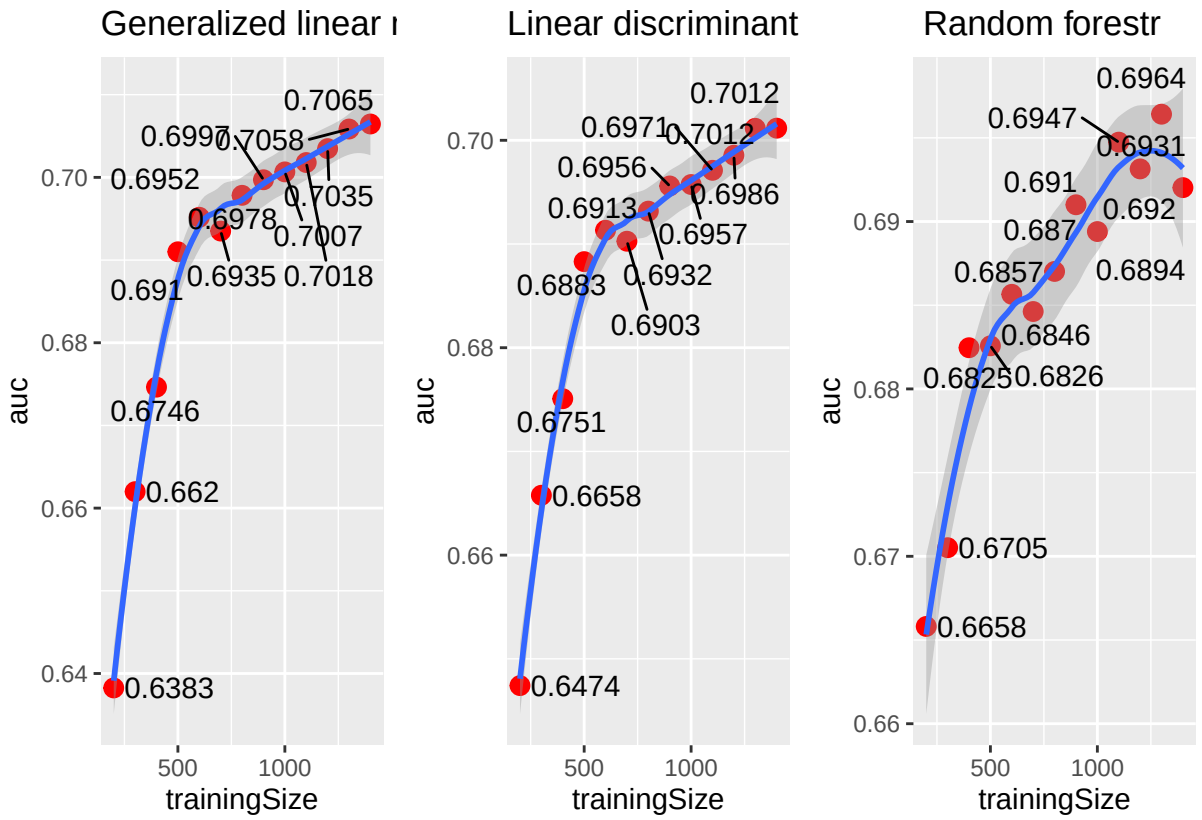
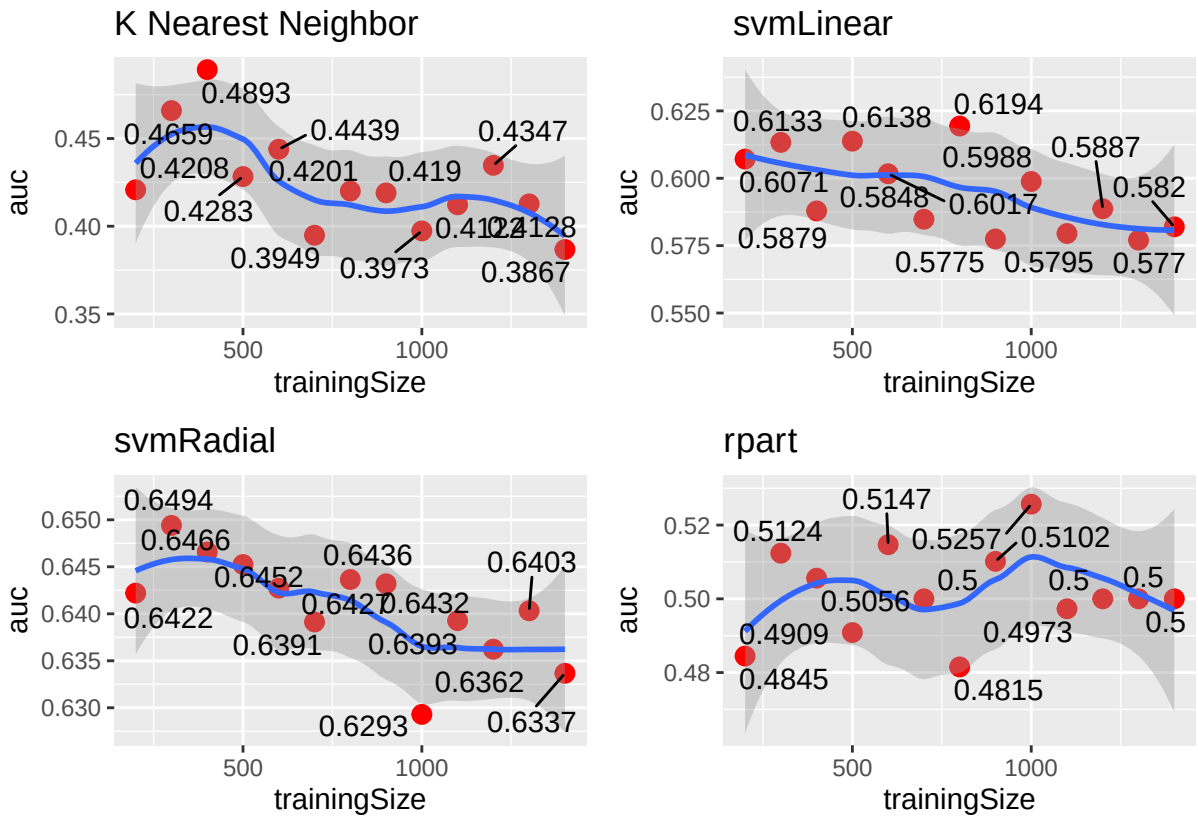


diagram von Knn ,svmLinear and svmRadial and rpart

```
g_Knn+g_svmLinear+g_svmRadial+g_rpart
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



performance

```
performance
```

```
##      models mean_unter500 mean_ueber500     variance   max_Auc   min_Auc
## 1       glm     0.6722106     0.6995463 4.041901e-04 0.7064567 0.6382543
## 2       lda     0.6735807     0.6952385 2.506780e-04 0.7011693 0.6474203
## 3        rf     0.6774027     0.6896600 8.191885e-05 0.6964124 0.6658099
## 4       knn     0.4496182     0.4149850 8.159574e-04 0.4892514 0.3867341
## 5 svmLinear     0.6047550     0.5923132 2.267764e-04 0.6193794 0.5770374
## 6 svmRadial     0.6452300     0.6392716 2.967603e-05 0.6494036 0.6292957
## 7     rpart     0.5015965     0.5020197 1.496269e-04 0.5256684 0.4815361
##   running_time
## 1    1.21 mins
## 2    1.03 mins
## 3   16.45 mins
## 4    1.18 mins
## 5    3.55 mins
## 6    6.18 mins
## 7    1.39 mins
```

##Interpretation:

Die Generalized linear model and Linear discriminant haben die beste Leistung im Vergleich zu anderen Methoden und die Auc für die Modelle , die mehr als 500 Trainingsdaten haben, wackelt nicht uund steigt mit einer leichten Steigung an.

Mit der Zunahme der Trainingsdaten steigt die Auc für die Modelle glm und lda an , im Gegensatz dazu sinkt die Auc für das Modell Support Vector Machine Radial/lineal und Knn ,die Auc bleibt fast konstant für das Modell rpart.

*Die Auc erhöht sich bei die Modelle rf und xgbTree , aber die ist für das Modell Randomforest im 4 Punkten und für das Modell xgbTree im 5 Punken kurz untergegengen.

Model Knn: die Auc steigt bis 500 Trainingsdaten und erreicht 0.489 und fällt danach am ende auf 0.3867.

Die Performancetabelle :

Der höchste Durchschnittswert des AUC (Trainingsdaten weniger als 500) beträgt 0,677 beim Random-Forest-Modell und das Modell lda mit der Auc 0.673. Der mindestens Durchschnittswert des AUC (Trainingsdaten weniger als 500) beträgt 0,44 beim Knn-Modell und das Modell rpart mit der Auc 0.50.

Das heißt, dass die Modelle rf und lda bei den wenigen Trainingsdaten eine bessere Leistung aufweisen.

Der höchste Durchschnittswert des AUC (Trainingsdaten mehr als 500) beträgt 0,699 beim glm-Modell und das Modell lda mit der Auc 0.695 . Der mindestens Durchschnittswert des AUC (Trainingsdaten mehr als 500) beträgt 0,44 beim Knn-Modell und das Modell rpart mit der Auc 0.50. Das heißt, dass die Modelle glm und lda bei den wenigen Trainingsdaten eine bessere Leistung aufweisen.

Der maximale Auc-Wert beträgt 0,7064 beim Glm-Modell und die nächst höhere Wert berträgt 0,7011 beim lda-Modell .

Der minimale AUC-Wert beträgt 0,386 beim Knn-Modell und der nächst niedrige Wert berträgt 0,7011 beim rpart .

Running Time: Das Modell lda hat die schnellste Ausführungsgeschwindigkeit von 1,03 Minuten und das Modell xgbTree hat die geringste Ausführungsgeschwindigkeit von 31,10 Minuten und das Modell rf mit 16.45 mins.

Literatur und Quellen

- Markdown Tuorial
- Markdown Cheatsheet
- R for Data Science
- R Introduction

https://www.kaggle.com/christofel04/cardiovascular-study-dataset-predict-heartdisea

https://www.kaggle.com/datasets/christofel04/cardiovascular-study-dataset-predict-heart-disea