Unüberwachtes Lernen mit dem Herz-Datensatz

Clustering und PCA auf die Herzdaten

Einlesen der Herz-Daten

Es werden wieder die Herzdaten aus der letzten Aufgabe genutzt. Lesen Sie diese als Data Frame ein.

```
# Ihre Lösung:
df = read.csv("https://oc.informatik.hs-mannheim.de/s/wyzFq34K9HiNjXR/download", header= TRUE, na.strin
kable(df[1:10 , ])
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	goal
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
53	1	4	140	203	1	2	155	1	3.1	3	0	7	1

Bedeutet "ähnliche Merkmale" auch "gleiche Diagnose"?

Für jeden Datensatz ist bekannt, zu welcher Klasse er gehört: 0 (gesund) und 1 (erkrankt). Wir wollen untersuchen, wie gut *ähnliche* Datensätze zur gleichen Klasse gehören. Dafür soll mit dem k-means-Clusterverfahren der Datensatz in zwei Cluster eingeteilt werden.

Nur reelle Merkmale

Zunächst sollen **nur die nummerischen Merkmale** benutzt werden und nicht jene, die Faktoren sind.

Clustering Clustern Sie diese Daten. Überlegen Sie, ob Sie die Daten standardisieren wollen.

```
# Ihre Lösung:

df_cleaned = na.omit(df) # Zeilen mit NA Werten entfernen

str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : int  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : int  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 0 1 ...
```

```
##  $ restecg : int  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : int  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : int  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : int  6 3 7 3 3 3 3 3 7 7 ...
##  $ goal    : int  0 2 1 0 0 0 3 0 2 1 ...
```

```r
df_cleaned$restecg = factor(df_cleaned$restecg)
df_cleaned$fbs = factor(df_cleaned$fbs)
df_cleaned$exang = factor(df_cleaned$exang)
df_cleaned$cp = factor(df_cleaned$cp)
df_cleaned$thal = factor(df_cleaned$thal)
df_cleaned$sex = factor(df_cleaned$sex)
df_cleaned$slope = factor(df_cleaned$slope) # the slop of the peak exersice ST segment
df_cleaned$ca = factor(df_cleaned$ca)
df_cleaned$goal = factor((df_cleaned$goal))

str(df_cleaned)
```

```
## 'data.frame':    297 obs. of  14 variables:
##  $ age     : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...
##  $ cp      : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
##  $ thal    : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ goal    : Factor w/ 5 levels "0","1","2","3",..: 1 3 2 1 1 1 4 1 3 2 ...
##  - attr(*, "na.action")= 'omit' Named int [1:6] 88 167 193 267 288 303
##   ..- attr(*, "names")= chr [1:6] "88" "167" "193" "267" ...
```

```r
# Ihre Lösung:


# neue disease Spalte erzeugen
# wenn goal == 0, dann wird neue Spalte auch 0 (gesund)
# wenn nicht, dann 1 (krank)
df_cleaned$disease = ifelse(df_cleaned$goal == 0, 0, 1)
df_cleaned$disease = factor(df_cleaned$disease)

# Spalte als Ground Truth speichern
gt = df_cleaned$disease


# für jede Spalte den Wert TRUE oder FALSE zurückgeben
# Namen der Spalten mit TRUE zurückgeben
numeric_cols = names(df_cleaned)[vapply(df_cleaned, is.numeric, logical(1))]
```

```r
# Daten standardisieren (Mittelwert 0, Standardabweichung 1)
# Spalten haben verschiedene Wertebereiche
# größere Zahlenwerte sollen aber nicht größeren Einfluss haben
df_numeric = data.frame(scale(df_cleaned[, numeric_cols, drop= FALSE]))

kable(df_numeric[1:10, ])
```

age	trestbps	chol	thalach	oldpeak
0.9346033	0.7491157	-0.2759776	0.0174650	1.0671642
1.3766051	1.5935769	0.7433005	-1.8132735	0.3811301
1.3766051	-0.6583195	-0.3529043	-0.8979043	1.3244270
-1.9384088	-0.0953454	0.0509607	1.6302584	2.0962153
-1.4964070	-0.0953454	-0.8336958	0.9764232	0.2953758
0.1611000	-0.6583195	-0.2182826	1.2379573	-0.2191498
0.8241028	0.4676287	0.3971306	0.4533551	2.1819696
0.2716005	-0.6583195	2.0510537	0.5841221	-0.3906583
0.9346033	-0.0953454	0.1278873	-0.1133021	0.2953758
-0.1704014	0.4676287	-0.8529275	0.2354100	1.7531983

```r
# Clustern auf zwei Cluster mit 20 Startcluster
km = kmeans(df_numeric, centers=2, nstart=20)
cluster_num = km$cluster
kable(cluster_num[1:30])
```

x
1
1
1
2
2
2
1
2
1
1
2
1
2
2
2
2
2
2
2
2
2
1
1
1
1
1

x
2
2
1
2
2

Richtig? Berechnen Sie, wie viel Prozent der Datensätze richtig einem Cluster eingeordnet wurden und geben Sie die Zahl auf zwei Nachkommastellen gerundet aus.

Hinweis: Berücksichtigen Sie, dass die Vergabe der Clusternummern zufällig ist. D.h. sowohl die Cluster (1, 2) wie auch (2, 1) sind möglich.

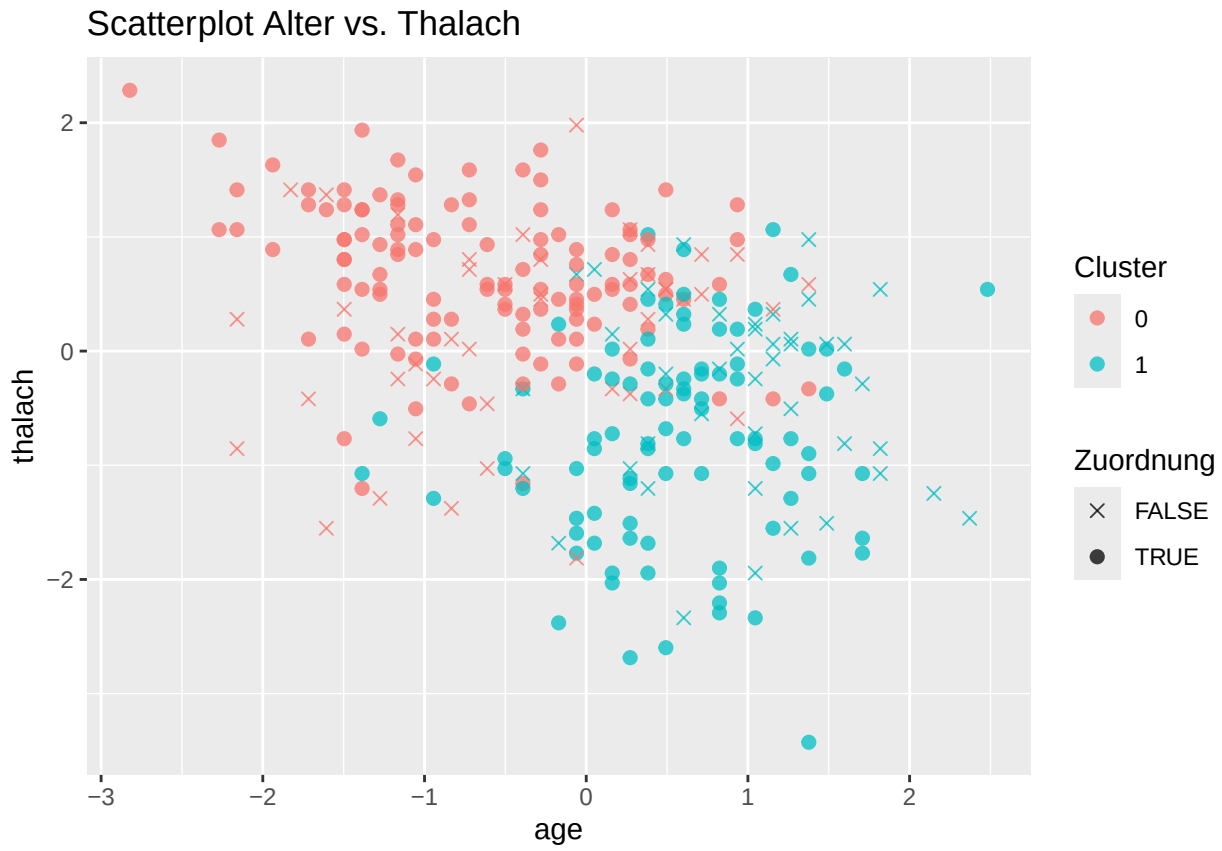
```
# Ihre Lösung:

# Clusternummern können vertauscht sein, deswegen beide Zuordnungen prüfen
match_clusters = function(cluster, gt) {
  gt = as.character(gt)

  # Beide Varianten prüfen
  # (cluster == 1) == (gt == "0") gibt boolsche Werte als Vektor zurück
  # mean berechnet den Anteil der TRUE Werte (Mittelwert von TRUE=1 und FALSE=0)

  acc_1_to_0 = mean((cluster == 1) == (gt == "0"))
  acc_1_to_1 = mean((cluster == 1) == (gt == "1"))


  # Klassenlabel (0, 1) richtig zuordnen
  if (acc_1_to_0 >= acc_1_to_1) {
    predicted = ifelse(cluster == 1, "0", "1")
  } else {
    predicted = ifelse(cluster == 1, "1", "0")
  }

  list(
    accuracy = max(acc_1_to_0, acc_1_to_1), # höhere Accuracy wählen
    predicted = predicted
  )
}

result_num = match_clusters(cluster_num, df_cleaned$disease)
correct_num = result_num$predicted == gt # TRUE FALSE Werte

sprintf("Richtig zugeordnet: %.2f%%", 100 * result_num$accuracy)
```

```
## [1] "Richtig zugeordnet: 70.03%"
```

Scatterplot age vs. thalach Plotten Sie die Merkmale `age` und `thalach` als Scatterplot. Färben Sie die Punkte gemäß ihrer Clusterzuordnung ein. Die Form (`shape`) eines Punkts soll zeigen, ob die Klassifikation (d.h. der Cluster) richtig oder falsch ist.

```
# Ihre Lösung:

# Dataframe mit allen Informationen für Plot erstellen
```

```r
plot_num = data.frame(
  age = df_numeric$age,
  thalach = df_numeric$thalach, # maximum heart rate
  cluster = factor(result_num$predicted), # Vorhersagen
  correct = factor(correct_num, levels = c(FALSE, TRUE)) # gt
)

ggplot(plot_num, aes(x = age, y = thalach,
                     color = cluster, shape = correct)) +
  geom_point(alpha = 0.75, size = 2) +
  scale_shape_manual(
    values = c("FALSE" = 4, "TRUE" = 19), # Formen festlegen
  ) +
  labs(title = "Scatterplot Alter vs. Thalach", color = "Cluster", shape = "Zuordnung", x = "age", y =
```



Mit Dummy-Variablen

Nun sollen **alle Merkmale** benutzt werden.

Clustering Clustern Sie diese Daten. Überlegen Sie, wie die Faktoren zu Zahlen werden.

```r
# Ihre Lösung:

# Vektor mit allen Merkmalen
target_cols = c("age","sex","cp","trestbps","chol","fbs","restecg","thalach",
                "exang", "oldpeak", "slope", "ca", "thal")
```

```r
# Spalten aus Dataframe auswählen
df_all = df_cleaned[, target_cols]

# nimmt alle Spalten des df
# erzeugt für alle Faktorenstufen eine eigene Spalte
# zwei Stufen = eine Spalte
mm = model.matrix(~ . -1, data=df_all)
kable(mm[1:10,])
```

age	sex0	sex1	cp2	cp3	cp4	trestbps	chol	fbs1	restecg1	restecg2	thalach	exang1	oldpeak	slope2	slope3	ca1	ca2	ca3	thal6	thal7
63	0	1	0	0	0	145	233	1	0	1	150	0	2.3	0	1	0	0	0	1	0
67	0	1	0	0	1	160	286	0	0	1	108	1	1.5	1	0	0	0	1	0	0
67	0	1	0	0	1	120	229	0	0	1	129	1	2.6	1	0	0	1	0	0	1
37	0	1	0	1	0	130	250	0	0	0	187	0	3.5	0	1	0	0	0	0	0
41	1	0	1	0	0	130	204	0	0	1	172	0	1.4	0	0	0	0	0	0	0
56	0	1	1	0	0	120	236	0	0	0	178	0	0.8	0	0	0	0	0	0	0
62	1	0	0	0	1	140	268	0	0	1	160	0	3.6	0	1	0	1	0	0	0
57	1	0	0	0	1	120	354	0	0	0	163	1	0.6	0	0	0	0	0	0	0
63	0	1	0	0	1	130	254	0	0	1	147	0	1.4	1	0	1	0	0	0	1
53	0	1	0	0	1	140	203	1	0	1	155	1	3.1	0	1	0	0	0	0	1

```r
mm_scaled = scale(mm)

# Clustering mit allen Merkmalen
km_all = kmeans(mm_scaled, centers = 2, nstart = 50)
```

Richtig? Berechnen Sie für diesen Fall, wie viel Prozent der Datensätze richtig einem Cluster eingeordnet wurden und geben Sie die Zahl auf zwei Nachkommastellen gerundet aus. Wie hat sich der Wert verändert? Warum ist dies so?

```r
# Ihre Lösung:

cluster_all = km_all$cluster

result_all = match_clusters(cluster_all, gt)
correct_all = result_all$predicted == gt

sprintf("Richtig zugeordnet: %.2f%%", 100 * mean(correct_all))
```

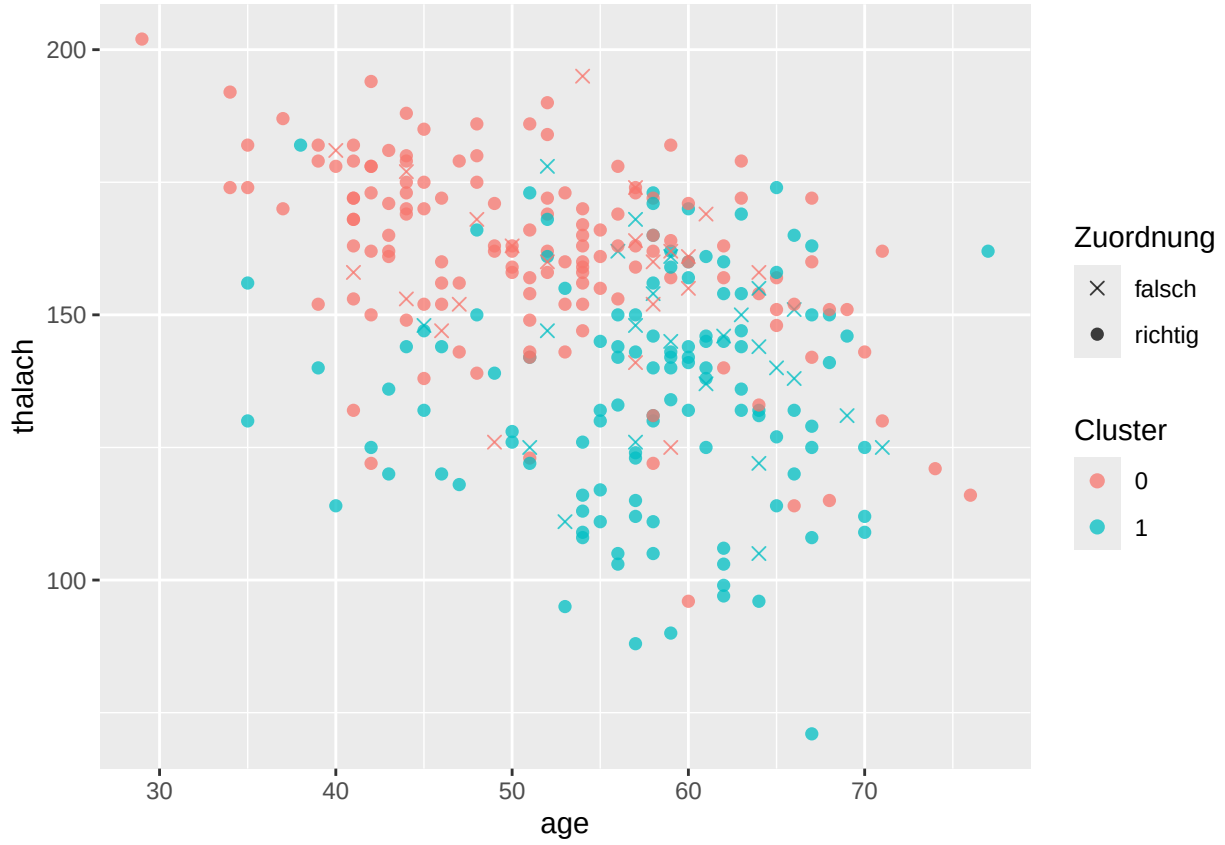
```
## [1] "Richtig zugeordnet: 84.51%"
```

Wie hat sich der Wert verändert und Warum? ANTWORT Der Wert hat sich deutlich verbessert, weil mehr Merkmale in die Berechnung eingeflossen sind und somit auch mehr Informationen. Faktoren wie das Geschlecht können wertvolle Informationen und Abhändigkeiten enthalten, die das Cluster Ergebniss verbessern.

Scatterplot age vs. thalach Plotten Sie erneut und schauen Sie, wie die richtigen nun Punkte verteilt sind.

```r
# Ihre Lösung:

plot_all = data.frame(
```

```
  age = df_all$age,
  thalach = df_all$thalach,
  cluster = factor(result_all$predicted), # Vorhersagen
  correct = factor(correct_all, levels = c(FALSE, TRUE)) # gt
)

ggplot(plot_all, aes(x = age, y = thalach,
                     color = cluster, shape = correct)) +
  geom_point(alpha = 0.75, size = 2) +
  scale_shape_manual(
    values = c("FALSE" = 4, "TRUE" = 16),
    labels = c("FALSE" = "falsch", "TRUE" = "richtig")
  ) +
  labs(color = "Cluster", shape = "Zuordnung", x = "age", y = "thalach")
```



PCA

Wenden Sie eine PCA auf diesen Datensatz an. Es sollen alle Merkmale berücksichtigt werden.

Wichtige Merkmale

Welche Merkmale der ersten Hauptkomponente tragen am meisten zur Varianz bei? Geben Sie die TOP-10-Merkmale an.

```
# Ihre Lösung:
```

```r
# Daten sind bereits zentriert und standardisiert
pca = prcomp(mm_scaled, center = FALSE, scale. = FALSE)

kable(pca$rotation[, 1])
```

	x
age	-0.2214695
sex0	0.1897896
sex1	-0.1897896
cp2	0.2496325
cp3	0.2146476
cp4	-0.3789151
trestbps	-0.1226259
chol	-0.0439295
fbs1	-0.0324478
restecg1	-0.0648217
restecg2	-0.1197593
thalach	0.3601178
exang1	-0.3387436
oldpeak	-0.3422748
slope2	-0.2640842
slope3	-0.0849604
ca1	-0.0898730
ca2	-0.1493219
ca3	-0.1394602
thal6	-0.1100339
thal7	-0.2998447

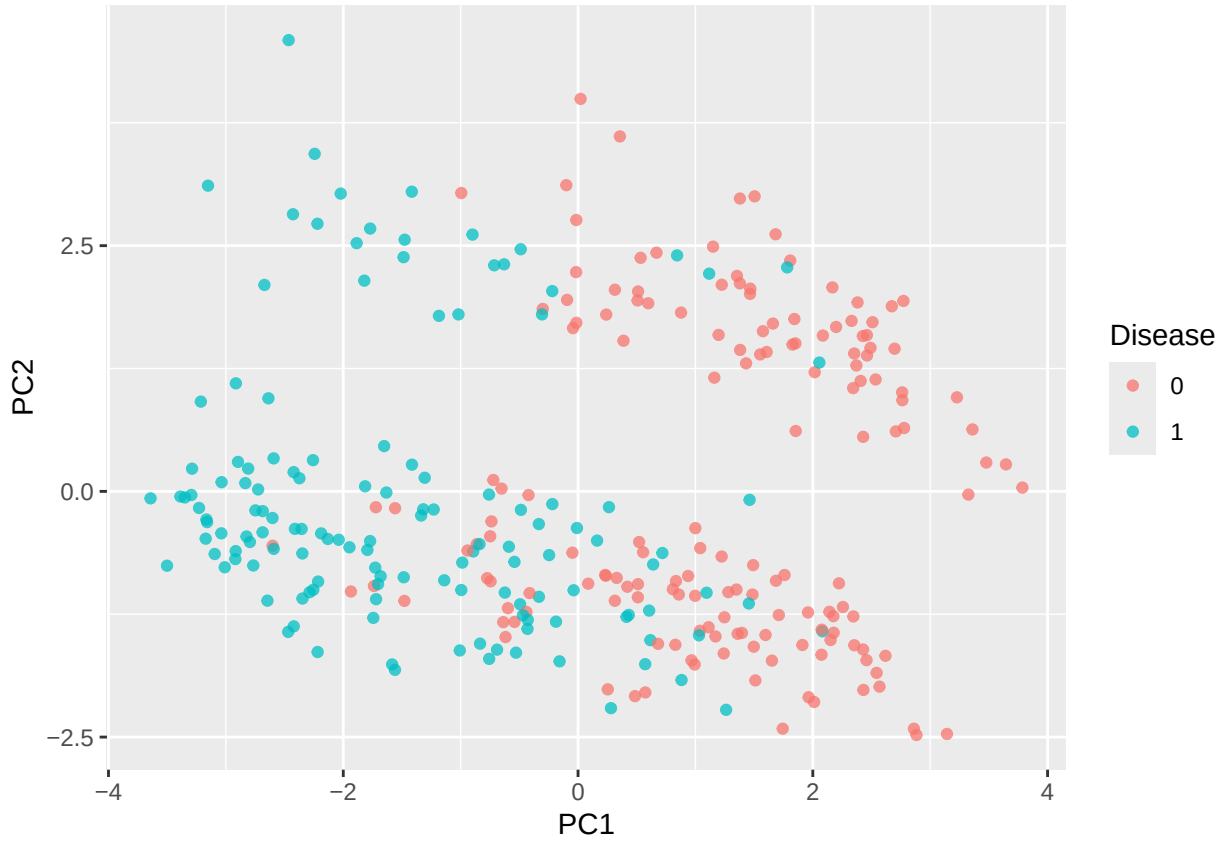
```r
# Beitrag der einzelnen Merkmale zur Varianz
# Aus Rotationsmatrix die Loadings (Eigenvektoren) der ersten Hauptkomponente quadrieren zurückgeben un
contribution_pc1 = pca$rotation[, 1]^2
top10 = sort(contribution_pc1, decreasing = TRUE)[1:10]

kable(top10)
```

	x
cp4	0.1435767
thalach	0.1296849
oldpeak	0.1171520
exang1	0.1147472
thal7	0.0899068
slope2	0.0697405
cp2	0.0623164
age	0.0490488
cp3	0.0460736
sex1	0.0360201

Erste und zweite Hauptkomponente

Plotten Sie die erste und zweite Hauptkomponente als Scatterplot. Färben Sie die Punkte gemäß ihrer Klasse (Disease) ein.

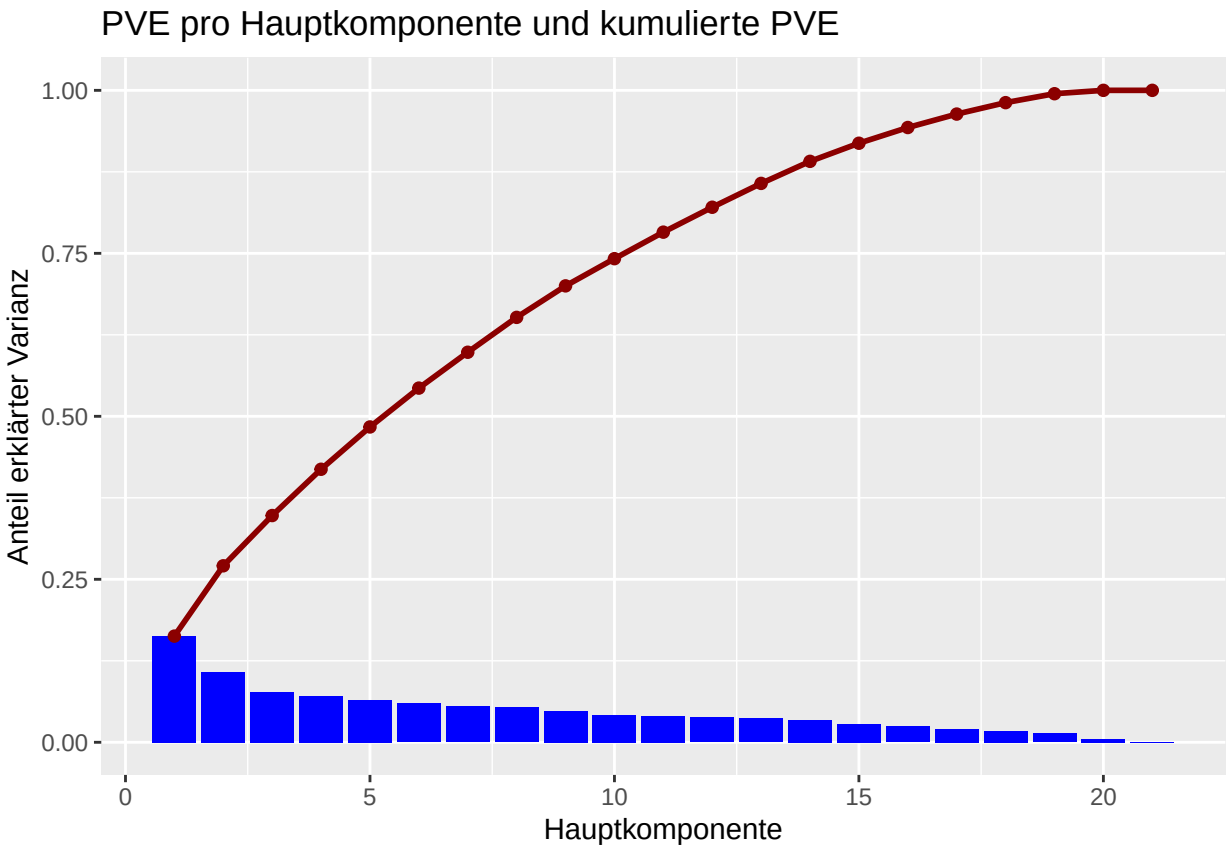
```r
# Ihre Lösung:

# wieder Dataframe für Plot erstellen
pca_plot = data.frame(
  PC1 = pca$x[, 1],
  PC2 = pca$x[, 2],
  Disease = factor(gt) # um Punkte anhand der Diagnose einzufärben
)

ggplot(pca_plot, aes(x = PC1, y = PC2, color = Disease)) +
  geom_point(alpha = 0.75)
```



Plot: die Punkte bilden erkennbare Gruppen, die sich aber überlagern Die ersten beiden Hautpkomponenten können die Unterschiede zwischen gesunden und erkrankten Patienten relativ gut sichtbar machen.

PVE

Plot Plotten Sie die Proportion of Variance explained (PVE) für jede Hauptkomponente sowie die akkumulierte PVE.



Wichtige Hauptkomponenten Wie viele Hauptkomponenten erklären mehr als 50% der Varianz?

Antwort: 6 Hauptkomponenten

Möglicherweise tragen bei Ihrem Ergebnis die letzten Hauptkomponenten keine Varianz mehr bei. Überlegen Sie, woran das liegen könnte.

Antowrt: Durch die Dummy-Codierung aller Faktorstufen entstehen lineare Abhängigkeiten zwischen den Variablen. Der df enthält also weniger unabhängige Informationen.