AMLHC-final-exam

Nico Hagmeier

2024-06-28

Datensatz: 8 unabhängige Variablen und 1 abhängige Variable. 768 Instances

unabhängige Variablen: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction (genetische Disposition), Age

abhängige Variable: Outcome

Da die abhängige Variable binär ist (0,1) liegt ein Klassifikationsproblem vor. Daher habe ich mich für Random Forest entschieden

Forschungsfrage: Kann man mit den angegebenen 8 Features vorhersagen, ob ein Diabetes vorliegt?

```
# Laden der notwendigen Bibliotheken
library(randomForest)
```

```
## Warning: Paket 'randomForest' wurde unter R Version 4.3.3 erstellt
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

```
## Warning: Paket 'caret' wurde unter R Version 4.3.3 erstellt
```

```
## Lade nötiges Paket: ggplot2
```

```
## Warning: Paket 'ggplot2' wurde unter R Version 4.3.3 erstellt
```

```
##
## Attache Paket: 'ggplot2'

## Das folgende Objekt ist maskiert 'package:randomForest':
##
##     margin

## Lade nötiges Paket: lattice

## Warning: Paket 'lattice' wurde unter R Version 4.3.3 erstellt
```

```
library(e1071)
```

```
## Warning: Paket 'e1071' wurde unter R Version 4.3.3 erstellt
```

```
library(pROC)
```

```
## Warning: Paket 'pROC' wurde unter R Version 4.3.3 erstellt

## Type 'citation("pROC")' for a citation.

##
## Attache Paket: 'pROC'

## Die folgenden Objekte sind maskiert von 'package:stats':
##
##     cov, smooth, var
```

```
library(ggplot2)
library(corrplot)
```

```
## Warning: Paket 'corrplot' wurde unter R Version 4.3.3 erstellt

## corrplot 0.92 loaded
```

```
library(reshape2)
```

```
## Warning: Paket 'reshape2' wurde unter R Version 4.3.3 erstellt
```

```
#Laden des Datensatz und Struktur der Daten
diab <- read.csv("C:/Users/Scele/OneDrive/Desktop/AMLHC-final-exam/diabetes.csv", header = TRUE)
str(diab)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

```r
head(diab)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

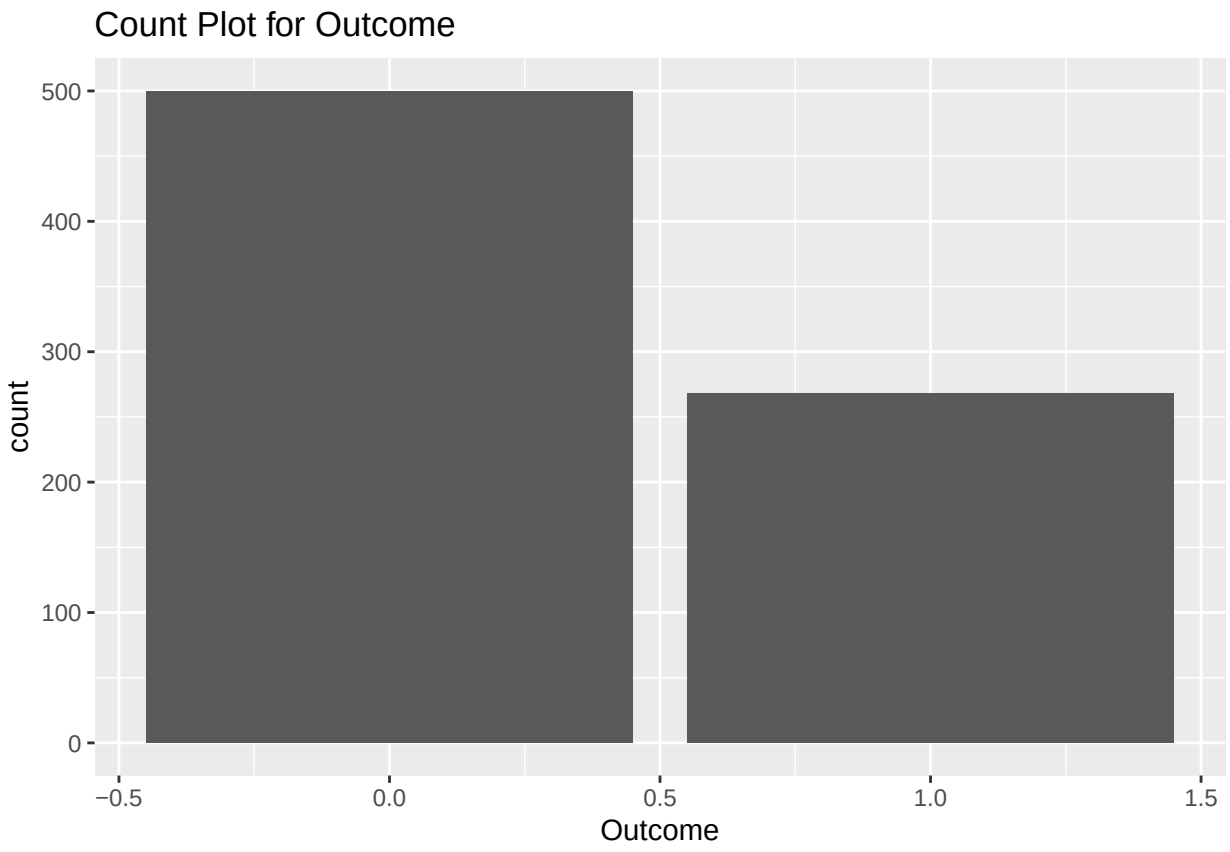
```r
summary(diab)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```r
#Prüfung auf NA-Werte
print(colSums(is.na(diab)))
```

```
##              Pregnancies                  Glucose            BloodPressure
##                        0                        0                        0
##            SkinThickness                  Insulin                      BMI
##                        0                        0                        0
## DiabetesPedigreeFunction                      Age                  Outcome
##                        0                        0                        0
```

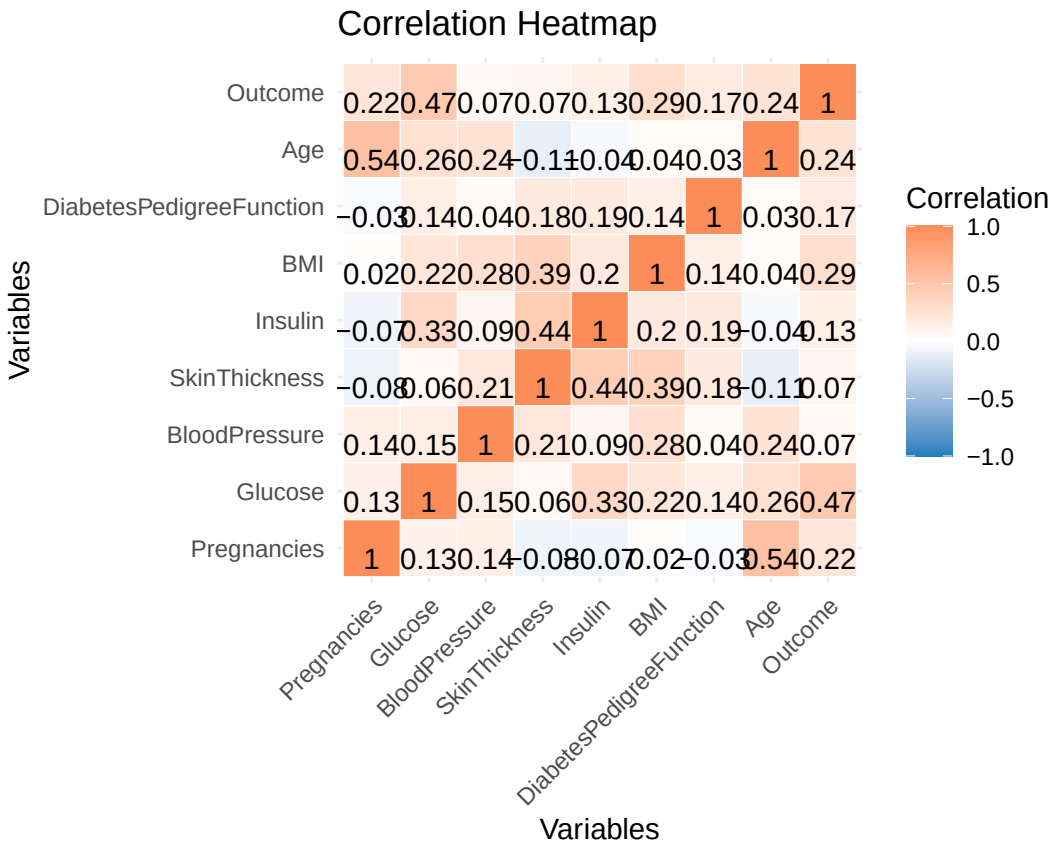
```r
#Anzeige wie viele Diabetiker vorhanden sind
ggplot(diab, aes(x = Outcome)) +
  geom_bar() +
  labs(x = "Outcome") +
  ggtitle("Count Plot for Outcome")
```




```r
# Korrelationsmatrix berechnen
cor_matrix <- cor(diab[, sapply(diab, is.numeric)])

# Umwandlung der Korrelationsmatrix in ein langformatiges Dataframe
cor_matrix_melted <- melt(cor_matrix)

# Erstellen der Korrelationsmatrix-Heatmap
ggplot(cor_matrix_melted, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "#1a7ebd", mid = "white", high = "#fc8d59", midpoint = 0,
                       limits = c(-1, 1), name = "Correlation") +
  geom_text(aes(label = round(value, 2)), vjust = 1) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Correlation Heatmap",
       x = "Variables", y = "Variables") +
  coord_fixed()
```




```
#Doppelte Einträge prüfen und entfernen
diab <- diab[!duplicated(diab), ]
```

```
#Anzeige der NA-Werte
colSums(is.na(diab))
```

```
##              Pregnancies                  Glucose            BloodPressure
##                        0                        0                        0
##            SkinThickness                  Insulin                      BMI
##                        0                        0                        0
## DiabetesPedigreeFunction                      Age                  Outcome
##                        0                        0                        0
```

```
#Anzeige der Werte die 0 entsprechen
print(sum(diab$BloodPressure == 0))
```

```
## [1] 35
```

```
print(sum(diab$Glucose == 0))
```

```
## [1] 5
```

```r
print(sum(diab$SkinThickness == 0))
```

```
## [1] 227
```

```r
print(sum(diab$Insulin == 0))
```

```
## [1] 374
```

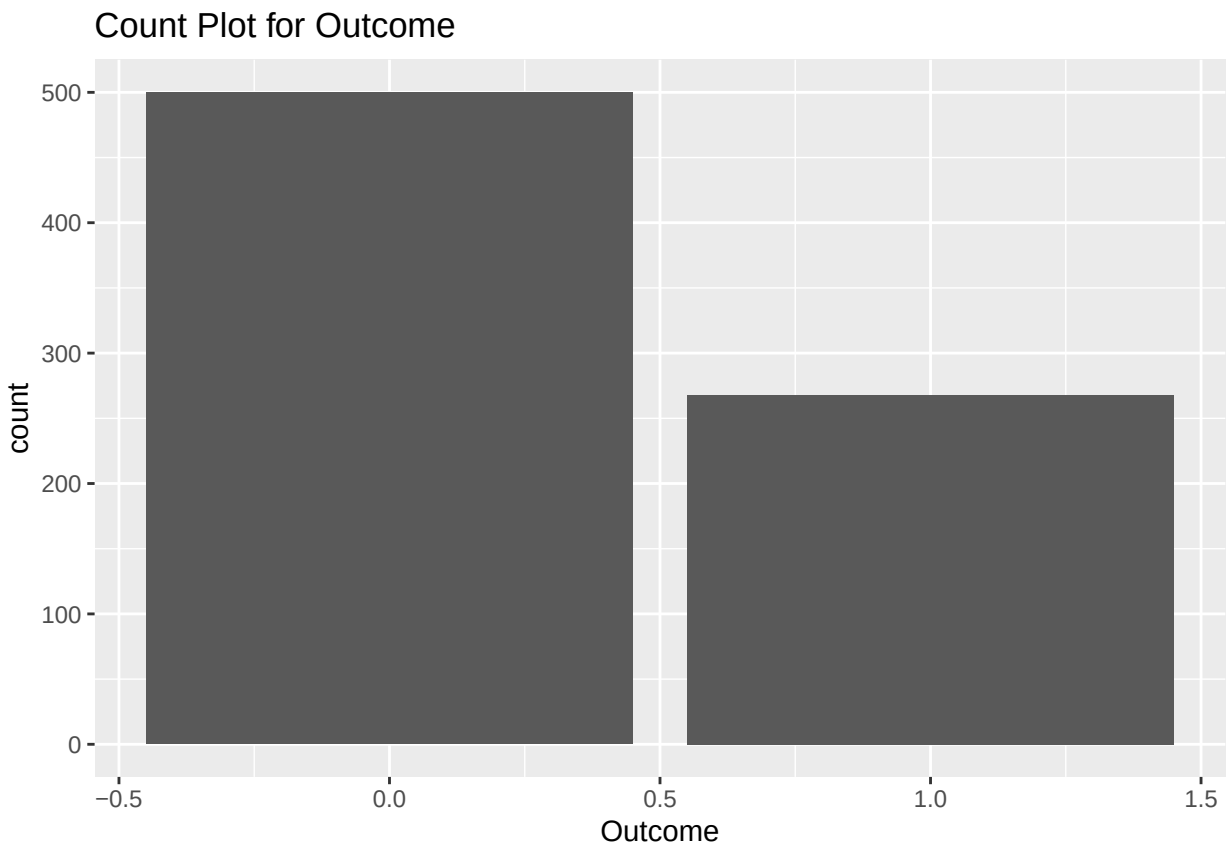
```r
print(sum(diab$BMI == 0))
```

```
## [1] 11
```

```r
#Ersetzen der 0-Werte
diab$Glucose[diab$Glucose == 0] <- mean(diab$Glucose[diab$Glucose != 0], na.rm = TRUE)
diab$BloodPressure[diab$BloodPressure == 0] <- mean(diab$BloodPressure[diab$BloodPressure != 0], na.rm =
diab$SkinThickness[diab$SkinThickness == 0] <- median(diab$SkinThickness[diab$SkinThickness != 0], na.r
diab$Insulin[diab$Insulin == 0] <- median(diab$Insulin[diab$Insulin != 0], na.rm = TRUE)
diab$BMI[diab$BMI == 0] <- median(diab$BMI[diab$BMI != 0], na.rm = TRUE)
```

neue Anzeige der Daten

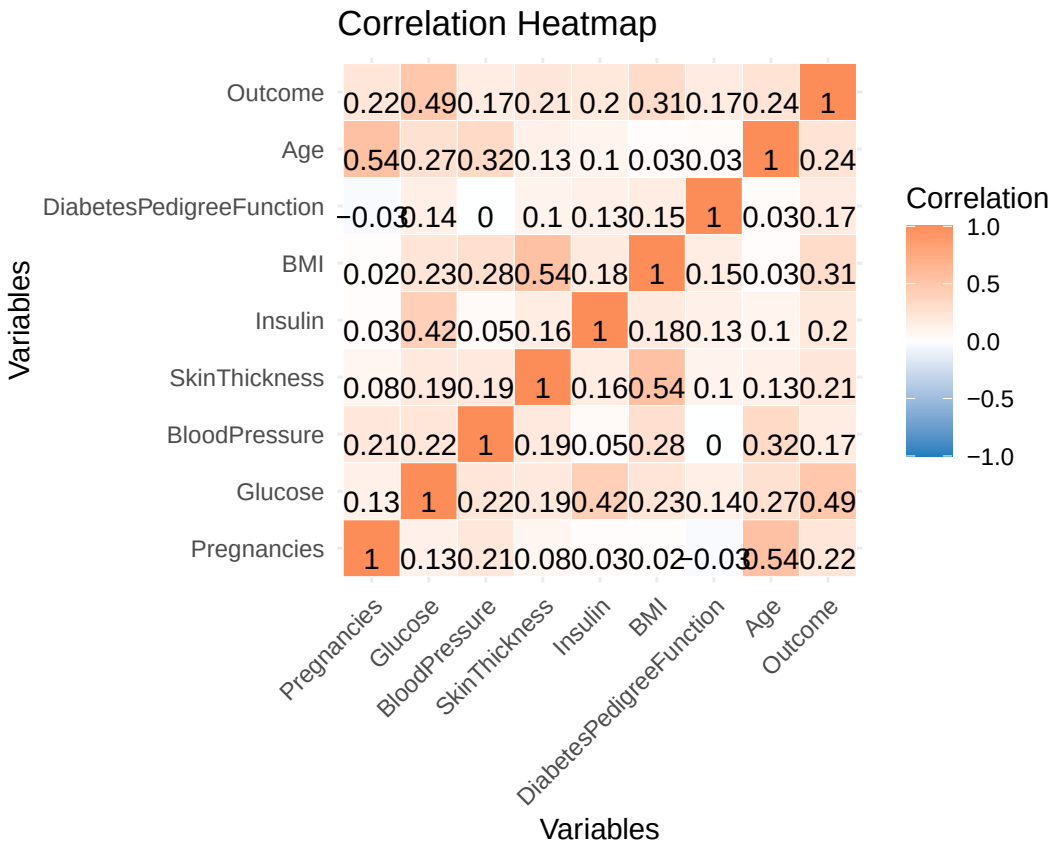
```r
ggplot(diab, aes(x = Outcome)) +
  geom_bar() +
  labs(x = "Outcome") +
  ggtitle("Count Plot for Outcome")
```




```r
# Korrelationsmatrix berechnen
cor_matrix <- cor(diab[, sapply(diab, is.numeric)])

# Umwandlung der Korrelationsmatrix in ein langformatiges Dataframe
cor_matrix_melted <- melt(cor_matrix)

# Erstellen der Korrelationsmatrix-Heatmap
ggplot(cor_matrix_melted, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "#1a7ebd", mid = "white", high = "#fc8d59", midpoint = 0,
                       limits = c(-1, 1), name = "Correlation") +
  geom_text(aes(label = round(value, 2)), vjust = 1) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Correlation Heatmap",
       x = "Variables", y = "Variables") +
  coord_fixed()
```




```
#Umwandlung in einen Faktor
diab$Outcome <- as.factor(diab$Outcome)
```

```
set.seed(123)  # Für Reproduzierbarkeit
trainIndex <- createDataPartition(diab$Outcome, p = 0.7, list = FALSE)
trainData <- diab[trainIndex, ]
testData <- diab[-trainIndex, ]

X_train <- trainData[, -ncol(trainData)]
y_train <- trainData$Outcome
X_test <- testData[, -ncol(testData)]
y_test <- testData$Outcome
```

```
# Definieren des Modells und der Parameter
model <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "grid")
tunegrid <- expand.grid(.mtry = c(sqrt(ncol(X_train)), log2(ncol(X_train))))
```

```
# Durchführung der Grid-Suche
set.seed(1)
rf_gridsearch <- train(X_train, y_train, method = "rf", metric = "Accuracy", tuneGrid = tunegrid, trCon
# Beste Parameter finden
best_model <- rf_gridsearch$finalModel
print(rf_gridsearch)
```

```
## Random Forest
```

```
##
## 538 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 484, 485, 484, 485, 484, 484, ...
## Resampling results across tuning parameters:
##
##   mtry      Accuracy   Kappa
##   2.828427  0.7458887  0.4257675
##   3.000000  0.7458887  0.4260871
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.828427.
```

```
# Vorhersagen mit dem besten Modell
rf_pred <- predict(rf_gridsearch, X_test)
```

```
# Klassifikationsbericht ausdrucken
conf_matrix <- confusionMatrix(rf_pred, y_test)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 129  31
##          1  21  49
##
##                Accuracy : 0.7739
##                  95% CI : (0.7143, 0.8263)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 4.208e-05
##
##                   Kappa : 0.4867
##
##  Mcnemar's Test P-Value : 0.212
##
##             Sensitivity : 0.8600
##             Specificity : 0.6125
##          Pos Pred Value : 0.8063
##          Neg Pred Value : 0.7000
##              Prevalence : 0.6522
##          Detection Rate : 0.5609
##    Detection Prevalence : 0.6957
##       Balanced Accuracy : 0.7363
##
##        'Positive' Class : 0
##
```

```r
precision <- posPredValue(rf_pred, y_test, positive = "1")
recall <- sensitivity(rf_pred, y_test, positive = "1")
f1 <- 2 * (precision * recall) / (precision + recall)

cat("\nPrecision:\n", precision)
```

```
##
## Precision:
##  0.7
```

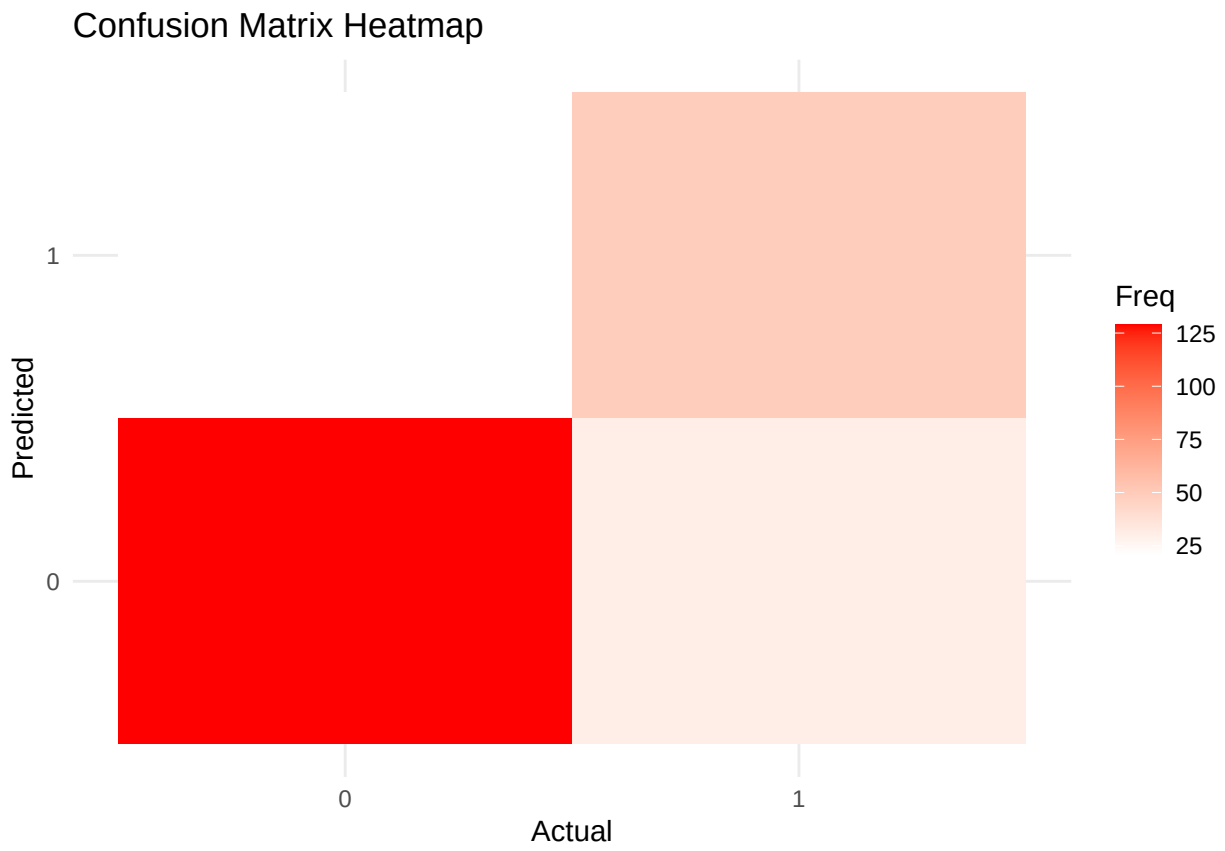
```r
cat("\nRecall:\n", recall)
```

```
##
## Recall:
##  0.6125
```

```r
cat("\nF1-Score:\n", f1)
```

```
##
## F1-Score:
##  0.6533333
```

```r
# Konfusionsmatrix als Heatmap darstellen
conf_matrix_table <- as.table(conf_matrix$table)
ggplot(data = as.data.frame(conf_matrix_table), aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted")
```



#andere p Verteilung um auf Overfitting zu testen

```
set.seed(123)  # Für Reproduzierbarkeit
trainIndex <- createDataPartition(diab$Outcome, p = 0.8, list = FALSE)
trainData <- diab[trainIndex, ]
testData <- diab[-trainIndex, ]

X_train <- trainData[, -ncol(trainData)]
y_train <- trainData$Outcome
X_test <- testData[, -ncol(testData)]
y_test <- testData$Outcome
```

```
# Definieren des Modells und der Parameter
model <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "grid")
tunegrid <- expand.grid(.mtry = c(sqrt(ncol(X_train)), log2(ncol(X_train))))
```

```
# Durchführung der Grid-Suche
set.seed(1)
rf_gridsearch <- train(X_train, y_train, method = "rf", metric = "Accuracy", tuneGrid = tunegrid, trCon

# Beste Parameter finden
best_model <- rf_gridsearch$finalModel
print(rf_gridsearch)
```

```
## Random Forest
```

```
##
## 615 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 554, 554, 553, 554, 554, 553, ...
## Resampling results across tuning parameters:
##
##   mtry      Accuracy   Kappa
##   2.828427  0.7495505  0.4300026
##   3.000000  0.7517275  0.4350362
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

```
# Vorhersagen mit dem besten Modell
rf_pred <- predict(rf_gridsearch, X_test)
```

```
# Klassifikationsbericht ausdrucken
conf_matrix <- confusionMatrix(rf_pred, y_test)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 88 17
##          1 12 36
##
##                Accuracy : 0.8105
##                  95% CI : (0.7393, 0.8692)
##     No Information Rate : 0.6536
##     P-Value [Acc > NIR] : 1.46e-05
##
##                   Kappa : 0.5719
##
##  Mcnemar's Test P-Value : 0.4576
##
##             Sensitivity : 0.8800
##             Specificity : 0.6792
##          Pos Pred Value : 0.8381
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6536
##          Detection Rate : 0.5752
##    Detection Prevalence : 0.6863
##       Balanced Accuracy : 0.7796
##
##        'Positive' Class : 0
##
```

```r
precision <- posPredValue(rf_pred, y_test, positive = "1")
recall <- sensitivity(rf_pred, y_test, positive = "1")
f1 <- 2 * (precision * recall) / (precision + recall)

cat("\nPrecision:\n", precision)
```

```
##
## Precision:
##  0.75
```

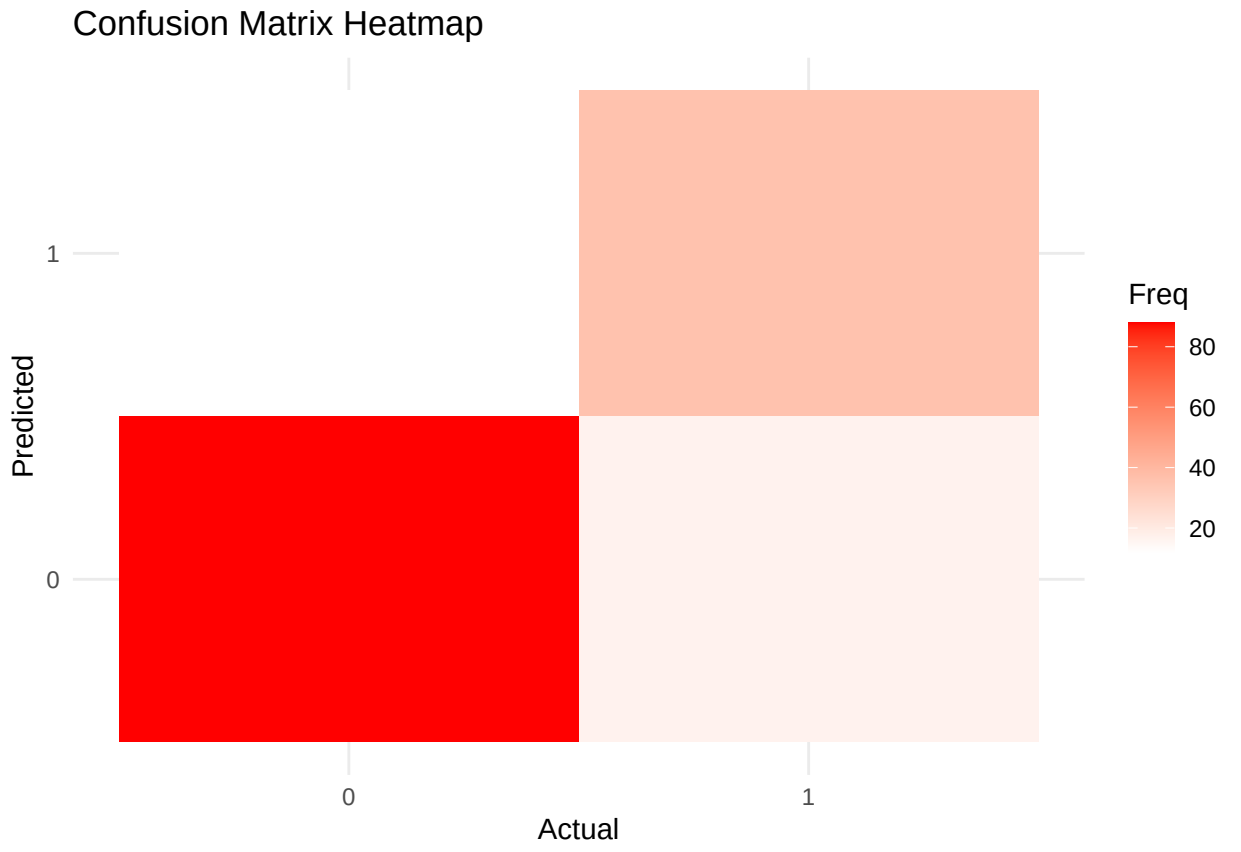
```r
cat("\nRecall:\n", recall)
```

```
##
## Recall:
##  0.6792453
```

```r
cat("\nF1-Score:\n", f1)
```

```
##
## F1-Score:
##  0.7128713
```

```r
# Konfusionsmatrix als Heatmap darstellen
conf_matrix_table <- as.table(conf_matrix$table)
ggplot(data = as.data.frame(conf_matrix_table), aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted")
```




```r
set.seed(123)  # Für Reproduzierbarkeit
trainIndex <- createDataPartition(diab$Outcome, p = 0.9, list = FALSE)
trainData <- diab[trainIndex, ]
testData <- diab[-trainIndex, ]

X_train <- trainData[, -ncol(trainData)]
y_train <- trainData$Outcome
X_test <- testData[, -ncol(testData)]
y_test <- testData$Outcome
```

```r
# Definieren des Modells und der Parameter
model <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "grid")
tunegrid <- expand.grid(.mtry = c(sqrt(ncol(X_train)), log2(ncol(X_train))))
```

```r
# Durchführung der Grid-Suche
set.seed(1)
rf_gridsearch <- train(X_train, y_train, method = "rf", metric = "Accuracy", tuneGrid = tunegrid, trCon

# Beste Parameter finden
best_model <- rf_gridsearch$finalModel
print(rf_gridsearch)
```

```
## Random Forest
##
## 692 samples
```

```
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 623, 623, 622, 622, 623, 623, ...
## Resampling results across tuning parameters:
##
##   mtry      Accuracy   Kappa
##   2.828427  0.7625259  0.4675059
##   3.000000  0.7620428  0.4646068
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.828427.
```

```
# Vorhersagen mit dem besten Modell
rf_pred <- predict(rf_gridsearch, X_test)
```

```
# Klassifikationsbericht ausdrucken
conf_matrix <- confusionMatrix(rf_pred, y_test)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 45  9
##          1  5 17
##
##                Accuracy : 0.8158
##                  95% CI : (0.7103, 0.8955)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.001846
##
##                   Kappa : 0.5751
##
##  Mcnemar's Test P-Value : 0.422678
##
##             Sensitivity : 0.9000
##             Specificity : 0.6538
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.7727
##              Prevalence : 0.6579
##          Detection Rate : 0.5921
##    Detection Prevalence : 0.7105
##       Balanced Accuracy : 0.7769
##
##        'Positive' Class : 0
##
```

```
precision <- posPredValue(rf_pred, y_test, positive = "1")
recall <- sensitivity(rf_pred, y_test, positive = "1")
f1 <- 2 * (precision * recall) / (precision + recall)
```

```r
cat("\nPrecision:\n", precision)
```

```
##
## Precision:
##  0.7727273
```

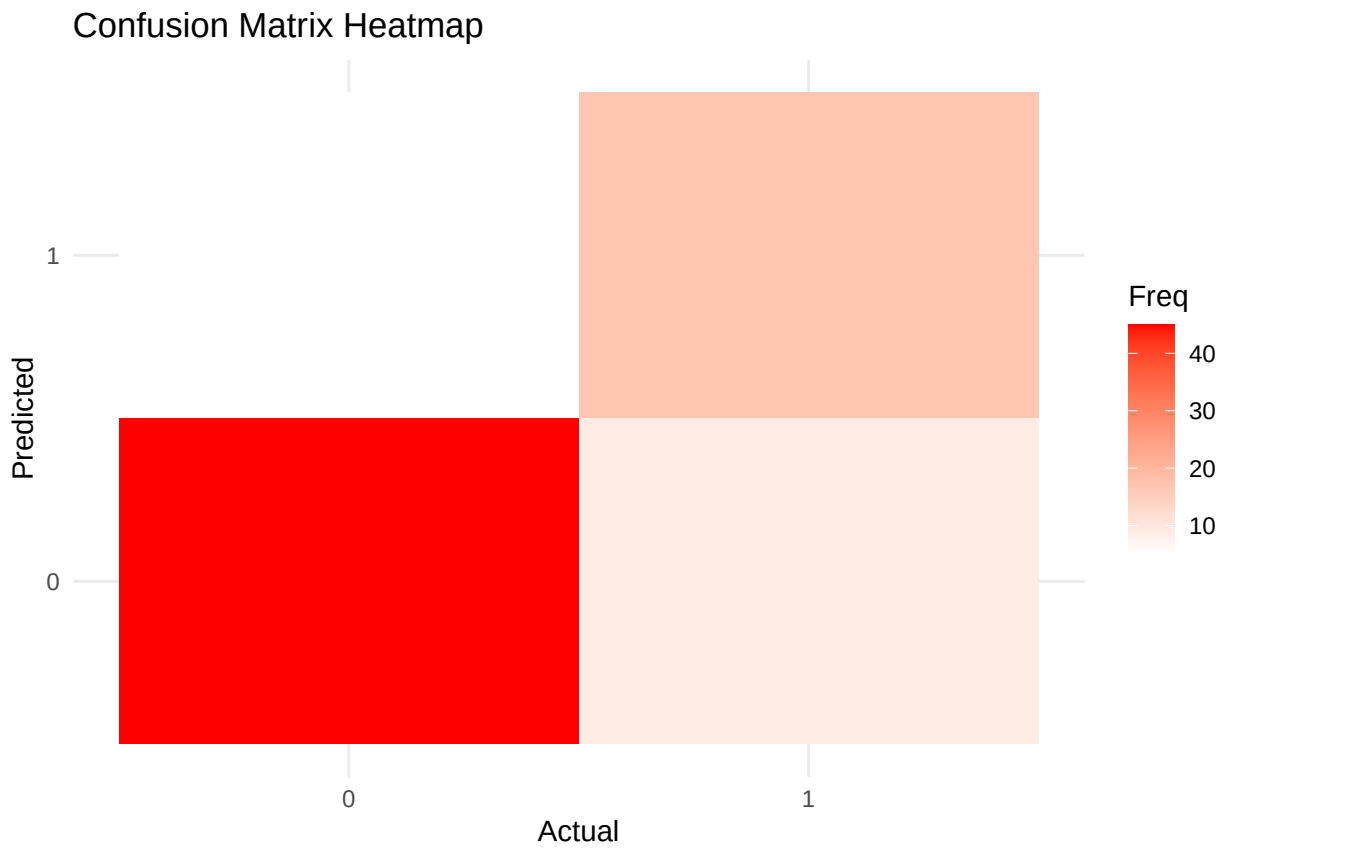
```r
cat("\nRecall:\n", recall)
```

```
##
## Recall:
##  0.6538462
```

```r
cat("\nF1-Score:\n", f1)
```

```
##
## F1-Score:
##  0.7083333
```

```r
# Konfusionsmatrix als Heatmap darstellen
conf_matrix_table <- as.table(conf_matrix$table)
ggplot(data = as.data.frame(conf_matrix_table), aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted")
```



```r
set.seed(123)  # Für Reproduzierbarkeit
trainIndex <- createDataPartition(diab$Outcome, p = 0.75, list = FALSE)
trainData <- diab[trainIndex, ]
testData <- diab[-trainIndex, ]

X_train <- trainData[, -ncol(trainData)]
y_train <- trainData$Outcome
X_test <- testData[, -ncol(testData)]
y_test <- testData$Outcome
```

```r
# Definieren des Modells und der Parameter
model <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "grid")
tunegrid <- expand.grid(.mtry = c(sqrt(ncol(X_train)), log2(ncol(X_train))))
```

```r
# Durchführung der Grid-Suche
set.seed(1)
rf_gridsearch <- train(X_train, y_train, method = "rf", metric = "Accuracy", tuneGrid = tunegrid, trCon

# Beste Parameter finden
best_model <- rf_gridsearch$finalModel
print(rf_gridsearch)
```

```
## Random Forest
##
## 576 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 519, 518, 519, 518, 518, 519, ...
## Resampling results across tuning parameters:
##
##   mtry      Accuracy   Kappa
##   2.828427  0.7505848  0.4364556
##   3.000000  0.7540532  0.4434844
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

```r
# Vorhersagen mit dem besten Modell
rf_pred <- predict(rf_gridsearch, X_test)
```

```r
# Klassifikationsbericht ausdrucken
conf_matrix <- confusionMatrix(rf_pred, y_test)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0 107  24
##          1  18  43
##
##                Accuracy : 0.7812
##                  95% CI : (0.716, 0.8376)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 6.137e-05
##
##                   Kappa : 0.5084
##
##  Mcnemar's Test P-Value : 0.4404
##
##             Sensitivity : 0.8560
##             Specificity : 0.6418
##          Pos Pred Value : 0.8168
##          Neg Pred Value : 0.7049
##              Prevalence : 0.6510
##          Detection Rate : 0.5573
##    Detection Prevalence : 0.6823
##       Balanced Accuracy : 0.7489
##
##        'Positive' Class : 0
##
```

```r
precision <- posPredValue(rf_pred, y_test, positive = "1")
recall <- sensitivity(rf_pred, y_test, positive = "1")
f1 <- 2 * (precision * recall) / (precision + recall)

cat("\nPrecision:\n", precision)
```

```
##
## Precision:
##  0.704918
```

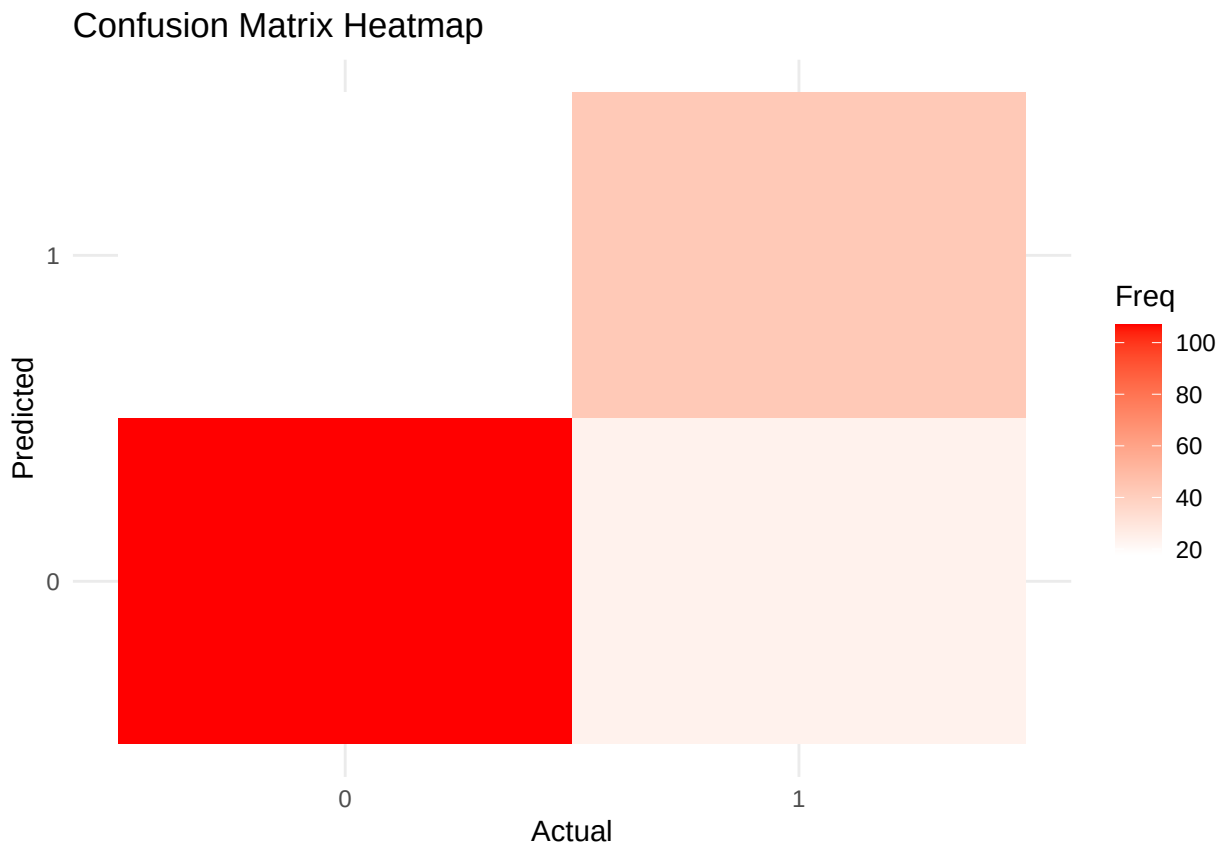
```r
cat("\nRecall:\n", recall)
```

```
##
## Recall:
##  0.641791
```

```r
cat("\nF1-Score:\n", f1)
```

```
##
## F1-Score:
##  0.671875
```

```r
# Konfusionsmatrix als Heatmap darstellen
conf_matrix_table <- as.table(conf_matrix$table)
ggplot(data = as.data.frame(conf_matrix_table), aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted")
```




```r
set.seed(123)  # Für Reproduzierbarkeit
trainIndex <- createDataPartition(diab$Outcome, p = 0.85, list = FALSE)
trainData <- diab[trainIndex, ]
testData <- diab[-trainIndex, ]

X_train <- trainData[, -ncol(trainData)]
y_train <- trainData$Outcome
X_test <- testData[, -ncol(testData)]
y_test <- testData$Outcome
```

```r
# Definieren des Modells und der Parameter
model <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "grid")
tunegrid <- expand.grid(.mtry = c(sqrt(ncol(X_train)), log2(ncol(X_train))))
```

```r
# Durchführung der Grid-Suche
set.seed(1)
rf_gridsearch <- train(X_train, y_train, method = "rf", metric = "Accuracy", tuneGrid = tunegrid, trCon
# Beste Parameter finden
best_model <- rf_gridsearch$finalModel
print(rf_gridsearch)
```

```
## Random Forest
##
## 653 samples
```

```
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 588, 588, 588, 588, 587, 588, ...
## Resampling results across tuning parameters:
##
##   mtry      Accuracy   Kappa
##   2.828427  0.7625597  0.4661485
##   3.000000  0.7620469  0.4639092
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.828427.
```

```
# Vorhersagen mit dem besten Modell
rf_pred <- predict(rf_gridsearch, X_test)
```

```
# Klassifikationsbericht ausdrucken
conf_matrix <- confusionMatrix(rf_pred, y_test)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 65 14
##          1 10 26
##
##                Accuracy : 0.7913
##                  95% CI : (0.7056, 0.8615)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 0.0008256
##
##                   Kappa : 0.529
##
##  Mcnemar's Test P-Value : 0.5402914
##
##             Sensitivity : 0.8667
##             Specificity : 0.6500
##          Pos Pred Value : 0.8228
##          Neg Pred Value : 0.7222
##              Prevalence : 0.6522
##          Detection Rate : 0.5652
##    Detection Prevalence : 0.6870
##       Balanced Accuracy : 0.7583
##
##        'Positive' Class : 0
##
```

```
precision <- posPredValue(rf_pred, y_test, positive = "1")
recall <- sensitivity(rf_pred, y_test, positive = "1")
f1 <- 2 * (precision * recall) / (precision + recall)
```

```r
cat("\nPrecision:\n", precision)
```

```
##
## Precision:
##  0.7222222
```

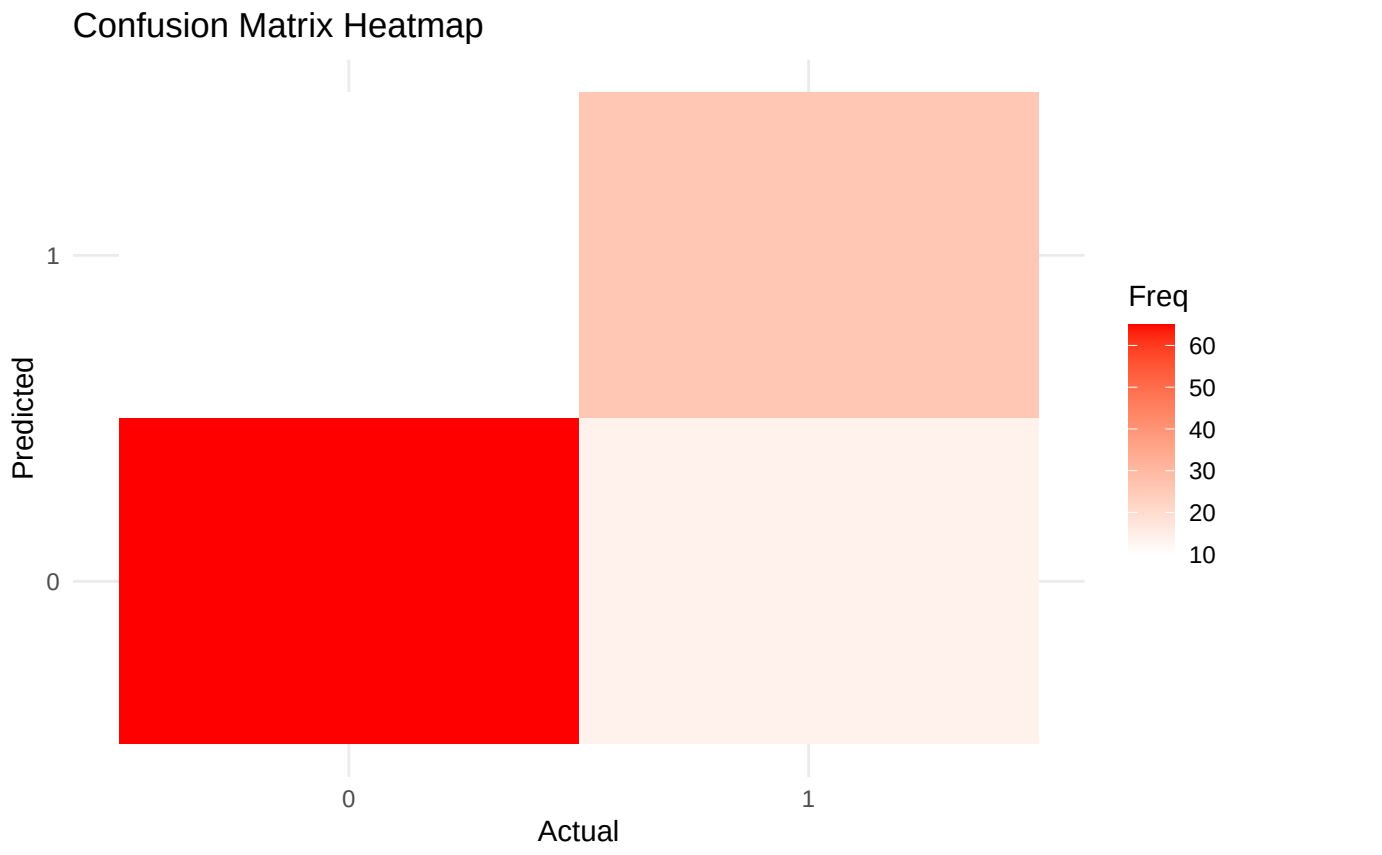
```r
cat("\nRecall:\n", recall)
```

```
##
## Recall:
##  0.65
```

```r
cat("\nF1-Score:\n", f1)
```

```
##
## F1-Score:
##  0.6842105
```

```r
# Konfusionsmatrix als Heatmap darstellen
conf_matrix_table <- as.table(conf_matrix$table)
ggplot(data = as.data.frame(conf_matrix_table), aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Confusion Matrix Heatmap", x = "Actual", y = "Predicted")
```



#Mit einer Präzision von knapp 81% kann vorhergesagt werden, ob ein Diabetes vorliegt.