Prediction

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

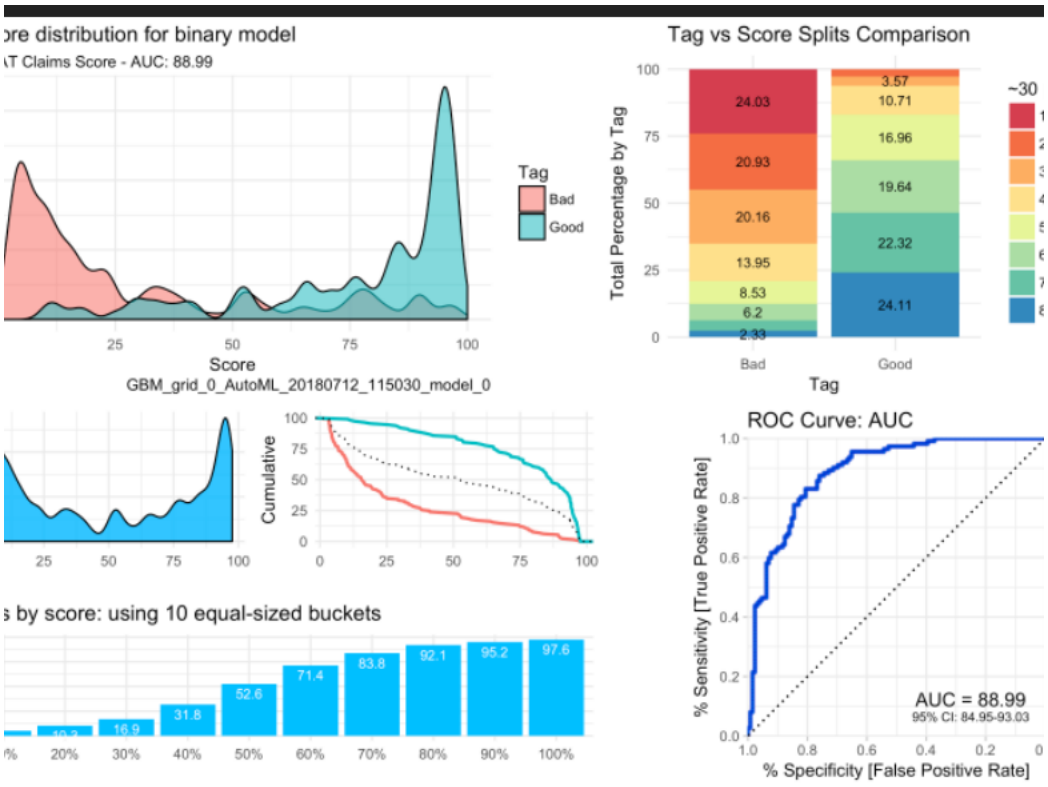


Figure 1: prediction

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
```

```
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
```

summarize all models

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

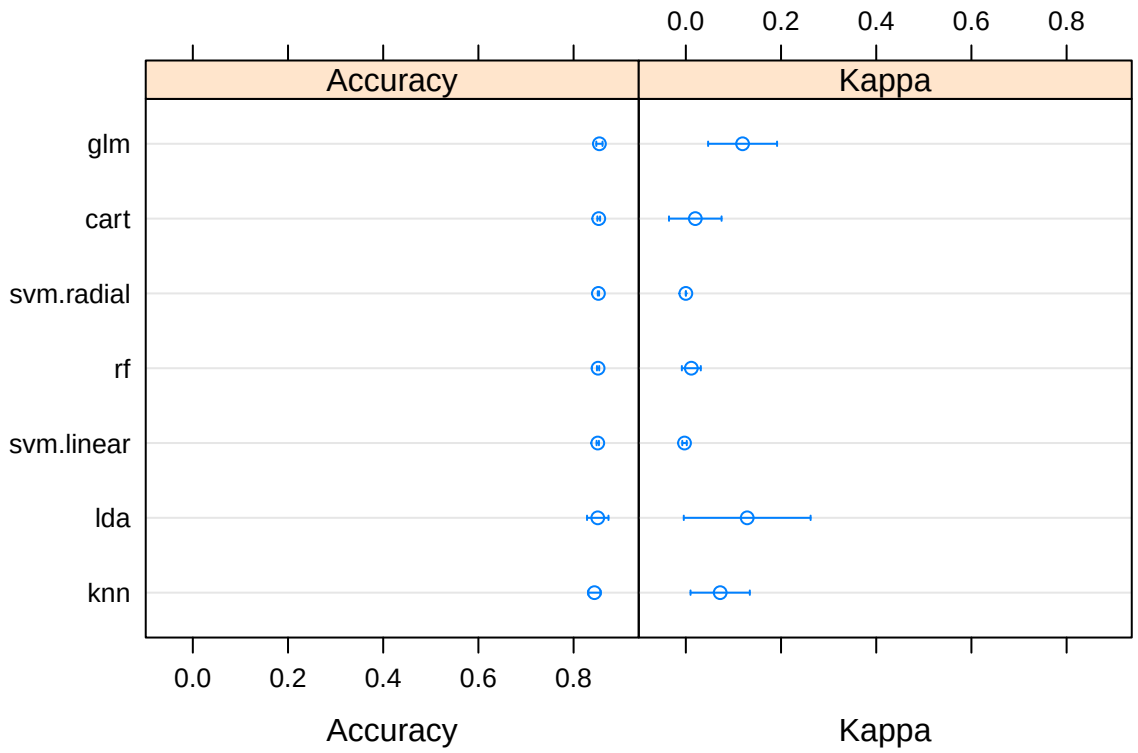
```
##
## Call:
## summary.resamples(object = results)
##
## Models: glm, lda, cart, knn, svm.radial, svm.linear, rf
## Number of resamples: 5
##
```

```
## 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.

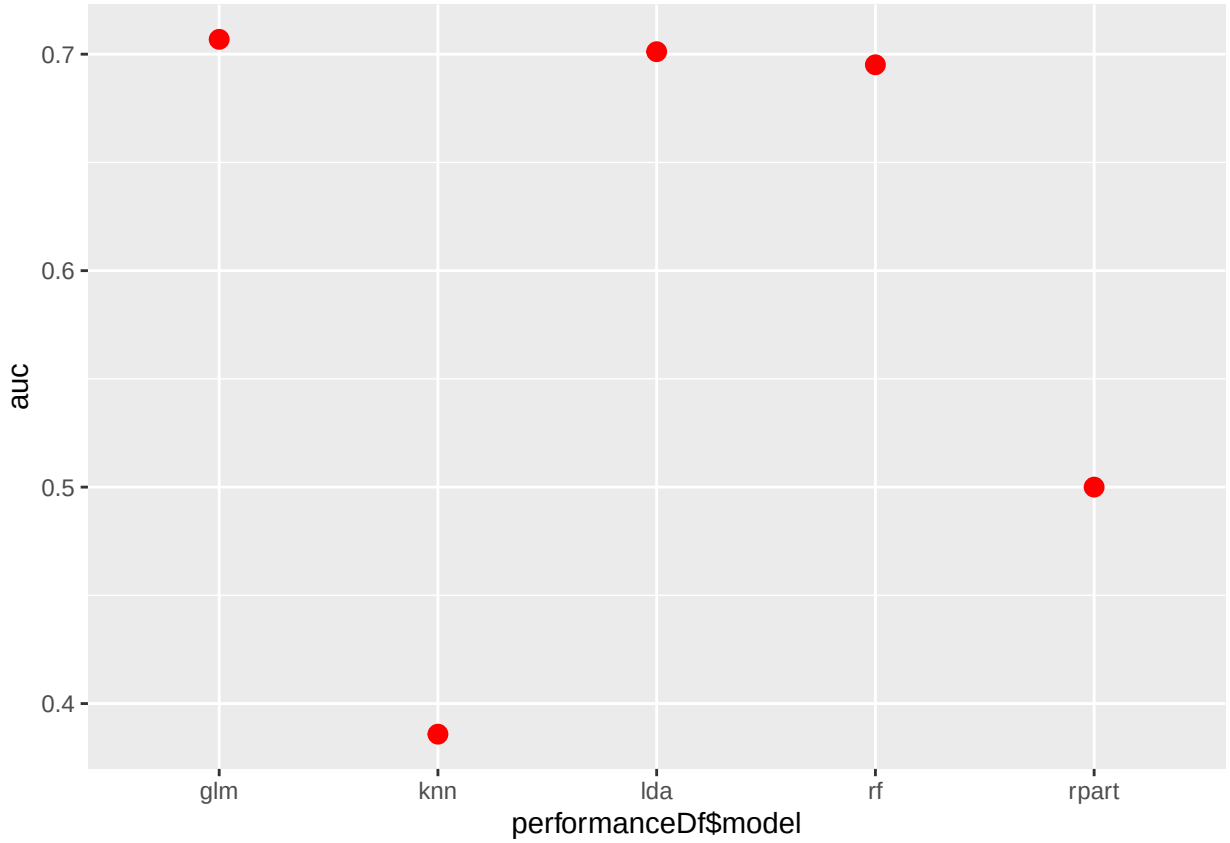
```
computeData <- F
models <- c("rf", "lda", "rpart", "knn", "glm")
workPath = "D:\\DataMining\\Seafile\\01_cardio\\Projekt\\"

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 0.8453333  0.01287554   0.9984215 0.6951062
## 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.