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'
```

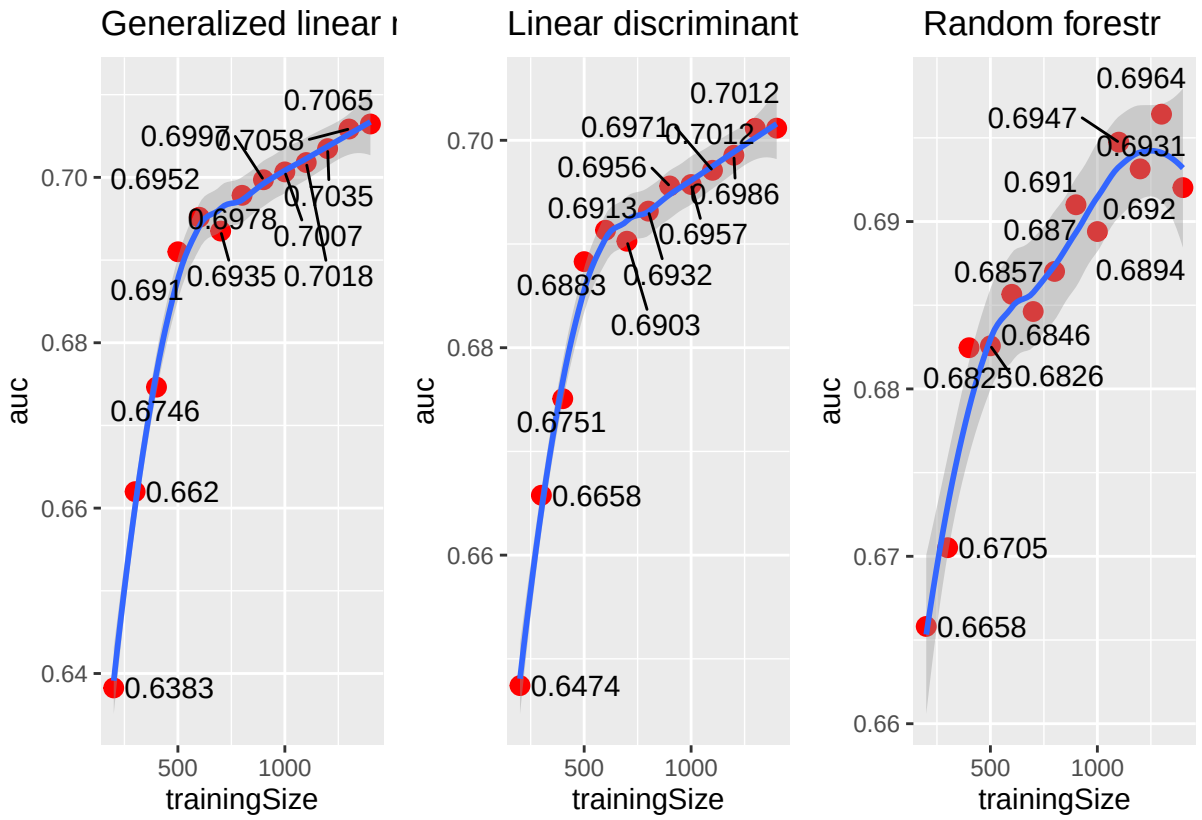
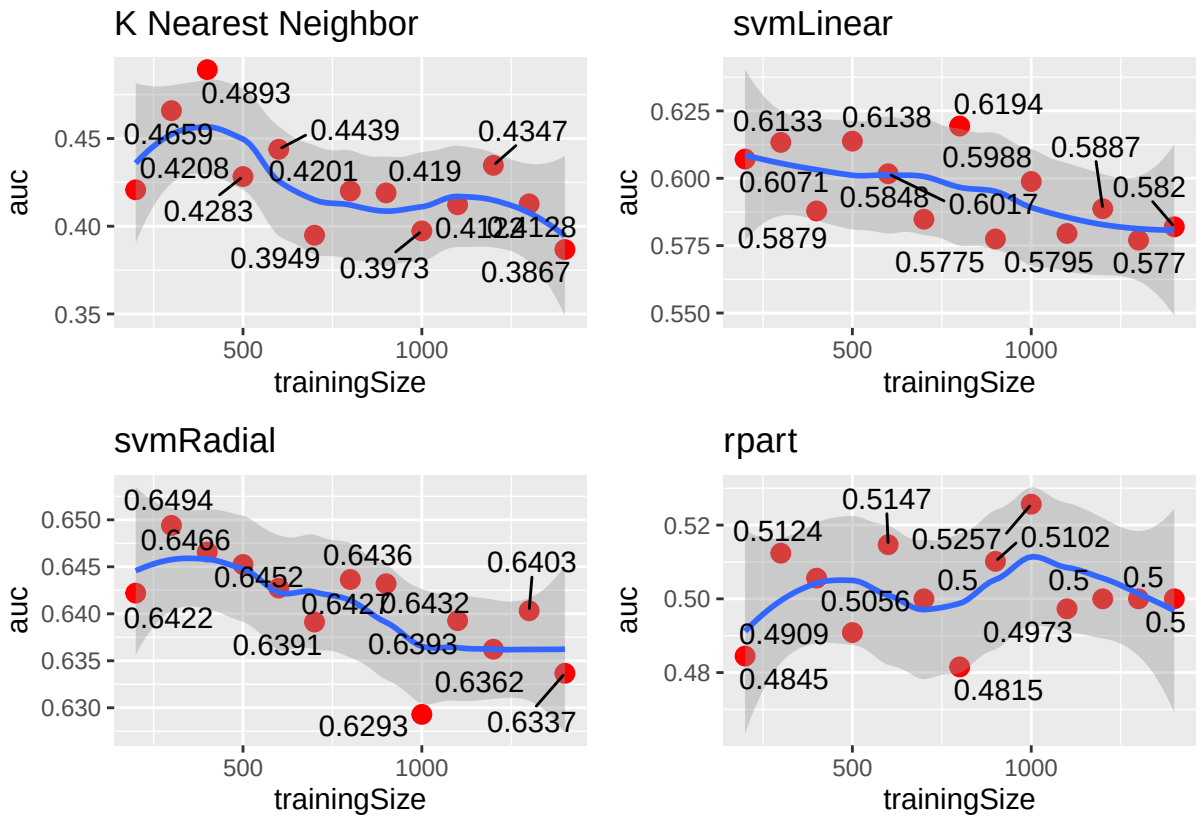


diagram von Knn ,svmLinear and svmRadial and rpart

```
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.