LAB Prostate LinReg 5.R Focus on the the description of the Ridge method in connection with cross-validation. Bonus: Apply the same method of the cross-validation to the Lasso method.

Behrooz Filzadeh

2025-05-15

#Prostata-Datenanalyse mit Ridge Regression und Kreuzvalidierung + Bonus: Lasso mit Kreuzvalidierung 1. Bibliotheken laden

```
library(data.table)
library(ggplot2)
library(leaps)      # Variable Selection
library(glmnet)     # Ridge & Lasso Modelle
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(DataExplorer)
```

Interpretation: Wir laden Pakete, die wir brauchen für Daten, Visualisierungen und für die Modellierung mit Ridge und Lasso.

#2. Daten laden und aufteilen

```
prostateData <- read.table(file="prostate_data.csv")
prostateData <- as.data.table(prostateData)
table(prostateData$train)
```

```
##
## FALSE  TRUE
##    30    67
```

```
prostateData_train <- prostateData[train==TRUE]
prostateData_test <- prostateData[train==FALSE]
prostateData_train$train <- NULL
prostateData_test$train <- NULL
```

Interpretation: Die Daten werden geladen und danach in Trainings- und Testdaten getrennt, damit wir Modelle trainieren und später überprüfen können.

#3. Standardisieren der Prädiktoren

```
prostateData_train_scaled <- scale(prostateData_train[, 1:8])
prostateData_train_scaled <- as.data.table(prostateData_train_scaled)
prostateData_train_scaled[, lpsa := prostateData_train$lpsa]

prostateData_test_scaled <- scale(prostateData_test[, 1:8])
prostateData_test_scaled <- as.data.table(prostateData_test_scaled)
prostateData_test_scaled[, lpsa := prostateData_test$lpsa]
```

Interpretation: Wir bringen alle Prädiktoren auf den gleichen Maßstab (Mittelwert 0, Standardabweichung 1). Das ist wichtig, damit die Regressionen korrekt funktionieren, besonders Ridge und Lasso.

#4. Ridge Regression mit Kreuzvalidierung

```
set.seed(123)  # Für Reproduzierbarkeit

x <- as.matrix(prostateData_train_scaled[,1:8])
y <- prostateData_train_scaled$lpsa

# Kreuzvalidierung mit 10 Folds für Ridge (alpha=0)
cvout <- cv.glmnet(x, y, alpha = 0, standardize = FALSE, intercept = TRUE, nfolds = 10)

# Informationen über das Ergebnis anschauen
typeof(cvout)
```

```
## [1] "list"
```

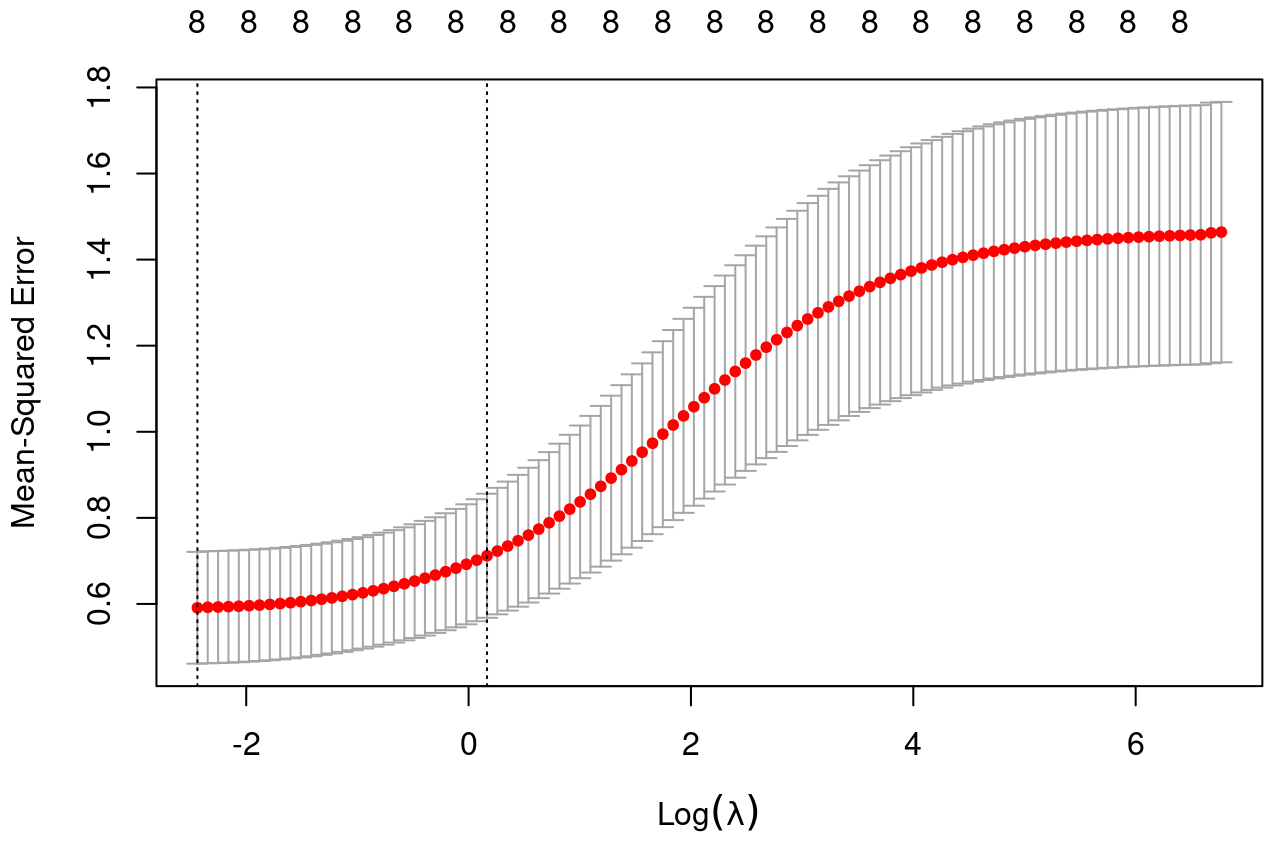
```
attributes(cvout)
```

```
## $names
##  [1] "lambda"     "cvm"        "cvsd"       "cvup"       "cvlo"
##  [6] "nzero"      "call"       "name"       "glmnet.fit" "lambda.min"
## [11] "lambda.1se" "index"
##
## $class
## [1] "cv.glmnet"
```

```
summary(cvout)
```

```
##            Length Class  Mode
## lambda     100    -none- numeric
## cvm        100    -none- numeric
## cvsd       100    -none- numeric
## cvup       100    -none- numeric
## cvlo       100    -none- numeric
## nzero      100    -none- numeric
## call         7    -none- call
## name         1    -none- character
## glmnet.fit  12    elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```
# Plot der Kreuzvalidierung: MSE in Abhängigkeit von Lambda
plot(cvout)
```




```
# Lambda-Wert mit der besten Leistung (1 Standardfehler Regel)
cvout$lambda.1se
```

```
## [1] 1.180259
```

Interpretation: Wir verwenden eine 10-fache Kreuzvalidierung, um den besten Wert von lambda (Schrumpfungsparameter) für Ridge zu finden.

cv.glmnet teilt die Daten in 10 Teile, trainiert auf 9 und prüft auf dem 10. Das wird 10-mal gemacht.

lambda.1se gibt das Lambda mit möglichst kleinem Fehler, aber höherer Einfachheit an (1 Standardfehler-Regel).

Der Plot zeigt, wie der Fehler (Mean Squared Error) sich mit verschiedenen Lambda ändert.