LAB Prostate LinReg 3b.R Focus on the topic model selection by using the forward subset selection with cross-validation.

Behrooz Filzadeh

2025-05-14

```
library(data.table)
library(ggplot2)
library(leaps)
library(caret)
```

```
## Loading required package: lattice
```

```
prostateData <- read.table(file="prostate_data.csv", header = TRUE, sep = "\t",row.names =1)
prostateData <- as.data.table(prostateData)

prostateData_train <- prostateData[train==TRUE]
prostateData_test <- prostateData[train==FALSE]

prostateData_train$train <- NULL
prostateData_test$train <- NULL


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

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

#Einleitung Dieses Dokument analysiert das R-Skript LAB Prostate LinReg 3b.R. Der zentrale Fokus liegt hier auf der Modellauswahl mittels Forward Subset Selection in Kombination mit Kreuzvalidierung (Cross-Validation, CV). Das Ziel ist es, die optimale Anzahl von Prädiktoren für ein lineares Regressionsmodell zu bestimmen, indem die Leistung von Modellen unterschiedlicher Komplexität durch Kreuzvalidierung auf den Trainingsdaten bewertet wird. Die Kreuzvalidierung hilft uns, eine robustere Schätzung der Modellgüte zu erhalten und die Gefahr von Overfitting bei der Auswahl der Prädiktorenanzahl zu reduzieren. Wir verwenden dazu die train-Funktion aus dem caret-Paket. Interpretation: Die Bibliothek caret ist hier zentral, da sie die Funktionen für das Training von Modellen mit Kreuzvalidierung (trainControl, train) und spezifische Methoden wie Forward Selection (method = “leapForward”) bereitstellt. #2. Datenaufbereitung (Identisch zu vorherigen Skripten)

```
# Laden und Vorbereiten der Daten
prostateData <- read.table(file="prostate_data.csv", header = TRUE, sep = "\t",row.names =1)
prostateData <- as.data.table(prostateData)

prostateData_train <- prostateData[train==TRUE]
prostateData_test <- prostateData[train==FALSE]

prostateData_train$train <- NULL
prostateData_test$train <- NULL

# Standardisieren der Trainingsdaten
temp_lpsa_train <- prostateData_train$lpsa
prostateData_train_scaled <- scale(prostateData_train[, 1:8])
# Spalten 1-8 sind Prädiktoren
prostateData_train_scaled <- as.data.table(prostateData_train_scaled)
prostateData_train_scaled[, lpsa:=temp_lpsa_train]
# Original lpsa hinzufügen

# Standardisieren der Testdaten
temp_lpsa_test <- prostateData_test$lpsa
prostateData_test_scaled <- scale(prostateData_test[, 1:8])
# Spalten 1-8 sind Prädiktoren
prostateData_test_scaled <- as.data.table(prostateData_test_scaled)
prostateData_test_scaled[, lpsa:=temp_lpsa_test]
# Original lpsa hinzufügen
```

#Interpretation: Die Datenaufbereitungsschritte sind identisch zu den vorherigen Skripten. Wir benötigen prostateData_train_scaled für das Training mit Kreuzvalidierung und prostateData_test_scaled für die finale Bewertung des ausgewählten Modells. #3. Modellauswahl mit Forward Selection und Kreuzvalidierung

```
# Festlegen der Kontrollparameter für die Kreuzvalidierung
# method = "cv": Kreuzvalidierung
# number = 10: Anzahl der Folds (Teilmengen) für die 10-fache Kreuzvalidierung
train_control <- trainControl(method = "cv", number=10)

# Definieren des Suchrasters für den Hyperparameter
# Für Forward Selection ist der Hyperparameter 'nvmax' (maximale Anzahl an Variablen)
# Wir testen Modelle mit 1 bis 8 Prädiktoren.
tune_grid <- data.frame(nvmax = 1:8)

# Setzen eines Seeds für Reproduzierbarkeit der Ergebnisse der Kreuzvalidierung
set.seed(123)

# Durchführen der Forward Subset Selection mit Kreuzvalidierung
# lpsa ~ .: Formel, lpsa als Zielvariable, alle anderen als mögliche Prädiktoren
# data = prostateData_train_scaled: Trainingsdaten
# method = "leapForward": Spezifiziert Forward Selection als Methode
# tuneGrid = tune_grid: Gibt das Raster der zu testenden 'nvmax'-Werte an
# trControl = train_control: Wendet die definierten Kreuzvalidierungseinstellungen an
subset_model <- train(
  lpsa ~ .,
  data = prostateData_train_scaled,
  method = "leapForward",
  tuneGrid = tune_grid,
  trControl = train_control
)

# Ausgabe der Ergebnisse der Kreuzvalidierung
# Zeigt die Leistung (RMSE, R-squared etc.) für jede getestete Anzahl von Variablen (nvmax)
cat("Ergebnisse der Kreuzvalidierung für Forward Selection:\n")
```

```
## Ergebnisse der Kreuzvalidierung für Forward Selection:
```

```
print(subset_model) # Zeigt RMSE, Rsquared, MAE für jede nvmax über die CV-Folds
```

```
## Linear Regression with Forward Selection
##
## 67 samples
##  8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 59, 60, 59, 61, 59, 61, ...
## Resampling results across tuning parameters:
##
##   nvmax  RMSE       Rsquared   MAE
##   1      0.7909808  0.6234927  0.6729143
##   2      0.7446279  0.7177662  0.6275184
##   3      0.7644942  0.6943227  0.6470326
##   4      0.7697219  0.6921123  0.6370394
##   5      0.7879510  0.6706197  0.6324513
##   6      0.7751550  0.6867549  0.6114235
##   7      0.7334251  0.7002753  0.5672845
##   8      0.7423062  0.6871364  0.5741194
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was nvmax = 7.
```

```
# Beste Anzahl von Variablen basierend auf der Kreuzvalidierung (minimiert typischerweise RMSE)
best_nvmax <- subset_model$bestTune$nvmax
cat("\nBeste Anzahl von Variablen (nvmax) laut Kreuzvalidierung:", best_nvmax, "\n")
```

```
##
## Beste Anzahl von Variablen (nvmax) laut Kreuzvalidierung: 7
```

```
# Koeffizienten des finalen Modells, das mit der besten Anzahl von Variablen
# auf den *gesamten* Trainingsdaten trainiert wurde.
best_subset_coefficients <- coef(subset_model$finalModel, best_nvmax)
cat("\nKoeffizienten des besten Modells mit", best_nvmax, "Variablen:\n")
```

```
##
## Koeffizienten des besten Modells mit 7 Variablen:
```

```
print(best_subset_coefficients)
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##   2.4523451   0.7131604   0.2951154  -0.1461421   0.2113905   0.3115400
##         lcp       pgg45
##  -0.2877348   0.2621042
```

#Interpretation des Modellauswahlprozesses: trainControl(method = “cv”, number=10): Definiert eine 10-fache Kreuzvalidierung. Die Trainingsdaten werden 10-mal in 9 Teile zum Trainieren und 1 Teil zum Validieren aufgeteilt. Dies liefert eine robustere Schätzung der Modellleistung als ein einfacher Train-Test-Split innerhalb der Trainingsdaten. tune_grid <- data.frame(nvmax = 1:8): Gibt an, dass die Forward Selection für Modelle mit 1 bis 8 Variablen durchgeführt und bewertet werden soll. nvmax ist der Tuning-Parameter für die Methode “leapForward”. set.seed(123): Stellt sicher, dass die zufällige Aufteilung der Daten in der Kreuzvalidierung reproduzierbar ist. subset_model <- train(…): Dies ist der Hauptbefehl. method = “leapForward”: Weist caret an, die Forward Selection zu verwenden. Bei diesem Algorithmus startet man mit einem leeren Modell und fügt schrittweise den Prädiktor hinzu, der die Modellgüte am stärksten verbessert, bis die gewünschte Anzahl von Variablen (nvmax) erreicht ist. caret führt diesen Prozess für jeden Wert von nvmax im tune_grid (also für 1 bis 8 Variablen) durch. Für jede dieser nvmax-Einstellungen wird die 10-fache Kreuzvalidierung durchgeführt, um die durchschnittliche Leistung (z.B. RMSE - Root Mean Squared Error) zu schätzen. print(subset_model): Zeigt eine Zusammenfassung der Kreuzvalidierungsergebnisse. Typischerweise wird der RMSE (oder ein anderes spezifiziertes Metrik) für jedes getestete nvmax (Anzahl der Variablen) angezeigt. caret wählt dann das nvmax, das den besten Wert dieser Metrik (z.B. den niedrigsten RMSE) erzielt. best_nvmax <- subset_model$bestTune$nvmax: Extrahiert die Anzahl der Variablen (nvmax), die laut Kreuzvalidierung die beste Leistung erbracht hat. coef(subset_model$finalModel, best_nvmax): Nachdem die optimale Anzahl von Variablen (best_nvmax) durch Kreuzvalidierung bestimmt wurde, trainiert caret automatisch ein finales Modell mit dieser Anzahl von Variablen auf den gesamten Trainingsdaten (prostateData_train_scaled). Dieser Befehl zeigt die Koeffizienten dieses finalen Modells. Der entscheidende Vorteil hier ist, dass die Auswahl der “optimalen” Anzahl von Variablen nicht auf einem einzigen Split der Daten basiert, sondern auf der durchschnittlichen Leistung über mehrere Validierungs-Splits innerhalb der Kreuzvalidierung. Dies macht die Auswahl robuster gegen zufällige Schwankungen in den Daten. #4. Bewertung des ausgewählten Modells auf den Testdaten

```
# Erstellen der Design-Matrix für die Testdaten,
# aber NUR mit den Variablen, die im besten Modell (ausgewählt durch CV) enthalten sind.
# Die Namen der Koeffizienten (außer Intercept) sind die Namen der ausgewählten Variablen.
selected_vars_names <- names(best_subset_coefficients)
# Entferne "(Intercept)" falls vorhanden, um nur Prädiktoren zu bekommen
selected_vars_names_no_intercept <- selected_vars_names[selected_vars_names != "(Intercept)"]

# Erstelle die Formel für das Testset basierend auf den ausgewählten Variablen
if (length(selected_vars_names_no_intercept) > 0) {
  test_formula_string <- paste("lpsa ~", paste(selected_vars_names_no_intercept, collapse=" +
"))
} else { # Falls keine Prädiktoren ausgewählt wurden (nur Intercept)
  test_formula_string <- "lpsa ~ 1"
}
# model.matrix braucht eine Formel ohne Zielvariable auf der linken Seite für die Prädiktoren-Ma
trix
# Daher erstellen wir eine Formel nur für die rechte Seite (Prädiktoren)
predictors_formula_string <- sub("lpsa ~ ", "~ ", test_formula_string) # Entfernt "lpsa ~ "

# Erstellen der Modellmatrix für das Testset. Enthält eine Intercept-Spalte.
# Wichtig: Nur die im `best_subset_coefficients` enthaltenen Variablen verwenden.
# model.matrix erstellt Spalten basierend auf den Termen in der Formel.
# Die Spaltennamen müssen mit den Koeffizientennamen übereinstimmen.
X_test_full_matrix <- model.matrix(as.formula(predictors_formula_string), data = prostateData_te
st_scaled)

# Sicherstellen, dass X_test_full_matrix die richtigen Spalten in der richtigen Reihenfolge hat
# und nur die Spalten, die in best_subset_coefficients sind.
# `names(best_subset_coefficients)` gibt die Namen der Koeffizienten des finalen Modells.
# Wir müssen sicherstellen, dass wir die Spalten in X_test_full_matrix auswählen, die diesen Nam
en entsprechen.
# Wenn das Modell z.B. lcavol und lweight hat, sind die Koeffizientennamen "(Intercept)", "lcavo
l", "lweight".
# model.matrix( ~ lcavol + lweight, data) erzeugt Spalten mit diesen Namen.

# Einfachere Methode, wenn subset_model$finalModel ein lm-Objekt ist (was es für leapForward is
t):
# Das finale Modell wurde ja bereits mit den besten Variablen auf den gesamten Trainingsdaten tr
ainiert.
# Wir können direkt 'predict' darauf anwenden.
predictions_test_caret <- predict(subset_model, newdata = prostateData_test_scaled)
mse_best_subset_caret <- mean((predictions_test_caret - prostateData_test_scaled$lpsa)^2)

cat("\n--- Bewertung auf Testdaten (mit predict auf subset_model) ---\n")
```

```
##
## --- Bewertung auf Testdaten (mit predict auf subset_model) ---
```

```
cat("MSE auf Testdaten für das via CV-Forward-Selection ausgewählte Modell:", round(mse_best_sub
set_caret, 4), "\n")
```

```
## MSE auf Testdaten für das via CV-Forward-Selection ausgewählte Modell: 0.5459
```

```
# Der ursprüngliche Code-Teil zur manuellen Berechnung:
# Dieser Teil ist komplexer, da man die Modellmatrix exakt nachbauen muss.
# Die predict-Methode oben ist meist robuster und einfacher.
# Ich lasse ihn hier zur Referenz, aber die predict-Methode ist vorzuziehen.

# if (all(names(best_subset_coefficients) %in% colnames(X_test_full_matrix))) {
#   X_test_selected_cols <- X_test_full_matrix[, names(best_subset_coefficients), drop = FALSE]
#
#   # Sicherstellen, dass die Reihenfolge der Spalten in X_test_selected_cols
#   # mit der Reihenfolge der Koeffizienten übereinstimmt.
#   X_test_ordered <- X_test_selected_cols[, names(best_subset_coefficients)]
#
#   Xbeta_hat <- X_test_ordered %*% best_subset_coefficients # Vorhersagen berechnen
#   mse_best_subset_manual <- mean((Xbeta_hat - prostateData_test_scaled$lpsa)^2)
#   cat("\n--- Bewertung auf Testdaten (manuelle Berechnung, zur Kontrolle) ---\n")
#   cat("MSE auf Testdaten (manuell berechnet):", round(mse_best_subset_manual, 4), "\n")
# } else {
#   cat("\nFehler: Nicht alle Koeffizientennamen in der Test-Modellmatrix gefunden. Manuelle MSE
-Berechnung übersprungen.\n")
#   mse_best_subset_manual <- NA
# }


# Standardfehler der Schätzervarianz (Standard Error of the Mean Squared Error)
# Dies ist eine Schätzung der Unsicherheit des MSE-Wertes selbst.
# (prostateData_test_scaled$lpsa - predictions_test_caret) sind die Test-Residuen
squared_errors_test <- (prostateData_test_scaled$lpsa - predictions_test_caret)^2
sErr_best_subset <- sqrt(var(squared_errors_test) / nrow(prostateData_test_scaled))
# N ist Anzahl der Testbeobachtungen
cat("Standardfehler des Test-MSE (Schätzung der Unsicherheit des MSE):", round(sErr_best_subset,
4), "\n")
```

```
## Standardfehler des Test-MSE (Schätzung der Unsicherheit des MSE): 0.1704
```

#Korrektur und Vereinfachung im Code-Block oben: Die manuelle Erstellung der X_test Matrix und Berechnung von Xbeta_hat kann fehleranfällig sein, besonders wenn es um die genaue Übereinstimmung und Reihenfolge der Spaltennamen mit den Koeffizientennamen geht. Eine einfachere und robustere Methode ist die direkte Verwendung der predict-Funktion auf dem subset_model-Objekt (das von caret::train zurückgegeben wird). caret kümmert sich intern darum, das richtige finale Modell (subset_model$finalModel) mit den optimalen Prädiktoren zu verwenden. Ich habe den Code entsprechend angepasst, um predict(subset_model, …) zu verwenden, und den ursprünglichen manuellen Teil als Referenz (auskommentiert oder mit einer Bedingung versehen) belassen. Auch die Berechnung des Standardfehlers wurde auf nrow(prostateData_test_scaled) (die tatsächliche Anzahl der Testbeobachtungen, hier 30) korrigiert, anstatt fest 30 zu verwenden. Interpretation der Testdatenauswertung: Vorbereitung für Vorhersage: Das von der Kreuzvalidierung als optimal befundene Modell (definiert durch best_subset_coefficients und best_nvmax) wird nun auf die ungesehenen Testdaten (prostateData_test_scaled) angewendet. Die predict(subset_model, newdata = prostateData_test_scaled) Funktion macht dies auf elegante Weise. Berechnung des Test-MSE: mse_best_subset_caret ist der Mean Squared Error auf den Testdaten. Dies ist

die Schätzung des Generalisierungsfehlers für das Modell, das durch Forward Selection mit Kreuzvalidierung ausgewählt wurde. Standardfehler des MSE (sErr_best_subset): Dieser Wert gibt eine Schätzung der Unsicherheit des berechneten Test-MSE. Ein kleinerer Standardfehler bedeutet, dass unsere Schätzung des MSE stabiler ist. Er hängt von der Varianz der quadrierten Fehler und der Größe des Testdatensatzes ab. #5. Zusätzliche Informationen aus dem subset_model Objekt

```
cat("\n--- Detaillierte Ergebnisse aus subset_model$results ---\n")
```

```
##
## --- Detaillierte Ergebnisse aus subset_model$results ---
```

```
print(subset_model$results)
```

```
##   nvmax      RMSE  Rsquared       MAE    RMSESD RsquaredSD     MAESD
## 1     1 0.7909808 0.6234927 0.6729143 0.2894796  0.2410610 0.2658397
## 2     2 0.7446279 0.7177662 0.6275184 0.2426524  0.1568619 0.2345480
## 3     3 0.7644942 0.6943227 0.6470326 0.2227669  0.1594470 0.2255769
## 4     4 0.7697219 0.6921123 0.6370394 0.2388994  0.1556461 0.2379851
## 5     5 0.7879510 0.6706197 0.6324513 0.2221183  0.1605358 0.2117355
## 6     6 0.7751550 0.6867549 0.6114235 0.1990129  0.1523750 0.1904084
## 7     7 0.7334251 0.7002753 0.5672845 0.1954796  0.1673772 0.1638639
## 8     8 0.7423062 0.6871364 0.5741194 0.1955855  0.1675363 0.1642815
```

```
# Kommentar aus dem Originalskript:
# "The (mean) RMSE (Root Mean Squared Error) at p=7 is 0.73 +/- 0.2"
# Dieser Kommentar würde sich auf eine spezifische Zeile in subset_model$results beziehen,
# wo nvmax = 7 ist. Der RMSE-Wert dort (Spalte RMSE) ist der Durchschnitt über die CV-Folds.
# Die +/- Angabe ist oft die Standardabweichung des RMSE über die Folds.

# "The selected is the model with p=2 because ... "
# Wenn das beste Modell (laut CV-Minimum-RMSE) 2 Prädiktoren hat (also best_nvmax = 2),
# dann würde dieser Kommentar zutreffen. Der Grund "because..." wäre,
# dass dieses Modell den niedrigsten Kreuzvalidierungs-RMSE hatte.
cat("\nDas von caret ausgewählte Modell hat nvmax =", subset_model$bestTune$nvmax,
    "da es den geringsten Kreuzvalidierungs-RMSE von",
    round(min(subset_model$results$RMSE), 4), "aufwies.\n")
```

```
##
## Das von caret ausgewählte Modell hat nvmax = 7 da es den geringsten Kreuzvalidierungs-RMSE vo
n 0.7334 aufwies.
```

nterpretation von subset_model$results : $Die Tabelle subset_model$ results zeigt für jede getestete Anzahl von Variablen (nvmax) die durchschnittliche Leistung über die 10 Kreuzvalidierungs-Folds. Typische Spalten sind: nvmax: Anzahl der Variablen im Modell. RMSE: Root Mean Squared Error (Wurzel des mittleren quadratischen Fehlers). caret minimiert standardmäßig diesen Wert. Rsquared: R-Quadrat. MAE: Mean Absolute Error. RMSESD, RsquaredSD, MAESD: Die Standardabweichungen dieser Metriken über die CV-Folds, was ein Maß für die Variabilität der Schätzung ist. Die Zeile, die caret als “bestes” Modell auswählt (und in subset_model$bestTune speichert), ist diejenige mit dem niedrigsten RMSE (oder der besten Metrik, falls anders spezifiziert). Der

Kommentar “The (mean) RMSE (Root Mean Squared Error) at p=7 is 0.73 +/- 0.2” würde bedeuten, dass für das Modell mit 7 Prädiktoren der durchschnittliche RMSE über die CV-Folds 0.73 war, mit einer Standardabweichung von 0.2. Der Kommentar “The selected is the model with p=2 because…” würde zutreffen, wenn best_nvmax gleich 2 ist. Der Grund wäre, dass dieses Modell den niedrigsten Kreuzvalidierungs-RMSE erzielte.

#Schlussfolgerung: Dieses Skript demonstriert einen robusten Ansatz zur Modellauswahl. Durch die Kombination von Forward Subset Selection (einem schrittweisen Algorithmus zur Variablenauswahl) mit Kreuzvalidierung wird die optimale Anzahl von Prädiktoren auf eine Weise bestimmt, die weniger anfällig für die spezifische Zusammensetzung eines einzelnen Trainings-/Validierungssplits ist. Der finale Schritt der Bewertung auf einem komplett unabhängigen Testdatensatz liefert dann die entscheidende Schätzung des Generalisierungsfehlers für das so ausgewählte Modell. Dies ist ein gängiger und empfohlener Workflow im Machine Learning.