Lösungen zu den Aufgaben

1. **Aufgabe**

   Eine logistische Regression wurde an einen Datensatz angepasst. Es ergaben sich folgende Koeffizienten (jeweils Punktschätzer):

   ```
   Konstante = -1.9 x = 0.7 z = 0.7
   ```

   `x` ist ein metrischer Prädiktor mit einem Range von 0 bis 10; `z` ist eine binärer Prädiktor (mit den Werten 0 und 1).

   Visualisieren Sie die Kurven in einem Diagramm für

   a. $\mathscr{L}$ vs. x
   b. $Pr(y = 1)$ vs. x

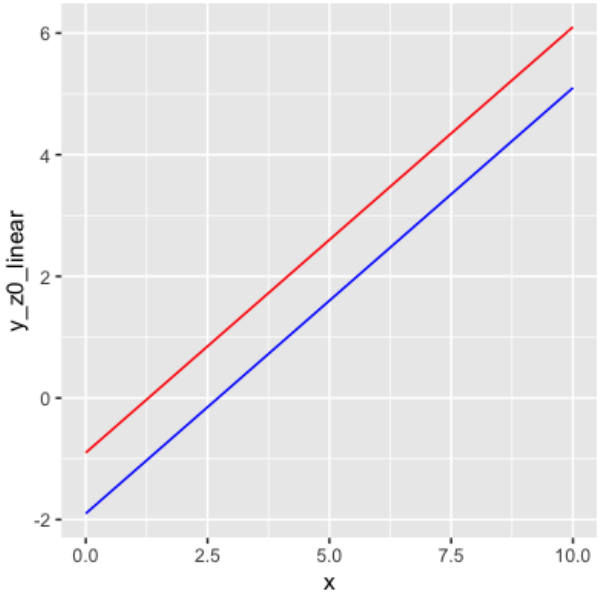
   Lösung

   Wir definieren die Variablen:

   ```
   d <-
     tibble(
       x = seq(from = 0, to = 10, by = 0.1),
       z = 1,
       y_z0_linear = -1.9 + 0.7 * x + 0*z,
       y_z1_linear = -1.9 + 0.7 * x + 1*z,
       p_y_z0 = plogis(y_z0_linear),
       p_y_z1 = plogis(y_z1_linear)
     )
   ```

   Hier ist das Diagramm mit *Logits* auf der Y-Achse:

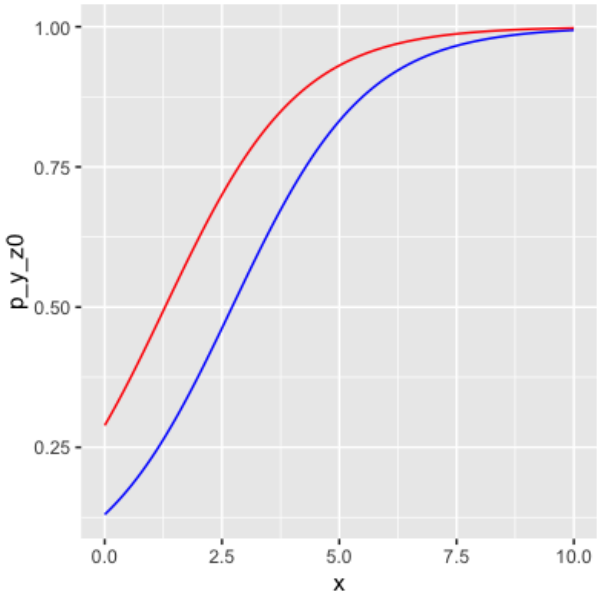
   ```
   d %>%
     ggplot() +
     aes(x = x) +
     geom_line(aes(y = y_z0_linear), color = "blue") +
     geom_line(aes(y = y_z1_linear), color = "red")
   ```

   plot of chunk unnamed-chunk-3

   Hier ist das Diagramm mit *Wahrscheinlichkeit* auf der Y-Achse:

   ```
   d %>%
     ggplot() +
     aes(x = x) +
     geom_line(aes(y = p_y_z0), color = "blue") +
     geom_line(aes(y = p_y_z1), color = "red")
   ```



plot of chunk unnamed-chunk-4

2. **Aufgabe**

Forschungsfrage: Ist der Zusammenhang von Körpergröße und 'Mann' positiv? Gehen also höhere Werte in Körpergröße `height` einher mit einer höheren Wahrscheinlichkeit, dass es sich um einen Mann `male` handelt?

Berechnen Sie ein Bayes-Modell mit `tidymodels` und geben Sie die Modellgüte an!

Hinweise:

- Rechnen Sie in Zentimeter um.
- Die AV sollte vom Typ `factor` sein bei einer Klassifikation, sonst beschwert sich Tidymodels.
- Stratifizieren Sie bei der Aufteilung von Train- und Test-Sample.
- Verwenden Sie eine 10-fache Kreuzvalidierung mit 10 Wiederholungen.
- Geben Sie die Modellgüte für folgende Koeffizienten an: ROC AUC, Sensitivität, Spezifität, PPV
- Verwenden Sie Rezept-Schritte (`steps`) nach eigenem Dafürhalten.
- Stoppen Sie die Zeit, die Ihr Computer braucht, um das Modell zu berechnen.
- Viel Spaß :-)

Lösung

Vorbereitung

```
library(tidymodels)
library(tidyverse)
library(tictoc)  # Zeitmessung der Rechenzeit

d <- read_csv(
"https://vincentarelbundock.github.io/Rdatasets/csv/openintro/speed_gender_height.csv")
```

```
## New names:
## Rows: 1325 Columns: 4
## ── Column specification
## ──────────────────── Delimiter: "," chr
## (1): gender dbl (3): ...1, speed, height
## ℹ Use `spec()` to retrieve the full column
## specification for this data. ℹ Specify the
## column types or set `show_col_types = FALSE`
## to quiet this message.
## • `` -> `...1`
```

Bereiten wir die Daten vor:

Datenaufteilung

Es ist praktisch, die AV vorab in einen Faktor umzuwandeln (s. weiter unten):

```
d <-
  d %>%
  mutate(gender = factor(gender))
```

Zur Erinnerung: `tidymodels` modelliert die *erste Stufe* der AV:

```
levels(d$gender)
```

```
## [1] "female" "male"
```

Also lieber re-leveln:

```
d <-
  d %>%
  mutate(gender = relevel(gender, ref = "male"))


levels(d$gender)  # check
```

```
## [1] "male"   "female"
```

Dann kommt die initial Datenaufteilung:

```
d_split <- initial_split(d, strata = "gender")
d_train <- training(d_split)
d_test <- testing(d_split)
```

Modelldefinition

```
logist_mod <-
  logistic_reg()

rsmpling <- vfold_cv(d_train, strata = "gender", repeats = 5)

recipe1 <- recipe(gender ~ height, data = d_train)  %>%
  step_mutate(height = height * 2.54) %>%
  step_impute_knn()

wf1 <-
  workflow() %>%
  add_model(logist_mod) %>%
  add_recipe(recipe1)
```

Modell fitten

```
tic()  # Stoppuhr an
fit1 <-
  fit_resamples(wf1,
                rsmpling,
                metrics = metric_set(roc_auc, sens, spec, ppv)
  )
toc()  # Stoppuhr aus
```

```
## 16.358 sec elapsed
```

```
fit1
```

```
## # Resampling results
## # 10-fold cross-validation repeated 5 times using stratification
## # A tibble: 50 × 5
##    splits            id      id2    .metrics
##    <list>            <chr>   <chr>  <list>
##  1 <split [892/101]> Repeat1 Fold01 <tibble>
##  2 <split [893/100]> Repeat1 Fold02 <tibble>
##  3 <split [894/99]>  Repeat1 Fold03 <tibble>
##  4 <split [894/99]>  Repeat1 Fold04 <tibble>
##  5 <split [894/99]>  Repeat1 Fold05 <tibble>
##  6 <split [894/99]>  Repeat1 Fold06 <tibble>
##  7 <split [894/99]>  Repeat1 Fold07 <tibble>
##  8 <split [894/99]>  Repeat1 Fold08 <tibble>
##  9 <split [894/99]>  Repeat1 Fold09 <tibble>
## 10 <split [894/99]>  Repeat1 Fold10 <tibble>
## # … with 40 more rows, and 1 more variable:
## #   .notes <list>
```

Ergebnisse im Train-Sample

```
collect_metrics(fit1)
```

```
## # A tibble: 4 × 6
##   .metric .estimator  mean     n std_err
##   <chr>   <chr>      <dbl> <int>   <dbl>
## 1 ppv     binary     0.808    50 0.00907
## 2 roc_auc binary     0.881    50 0.00443
## 3 sens    binary     0.650    50 0.0102
## 4 spec    binary     0.921    50 0.00432
## # … with 1 more variable: .config <chr>
```

Da wir *nicht* gesagt haben, dass die Vorhersagen gespeichert werden sollen, können wir Sie uns auch nicht anschauen:

```
collect_predictions(fit1)
```

```
## Error in `collect_predictions()`:
## ! The `.predictions` column does not exist. Refit with the control argument `save_pred = TRUE` to save predictions.
```

Das hätten wir so machen können, also mit `control()`, dann können wir die Vorhersagen speichern:

```
tic()  # Stoppuhr an
fit2 <-
  fit_resamples(wf1,
                rsmpling,
                metrics = metric_set(roc_auc, sens, spec, ppv),
                control = control_resamples(save_pred = TRUE)  # NEUE ZEILE

  )
toc()  # Stoppuhr aus
```

```
## 16.726 sec elapsed
```

Dauert schon ein bisschen…

Dabei ist die logistische Regression sehr *wenig* rechenintensiv.

```
fit2
```

```
## # Resampling results
## # 10-fold cross-validation repeated 5 times using stratification
## # A tibble: 50 × 6
##    splits            id      id2    .metrics
##    <list>            <chr>   <chr>  <list>
##  1 <split [892/101]> Repeat1 Fold01 <tibble>
##  2 <split [893/100]> Repeat1 Fold02 <tibble>
##  3 <split [894/99]>  Repeat1 Fold03 <tibble>
##  4 <split [894/99]>  Repeat1 Fold04 <tibble>
##  5 <split [894/99]>  Repeat1 Fold05 <tibble>
##  6 <split [894/99]>  Repeat1 Fold06 <tibble>
##  7 <split [894/99]>  Repeat1 Fold07 <tibble>
##  8 <split [894/99]>  Repeat1 Fold08 <tibble>
##  9 <split [894/99]>  Repeat1 Fold09 <tibble>
## 10 <split [894/99]>  Repeat1 Fold10 <tibble>
## # … with 40 more rows, and 2 more variables:
## #   .notes <list>, .predictions <list>
```

```
collect_metrics(fit2)
```

```
## # A tibble: 4 × 6
##   .metric .estimator  mean     n std_err
##   <chr>   <chr>      <dbl> <int>   <dbl>
## 1 ppv     binary     0.808    50 0.00907
## 2 roc_auc binary     0.881    50 0.00443
## 3 sens    binary     0.650    50 0.0102
## 4 spec    binary     0.921    50 0.00432
## # … with 1 more variable: .config <chr>
```

Vorhersagen im Train-Sample

Schauen wir uns die Vorhersagen im Train-Sample, zusammengefasst über die alle Faltungen und Wiederholungen (`? collect_predictions`).

```
collect_predictions(fit2, summarize = TRUE)
```

```
## # A tibble: 993 × 6
##     .row gender .config            .pred_male
##    <int> <fct>  <chr>                   <dbl>
##  1     1 female Preprocessor1_Mod…     0.589
##  2     2 female Preprocessor1_Mod…     0.104
##  3     3 female Preprocessor1_Mod…     0.0159
##  4     4 female Preprocessor1_Mod…     0.0253
##  5     5 female Preprocessor1_Mod…     0.166
##  6     6 female Preprocessor1_Mod…     0.161
##  7     7 female Preprocessor1_Mod…     0.0702
##  8     8 female Preprocessor1_Mod…     0.0683
```

```
##  9     9 female Preprocessor1_Mod…     0.161
## 10    10 female Preprocessor1_Mod…     0.108
## # … with 983 more rows, and 2 more
## #   variables: .pred_female <dbl>,
## #   .pred_class <fct>
```

ROC-Kurve im Train-Sample

```
roc_data <-
  fit2 %>%
  collect_predictions(summarize = TRUE) %>%
  select(.pred_male) %>%
  bind_cols(d_train) %>%
  roc_curve(truth = gender, estimate = .pred_male)
```

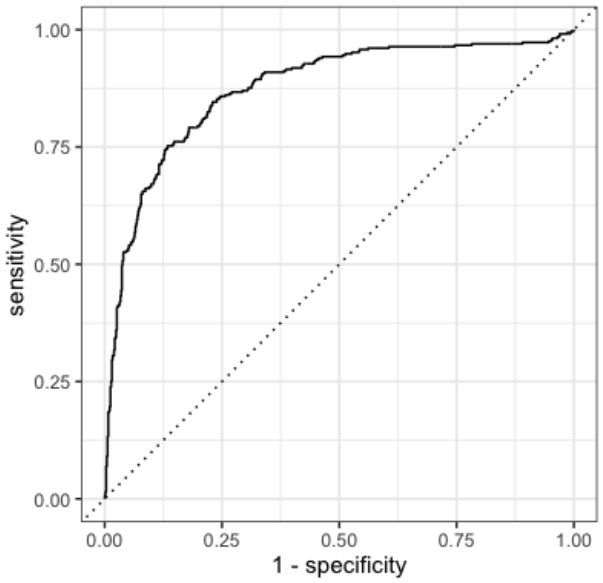
```
## New names:
## • `...1` -> `...2`
```

```
roc_data
```

```
## # A tibble: 989 × 3
##      .threshold specificity sensitivity
##           <dbl>       <dbl>       <dbl>
##  1 -Inf            0             1
##  2    0.000262     0             1
##  3    0.000289     0             0.997
##  4    0.000304     0.00152       0.997
##  5    0.00130      0.00304       0.997
##  6    0.00200      0.00457       0.997
##  7    0.00217      0.00457       0.994
##  8    0.00382      0.00609       0.994
##  9    0.00403      0.00761       0.994
## 10    0.00405      0.00913       0.994
## # … with 979 more rows
```

Plotten:

```
roc_data %>%
  ggplot(aes(x = 1- specificity, sensitivity)) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal() +
  theme_bw()
```



plot of chunk unnamed-chunk-16

Gar nicht schlecht!

Vorhersagegüte im Test-Sample

```
lm1_lastfit <-
  last_fit(wf1, d_split,
           metrics = metric_set(roc_auc, sens, spec, ppv)
  )
```

```
lm1_lastfit
```

```
## # Resampling results
## # Manual resampling
## # A tibble: 1 × 6
##   splits            id      .metrics .notes
##   <list>            <chr>   <list>   <list>
## 1 <split [993/332]> train/… <tibble> <tibble>
## # … with 2 more variables:
## #   .predictions <list>, .workflow <list>
```

```
collect_metrics(lm1_lastfit)
```

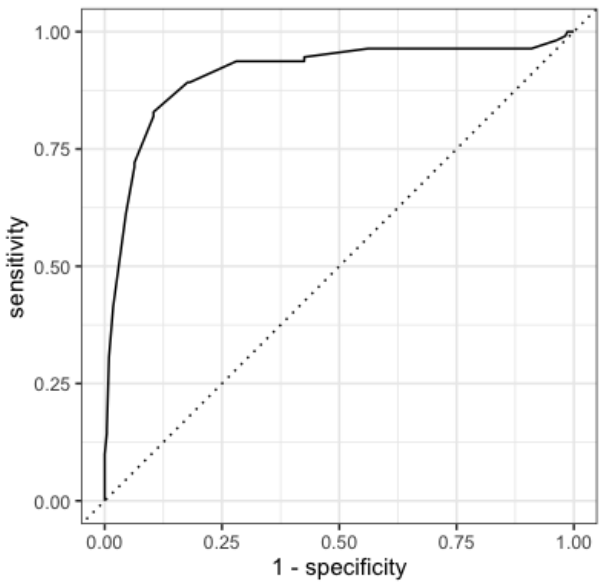
```
## # A tibble: 4 × 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 sens    binary         0.721 Preprocessor1…
## 2 spec    binary         0.937 Preprocessor1…
## 3 ppv     binary         0.851 Preprocessor1…
## 4 roc_auc binary         0.909 Preprocessor1…
```

```
collect_predictions(lm1_lastfit)
```

```
## # A tibble: 332 × 7
##    id            .pred_male .pred_female  .row
##    <chr>              <dbl>        <dbl> <int>
##  1 train/test …     0.0261        0.974     9
##  2 train/test …     0.0673        0.933    37
##  3 train/test …     0.344         0.656    45
##  4 train/test …     0.0261        0.974    47
##  5 train/test …     0.698         0.302    49
##  6 train/test …     0.106         0.894    51
##  7 train/test …     0.242         0.758    52
##  8 train/test …     0.698         0.302    55
##  9 train/test …     0.0421        0.958    56
## 10 train/test …     0.106         0.894    58
## # … with 322 more rows, and 3 more
## #   variables: .pred_class <fct>,
## #   gender <fct>, .config <chr>
```

ROC-Kurve

```
lm1_lastfit %>%
  collect_predictions() %>%
  roc_curve(truth = gender, estimate = .pred_male) %>%
  ggplot(aes(x = 1 - specificity, sensitivity )) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal() +
  theme_bw()
```



plot of chunk unnamed-chunk-20

Shinymodels

Mit (dem R-Paket) `{{shinymodels}}` kann man sich eine App ausgeben lassen, zur Erkundung der Modellergebnisse:

```
library(shinymodels)
```

```
explore(fit2)
```