Modele regresji i ich zastosowania - raport 4.

Patryk Krukowski(249824)

19 maja 2021

Spis treści

1 Zadanie 1.

1.1 Wstęp

W pierwszym zadaniu do pakietu statystycznego *R* wczytujemy dane z pliku *logistyczna.txt* dotyczące informacji o stanie pacjenta i parametrach charakteryzujących pracę jego serca. Zbiór danych zawiera dziesięć zmiennych o nazwach:

- STAN (objaśniania zmienna binarna)
- RMS10
- RMS20
- RMS30
- PWD
- A
- DD
- YA

- YDD
- AR

oraz 267 obserwacji. Dane są poprawnie zakodowane, nie występują obserwacje zakodowane jako *NA* oraz dane są kompletne.

```
dane <- read.table('logistyczna.txt', skip=1, header=T)
dane <- as.data.frame(dane)
```

Przejdźmy teraz do wykonania zadań laboratoryjnych dotyczących pierwszego zadania.

1.2 (a)

Dzielimy losowo zbiór danych w proporcji 70:30 na zbiór treningowy i zbiór testowy w taki sposób, by frakcja pacjentów ze zmienną $STAN = 1$ niewiele różniła się od frakcji takich pacjentów w zbiorze testowym. Warunek ten gwarantuje nam możliwość rozsądnego sprawdzenia zdolności predykcyjnej modelu regresji logistycznej na zbiorze testowym.

Losujemy indeksy, korzystając z funkcji *sample*, tworzymy podział na zbiór testowy i treningowy i liczymy odpowiednie frakcje.

```
set.seed(12)
indeks <- sample(seq_len(nrow(dane)), size=0.7*nrow(dane)) #w rozmiarze jest wzie-
ta podloga
treningowy <- dane[indeks,]
testowy <- dane[-indeks,]
freq_train <- count(treningowy, vars='STAN')
freq_test <- count(testowy, vars='STAN')

freq_test

##   STAN freq
## 1    0    9
## 2    1   72

freq_train

##   STAN freq
## 1    0   26
## 2    1  160
```

Jak możemy zauważyć, podział danych jest dla nas zadowalający, ponieważ frakcje pacjentów ze zmienną $STAN = 1$ wynoszą:

- $\frac{72}{81} \approx 0.89$ w zbiorze testowym
- $\frac{160}{186} \approx 0.86$ w zbiorze treningowym.

Uwaga 1 *Parametr size w funkcji sample bierze podłogę z wartości tego parametru.*

1.3 (b)

Konstruujemy model regresji logistycznej na zbiorze treningowym, gdzie zmienna *STAN* jest binarną zmienną objaśnianą, a **wszystkie** pozostałe zmienne są zmiennymi objaśniającymi. Użyjemy do tego funkcji *glm*.

```
model <- glm(STAN~RMS10+RMS20+RMS30+PWD+A+DD+YA+YDD+AR, data=treningowy, method='glm.fit
             family = 'binomial')
```

Ważną rzeczą jest, by w funkcji *glm* ustawić wartość parametru *family* jako *'binomial'*. Gwarantuje to nam dopasowanie modelu regresji logistycznej.

1.4 (c)

Za pomocą testu ilorazu wiarogodności na poziomie istotności $\alpha = 0.05$ weryfikujemy hipotezę zerową $H_0 : \beta_2 = \ldots = \beta_{10} = 0$ przeciwko hipotezie H_1 mówiącej o tym, że hipoteza zerowa jest fałszywa. Budujemy model regresji logistycznej *model_0*, w którym predyktorem jest jedynie stała i używamy testu ilorazu wiarogodności, korzystając z funkcji *lrtest* z paczki *lmtest*.

```
model_0 <- glm(STAN~1, data=treningowy, method='glm.fit',
               family = 'binomial')
#odrzucamy H_0
lrtest(model, model_0)

## Likelihood ratio test
##
## Model 1: STAN ~ RMS10 + RMS20 + RMS30 + PWD + A + DD + YA + YDD + AR
## Model 2: STAN ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1  10 -61.978
## 2   1 -75.251 -9 26.545   0.001663 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Z powyższego kodu odczytujemy, że powinniśmy odrzucić hipotezę H_0 na poziomie istotności $\alpha = 0.05$, ponieważ p-wartość użytego testu wynosi 0.001663.

Wniosek 1 *Model regresji logistycznej ma sens, tzn. istnieją zmienne mające wpływ na zmienną objaśnianą.*

1.5 (d)

Dla każdego z parametrów β_i, $i = 2, \ldots, 10$,

- wyznaczamy wartości estymatorów $\hat{\beta}_i$ oraz SE_{β_i}.

```
#bez wyrazu wolnego
beta_hat <- model$coefficients[2:10]
se_hat <- summary(model)$coefficients[,2][2:10]
data.frame(beta_hat, se_hat)
```

```
##            beta_hat      se_hat
## RMS10 -27.966741714 20.98252945
## RMS20  21.270943306 27.93266367
## RMS30   5.908120443 18.51330127
## PWD     6.128564650  1.86124685
## A       0.008964789  0.01230421
## DD     -4.528598262  3.24772249
## YA      0.009086908  0.02010515
## YDD     3.239065912  4.24101210
## AR      0.150204493  1.49528319
```

W szczególności zauważmy, że estymatory SE_{β_i} dla zmiennych *RMS10* oraz *RMS20*, *RMS30* są bardzo duże. Może to być spowodowane niezbyt liczną liczbą obserwacji zbioru treningowego.

- konstruujemy przedziały ufności Walda na poziomie ufności $1-\alpha$.

```
CI <- function(beta, se, alpha) {
  L <- rep(0, length(beta))
  R <- rep(0, length(beta))
  for (k in 1:length(beta)) {
    L[k] <- beta[k]+qnorm(alpha/2)*se[k]
    R[k] <- beta[k]-qnorm(alpha/2)*se[k]
  }
  return(data.frame(L,R, beta))
}
```

Możemy teraz wyznaczyć te przedziały np. dla $\alpha = 0.05$.

```
CI(beta_hat, se_hat, 0.05)
```

```
##                   L           R           beta
## RMS10 -69.09174375 13.15826032 -27.966741714
## RMS20 -33.47607149 76.01795810  21.270943306
## RMS30 -30.37728328 42.19352417   5.908120443
## PWD     2.48058787  9.77654143   6.128564650
## A      -0.01515102  0.03308060   0.008964789
## DD    -10.89401737  1.83682085  -4.528598262
## YA     -0.03031846  0.04849228   0.009086908
## YDD    -5.07316506 11.55129689   3.239065912
## AR     -2.78049671  3.08090570   0.150204493
```

- weryfikujemy $H_0 : \beta_i = 0$ vs $H_1 : \beta_i \neq 0$ na poziomie istotności α. W tym celu możemy albo skorzystać z funkcji *summary* i odczytać odpowiednie p-wartości uzyskane dla statystyki Walda, albo sprawdzić, dla których zmiennych 0 należy do przedziału ufności z poprzedniej kropki. Niech $\alpha = 0.05$.

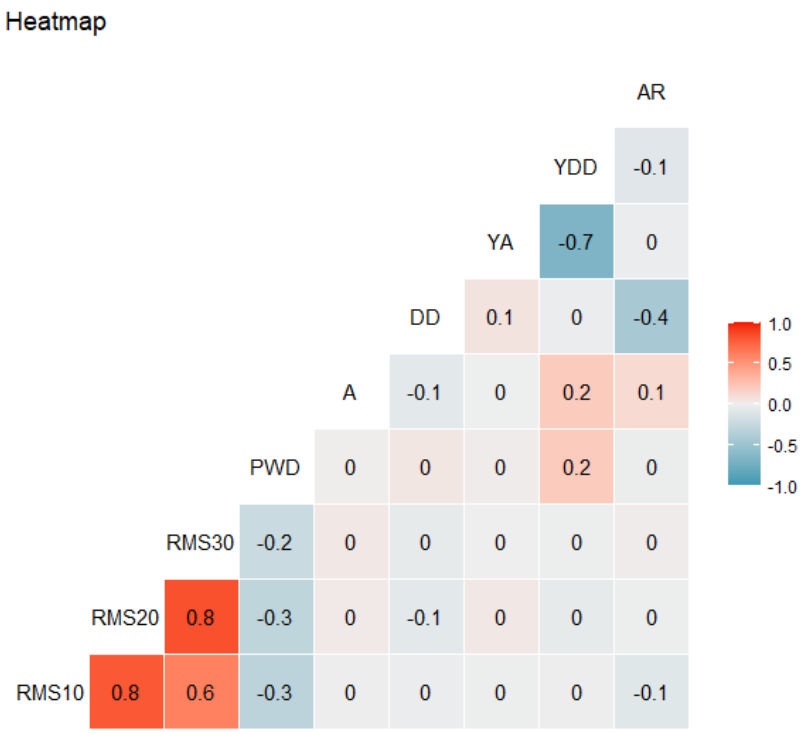
```
summary(model)

##
## Call:
## glm(formula = STAN ~ RMS10 + RMS20 + RMS30 + PWD + A + DD + YA +
##     YDD + AR, family = "binomial", data = treningowy, method = "glm.fit")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5236   0.2403   0.3513   0.5375   1.5048
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.180353   3.100740  -2.316 0.020575 *
## RMS10       -27.966742  20.982529  -1.333 0.182578
## RMS20        21.270943  27.932664   0.762 0.446354
## RMS30         5.908120  18.513301   0.319 0.749629
## PWD           6.128565   1.861247   3.293 0.000992 ***
## A             0.008965   0.012304   0.729 0.466249
## DD           -4.528598   3.247722  -1.394 0.163199
## YA            0.009087   0.020105   0.452 0.651291
## YDD           3.239066   4.241012   0.764 0.445017
## AR            0.150204   1.495283   0.100 0.919985
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 150.50  on 185  degrees of freedom
## Residual deviance: 123.96  on 176  degrees of freedom
## AIC: 143.96
##
## Number of Fisher Scoring iterations: 5
```

Korzystając z jednej lub drugiej metody, odczytujemy, że jedynie zmienna *PWD* jest istotna statystycznie, tzn. tylko dla β_5 odrzucamy hipotezę zerową na poziomie istotności $\alpha = 0.05$.

Wniosek 2 *Do dalszej analizy tworzymy model regresji logistycznej, w którym predyktorem jest jedynie zmienna PWD.*

Nim jednak przystąpimy do zbudowania nowego modelu, zastanówmy się, czy we wcześniejszym modelu (oznaczonym jako *model*) nie mamy problemu ze **współliniowością**. Gdybyśmy mieli z nią do czynienia, to mogłoby się okazać, że istnieje grupa silnie skorelowanych ze sobą zmiennych objaśniających i usunięcie np. jednej z nich wystarczyłoby, aby pozostałe zmienne były statystycznie istotne. Wykonajmy rysunek typu heatmap z wyznaczonymi korelacjami próbkowymi.



Rysunek 1: Heatmap korelacji próbkowych dla zmiennych *RMS10*,..., *AR*

Rysunek nr 1 sugeruje, ze pary zmiennych (*RMS10*, *RMS20*) oraz (*RMS20*, *RMS30*) wydają się być mocno skorelowane, lecz raczej nie świadczy to o współliniowości. Na wszelki wypadek wyznaczmy jeszcze wskaźniki podbicia wariancji dla badanego modelu.

```
vif(model)

##    RMS10    RMS20    RMS30      PWD        A       DD       YA      YDD
## 3.419502 7.597854 4.131140 1.136345 1.195847 1.151139 2.063128 2.226661
##       AR
## 1.174443
```

Żaden ze wskaźników podbicia wariancji nie przekracza 10, zatem zakładamy, że nie mamy problemu ze współliniowością w modelu *model*. W związku z tym za model do dalszej analizy wybieramy model opisany we wcześniejszym wniosku.

```
model_1 <- glm(STAN~PWD, data=treningowy, method='glm.fit',
               family = 'binomial')
summary(model_1)

##
## Call:
## glm(formula = STAN ~ PWD, family = "binomial", data = treningowy,
##     method = "glm.fit")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.4230   0.2546   0.4137   0.5489   1.4304
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -6.164      1.995  -3.090    0.002 **
## PWD            6.651      1.708   3.893 9.91e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 150.50  on 185  degrees of freedom
## Residual deviance: 131.97  on 184  degrees of freedom
## AIC: 135.97
##
## Number of Fisher Scoring iterations: 5
```

1.6 (e)

Teraz dla progów odcięcia π_0 równych 0.5 i 0.7 wykorzystamy zbiór testowy do skonstruowania *classification table* oraz wyznaczymy parametry *sensivity*, *specifity*, *false positive rate*, *false negative rate*, *overall proportion of correct classifications*. Niech

- $\pi_0 = 0.5$. Wykonajmy odpowiednie prognozy (na zbiorze testowym) i zbudujmy *classification table*. W tym przypadku mamy

$$\hat{Y}(\boldsymbol{x}) = \begin{cases} 1, & \hat{\pi}(\boldsymbol{x}) > 0.5, \\ 0, & \hat{\pi}(\boldsymbol{x}) \leq 0.5, \end{cases}$$

  gdzie oznaczenia są zgodne z wykładem.

```
#wyznaczamy estymator wektora pi(x)
prob_0.5 <- predict(model_1, testowy, type='response')

#wyznaczamy prognozowane klasy
pred_0.5 <- rep(0, dim(testowy)[1])
pred_0.5[prob_0.5 > 0.5] <- 1

#classification table
tab_0.5 <- table(pred_0.5, testowy$STAN)
tab_0.5


##
## pred_0.5  0  1
##        0  0  1
##        1  9 71
```

```
TP_0.5 <- tab_0.5[2,2] #True Positives
TN_0.5 <- tab_0.5[1,1] #True Negatives
FP_0.5 <- tab_0.5[2,1] #False Positives
FN_0.5 <- tab_0.5[1,2] #False Negatives
```

Wyznaczamy estymatory parametrów charakteryzujących zdolność predykcyjną modelu.

- sensivity

```
sens_0.5 <- TP_0.5/(TP_0.5+FN_0.5)
sens_0.5

## [1] 0.9861111
```

- specifity

```
spec_0.5 <- TN_0.5/(TN_0.5+FP_0.5)
spec_0.5

## [1] 0
```

- false positive rate

```
FPR_0.5 <- FP_0.5/(FP_0.5+TN_0.5)
FPR_0.5

## [1] 1
```

- false negative rate

```
FNR_0.5 <- FN_0.5/(FN_0.5+TP_0.5)
FNR_0.5

## [1] 0.01388889
```

- overall proportion of correct classifications

```
prop_correct_0.5 <- mean(pred_0.5 == testowy$STAN)
prop_correct_0.5

## [1] 0.8765432
```

Powyższe parametry mówią nam o tym, że utworzony model regresji logistycznej z progiem odcięcia π_0 równym 0.5 posiada dobrą zdolność klasyfikacji w przypadku, gdy $STAN = 1$ (parametr sensivity i false negative rate).

Natomiast model nie jest w stanie poprawnie sklasyfikować przypadków, gdy $STAN = 0$ (parametr specifity i false positive rate). To znak, że próg odcięcia równy 0.5 nie jest najlepszym wyborem.

Oszacowanie prawdopodobieństwa poprawnej klasyfikacji, z dokładnością do powyższych uwag, wynosi w przybliżeniu 0.877.

Wniosek 3 *W ocenie dokładności modelu nie możemy sugerować się jedną miarą dokładności, jak np. oszacowaniem prawdopodobieństwa poprawnej klasyfikacji, gdyż może to nas doprowadzić do złych wniosków.*

- $\pi_0 = 0.7$. Wykonajmy odpowiednie prognozy (na zbiorze testowym) i zbudujmy *classification table*. W tym przypadku mamy

$$\hat{Y}(\boldsymbol{x}) = \begin{cases} 1, & \hat{\pi}(\boldsymbol{x}) > 0.7, \\ 0, & \hat{\pi}(\boldsymbol{x}) \leq 0.7, \end{cases}$$

gdzie oznaczenia są zgodne z wykładem.

```
#wyznaczamy estymator wektora pi(x)
prob_0.7 <- predict(model_1, testowy, type='response')

#wyznaczamy prognozowane klasy
pred_0.7 <- rep(0, dim(testowy)[1])
pred_0.7[prob_0.7 > 0.7] <- 1

#classification table
tab_0.7 <- table(pred_0.7, testowy$STAN)
tab_0.7

##
## pred_0.7  0  1
##        0  2  5
##        1  7 67

TP_0.7 <- tab_0.7[2,2] #True Positives
TN_0.7 <- tab_0.7[1,1] #True Negatives
FP_0.7 <- tab_0.7[2,1] #False Positives
FN_0.7 <- tab_0.7[1,2] #False Negatives
```

Wyznaczamy estymatory parametrów charakteryzujących zdolność predykcyjną modelu.

  - sensivity

```
sens_0.7 <- TP_0.7/(TP_0.7+FN_0.7)
sens_0.7

## [1] 0.9305556
```

  - specifity

```
spec_0.7 <- TN_0.7/(TN_0.7+FP_0.7)
spec_0.7

## [1] 0.2222222
```

– false positive rate

```
FPR_0.7 <- FP_0.7/(FP_0.7+TN_0.7)
FPR_0.7

## [1] 0.7777778
```

– false negative rate

```
FNR_0.7 <- FN_0.7/(FN_0.7+TP_0.7)
FNR_0.7

## [1] 0.06944444
```

– overall proportion of correct classifications

```
prop_correct_0.7 <- mean(pred_0.7 == testowy$STAN)
prop_correct_0.7

## [1] 0.8518519
```

Kosztem małego spadku overall proportion of correct classifications, model zyskał lepszą zdolność predykcyjną w przypadku, gdy zmienna *STAN* przyjmuje wartość równą 0, co możemy zauważyć, patrząc na wartości estymatorów parametrów specifity oraz false positive rate. Jednak nieco spadła zdolność predykcyjna, gdy *STAN* jest równe 1 (sensivity, false negative rate).

Wniosek 4 *Model z progiem odcięcia $\pi_0 = 0.7$ wydaje się być lepszym wyborem z uwagi na lepszą zdolność klasyfikacyjną "zer" przy niewielkim spadku zdolności klasyfikacyjnych "jedynek".*

1.7 (f)

Używamy zbioru testowego do narysowania krzywej ROC. W tym celu posłużymy się funkcją *rocit* z pakietu *ROCit*.

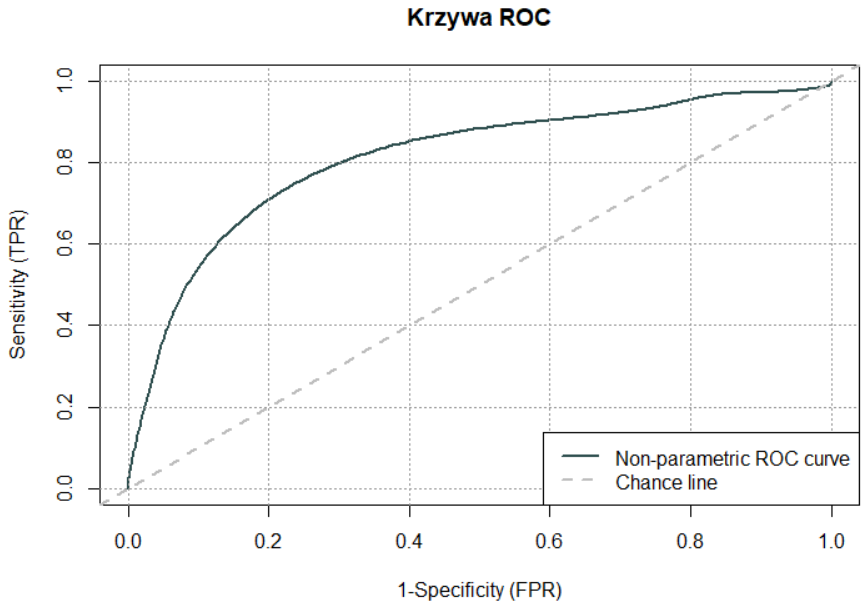
```
ROC <- rocit(predict(model_1, testowy,type='response'), class=testowy$STAN,
             method='non')
plot(ROC, YIndex=FALSE)
title(main='Krzywa ROC')
```

Z rysunku nr 2 wynika, że model ma dobre zdolności predykcyjne (krzywa ROC przebiega dość wysoko). Tę obserwację potwierdza AUC, czyli pole pod wykresem narysowanej krzywej.

```
ROC$AUC

## [1] 0.8128709
```

Uwaga 2 *Oczywiście, narysowana krzywa na rysunku nr 2 jest **estymatorem** krzywej ROC uzyskanym metodą nieparametryczną.*



Rysunek 2: Krzywa ROC narysowana na podstawie zbioru testowego dla danych z pliku *logistyczna.txt*

2 Zadanie 2.

2.1 Wstęp

Teraz badanym zbiorem danych jest zbiór danych z pliku *cancer remission.xlsx*. Binarna zmienna objaśniana *remission* przyjmuje wartość 1, gdy choroba jest w stanie remisji i wartość 0, gdy nie jest w tym stanie. Zmienne objaśniające *cell*, *smear.*, *infil*, *li*, *blast*, *temp* to czynniki ryzyka, których wartości mogą wpływać na to, czy choroba jest w stanie remisji, czy też nie.

W zbiorze tym obserwacje nie są dobrze zakodowane, ponieważ liczby z obserwacji z niezerową częścią ułamkową są oddzielone od części całkowitej poprzez przecinek, a nie kropkę. Problem ten naprawiamy, konwertując obserwacje do typu double poniższym kodem.

```
dane_2 <- read.xlsx2('C:/Users/Lenovo/Desktop/Modele regresji i ich zastosowania/cancer
mission.xlsx',
                     sheetName = 1)
for (k in 1:ncol(dane_2)) {
    dane_2[,k] <- as.numeric(dane_2[,k])
}
```

W zbiorze danych nie występują obserwacje zakodowane jako *NA* oraz mamy 27 obserwacji i 7 zmiennych:

- remission (binarna zmienna objaśniana)
- cell
- smear.
- infil

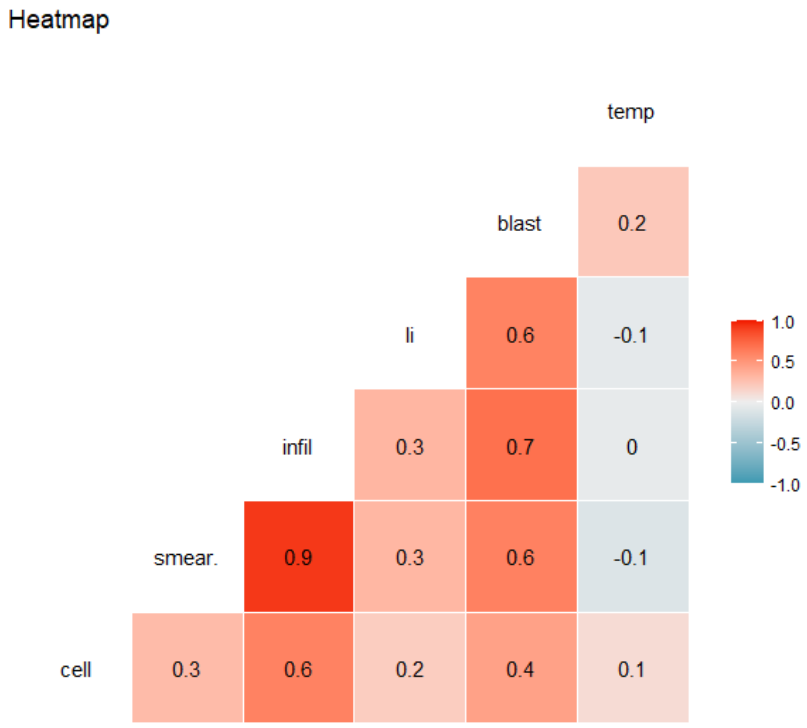
- li
- blast
- temp

Przystąpmy teraz do wykonania zadań laboratoryjnych.

2.2 (a) i (b)

Nim przystąpimy do rozwiązania zadania (podpunkty (a) i (b) jednocześnie), zastanówmy się, czy nie mamy problemu ze współliniowością. W tym celu stwórzmy wykres typu heatmap, ilustrujący korelacje próbkowe między zmiennymi.

```
ggcorr(dane_2[-1], method = c("everything", "pearson"), label =T) +
  ggtitle('Heatmap')
```



Rysunek 3: Heatmap ilustrujący korelacje próbkowe między zmiennymi objaśniającymi pochodzącymi ze zbioru danych *cancer remission.xlsx*

Para zmiennych $(smear., infil)$ jest dość silnie skorelowana ze sobą. Użyjmy współczynników podbicia wariancji do sprawdzenia, czy faktycznie mamy problem ze współliniowością.

```
vif(glm(remission~cell+temp+smear.+li+blast+infil, data=dane_2, method='glm.fit',
        family = 'binomial'))

##       cell       temp     smear.         li      blast      infil
##  50.789567   2.362035 392.521000   3.424815   4.227980 433.705073
```

Zdecydowanie mamy problem ze współliniowością. Usuńmy zatem zmienną *infil* z modelu, jako że dla tej zmiennej współczynnik podbicia wariancji jest największy i większy od 10, a następnie znów policzymy *VIF*.

```
vif(glm(remission~cell+temp+smear.+li+blast, data=dane_2, method='glm.fit',
        family = 'binomial'))

##     cell     temp   smear.       li    blast
## 2.097570 2.277070 1.495639 3.298644 3.817588
```

Wyliczone współczynniki podbicia wariancji są mniejsze od 10, zatem zakładamy, że dla takiego zestawu zmiennych nie mamy już problemu ze współliniowością. Jak wykorzystamy tę wiedzę w praktyce? Wykorzystamy ją w taki sposób, że w regresji krokowej nie uwzględnimy zmiennej *infil*, gdyż może to nas doprowadzić do uzyskania mniej optymalnego modelu lub nawet zaburzyć wybór modelu.

Przejdźmy teraz do skonstruowania modelu regresji logistycznej, używając **regresji krokowej**. Przyjmijmy poziom istotności równy 0.3 dla zmiennej, która ma być wprowadzona do modelu i poziom istotności równy 0.35 dla zmiennej, która ma pozostać w modelu. Na każdym kroku procedury przy pomocy testu **Walda** weryfikujemy hipotezę H_0: prawdziwy jest model M_0 vs H_1: prawdziwy jest model M_1, gdzie M_0 to prostszy model, a M_1 to model zawierający wszystkie zmienne, które są w M_0 wraz z badaną zmienną na danym kroku procedury.

Wyniki z (b, ii.) i (b, iii.) umieścimy pod koniec tejże sekcji w formie tabeli dla lepszej wizualizacji.

1. krok procedury - zaczynamy od modelu, w którym występuje jedynie stała i dołączamy kolejną zmienną zgodnie z procedurą selekcji postępującej.

```
model_2 <- glm(remission~1, data=dane_2, method='glm.fit',
                    family = 'binomial')
model_2_1 <- glm(remission~cell, data=dane_2, method='glm.fit',
                  family = 'binomial')
model_2_2 <- glm(remission~smear., data=dane_2, method='glm.fit',
                  family = 'binomial')
model_2_3 <- glm(remission~li, data=dane_2, method='glm.fit',
                  family = 'binomial')
model_2_4 <- glm(remission~blast, data=dane_2, method='glm.fit',
                  family = 'binomial')
model_2_5 <- glm(remission~temp, data=dane_2, method='glm.fit',
                  family = 'binomial')
waldtest(model_2, model_2_1)

## Wald test
##
## Model 1: remission ~ 1
## Model 2: remission ~ cell
##   Res.Df Df      F Pr(>F)
## 1     26
## 2     25  1 1.5515 0.2245
```

```r
waldtest(model_2, model_2_2)

## Wald test
##
## Model 1: remission ~ 1
## Model 2: remission ~ smear.
##   Res.Df Df      F Pr(>F)
## 1     26
## 2     25  1 1.0455 0.3163

waldtest(model_2, model_2_3) #bierzemy te zmienną do modelu

## Wald test
##
## Model 1: remission ~ 1
## Model 2: remission ~ li
##   Res.Df Df      F  Pr(>F)
## 1     26
## 2     25  1 5.9595 0.02206 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

waldtest(model_2, model_2_4)

## Wald test
##
## Model 1: remission ~ 1
## Model 2: remission ~ blast
##   Res.Df Df      F  Pr(>F)
## 1     26
## 2     25  1 3.0894 0.09105 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

waldtest(model_2, model_2_5)

## Wald test
##
## Model 1: remission ~ 1
## Model 2: remission ~ temp
##   Res.Df Df      F Pr(>F)
## 1     26
## 2     25  1 0.6384 0.4318
```

Do modelu dołączamy zmienną *li*, jako że jej p-value w teście Walda jest najmniejsze spośród pozostałych p-wartości i jest mniejsze, niż 0.30.

2. krok procedury - kolejny krok selekcji postępującej.

```
model_2 <- glm(remission~li, data=dane_2, method='glm.fit',
               family = 'binomial')
model_2_1 <- glm(remission~li+cell, data=dane_2, method='glm.fit',
                 family = 'binomial')
model_2_2 <- glm(remission~li+smear., data=dane_2, method='glm.fit',
                 family = 'binomial')
model_2_3 <- glm(remission~li+temp, data=dane_2, method='glm.fit',
                 family = 'binomial')
model_2_4 <- glm(remission~li+blast, data=dane_2, method='glm.fit',
                 family = 'binomial')
waldtest(model_2, model_2_1)

## Wald test
##
## Model 1: remission ~ li
## Model 2: remission ~ li + cell
##   Res.Df Df      F Pr(>F)
## 1     25
## 2     24  1 1.0457 0.3167

waldtest(model_2, model_2_2)

## Wald test
##
## Model 1: remission ~ li
## Model 2: remission ~ li + smear.
##   Res.Df Df      F Pr(>F)
## 1     25
## 2     24  1 0.1361 0.7155

waldtest(model_2, model_2_3) #bierzemy te zmienną do modelu

## Wald test
##
## Model 1: remission ~ li
## Model 2: remission ~ li + temp
##   Res.Df Df      F Pr(>F)
## 1     25
## 2     24  1 1.2189 0.2805

waldtest(model_2, model_2_4)

## Wald test
##
## Model 1: remission ~ li
```

```
## Model 2: remission ~ li + blast
##   Res.Df Df      F Pr(>F)
## 1     25
## 2     24  1 0.0929 0.7631

#wartości do (b, ii.) i (b, iii.)
beta_hat_2 <- model_2$coefficients[2]
se_hat_2 <- summary(model_2)$coefficients[,2][2]
p_value_2 <- summary(model_2)$coefficients[,4][2]
aic_2 <- model_2$aic
beta_hat_2

##       li
## 2.897264

se_hat_2

##      li
## 1.18682

p_value_2

##         li
## 0.01463855

aic_2

## [1] 30.07296
```

Tym razem do modelu dołączamy zmienną *temp* - jej p-value jest najmniejsze i mniejsze, niż 0.3.

3. krok procedury - wykonujemy krok **eliminacji wstecznej**

```
model_2 <- glm(remission~li+temp, data=dane_2, method='glm.fit',
               family = 'binomial')

#czy usunąć którąś ze zmiennych?
model_2_1 <- glm(remission~li, data=dane_2, method='glm.fit',
                 family = 'binomial')
model_2_2 <- glm(remission~temp, data=dane_2, method='glm.fit',
                 family = 'binomial')
#nie usuwamy
waldtest(model_2, model_2_1)

## Wald test
```

```
##
## Model 1: remission ~ li + temp
## Model 2: remission ~ li
##   Res.Df Df      F Pr(>F)
## 1     24
## 2     25 -1 1.2189 0.2805

waldtest(model_2, model_2_2)

## Wald test
##
## Model 1: remission ~ li + temp
## Model 2: remission ~ temp
##   Res.Df Df      F  Pr(>F)
## 1     24
## 2     25 -1 5.9007 0.02299 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#wartości do (b, ii.) i (b, iii.)
beta_hat_2 <- model_2$coefficients[2:3]
se_hat_2 <- summary(model_2)$coefficients[,2][2:3]
p_value_2 <- summary(model_2)$coefficients[,4][2:3]
aic_2 <- model_2$aic
beta_hat_2

##         li       temp
##   3.302045 -52.433116

se_hat_2

##        li      temp
##  1.359346 47.492492

p_value_2

##         li       temp
## 0.01513458 0.26958031

aic_2

## [1] 30.64782
```

P-wartości pochodzące z testów Walda są mniejsze od poziomu istotności 0.35, zatem odrzucamy hipotezę zerową (nie usuwamy żadnych zmiennych z modelu).

4. krok procedury - wykonujemy krok selekcji postępującej.

```
model_2_1 <- glm(remission~li+temp+cell, data=dane_2, method='glm.fit',
                 family='binomial')
model_2_2 <- glm(remission~li+temp+smear., data=dane_2, method='glm.fit',
                 family='binomial')
model_2_3 <- glm(remission~li+temp+blast, data=dane_2, method='glm.fit',
                 family='binomial')
waldtest(model_2, model_2_1) #bierzemy do modelu

## Wald test
##
## Model 1: remission ~ li + temp
## Model 2: remission ~ li + temp + cell
##   Res.Df Df      F Pr(>F)
## 1     24
## 2     23  1 1.5507 0.2256

waldtest(model_2, model_2_2)

## Wald test
##
## Model 1: remission ~ li + temp
## Model 2: remission ~ li + temp + smear.
##   Res.Df Df      F Pr(>F)
## 1     24
## 2     23  1 0.1715 0.6826

waldtest(model_2, model_2_3)

## Wald test
##
## Model 1: remission ~ li + temp
## Model 2: remission ~ li + temp + blast
##   Res.Df Df      F Pr(>F)
## 1     24
## 2     23  1 1.0542 0.3152
```

Na poziomie istotności 0.3 odrzucamy hipotezę zerową i do modelu dołączamy zmienną *cell*.

5. krok procedury - wykonujemy krok eliminacji wstecznej.

```
model_2 <- glm(remission~li+temp+cell, data=dane_2, method='glm.fit',
               family = 'binomial')

#czy usuwamy jakąś zmienną?
```

```r
model_2_1 <- glm(remission~li+temp, data=dane_2, method='glm.fit',
                 family='binomial')
model_2_2 <- glm(remission~li+cell, data=dane_2, method='glm.fit',
                 family='binomial')
model_2_3 <- glm(remission~temp+cell, data=dane_2, method='glm.fit',
                 family='binomial')

#nie usuwamy
waldtest(model_2, model_2_1)

## Wald test
##
## Model 1: remission ~ li + temp + cell
## Model 2: remission ~ li + temp
##   Res.Df Df      F Pr(>F)
## 1     23
## 2     24 -1 1.5507 0.2256

waldtest(model_2, model_2_2)

## Wald test
##
## Model 1: remission ~ li + temp + cell
## Model 2: remission ~ li + cell
##   Res.Df Df      F Pr(>F)
## 1     23
## 2     24 -1 1.7687 0.1966

waldtest(model_2, model_2_3)

## Wald test
##
## Model 1: remission ~ li + temp + cell
## Model 2: remission ~ temp + cell
##   Res.Df Df     F  Pr(>F)
## 1     23
## 2     24 -1 4.729 0.04019 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

#wartości do (b, ii.) i (b, iii.)
beta_hat_2 <- model_2$coefficients[2:4]
se_hat_2 <- summary(model_2)$coefficients[,2][2:4]
p_value_2 <- summary(model_2)$coefficients[,4][2:4]
aic_2 <- model_2$aic
beta_hat_2
```

```
##         li       temp       cell
##   3.867100 -82.073774   9.652152

se_hat_2

##        li      temp      cell
##  1.778278 61.712382  7.751076

p_value_2

##         li       temp       cell
## 0.02965768 0.18353799 0.21303394

aic_2

## [1] 29.95337
```

Na poziomie istotności 0.35 odrzucamy H_0 i nie usuwamy żadnej ze zmiennych z modelu.

6. krok procedury - wykonujemy krok selekcji postępującej.

```
model_2_1 <- glm(remission~li+temp+cell+smear., data=dane_2, method='glm.fit',
                 family='binomial')
model_2_2 <- glm(remission~li+temp+cell+blast, data=dane_2, method='glm.fit',
                 family='binomial')

#nie dołączamy kolejnych zmiennych
waldtest(model_2, model_2_1)

## Wald test
##
## Model 1: remission ~ li + temp + cell
## Model 2: remission ~ li + temp + cell + smear.
##   Res.Df Df     F Pr(>F)
## 1     23
## 2     22  1 0.095 0.7608

waldtest(model_2, model_2_2)

## Wald test
##
## Model 1: remission ~ li + temp + cell
## Model 2: remission ~ li + temp + cell + blast
##   Res.Df Df      F Pr(>F)
## 1     23
## 2     22  1 0.0208 0.8866
```

Teraz na poziomie istotności 0.3 nie mamy podstaw do odrzucenia hipotezy zerowej - co oznacza, że nie dołączamy kolejnej zmiennej do modelu. Ten krok kończy zatem procedurę regresji krokowej.

Wniosek 5 *Przyjmujemy model regresji logistycznej, w którym predyktorami są zmienne li, temp oraz cell.*

Stworzymy teraz tabelkę z wartościami estymatorów $\hat{\beta}_i$ i SE_{β_i} dla $i = 2, 3, 4$, p-wartościami testu Walda oraz wartościami kryetrium AIC wyliczonymi podczas procedury regresji krokowej.

kroki procedury	zmienne w modelu	$\hat{\boldsymbol{\beta}}$	$SE_{\boldsymbol{\beta}}$	p-wartość testu Walda	wartość AIC
1.	*li*	2.89	1.19	0.015	30.07
2.	*li, temp*	(3.3, -52.43)	(1.36, 47.5)	(0.015, 0.27)	30.65
3.	*li, temp*	(3.3, -52.43)	(1.36, 47.5)	(0.015, 0.27)	30.65
4.	*li, temp, cell*	(3.87, -82.07, 9.65)	(1.78, 61.71, 7.75)	(0.03, 0.18, 0.21)	29.95
5.	*li, temp, cell*	(3.87, -82.07, 9.65)	(1.78, 61.71, 7.75)	(0.03, 0.18, 0.21)	29.95
6.	*li, temp, cell*	(3.87, -82.07, 9.65)	(1.78, 61.71, 7.75)	(0.03, 0.18, 0.21)	29.95

Tabela 1: Tabelka ilustrująca zmianę wartości estymatorów $\hat{\boldsymbol{\beta}}$ i $SE_{\boldsymbol{\beta}}$, p-wartości (podanych jako wektory) oraz AIC w ciągu kolejnych kroków regresji krokowej

Wniosek 6 *Optymalny model odpowiada zestawowi zmiennych z najmniejszą wartością AIC.*

2.3 (c)

Konstruujemy tabelę, o którą poproszono nas w zadaniu. Uprzednio jednak liczymy odpowiednie wartości/parametry w poniższym kodzie. Zacznijmy od prognozowanego przez model prawdopodobieństwa $\pi(\boldsymbol{x})$, zdefiniowanego jako

$$\pi(\boldsymbol{x}) := P\left(remiss = 1 | \boldsymbol{X} = \boldsymbol{x}\right).$$

```
#prognozowane przez model p-ństwo pi(x)
pi_prob <- predict(model_2, dane_2, type='response')
```

Teraz wyliczymy 95% przedział ufności Walda dla $\pi(\boldsymbol{x})$. W tym celu przypomnijmy zdefiniowaną na wykładzie funkcję **logit**, zdefiniowaną jako

$$logit\left(\pi(\boldsymbol{x})\right) := \log \frac{\pi(\boldsymbol{x})}{1 - \pi(\boldsymbol{x})} = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p.$$

Wartości tej funkcji możemy łatwo uzyskać, korzystając z opcji *type = 'link'* w funkcji *predict*.

```
#liczymy logit
logit <- predict(model_2, dane_2, type='link', se.fit=TRUE)
```

Mając $logit\left(\pi\left(\boldsymbol{x}\right)\right)$, wyliczamy przedziały ufności Walda dla $\beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$ (u nas p=3) i używając funkcji **odwrotnej** do funkcji $logit\left(\pi\left(\boldsymbol{x}\right)\right)$ do końców wyliczonych przedziałów, dostajemy przedział ufności Walda dla $\pi\left(\boldsymbol{x}\right)$. Dlaczego? Łatwo sprawdzić, że funkcją odwrotną do $logit\left(x\right)$ jest funkcja

$$f\left(x\right) := \frac{e^x}{1+e^x}.$$

Zatem znajdując przedział ufności dla kombinacji liniowej

$$\beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p$$

i używając jej jako argument zdefiniowanej wyżej funkcji f, dostajemy przedział ufności Walda dla $\pi\left(\boldsymbol{x}\right)$.

```
inverse_logit <- function(x){
  return(exp(x)/(1+exp(x)))
}
CI_R <- inverse_logit(logit$fit + (logit$se.fit*qnorm(1-0.05/2)))
CI_L <- inverse_logit(logit$fit - (logit$se.fit*qnorm(1-0.05/2)))
df <- data.frame(model_2$model,
                 pi_prob,
                 CI_L,
                 CI_R)
names(df)[5] <- 'estimated_pi(x)'
names(df)[6] <- 'CI_Wald_left'
names(df)[7] <- 'CI_Wald_right'
head(df)
```

```
##   remission  li  temp cell estimated_pi(x) CI_Wald_left CI_Wald_right
## 1         1 1.9 0.996  0.8       0.7226489  0.168920305     0.9709308
## 2         1 1.4 0.992  0.9       0.5787391  0.267876917     0.8376193
## 3         0 0.8 0.982  0.8       0.1045990  0.007810045     0.6341884
## 4         0 0.7 0.986  1.0       0.2825773  0.074979321     0.6568252
## 5         1 1.3 0.980  0.9       0.7141804  0.252179466     0.9487569
## 6         0 0.6 0.982  1.0       0.2708868  0.058519423     0.6895121
```

Funkcja *xtable* z biblioteki *xtable* w prosty sposób pozwala zbudować nam pożądaną tabelę.

Wniosek 7 *Skonstruowane przedziały ufności Walda z tabeli nr 2 są "szerokie", dlatego nie są zbyt wiarygodne. Przyczyną najpewniej jest mało liczna próba.*

	remission	li	temp	cell	estimated_pi(x)	CI_Wald_left	CI_Wald_right
1	1	1.90	1.00	0.80	0.72	0.17	0.97
2	1	1.40	0.99	0.90	0.58	0.27	0.84
3	0	0.80	0.98	0.80	0.10	0.01	0.63
4	0	0.70	0.99	1.00	0.28	0.07	0.66
5	1	1.30	0.98	0.90	0.71	0.25	0.95
6	0	0.60	0.98	1.00	0.27	0.06	0.69
7	1	1.00	0.99	0.95	0.32	0.13	0.60
8	0	1.90	1.02	0.95	0.61	0.11	0.95
9	0	0.80	1.00	1.00	0.17	0.03	0.56
10	0	0.50	1.04	0.95	0.00	0.00	0.69
11	0	0.70	0.99	0.85	0.07	0.01	0.50
12	0	1.20	0.98	0.70	0.17	0.01	0.87
13	0	0.40	1.01	0.80	0.00	0.00	0.47
14	0	0.80	0.99	0.20	0.00	0.00	0.96
15	0	1.10	0.99	1.00	0.57	0.25	0.84
16	1	1.90	1.02	1.00	0.71	0.15	0.97
17	0	0.50	1.01	0.65	0.00	0.00	0.63
18	0	1.00	1.00	1.00	0.22	0.04	0.64
19	0	0.60	0.99	0.50	0.00	0.00	0.80
20	1	1.10	0.99	1.00	0.65	0.26	0.91
21	0	0.40	1.01	1.00	0.02	0.00	0.50
22	0	0.60	1.02	0.90	0.01	0.00	0.56
23	1	1.00	1.00	1.00	0.25	0.06	0.64
24	0	1.60	0.99	0.95	0.87	0.41	0.98
25	1	1.70	0.99	1.00	0.93	0.44	1.00
26	1	0.90	0.99	1.00	0.46	0.17	0.79
27	0	0.70	0.99	1.00	0.28	0.07	0.66

Tabela 2: Tabela przedstawiająca wartości zmiennych (remission, li, temp, cell), prognozowane przez model p-ństwo $\pi(\boldsymbol{x})$ oraz lewy i prawy koniec 95% przedziału ufności Walda dla $\pi(\boldsymbol{x})$