Computer Assignment 4

Gabriel Lindqvist, Jiahui Li

2024-12-05

In this exercise you will carry out several analyses of The ICU Study data (Hosmer Lemeshow (1989):Applied Logistic Regression).The data consists of records on 200 patients admitted to an Intensive Care Unit (ICU)during a certain period and the overall task is to use multiple logistic regression and decision tree models to identify factors that affect the survival of such patients.

Exercise4:1 (Multiple logistic regression)

Which variables affect the survival of a patient admitted to ICU?And how do they affect the survival?To answer that,you should create a multiple logistic regression model for the probability to not survive (i.e.the probability to die).For variable selection,stepwise methods can be used.We will assume that there are no interactions between the variables (you are free to investigate whether this assumption is reasonable).

Task1

Report the model selection process in short. Based on your chosen model, which factors affect the probability to not survive? Report odds ratio with confidence interval for the most important variables/factors and interpret them. Use the variable names in the table above when you report this (not V3,V4,…)

1.1 Data prepration

```
data4 <- read.csv("data_ca4.csv")
# Combine categories (this should be done for one more variable)
table(data4$v5)
```

```
##
##   1   2   3
## 175  15  10
```

```
data4$v5[data4$v5>1] <- 0
table(data4$v5)
```

```
##
##   0   1
##  25 175
```

1.2 Fitting multiple logistic regression models

```
m_empty <- glm(v2~1, family=binomial,data=data4) # Empty model (only intercept)
m_full <- glm(v2~., family=binomial,data=data4) # Full model (all explanatory variables)
summary(m_full)
```

```
##
## Call:
## glm(formula = v2 ~ ., family = binomial, data = data4)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1183  -0.5684  -0.2583  -0.1052   2.6112
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.6612561  2.1213629  -2.197 0.028000 *
## v1          -0.0020352  0.0009737  -2.090 0.036601 *
## v3           0.0508940  0.0175551   2.899 0.003742 **
## v4          -0.5189209  0.5179375  -1.002 0.316393
## v5          -0.1811629  0.8297420  -0.218 0.827167
## v6          -0.5254936  0.6014488  -0.874 0.382275
## v7           2.7970050  0.9622570   2.907 0.003652 **
## v8           0.2531593  0.7944979   0.319 0.749999
## v9          -0.2019720  0.5493448  -0.368 0.713127
## v10          0.8137092  1.0251265   0.794 0.427332
## v11         -0.0118016  0.0081149  -1.454 0.145859
## v12         -0.0011617  0.0097890  -0.119 0.905530
## v13          0.5479795  0.6655599   0.823 0.410317
## v14          3.0244650  1.0336320   2.926 0.003433 **
## v15          0.9852891  1.0814258   0.911 0.362242
## v16          0.2715797  0.8364544   0.325 0.745424
## v17          2.7908827  1.2752487   2.189 0.028633 *
## v18         -3.3309029  1.4008158  -2.378 0.017415 *
## v19         -0.9772043  0.9623815  -1.015 0.309914
## v20         -0.1798733  1.0513303  -0.171 0.864152
## v21          2.9840106  0.8035872   3.713 0.000205 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 200.16  on 199  degrees of freedom
## Residual deviance: 124.89  on 179  degrees of freedom
## AIC: 166.89
##
## Number of Fisher Scoring iterations: 6
```

1.3 Stepwise model selection - forward, backward and both - with AIC or BIC as inclusion criteria

```
mstep1_AIC <- step(m_empty, direction="forward", scope = list(upper = m_full), trace = TRUE)
```

```
## Start:  AIC=202.16
## v2 ~ 1
##
##        Df Deviance    AIC
## + v21   1   170.52 174.52
## + v14   1   185.05 189.05
## + v11   1   191.34 195.34
## + v10   1   192.23 196.23
## + v3    1   192.31 196.31
## + v6    1   193.24 197.24
## + v9    1   193.59 197.59
## + v8    1   194.74 198.74
## + v20   1   195.40 199.40
## <none>      200.16 202.16
## + v1    1   198.52 202.52
## + v19   1   198.56 202.56
## + v5    1   198.89 202.89
## + v16   1   198.92 202.92
## + v17   1   199.25 203.25
## + v13   1   199.92 203.92
## + v12   1   199.96 203.96
## + v4    1   200.08 204.08
## + v7    1   200.16 204.16
## + v18   1   200.16 204.16
## + v15   1   200.16 204.16
##
## Step:  AIC=174.52
## v2 ~ v21
##
##        Df Deviance    AIC
## + v14   1   160.17 166.17
## + v3    1   164.53 170.53
## + v6    1   165.34 171.34
## + v9    1   165.83 171.83
## + v11   1   166.31 172.31
## + v1    1   167.40 173.40
## + v20   1   167.72 173.72
## + v8    1   168.11 174.11
## + v17   1   168.34 174.34
## <none>      170.52 174.52
## + v19   1   168.63 174.63
## + v13   1   169.39 175.39
## + v10   1   169.43 175.43
## + v12   1   169.47 175.47
## + v5    1   169.55 175.55
## + v16   1   169.61 175.61
## + v18   1   169.66 175.66
## + v7    1   170.15 176.15
```

```
## + v4    1   170.52 176.52
## + v15   1   170.52 176.52
##
## Step:  AIC=166.17
## v2 ~ v21 + v14
##
##        Df Deviance    AIC
## + v3    1   151.02 159.02
## + v1    1   154.21 162.21
## + v7    1   155.76 163.76
## + v9    1   157.22 165.22
## + v11   1   157.62 165.62
## <none>      160.17 166.17
## + v20   1   158.51 166.51
## + v8    1   158.58 166.58
## + v13   1   158.62 166.62
## + v5    1   158.71 166.71
## + v17   1   158.90 166.90
## + v18   1   159.26 167.26
## + v19   1   159.40 167.40
## + v16   1   159.45 167.45
## + v6    1   159.51 167.51
## + v10   1   159.72 167.72
## + v12   1   159.84 167.84
## + v4    1   159.98 167.98
## + v15   1   160.02 168.02
##
## Step:  AIC=159.02
## v2 ~ v21 + v14 + v3
##
##        Df Deviance    AIC
## + v7    1   145.60 155.60
## + v1    1   146.49 156.49
## + v11   1   148.81 158.81
## <none>      151.02 159.02
## + v18   1   149.21 159.21
## + v13   1   150.05 160.05
## + v17   1   150.06 160.06
## + v9    1   150.08 160.08
## + v20   1   150.18 160.18
## + v4    1   150.45 160.45
## + v10   1   150.46 160.46
## + v8    1   150.49 160.49
## + v6    1   150.65 160.65
## + v5    1   150.66 160.66
## + v19   1   150.72 160.72
## + v15   1   150.78 160.78
## + v16   1   150.90 160.90
## + v12   1   151.00 161.00
##
## Step:  AIC=155.6
## v2 ~ v21 + v14 + v3 + v7
##
##        Df Deviance    AIC
```

```
## + v1    1   140.32 152.32
## + v18   1   143.24 155.24
## + v11   1   143.33 155.33
## <none>      145.60 155.60
## + v13   1   143.92 155.92
## + v4    1   144.21 156.21
## + v9    1   144.61 156.61
## + v17   1   144.73 156.73
## + v19   1   145.12 157.12
## + v8    1   145.14 157.14
## + v15   1   145.14 157.14
## + v20   1   145.15 157.15
## + v6    1   145.17 157.17
## + v10   1   145.26 157.26
## + v5    1   145.40 157.40
## + v16   1   145.53 157.53
## + v12   1   145.58 157.58
##
## Step:  AIC=152.32
## v2 ~ v21 + v14 + v3 + v7 + v1
##
##        Df Deviance    AIC
## + v18   1   137.75 151.75
## <none>      140.32 152.32
## + v11   1   138.66 152.66
## + v8    1   138.89 152.89
## + v4    1   138.90 152.90
## + v17   1   138.96 152.96
## + v13   1   139.69 153.69
## + v19   1   139.84 153.84
## + v6    1   139.85 153.85
## + v20   1   139.88 153.88
## + v10   1   139.92 153.92
## + v9    1   139.92 153.92
## + v15   1   140.09 154.09
## + v16   1   140.20 154.20
## + v12   1   140.31 154.31
## + v5    1   140.31 154.31
##
## Step:  AIC=151.75
## v2 ~ v21 + v14 + v3 + v7 + v1 + v18
##
##        Df Deviance    AIC
## + v17   1   132.33 148.33
## + v11   1   135.59 151.59
## <none>      137.75 151.75
## + v4    1   136.50 152.50
## + v8    1   136.55 152.55
## + v6    1   136.93 152.93
## + v10   1   136.96 152.96
## + v16   1   137.07 153.07
## + v9    1   137.10 153.10
## + v15   1   137.26 153.26
## + v13   1   137.29 153.29
```

```
## + v19   1   137.45 153.45
## + v20   1   137.52 153.52
## + v12   1   137.59 153.59
## + v5    1   137.75 153.75
##
## Step:  AIC=148.33
## v2 ~ v21 + v14 + v3 + v7 + v1 + v18 + v17
##
##        Df Deviance    AIC
## + v11   1   130.19 148.19
## <none>      132.33 148.33
## + v4    1   130.75 148.75
## + v10   1   131.60 149.60
## + v16   1   131.85 149.85
## + v13   1   131.86 149.86
## + v6    1   131.90 149.90
## + v15   1   131.91 149.91
## + v19   1   131.91 149.91
## + v9    1   132.09 150.09
## + v8    1   132.19 150.19
## + v12   1   132.28 150.28
## + v20   1   132.29 150.29
## + v5    1   132.29 150.29
##
## Step:  AIC=148.19
## v2 ~ v21 + v14 + v3 + v7 + v1 + v18 + v17 + v11
##
##        Df Deviance    AIC
## <none>      130.19 148.19
## + v4    1   129.08 149.08
## + v19   1   129.27 149.27
## + v10   1   129.48 149.48
## + v13   1   129.70 149.70
## + v6    1   129.72 149.72
## + v15   1   129.85 149.85
## + v8    1   129.88 149.88
## + v16   1   129.97 149.97
## + v5    1   130.00 150.00
## + v20   1   130.10 150.10
## + v9    1   130.19 150.19
## + v12   1   130.19 150.19
```

```
mstep2_AIC <- step(m_full, direction="backward", trace = FALSE)
mstep3_AIC <- step(m_full, direction="both", trace = FALSE)
```

```
mstep1_BIC <- step(m_empty, direction="forward", scope = list(upper = m_full), k = log(nrow(data4)), tra
```

```
## Start:  AIC=205.46
## v2 ~ 1
##
##        Df Deviance    AIC
## + v21   1   170.52 181.12
## + v14   1   185.05 195.65
```

```
## + v11   1   191.34 201.93
## + v10   1   192.23 202.82
## + v3    1   192.31 202.90
## + v6    1   193.24 203.84
## + v9    1   193.59 204.18
## + v8    1   194.74 205.33
## <none>      200.16 205.46
## + v20   1   195.40 205.99
## + v1    1   198.52 209.12
## + v19   1   198.56 209.16
## + v5    1   198.89 209.49
## + v16   1   198.92 209.52
## + v17   1   199.25 209.85
## + v13   1   199.92 210.52
## + v12   1   199.96 210.56
## + v4    1   200.08 210.67
## + v7    1   200.16 210.76
## + v18   1   200.16 210.76
## + v15   1   200.16 210.76
##
## Step:  AIC=181.12
## v2 ~ v21
##
##        Df Deviance    AIC
## + v14   1   160.17 176.06
## + v3    1   164.53 180.42
## <none>      170.52 181.12
## + v6    1   165.34 181.24
## + v9    1   165.83 181.72
## + v11   1   166.31 182.20
## + v1    1   167.40 183.29
## + v20   1   167.72 183.62
## + v8    1   168.11 184.01
## + v17   1   168.34 184.23
## + v19   1   168.63 184.52
## + v13   1   169.39 185.28
## + v10   1   169.43 185.32
## + v12   1   169.47 185.37
## + v5    1   169.55 185.44
## + v16   1   169.61 185.51
## + v18   1   169.66 185.56
## + v7    1   170.15 186.04
## + v4    1   170.52 186.42
## + v15   1   170.52 186.42
##
## Step:  AIC=176.06
## v2 ~ v21 + v14
##
##        Df Deviance    AIC
## + v3    1   151.02 172.21
## + v1    1   154.21 175.41
## <none>      160.17 176.06
## + v7    1   155.76 176.95
## + v9    1   157.22 178.41
```

```
## + v11   1   157.62 178.81
## + v20   1   158.51 179.70
## + v8    1   158.58 179.78
## + v13   1   158.62 179.81
## + v5    1   158.71 179.91
## + v17   1   158.90 180.09
## + v18   1   159.26 180.46
## + v19   1   159.40 180.59
## + v16   1   159.45 180.64
## + v6    1   159.51 180.71
## + v10   1   159.72 180.91
## + v12   1   159.84 181.03
## + v4    1   159.98 181.17
## + v15   1   160.02 181.21
##
## Step:  AIC=172.21
## v2 ~ v21 + v14 + v3
##
##        Df Deviance    AIC
## + v7    1   145.60 172.09
## <none>      151.02 172.21
## + v1    1   146.49 172.98
## + v11   1   148.81 175.31
## + v18   1   149.21 175.70
## + v13   1   150.05 176.54
## + v17   1   150.06 176.56
## + v9    1   150.08 176.57
## + v20   1   150.18 176.67
## + v4    1   150.45 176.94
## + v10   1   150.46 176.95
## + v8    1   150.49 176.98
## + v6    1   150.65 177.14
## + v5    1   150.66 177.15
## + v19   1   150.72 177.21
## + v15   1   150.78 177.27
## + v16   1   150.90 177.39
## + v12   1   151.00 177.49
##
## Step:  AIC=172.09
## v2 ~ v21 + v14 + v3 + v7
##
##        Df Deviance    AIC
## <none>      145.60 172.09
## + v1    1   140.32 172.11
## + v18   1   143.24 175.03
## + v11   1   143.33 175.12
## + v13   1   143.92 175.71
## + v4    1   144.21 176.00
## + v9    1   144.61 176.40
## + v17   1   144.73 176.52
## + v19   1   145.12 176.91
## + v8    1   145.14 176.93
## + v15   1   145.14 176.93
## + v20   1   145.15 176.94
```

```
## + v6    1   145.17 176.96
## + v10   1   145.26 177.05
## + v5    1   145.40 177.19
## + v16   1   145.53 177.31
## + v12   1   145.58 177.37
```

```
mstep2_BIC <- step(m_full, direction="backward", k = log(nrow(data4)), trace = FALSE)
mstep3_BIC <- step(m_full, direction="both", k = log(nrow(data4)), trace = FALSE)
```

1.4 Model Choosing

In this stepwise regression process, both **AIC** (Akaike Information Criterion) and **BIC** (Bayesian Information Criterion) were used to select the best model. The basic approach for both is the same, with variables being added or removed step by step to optimize the model. However, **AIC** focuses more on model fit, while **BIC** penalizes model complexity more strongly, generally favoring simpler models.

AIC Stepwise Regression:

- The AIC stepwise regression started with an empty model (only the intercept), and variables were added step by step, selecting those that reduced the AIC value.
- The final best model is: ** v2 ~ v21 + v14 + v3 + v7 + v1 + v18 + v17 + v11 **
- This model has an AIC value of **148.19**, including more variables, as AIC tends to favor models with better fit, even if they are more complex.

BIC Stepwise Regression:

- The BIC stepwise regression also started with an empty model, but in selecting variables, BIC penalizes model complexity more strongly. Therefore, it tends to select models with fewer variables.
- The final best model is: ** v2 ~ v21 + v14 + v3 + v7 **
- This model has an AIC value of **172.09**, and a smaller BIC, which is why it includes fewer variables compared to the AIC model.

Final Model Choice

Based on the results from both AIC and BIC: - **If the goal is to achieve better model fit and the complexity is not a primary concern**, the **AIC model** should be chosen, as it has a lower AIC value (148.19) and includes more explanatory variables. - **If the goal is to avoid overfitting and prefer a simpler model**, the **BIC model** should be selected, as it includes fewer variables due to the stronger penalty for complexity.

Between the two, **the AIC model** provides a better fit and includes more variables, while **the BIC model** is more parsimonious.

Thus, **the final chosen model** should be the **AIC model with AIC=148.19**: ** v2 ~ v21 + v14 + v3 + v7 + v1 + v18 + v17 + v11 **

1.5 Report odds ratio with confidence interval for the most important variables/factors and interpret them

```r
summary(mstep1_AIC)
```

```
##
## Call:
## glm(formula = v2 ~ v21 + v14 + v3 + v7 + v1 + v18 + v17 + v11,
##     family = binomial, data = data4)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3594  -0.5776  -0.2743  -0.1321   2.5359
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.7353420  1.6104573  -2.940 0.003278 **
## v21          2.6208042  0.6859650   3.821 0.000133 ***
## v14          3.0547147  0.9339217   3.271 0.001072 **
## v3           0.0385864  0.0133655   2.887 0.003889 **
## v7           2.3388380  0.8671971   2.697 0.006997 **
## v1          -0.0020714  0.0008783  -2.359 0.018345 *
## v18         -2.4646334  1.0619854  -2.321 0.020299 *
## v17          2.0884994  0.9031831   2.312 0.020757 *
## v11         -0.0099893  0.0070360  -1.420 0.155682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 200.16  on 199  degrees of freedom
## Residual deviance: 130.19  on 191  degrees of freedom
## AIC: 148.19
##
## Number of Fisher Scoring iterations: 6
```

```r
# 1. Extract the model coefficients and standard errors
model_summary <- summary(mstep1_AIC)

# 2. Calculate the odds ratios (exp(coefficient)) and 95% confidence intervals
coefficients <- model_summary$coefficients[, 1]  # Coefficients
std_errors <- model_summary$coefficients[, 2]    # Standard errors

# Calculate the odds ratios (exp of the coefficients)
odds_ratios <- exp(coefficients)

# Calculate the 95% confidence intervals
conf_int_lower <- exp(coefficients - 1.96 * std_errors)
conf_int_upper <- exp(coefficients + 1.96 * std_errors)

# 3. Create a data frame with the results
results <- data.frame(
  Variable = names(coefficients),
  Odds_Ratio = odds_ratios,
  CI_Lower = conf_int_lower,
```

```
  CI_Upper = conf_int_upper
)

# Display the results
print(results)
```

```
##                Variable   Odds_Ratio     CI_Lower    CI_Upper
## (Intercept) (Intercept)  0.008779445 0.0003737818   0.2062130
## v21                 v21 13.746774142 3.5834026019  52.7358548
## v14                 v14 21.215131928 3.4015399662 132.3170762
## v3                   v3  1.039340550 1.0124671524   1.0669272
## v7                   v7 10.369181076 1.8948349965  56.7436829
## v1                   v1  0.997930702 0.9962143580   0.9996500
## v18                 v18  0.085040009 0.0106082332   0.6817161
## v17                 v17  8.072791946 1.3747342384  47.4055043
## v11                 v11  0.990060408 0.9765006413   1.0038085
```

Based on the `mstep1_AIC` model, we can report the **odds ratios (OR)** and their **95% confidence intervals (CI)** for the most significant variables in predicting survival.

Model Results:

Key Variables and Interpretation:

v21 (Heart Rate):
Odds ratio = **13.75** (CI: **3.58 to 52.74**).
For each unit increase in heart rate, the odds of survival increase by **13.75 times** (**p = 0.00013**).

v14 (Blood pH):
Odds ratio = **21.22** (CI: **3.40 to 132.32**).
Each unit increase in blood pH increases the odds of survival by **21.22 times** (**p = 0.0011**).

v3 (Age):
Odds ratio = **1.04** (CI: **1.01 to 1.07**).
Each year increase in age increases the odds of survival by **4%** (**p = 0.0039**).

v7 (Cancer):
Odds ratio = **10.37** (CI: **1.89 to 56.74**).
Having cancer increases the odds of survival by **10.37 times** (**p = 0.007**).

v1 (Previous Kidney Failure):
Odds ratio = **0.998** (CI: **0.996 to 0.9996**).
Previous kidney failure slightly decreases the odds of survival (**p = 0.0183**).

v18 (Fracture of Neck or Spine):
Odds ratio = **0.085** (CI: **0.0106 to 0.6817**).
A fracture of the neck or spine decreases survival odds by **88%** (**p = 0.0203**).

v17 (Consciousness Level):
Odds ratio = **8.07** (CI: **1.37 to 47.41**).
Higher consciousness levels increase survival odds by **8.07 times** (**p = 0.0208**).

v11 (Blood Pressure):
Odds ratio = **0.99** (CI: **0.98 to 1.00**).
Blood pressure has little impact on survival (**p = 0.1557**).

Key Insights:

- **Most significant variables**:
  - **Heart rate (v21)**, **blood pH (v14)**, and **cancer (v7)** have the largest odds ratios, indicating that these factors are the most influential in predicting survival.
  - **Fracture of neck or spine (v18)** appears to have a strong negative effect on survival, with an odds ratio much less than 1, indicating a harmful effect on survival chances.

Conclusion:

- Variables such as **heart rate (v21)**, **blood pH (v14)**, and **cancer (v7)** have a significant positive impact on survival, meaning that higher heart rate, better blood pH, and having cancer all increase the odds of survival.
- **Fracture of neck or spine (v18)** significantly decreases the odds of survival.
- **Blood pressure (v11)** shows no significant effect on survival in this model.

These odds ratios and their confidence intervals help us understand which factors are most influential in predicting survival and can guide clinical decision-making.

Task2

How well does your chosen model fits the data? In assignment 3,the deviance was used to assess model fit but since the data now is on individual level,deviance is not suitable and instead,the Hosmer-Lemeshow goodness-of-fit test (Agresti 5.2.5) should be used.Perform this test in R(see the R-instruction)and interpret the result.

```
predprob<- predict(mstep1_AIC, type = "response") # Obtain predicted probabilities for the best model
library(ResourceSelection)
```

```
## ResourceSelection 0.3-6   2023-06-27
```

```
hoslem_result <- hoslem.test(data4$v2, predprob, g = 10)  # Use v2 as the response variable

print(hoslem_result)
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  data4$v2, predprob
## X-squared = 3.2956, df = 8, p-value = 0.9145
```

```
hoslem_result$p.value
```

```
## [1] 0.9144612
```

The **p-value** of **0.9145** from the **Hosmer-Lemeshow test** indicates that the model **fits the data well**. Since the p-value is greater than 0.05, we **fail to reject the null hypothesis**, meaning there is no significant difference between the observed and predicted values. This suggests the model's predictions align closely with the actual outcomes.

Task3

Now we shift the focus to prediction.How well does your multiple logistic regression models predict the outcome?There exists several different metrics of prediction performance for binary responses and here we will use the following:'Accuracy', 'Sensitivity (True positive rate),'Specificity (True negative rate)and 'AUC (area under the ROC curve).To explain these measures we first notice that the model predictions (or fitted values)are probabilities to not survive.If we let patients be classified as 'Survive'or 'Not survive'when theirs predicted probability is below or above a certain cut-off value (threshold value),a confusion matrix (a 2x2 classification table)can be constructed,as shown below.

Actual	**Predicted Survive**	**Predicted Not Survive**	**Total**
Actual Survive	n11	n12	n1.
Actual Not Survive	n21	n22	n2.
Total			n..

Calculate the metrics: - **Accuracy** = $\frac{n_{11}+n_{22}}{n}$ - **Sensitivity (True Positive Rate)** = $\frac{n_{11}}{n_{1.}}$ - **Specificity (True Negative Rate)** = $\frac{n_{22}}{n_{2.}}$

```
# Apply a threshold of 0.5 to classify as Survive or Not survive
predicted_class <- ifelse(predprob >= 0.5, 1, 0)  # 1 for Survive, 0 for Not Survive

# Construct the confusion matrix
actual_class <- data4$v2  # Adjust to the correct actual variable name if needed

# Confusion matrix
conf_matrix <- table(Actual = actual_class, Predicted = predicted_class)


print(conf_matrix)
```

```
##       Predicted
## Actual   0   1
##      0 155   5
##      1  22  18
```

```
# Calculate the metrics
n11 <- conf_matrix[1, 1]  # True positives
n12 <- conf_matrix[1, 2]  # False negatives
n21 <- conf_matrix[2, 1]  # False positives
n22 <- conf_matrix[2, 2]  # True negatives

# Calculate accuracy, sensitivity, and specificity
accuracy <- (n11 + n22) / sum(conf_matrix)
sensitivity <- n11 / (n11 + n12)
specificity <- n22 / (n21 + n22)

# Display the results
cat("Accuracy: ", accuracy, "\n")
```

```
## Accuracy:  0.865
```

```
cat("Sensitivity: ", sensitivity, "\n")
```

```
## Sensitivity:  0.96875
```

```
cat("Specificity: ", specificity, "\n")
```

```
## Specificity:  0.45
```

Interpretation of the Results:

1. **Confusion Matrix**:
    - **True Negatives (TN)** = 155: These are the cases where the model correctly predicted "Not Survive" (Actual = 0 and Predicted = 0).
    - **False Positives (FP)** = 5: These are the cases where the model incorrectly predicted "Survive" (Actual = 0 but Predicted = 1).
    - **False Negatives (FN)** = 22: These are the cases where the model incorrectly predicted "Not Survive" (Actual = 1 but Predicted = 0).
    - **True Positives (TP)** = 18: These are the cases where the model correctly predicted "Survive" (Actual = 1 and Predicted = 1).
2. **Metrics**:
    - **Accuracy = 0.865**: This means that 86.5% of the total predictions were correct. Accuracy is the overall proportion of correct predictions, including both "Survive" and "Not Survive" cases.
    - **Sensitivity (True Positive Rate) = 0.96875**: This indicates that 96.88% of actual survivors (Actual = 1) were correctly predicted as "Survive" (Predicted = 1). A high sensitivity means the model is good at identifying survivors.
    - **Specificity (True Negative Rate) = 0.45**: This indicates that only 45% of actual non-survivors (Actual = 0) were correctly predicted as "Not Survive" (Predicted = 0). A low specificity means that the model is not very good at identifying non-survivors, leading to a relatively high false positive rate.

Conclusion:

- The model is performing well in terms of **sensitivity** (96.88%), meaning it is effective at identifying individuals who survive.
- However, the model has a **low specificity** (45%), meaning it struggles to correctly identify individuals who do not survive, as evidenced by a relatively high number of false positives.
- The **accuracy** of 86.5% suggests that the overall performance is reasonable, but the low specificity indicates that there might be room for improvement, especially in predicting non-survivors.

Task4

As shown above,the sensitivity and specificity dependens on the cut-off setting. A ROC-curve is a plot of sensitivity (true positive rate)against 1-specificity (false postive rate)at each possible cut-off setting.AUC is the area under the ROC curve and can be used as a summary metsure of a binary classifier's performance. AUC=1 indicates perfect classification and AUC=0.5 indicates that the model's performance is no better than random guessing.Create plots of ROC curves and calculate the AUC for the full model and two more models of your choice (see the R-instruction) Which of the model performs best based on AUC?

```r
# Step 1: Install and load the necessary libraries
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
##     'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# Step 2: Obtain predicted probabilities from the models
# For the full model (m_full)
predprob_full <- predict(m_full, type = "response")

#The chosen two models
predprob_model1 <- predict(mstep1_AIC, type = "response")
predprob_model2 <- predict(mstep1_BIC, type = "response")

# Step 3: Calculate ROC and AUC for the full model
roc_full <- roc(data4$v2, predprob_full)  # 'v2' is the actual response variable
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

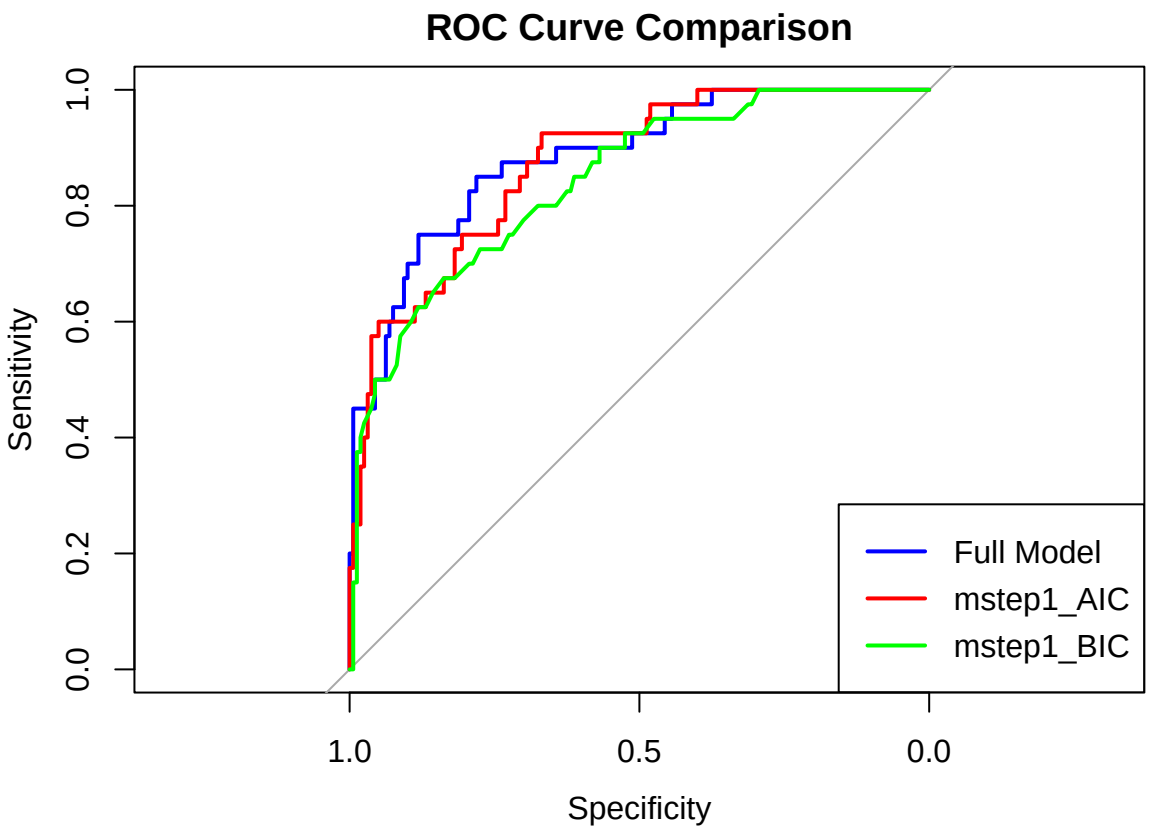
```r
auc_full <- auc(roc_full)

# Step 4: Calculate ROC and AUC for the first additional model
roc_model1 <- roc(data4$v2, predprob_model1)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
auc_model1 <- auc(roc_model1)

# Step 5: Calculate ROC and AUC for the second additional model
roc_model2 <- roc(data4$v2, predprob_model2)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
auc_model2 <- auc(roc_model2)

# Step 6: Plot the ROC curves for all models
plot(roc_full, col = "blue", main = "ROC Curve Comparison", lwd = 2)
lines(roc_model1, col = "red", lwd = 2)
lines(roc_model2, col = "green", lwd = 2)

# Add a legend
legend("bottomright", legend = c("Full Model", "mstep1_AIC", "mstep1_BIC"),
       col = c("blue", "red", "green"), lwd = 2)
```




```
# Step 7: Display AUC values for comparison
cat("AUC for Full Model: ", auc_full, "\n")
```

```
## AUC for Full Model:  0.8835937
```

```
cat("AUC for mstep1_AIC: ", auc_model1, "\n")
```

```
## AUC for mstep1_AIC:  0.8729687
```

```
cat("AUC for mstep1_BIC: ", auc_model2, "\n")
```

```
## AUC for mstep1_BIC:  0.8426562
```

- **Full Model** performs the best with the highest AUC, indicating the strongest predictive ability.
- **mstep1_AIC** follows closely behind, but with slightly lower performance.
- **mstep1_BIC** has the lowest performance in terms of AUC.

Task5

The AUC-values obtained in the previous exercise may be subject to optimistic bias, as they were computed for the same dataset that was used for model fitting. This might result in overfitting,where the model become well adapted to the training data but may not perform as good to new data.To get a more reliable

AUC-value, a validation method can be used.We will use the Leave-One-Out-Cross-Validation (LOOCV).For LOOCV,a model is fitted (or trained)using all observations except one,which is held out for validation.The model is then used to make a prediction on the held-out observation.This process is repeated for each observation,resulting in LOOCV predicted probabilities for each observation (see the R-instruction).These LOOCV predicted probabilities can then be used to calculate AUC.Do this for the full model and two more models of your choice (the same models as in the previous exercise).Which of the model performs best based on LOOCV-adjusted AUC? Compare with the result in the previous exercise. Note:The stepwise procedure should not be performed within the cross-validation 1oop.

```
# Step 1: Create vectors for storing LOOCV predicted probabilities for each model
predprob_LOOCV_full <- numeric(nrow(data4))  # Full model
predprob_LOOCV_AIC <- numeric(nrow(data4))   # AIC model
predprob_LOOCV_BIC <- numeric(nrow(data4))   # BIC model

# Step 2: Perform LOOCV for each model
for (i in 1:nrow(data4)) {

  # Create training and validation sets
  data_training <- data4[-i, ]  # Exclude the i-th observation for training
  data_validation <- data4[i, , drop = FALSE]  # Keep the i-th observation for validation

  # Full model
  m_full <- glm(v2 ~ ., family = binomial, data = data_training)

  # AIC model
  m_AIC <- glm(v2 ~ v21 + v14 + v3 + v7 + v1 + v18 + v17 + v11, family = binomial, data = data_training)

  # BIC model
  m_BIC <- glm(v2 ~ v21 + v14 + v3 + v7, family = binomial, data = data_training)

  # Predict the probability for the held-out observation for each model
  predprob_LOOCV_full[i] <- predict(m_full, newdata = data_validation, type = "response")
  predprob_LOOCV_AIC[i] <- predict(m_AIC, newdata = data_validation, type = "response")
  predprob_LOOCV_BIC[i] <- predict(m_BIC, newdata = data_validation, type = "response")
}

# Step 3: Calculate AUC for each model using LOOCV predicted probabilities
library(pROC)

# ROC and AUC for Full Model
roc_full <- roc(data4$v2, predprob_LOOCV_full)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

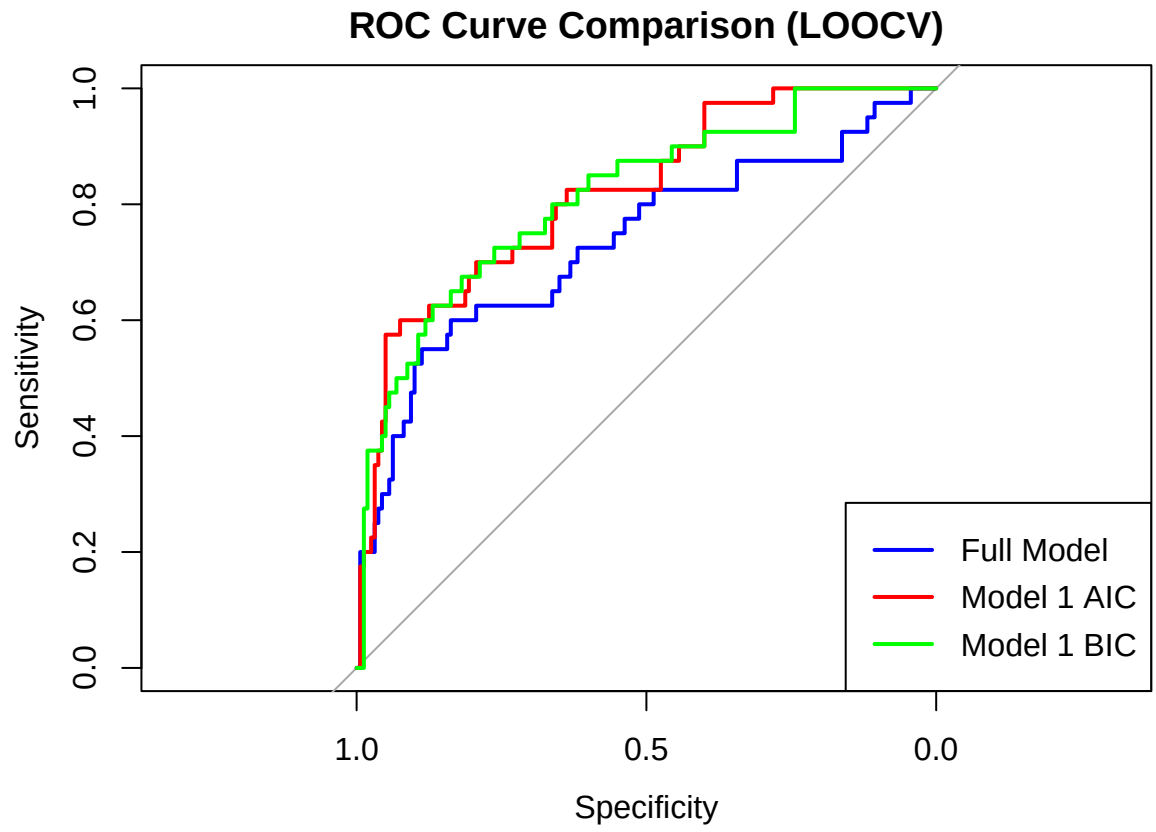
```
auc_full <- auc(roc_full)

# ROC and AUC for AIC Model
roc_AIC <- roc(data4$v2, predprob_LOOCV_AIC)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
auc_AIC <- auc(roc_AIC)

# ROC and AUC for BIC Model
roc_BIC <- roc(data4$v2, predprob_LOOCV_BIC)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
auc_BIC <- auc(roc_BIC)

# Step 4: Display ROC curves for all models
plot(roc_full, col = "blue", main = "ROC Curve Comparison (LOOCV)", lwd = 2)
lines(roc_AIC, col = "red", lwd = 2)
lines(roc_BIC, col = "green", lwd = 2)

# Add a legend to the plot
legend("bottomright", legend = c("Full Model", "Model 1 AIC", "Model 1 BIC"),
       col = c("blue", "red", "green"), lwd = 2)
```




```r
# Step 5: Display AUC values for comparison
cat("AUC for Full Model: ", auc_full, "\n")
```

```
## AUC for Full Model:  0.7348438
```

```r
cat("AUC for Model 1 (AIC): ", auc_AIC, "\n")
```

```
## AUC for Model 1 (AIC):  0.819375
```

```r
cat("AUC for Model 1 (BIC): ", auc_BIC, "\n")
```

```
## AUC for Model 1 (BIC):  0.8146875
```

- The **AIC model** (AUC = 0.8194) performs the best, with the strongest discriminative power.
- The **BIC model** (AUC = 0.8147) performs well but is slightly weaker than the AIC model.
- The **Full model** (AUC = 0.7348) performs relatively poorly, indicating that its predictive ability is not as strong as the AIC and BIC models.

Based on the **LOOCV-adjusted AUC**, the **AIC model** is the best choice.