2) Logistic Regression

2023-04-04

Pre-processing the data-set

```
library(MASS)
data <- Boston

processed_data <- na.omit(data)

processed_data$high_medv <- ifelse(processed_data$medv > median(processed_data$medv),
1, 0)
head(processed_data)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv high_medv
## 1 24.0         1
## 2 21.6         1
## 3 34.7         1
## 4 33.4         1
## 5 36.2         1
## 6 28.7         1
```

```
summary(processed_data)
```

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm             age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv         high_medv     
##  Min.   : 1.73   Min.   : 5.00   Min.   :0.0000  
##  1st Qu.: 6.95   1st Qu.:17.02   1st Qu.:0.0000  
##  Median :11.36   Median :21.20   Median :0.0000  
##  Mean   :12.65   Mean   :22.53   Mean   :0.4941  
##  3rd Qu.:16.95   3rd Qu.:25.00   3rd Qu.:1.0000  
##  Max.   :37.97   Max.   :50.00   Max.   :1.0000  
```

```
str(processed_data)
```

```
## 'data.frame':    506 obs. of  15 variables:
##  $ crim     : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn       : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus    : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox      : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm       : num  6.58 6.42 7.18 7 7.15 ...
##  $ age      : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis      : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad      : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax      : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio  : num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black    : num  397 397 393 395 397 ...
##  $ lstat    : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv     : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
##  $ high_medv: num  1 1 1 1 1 1 1 1 0 0 ...
```

```
nrow(processed_data)
```

```
## [1] 506
```

Splitting the model

```
library(caret)
```

```
## Loading required package: ggplot2
```

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

```
indexs = createDataPartition(processed_data$high_medv, times = 1, p = 0.7, list = F)
#times = no. of times to be split
#p = percentage of data to be used for training, here 70% is used of training and 30%
for testing

train = processed_data[indexs, ]
nrow(train)
```

```
## [1] 355
```

```
test = processed_data[-indexs, ]
nrow(test)
```

```
## [1] 151
```

Creating the model

```
# y - high_medv - dependent
# x - lstat - independent
# dependent ~ independent
model <- glm(processed_data$high_medv ~ processed_data$lstat, data = train)
model
```

```
##
## Call:  glm(formula = processed_data$high_medv ~ processed_data$lstat,
##     data = train)
##
## Coefficients:
##          (Intercept)  processed_data$lstat
##              1.08228              -0.04649
##
## Degrees of Freedom: 505 Total (i.e. Null);  504 Residual
## Null Deviance:       126.5
## Residual Deviance: 70.83     AIC: 447
```

```
summary(model)
```

```
## 
## Call:
## glm(formula = processed_data$high_medv ~ processed_data$lstat, 
##     data = train)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -0.8233  -0.3249   0.0969   0.2675   1.2914  
## 
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)    
## (Intercept)           1.082277   0.033933    31.9   <2e-16 ***
## processed_data$lstat -0.046487   0.002336   -19.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 0.1405352)
## 
##     Null deviance: 126.48  on 505  degrees of freedom
## Residual deviance:  70.83  on 504  degrees of freedom
## AIC: 447.04
## 
## Number of Fisher Scoring iterations: 2
```

Predicting the values using the model

```
predicted <- predict(model, newdata = test)
```

```
## Warning: 'newdata' had 151 rows but variables found have 506 rows
```

```
predicted <- ifelse(predicted>mean(predicted),1,0)
predicted
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20 
##   1   1   1   1   1   1   1   0   0   0   0   0   0   1   1   1   1   0   1   1 
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 
##   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   1   1   1   1   1 
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60 
##   1   1   1   1   1   1   0   0   0   0   0   1   1   1   0   1   1   1   1   1 
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 
##   0   0   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   1   1 
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 
##   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   1   1   1   1 
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 
##   1   1   1   0   1   0   0   0   1   0   0   1   0   0   1   0   1   1   0   0 
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 
##   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0   0 
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 
##   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   1   1   1 
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 
##   1   1   1   1   1   1   1   1   1   1   0   1   0   1   1   1   1   1   1   1 
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 
##   1   1   1   1   0   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 
##   1   1   1   1   1   1   1   0   0   0   0   0   0   1   0   1   0   1   0   1 
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 
##   1   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 
##   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1 
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 
##   1   1   1   1   1   1   0   1   1   0   0   1   1   1   1   1   1   1   1   1 
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 
##   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   1   0   1   1 
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   0   0   1   0 
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 
##   1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   1   1   1 
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 
##   1   1   1   1   1   1   0   1   1   1   1   1   1   1   1   1   0   0   1   0 
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 
##   1   0   1   0   1   1   0   0   1   1   1   1   1   0   0   0   0   0   0   0 
## 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 
## 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 
##   0   0   0   0   0   0   0   1   0   0   1   0   0   0   0   0   0   0   0   0 
## 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 
##   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0 
## 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 
## 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 
##   0   0   0   1   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0 
## 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 
##   1   1   1   1   0   1   0   1   0   0   0   0   0   1   0   0   0   0   0   0 
## 501 502 503 504 505 506 
##   0   1   1   1   1   1
```

```
length(predicted)
```

```
## [1] 506
```

```
length(processed_data$high_medv)
```

```
## [1] 506
```

```
#acc<- mean(predicted== test$high_medv)
#acc

#cm <- table(test$high_medv, predicted)
cm <- table(processed_data$high_medv, predicted)
cm
```

```
##    predicted
##       0   1
##   0 192  64
##   1  31 219
```

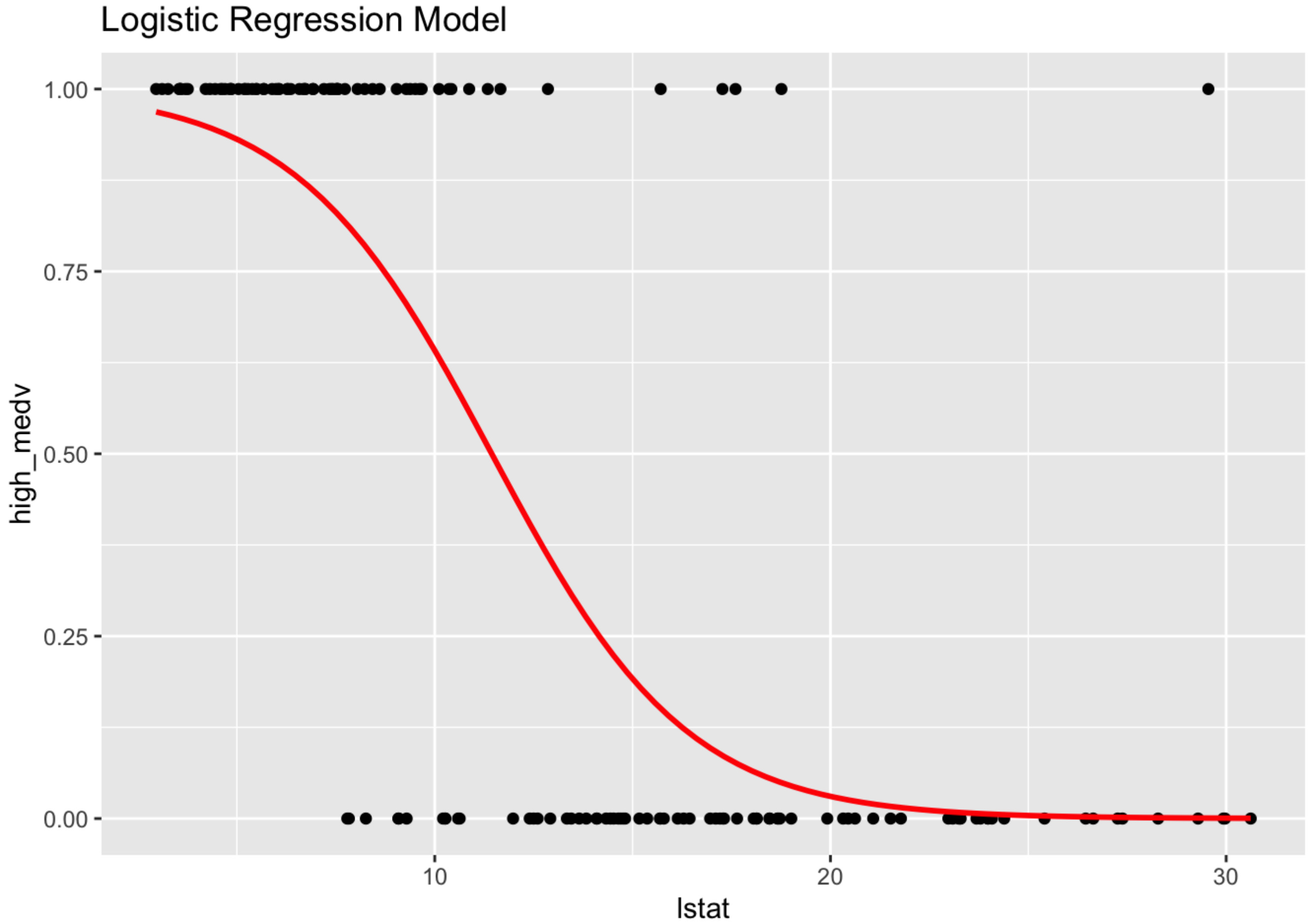
```
#confusionMatrix(processed_data$high_medv, predicted)
```

Plotting the logistic regression curve

```
library(ggplot2)

ggplot(data = test, aes(x = lstat, y = high_medv)) +
  geom_point() +
  stat_smooth(method = "glm", method.args = list(family = "binomial"), se = FALSE, co
lor = "red") +
  labs(title = "Logistic Regression Model", x = "lstat", y = "high_medv")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Conclusion: We can observe that the accuracy of the logistic model is 79% which is an acceptable one in terms of the data provided. The model can be further optimized with more number of dataset and applying proper data cleaning methods. From the significance of the model we can also see that the PClass attribute, SexMale and Age are the most significant predictors in this dataset and it can be inferred that persons with higher passenger class and female passengers were mostly survived in the Titanic crash.