Solutions Exercise #7

Logistic Regression with R

Pasquale De Rosa

03/05/2022

Preliminaries (Problem 1)

Load the "Cars.txt" dataset from the ILIAS website.

```r
cars <- read.table("Cars.txt", header = T)
summary(cars)
```

```
##       mpg          cylinders      displacement     horsepower        weight    
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613  
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.2   1st Qu.: 75.0   1st Qu.:2224  
##  Median :23.00   Median :4.000   Median :148.5   Median : 93.5   Median :2804  
##  Mean   :23.51   Mean   :5.455   Mean   :193.4   Mean   :104.5   Mean   :2970  
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0   3rd Qu.:126.0   3rd Qu.:3608  
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140  
##                                                  NA's   :6                     
##   acceleration        year           origin                   name    
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   ford pinto    :  6  
##  1st Qu.:13.82   1st Qu.:73.00   1st Qu.:1.000   amc matador   :  5  
##  Median :15.50   Median :76.00   Median :1.000   ford maverick :  5  
##  Mean   :15.57   Mean   :76.01   Mean   :1.573   toyota corolla:  5  
##  3rd Qu.:17.18   3rd Qu.:79.00   3rd Qu.:2.000   amc gremlin   :  4  
##  Max.   :24.80   Max.   :82.00   Max.   :3.000   amc hornet    :  4  
##                                                  (Other)       :369  
```

Problem 1: Consider the Cars dataset:

a. Build three different (generalized) linear regression models to predict mpg (at least one of them must be a multiple regression model).

Analyzing the dataset, we can assume that the model name is not useful if our aim is to predict the car system performance. Therefore, we can eliminate this variable.

```r
cars_new <- cars[, c("mpg", "cylinders", "displacement", "horsepower", "weight",
                     "acceleration", "year", "origin")]
summary(cars_new)
```

```
##       mpg          cylinders      displacement     horsepower        weight    
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613  
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.2   1st Qu.: 75.0   1st Qu.:2224  
##  Median :23.00   Median :4.000   Median :148.5   Median : 93.5   Median :2804  
##  Mean   :23.51   Mean   :5.455   Mean   :193.4   Mean   :104.5   Mean   :2970  
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0   3rd Qu.:126.0   3rd Qu.:3608  
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140  
##                                                  NA's   :6
##   acceleration        year           origin
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:13.82   1st Qu.:73.00   1st Qu.:1.000
##  Median :15.50   Median :76.00   Median :1.000
##  Mean   :15.57   Mean   :76.01   Mean   :1.573
##  3rd Qu.:17.18   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :24.80   Max.   :82.00   Max.   :3.000
##
```

Before proceeding with our analyses, we can standardize our values in order to have all the predictors in the same measurement scale and speed up the model training:

```
to_remove <- which(is.na(cars_new$horsepower))
cars_new <- cars_new[-to_remove, ]
means_cars <- lapply(cars_new, mean)
sd_cars <- lapply(cars_new, sd)
cars_standardized <- (cars_new - means_cars) / sd_cars
attach(cars_standardized)
```

We can select the two most important predictors to build the first two generalized regression models. We first choose the one that minimizes the residual sum of squares (RSS):

```
predictors_cars <- names(cars_standardized)
predictors_cars <- predictors_cars[predictors_cars != "mpg"]
n_predictors_cars <- length(predictors_cars)
RSS_cars <- numeric(n_predictors_cars)

for (i in 1:n_predictors_cars) {
  cars_lm <- lm(mpg ~ cars_standardized[predictors_cars][, i],
                data=cars_standardized)
  RSS_cars[i] <- sum(cars_lm$residuals^2)
}

predictors_cars[which(RSS_cars== min(RSS_cars))]
```

```
## [1] "weight"
```

```
min(RSS_cars)
```

```
## [1] 120.1815
```

So weight is the best predictor, that minimizes the residual sum of squares. We can then find the second best one:

```
predictors_cars <- names(cars_standardized)
predictors_cars <- predictors_cars[predictors_cars != "mpg"]
predictors_cars <- predictors_cars[predictors_cars != "weight"]
n_predictors_cars <- length(predictors_cars)
RSS_cars <- numeric(n_predictors_cars)

for (i in 1:n_predictors_cars) {
  cars_lm <- lm(mpg ~ cars_standardized[predictors_cars][, i],
                data=cars_standardized)
  RSS_cars[i] <- sum(cars_lm$residuals^2)
}

predictors_cars[which(RSS_cars== min(RSS_cars))]
```

```
## [1] "displacement"
```

```
min(RSS_cars)
```

```
## [1] 137.5423
```

We found that weight and displacement are the two variables that minimize the RSS. We proceed with building our two single regression models:

```
weight_glm <- glm(mpg ~ weight, data=cars_standardized)
summary(weight_glm)
```

```
## 
## Call:
## glm(formula = mpg ~ weight, data = cars_standardized)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -1.53409  -0.35305  -0.04303   0.27391   2.11651  
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.663e-16  2.804e-02    0.00        1    
## weight      -8.322e-01  2.807e-02  -29.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 0.3081577)
## 
##     Null deviance: 391.00  on 391  degrees of freedom
## Residual deviance: 120.18  on 390  degrees of freedom
## AIC: 655
## 
## Number of Fisher Scoring iterations: 2
```

```
displacement_glm <- glm(mpg ~ displacement, data=cars_standardized)
summary(displacement_glm)
```

```
## 
## Call:
## glm(formula = mpg ~ displacement, data = cars_standardized)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -1.65497  -0.38748  -0.06433   0.30124   2.38473  
## 
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)   5.584e-16  2.999e-02    0.00        1    
## displacement -8.051e-01  3.003e-02  -26.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 0.3526726)
## 
```

```
##     Null deviance: 391.00  on 391  degrees of freedom
## Residual deviance: 137.54  on 390  degrees of freedom
## AIC: 707.89
##
## Number of Fisher Scoring iterations: 2
```

We can see that both model are statistically significant. Let's create a third glm, this time multiple:

```
cars_multiple_glm <- glm(mpg ~ ., data=cars_standardized)
summary(cars_multiple_glm)
```

```
##
## Call:
## glm(formula = mpg ~ ., data = cars_standardized)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.22873  -0.27630  -0.01498   0.23946   1.67334
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.046e-15  2.153e-02   0.000  1.00000
## cylinders    -1.078e-01  7.065e-02  -1.526  0.12780
## displacement  2.667e-01  1.008e-01   2.647  0.00844 **
## horsepower   -8.360e-02  6.799e-02  -1.230  0.21963
## weight       -7.046e-01  7.096e-02  -9.929  < 2e-16 ***
## acceleration  2.848e-02  3.494e-02   0.815  0.41548
## year          3.543e-01  2.406e-02  14.729  < 2e-16 ***
## origin        1.472e-01  2.871e-02   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1817762)
##
##     Null deviance: 391.000  on 391  degrees of freedom
## Residual deviance:  69.802  on 384  degrees of freedom
## AIC: 454.01
##
## Number of Fisher Scoring iterations: 2
```

Let's remove the non-significant variables (cylinders, horsepower and acceleration:

```
cars_multiple_glm <- glm(mpg ~ displacement + weight + year + origin,
                         data=cars_standardized)
summary(cars_multiple_glm)
```

```
##
## Call:
## glm(formula = mpg ~ displacement + weight + year + origin, data = cars_standardized)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.25691  -0.27071  -0.00497   0.22710   1.69232
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.128e-15  2.165e-02   0.000    1.000    
## displacement  7.492e-02  6.393e-02   1.172    0.242    
## weight       -7.156e-01  6.063e-02 -11.802  < 2e-16 ***
## year          3.641e-01  2.351e-02  15.486  < 2e-16 ***
## origin        1.265e-01  2.755e-02   4.593 5.92e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 0.1837824)
## 
##     Null deviance: 391.000  on 391  degrees of freedom
## Residual deviance:  71.124  on 387  degrees of freedom
## AIC: 455.37
## 
## Number of Fisher Scoring iterations: 2
```

Now the displacement is not significant anymore. We will remove it to create our final model:

```r
cars_multiple_glm <- glm(mpg ~ weight + year + origin, data=cars_standardized)
summary(cars_multiple_glm)
```

```
## 
## Call:
## glm(formula = mpg ~ weight + year + origin, data = cars_standardized)
## 
## Deviance Residuals: 
##      Min        1Q    Median        3Q       Max  
## -1.27406  -0.26839  -0.00499   0.22108   1.70047  
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  1.135e-15  2.166e-02   0.000        1    
## weight      -6.523e-01  2.765e-02 -23.588  < 2e-16 ***
## year         3.573e-01  2.281e-02  15.668  < 2e-16 ***
## origin       1.187e-01  2.674e-02   4.439 1.18e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 0.1839593)
## 
##     Null deviance: 391.000  on 391  degrees of freedom
## Residual deviance:  71.376  on 388  degrees of freedom
## AIC: 454.76
## 
## Number of Fisher Scoring iterations: 2
```

b. Perform 10-fold cross validation to estimate the test error of the models you built in a).

We can easily compare the three models performing a 10-fold cv using the cv.glm() function from the library boot:

```r
library(boot)
set.seed(123)
weight_cv <- cv.glm(cars_standardized, weight_glm, K=10)
weight_cv$delta[1]
```

```
## [1] 0.3091372
```

```
displacement_cv <- cv.glm(cars_standardized, displacement_glm, K=10)
displacement_cv$delta[1]
```

```
## [1] 0.3539645
```

```
cars_multiple_cv <- cv.glm(cars_standardized, cars_multiple_glm, K=10)
cars_multiple_cv$delta[1]
```

```
## [1] 0.186302
```

We see that the multiple regression model is the best performing one with an average RMSE of ≈ 0.18.

Preliminaries (Problem 2)

Load the "Cancer.txt" dataset from the ILIAS website.

```
cancer <- read.table("Cancer.txt", header = T)
summary(cancer)
```

```
##        ID             Diagnostic     Radius          Texture
##  Min.   :     8670   B:357      Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   M:212      1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024              Median :13.370   Median :18.84
##  Mean   : 30371831              Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129              3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502              Max.   :28.110   Max.   :39.28
##    Perimeter           Area            Smooth           Compact
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##    Concavity          Concave           Symmetry         Fractal
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02956   1st Qu.:0.02031   1st Qu.:0.1619   1st Qu.:0.05770
##  Median :0.06154   Median :0.03350   Median :0.1792   Median :0.06154
##  Mean   :0.08880   Mean   :0.04892   Mean   :0.1812   Mean   :0.06280
##  3rd Qu.:0.13070   3rd Qu.:0.07400   3rd Qu.:0.1957   3rd Qu.:0.06612
##  Max.   :0.42680   Max.   :0.20120   Max.   :0.3040   Max.   :0.09744
##     RadiusSE        TextureSE       PerimeterSE         AreaSE
##  Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
##  1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
##  Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
##  Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
##  3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
##  Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
##     SmoothSE          CompactSE         ConcavitySE        ConcaveSE
##  Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
##  Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930
##  Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
##  3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
##  Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
##    SymmetrySE         FractalSE           RadiusMax       TextureMax
##  Min.   :0.007882   Min.   :0.0008948   Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480   1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870   Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949   Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580   3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400   Max.   :36.04   Max.   :49.54
##   PerimeterMax       AreaMax          SmoothMax         CompactMax
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
##  Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##   ConcavityMax      ConcaveMax        SymmetryMax       FractalMax
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1145   1st Qu.:0.06493   1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993   Median :0.2822   Median :0.08004
##  Mean   :0.2722   Mean   :0.11461   Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140   3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100   Max.   :0.6638   Max.   :0.20750
```

Problem 2: Apply the logistic regression to predict the category diagnosis and interpret the most important values of the model that you obtained. Can you estimate the error rate of your model?

We can get rid of the ID variable that cannot be used as a model predictor:

```
cancer_new <- cancer[,-1]
attach(cancer_new)
```

Let's create our logistic model to predict the variable Diagnostic using glm() with family=binomial as an argument:

```
cancer_logr <- glm(Diagnostic ~ ., data=cancer_new, family=binomial)
summary(cancer_logr)
```

```
##
## Call:
## glm(formula = Diagnostic ~ ., family = binomial, data = cancer_new)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -8.49   -8.49   -8.49    8.49    8.49
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.881e+06  2.816e+05 -10.233  < 2e-16 ***
## Radius        2.427e+06  2.693e+05   9.014  < 2e-16 ***
## Texture       1.958e+05  1.471e+04  13.313  < 2e-16 ***
## Perimeter     1.473e+06  2.464e+04  59.791  < 2e-16 ***
## Area         -1.301e+05  3.907e+03 -33.301  < 2e-16 ***
## Smooth       -1.525e+08  8.361e+06 -18.234  < 2e-16 ***
## Compact      -6.428e+06  3.213e+06  -2.001  0.04539 *
## Concavity     1.042e+06  1.408e+06   0.740  0.45959
## Concave      -1.716e+07  5.382e+06  -3.188  0.00143 **
## Symmetry      4.049e+07  7.772e+05  52.093  < 2e-16 ***
## Fractal      -4.233e+07  2.169e+06 -19.519  < 2e-16 ***
## RadiusSE      3.328e+07  1.169e+06  28.478  < 2e-16 ***
## TextureSE     6.368e+06  2.005e+05  31.763  < 2e-16 ***
## PerimeterSE   1.701e+06  4.720e+04  36.032  < 2e-16 ***
## AreaSE       -6.393e+05  1.835e+04 -34.840  < 2e-16 ***
## SmoothSE      7.492e+08  1.224e+07  61.213  < 2e-16 ***
## CompactSE    -1.773e+08  5.732e+06 -30.931  < 2e-16 ***
## ConcavitySE   1.529e+08  5.340e+06  28.624  < 2e-16 ***
## ConcaveSE    -1.260e+09  4.012e+07 -31.398  < 2e-16 ***
## SymmetrySE    2.890e+08  4.126e+06  70.055  < 2e-16 ***
## FractalSE     1.512e+09  6.597e+07  22.921  < 2e-16 ***
## RadiusMax    -6.130e+06  2.143e+05 -28.606  < 2e-16 ***
## TextureMax   -5.832e+05  2.437e+04 -23.935  < 2e-16 ***
## PerimeterMax -3.538e+05  1.219e+04 -29.023  < 2e-16 ***
## AreaMax       8.950e+04  2.741e+03  32.658  < 2e-16 ***
## SmoothMax    -2.161e+07  3.298e+06  -6.553 5.66e-11 ***
## CompactMax    8.986e+06  3.999e+05  22.470  < 2e-16 ***
## ConcavityMax -3.028e+07  1.523e+06 -19.875  < 2e-16 ***
## ConcaveMax    1.431e+08  5.471e+06  26.162  < 2e-16 ***
## SymmetryMax  -2.474e+07  3.392e+05 -72.923  < 2e-16 ***
## FractalMax   -3.698e+07  5.340e+06  -6.926 4.33e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:   751.44  on 568  degrees of freedom
## Residual deviance: 32006.76  on 538  degrees of freedom
## AIC: 32069
##
## Number of Fisher Scoring iterations: 25
```

As we can see, the Concavity is not significant. We can then remove it and re-create our model:

```
cancer_new <- cancer_new[,-8]
cancer_logr <- glm(Diagnostic ~ ., data=cancer_new, family=binomial)
summary(cancer_logr)
```

```
##
## Call:
## glm(formula = Diagnostic ~ ., family = binomial, data = cancer_new)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -8.49    0.00    0.00    0.00    8.49
##
## Coefficients:
##               Estimate Std. Error   z value Pr(>|z|)
## (Intercept)  -1.217e+14  1.213e+08  -1003344   <2e-16 ***
## Radius       -2.927e+15  4.760e+07 -61502056   <2e-16 ***
## Texture       6.820e+13  2.257e+06  30219323   <2e-16 ***
## Perimeter     1.702e+14  6.892e+06  24690108   <2e-16 ***
## Area          1.624e+13  1.491e+05 108946317   <2e-16 ***
## Smooth        1.746e+16  5.657e+08  30867575   <2e-16 ***
## Compact      -1.771e+16  3.776e+08 -46912470   <2e-16 ***
## Concave       2.429e+16  4.045e+08  60040200   <2e-16 ***
## Symmetry     -7.072e+15  2.109e+08 -33536226   <2e-16 ***
## Fractal      -7.459e+15  1.580e+09  -4721581   <2e-16 ***
## RadiusSE      6.869e+14  8.812e+07   7794767   <2e-16 ***
## TextureSE     2.922e+12  1.046e+07    279437   <2e-16 ***
## PerimeterSE  -4.643e+14  1.163e+07 -39917322   <2e-16 ***
## AreaSE        3.466e+13  3.945e+05  87868453   <2e-16 ***
## SmoothSE     -1.701e+16  1.877e+09  -9062133   <2e-16 ***
## CompactSE     3.244e+16  6.099e+08  53194503   <2e-16 ***
## ConcavitySE  -2.676e+16  3.234e+08 -82738584   <2e-16 ***
## ConcaveSE     1.175e+17  1.534e+09  76621241   <2e-16 ***
## SymmetrySE   -2.541e+16  7.748e+08 -32789711   <2e-16 ***
## FractalSE    -2.677e+17  3.314e+09 -80763694   <2e-16 ***
## RadiusMax     7.154e+14  1.647e+07  43440178   <2e-16 ***
## TextureMax   -6.198e+12  1.974e+06  -3139500   <2e-16 ***
## PerimeterMax  3.523e+13  1.686e+06  20891171   <2e-16 ***
## AreaMax      -5.185e+12  9.069e+04 -57177752   <2e-16 ***
## SmoothMax     7.875e+14  4.076e+08   1932247   <2e-16 ***
## CompactMax   -2.923e+15  1.086e+08 -26910623   <2e-16 ***
## ConcavityMax  3.729e+15  6.372e+07  58528461   <2e-16 ***
## ConcaveMax   -5.255e+15  2.541e+08 -20685733   <2e-16 ***
## SymmetryMax   6.714e+15  1.404e+08  47822073   <2e-16 ***
## FractalMax    2.888e+16  6.748e+08  42795823   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 751.44  on 568  degrees of freedom
## Residual deviance: 865.05  on 539  degrees of freedom
## AIC: 925.05
##
## Number of Fisher Scoring iterations: 25
```

We can observe that the computation was done after 25 iterations. The coefficients are now all statistically significant, with values either positive or negative (if $\beta_n > 0$, for higher values of x_n increases the probability that $y = 1$, and vice-versa). To estimate the error rate of our model, let's recall the formula of the (multiple) logistic regression:

$\Pi = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n)}}$

Where Π is the probability of y to belong to a binary class and assumes values between [0, 1]. Since the output is binary, the error for each value of y will be:

$$\epsilon = \begin{cases} -\pi, & \text{if } y = 0 \\ 1-\pi, & \text{if } y = 1 \end{cases}$$

Using the cv.glm() function, we can perform the 10-fold cross-validation and estimate the error rate (averaged on the 10 folds) of our logistic model:

```
cancer_cv <- cv.glm(cancer_new, cancer_logr, K=10)
cancer_cv$delta[1]
```

```
## [1] 0.05448154
```

So our model has an average error rate on the 10 folds of $\approx 5\%$, that is fairly low. We can then consider this logistic model as a good predictor for the cancer diagnosis.