Simple Linear Regression an Logistic Regression

EBRU KILIÇ

Last compiled on 5 November 2022

Contents

1 Part 1: Simple Linear Regression **1**

2 Part 1: Solution **1**

3 Part 2: Logistic Regression **8**

4 Part 2: Solution **9**

5 References **12**

1 Part 1: Simple Linear Regression

Exam grades and weekly spent time on self study x (in hours) of 14 statistics students are given in the following table.

Self study	25.0	26.2	24.9	23.7	22.8	24.6	23.6	23.0	22.5	26.2	25.8	24.0	22.1	21.7
Exam Grades	63	53	52	46	34	47	43	37	40	45	53	42	32	49

1. Create a data frame in `R` with the above data. Plot the data with the weekly spent time on self study in the x-axis and exam grades on the y-axis (You should include labels for your axes and a title for the plot)
2. Obtain the least squares regression line of exam grades on weekly spent time on self study. Interpret your model result (Using the whole data set)
3. Fit the linear model after partitioning your data set into training and testing (round the number of observations when it is necessary). After fitting the model, compare your parameter estimates with the model result in Question 2. Then, make predictions on testing data and compare with the original observations.
4. Using the `plot` command, comment on the validity of the assumption of the model that you fit in Question 3 (Note before using the `plot` command you may wish to specify a 2x2 graphics window using `par(mfrow = c(2, 2))`).
5. Calculate a 95% confidence interval for the slope regression parameter for the last model you fit in Question 3. (Note that the number of degrees of freedom should be obtained from the `R` output). For this you can use a built-in function in R

2 Part 1: Solution

```
#install.packages("modelr")
#install.packages("tinytex")
#tinytex::install_tinytex()
#install.packages("mlbench")
#install.packages("caret")
#install.packages("ggplot2")
#install.packages("lattice")

# Last 5 digits of "14449202680"
# and par() method are used because of the reproducibility and better plots.

set.seed(02680)
par(mfrow =c(2, 3))
```

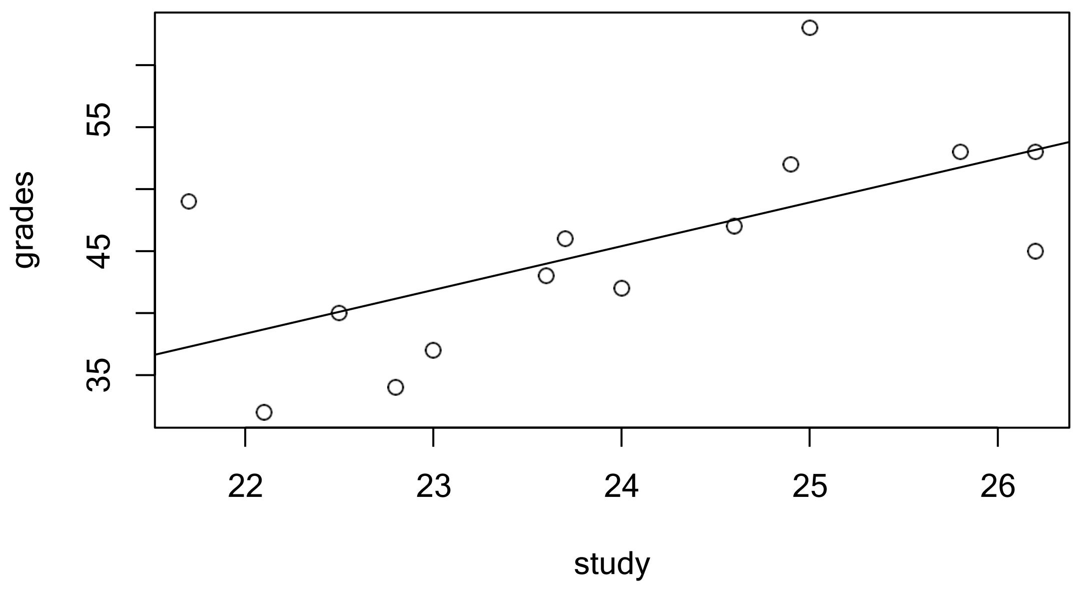
```
# Data set on the given pdf file
datas <- data.frame(
  study = c(25.0,26.2,24.9,23.7,22.8,24.6,23.6,23.0,22.5,26.2,25.8,24.0,22.1,21.7),
  grades = c(63,53,52,46,34,47,43,37,40,45,53,42,32,49))

# All data"s regression
allreggrades <- lm(grades ~ study, data = datas)

# Control the outliners. There is no problem since it is not outliner.
z_score  <-  as.data.frame(sapply(datas,  function(datas)
  (abs(datas-mean(datas))/sd(datas))))
dim(datas)
```

```
## [1] 14  2
```

```
except_outliers <- z_score[!rowSums(3 < z_score), ]
dim(except_outliers)
```

```
## [1] 14  2
```

```
# Plot of Study and grades
# least squares regression line
# plot() and abline() has used
plot(datas)
abline(allreggrades)
```



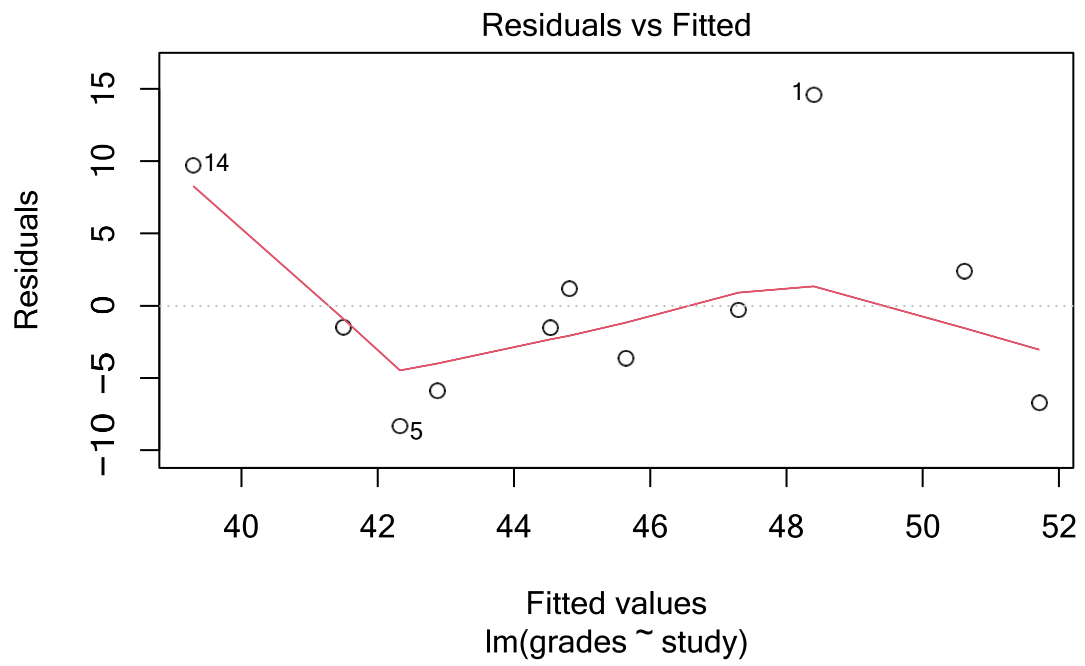
```
# Data partitioning as %80.
index <- sample(seq_len(nrow(datas)), round(8/10*nrow(datas)))
length(index)
```

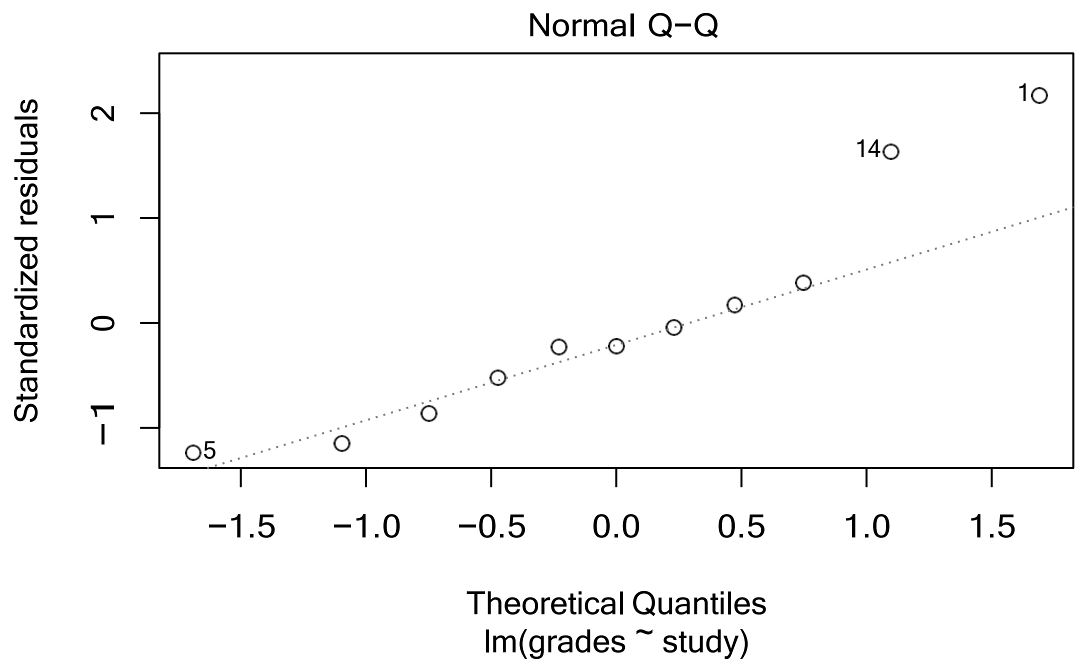
```
## [1] 11
```

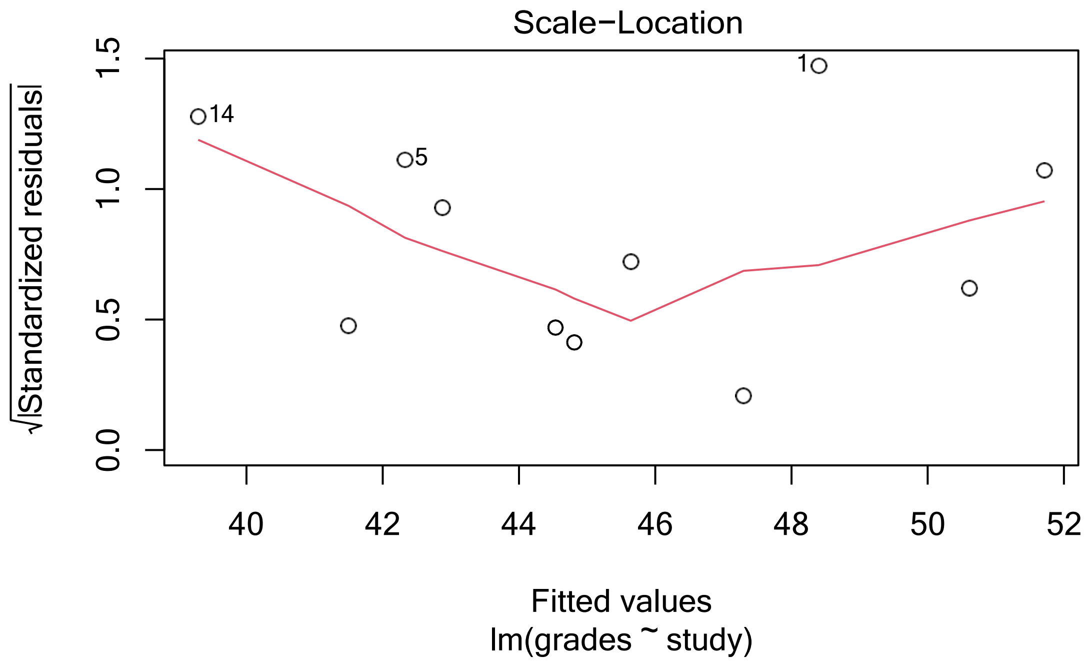
```
# Training data
train = datas[index, ]
# Test data
test <- datas[-index, ]

### FITTING THE MODEL
regressionDatas <- lm(grades ~ study, data = train)

# the P-Value < 0.05
# it is more meaningful since it has got lower predictor value then p-value
summary(regressionDatas)
```

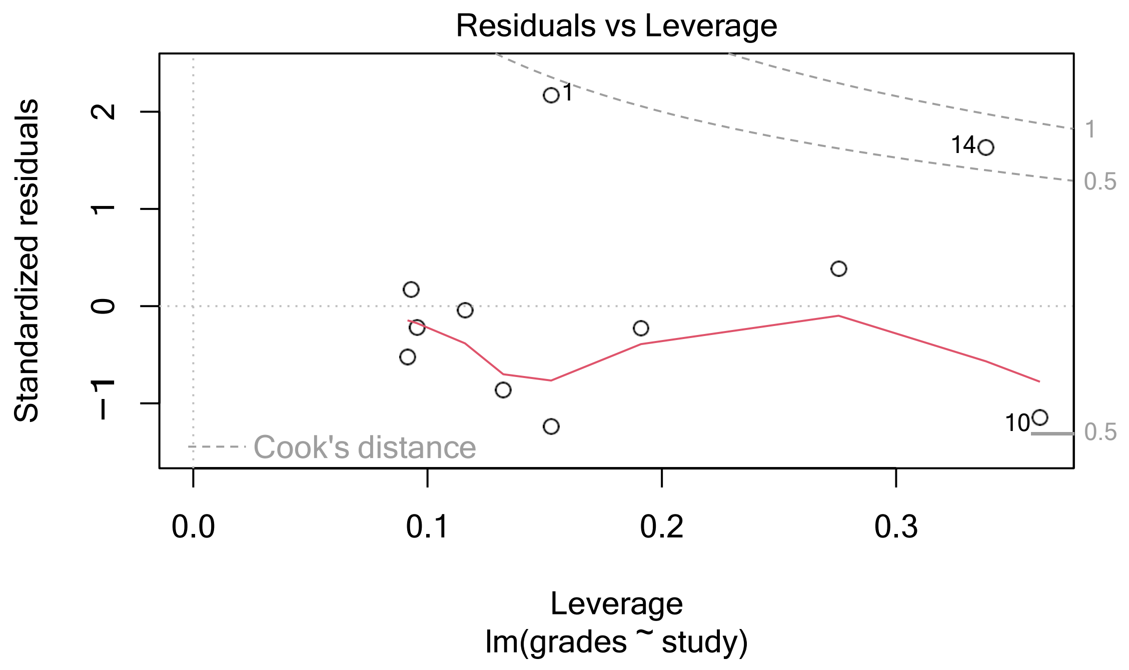
```
##
## Call:
## lm(formula = grades ~ study, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.327 -4.759 -1.499  1.790 14.600
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -20.618     39.587  -0.521    0.615
## study          2.761      1.654   1.669    0.129
##
## Residual standard error: 7.314 on 9 degrees of freedom
## Multiple R-squared:  0.2364, Adjusted R-squared:  0.1516
## F-statistic: 2.787 on 1 and 9 DF,  p-value: 0.1294
```

```
plot(regressionDatas)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(grades ~ study)
1
14
5
−1
0
1
2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Scale−Location
√|Standardized residuals|
Fitted values
lm(grades ~ study)
1
14
5
0.0
0.5
1.0
1.5
40
42
44
46
48
50
52



```
# confidence interval = 95%
# slope regression parameter for the model.
confint(regressionDatas, level = 95/100)
```

```
##                   2.5 %   97.5 %
## (Intercept) -110.1694047 68.93349
## study         -0.9803677  6.50184
```

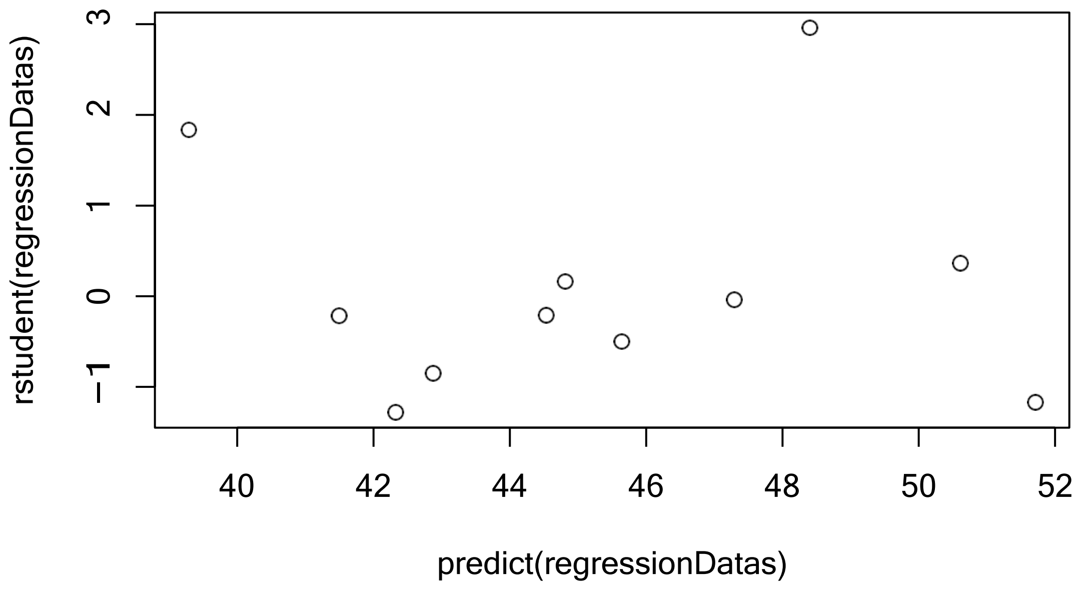
```
# Shows the predictions on testing data.
# It shows the differences between predicted values
# While we are making prediction
predict(regressionDatas, train)
```

```
##        8        1        6       14        5       12       11        4
## 42.87897 48.40045 47.29615 39.29002 42.32683 45.63971 50.60904 44.81149
##       10        9        7
## 51.71333 41.49861 44.53542
```

```
train$grades
```

```
##  [1] 37 63 47 49 34 42 53 46 45 40 43
```

```
# Shows the residuals that coming from linear regression.
plot(predict(regressionDatas), rstudent(regressionDatas))
```



```
# Model's performance
library(modelr)

summary(regressionDatas)
```

```
##
## Call:
## lm(formula = grades ~ study, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.327 -4.759 -1.499  1.790 14.600
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -20.618     39.587  -0.521    0.615
## study          2.761      1.654   1.669    0.129
##
## Residual standard error: 7.314 on 9 degrees of freedom
## Multiple R-squared:  0.2364, Adjusted R-squared:  0.1516
## F-statistic: 2.787 on 1 and 9 DF,  p-value: 0.1294
```

```
data.frame(
  #Root Mean Square Error
  RMSE = rmse(regressionDatas, data = train),
  #Mean Absolute Error
  MAE = mae(regressionDatas, data = train), AIC(regressionDatas),
  BIC(regressionDatas)
)
```

```
##      RMSE     MAE AIC.regressionDatas. BIC.regressionDatas.
## 1 6.61586 5.07073             78.78498             79.97867
```

```
# Relationship‴s strength between the dependent variable on a convenient
# and my model is measuring by r-squared.
```

3 Part 2: Logistic Regression

Consider the available example data set below

```
# Last 5 digits of "14449202680"
# and par() method are used because of the reproducibility and better plots.
set.seed(02680)
par(mfrow =c(2, 3))

# install.packages("mlbench")
library(mlbench)
data(BreastCancer)
summary(BreastCancer)
```

```
##       Id              Cl.thickness   Cell.size     Cell.shape  Marg.adhesion
##  Length:699         1      :145   1      :384   1      :353   1      :407
##  Class :character   5      :130   10     : 67   2      : 59   2      : 58
##  Mode  :character   3      :108   3      : 52   10     : 58   3      : 58
##                     4      : 80   2      : 45   3      : 56   10     : 55
##                     10     : 69   4      : 40   4      : 44   4      : 33
##                     2      : 50   5      : 30   5      : 34   8      : 25
##                     (Other):117   (Other): 81   (Other): 95   (Other): 63
##   Epith.c.size  Bare.nuclei   Bl.cromatin  Normal.nucleoli    Mitoses
##  2      :386   1      :402   2      :166   1      :443     1      :579
##  3      : 72   10     :132   3      :165   10     : 61     2      : 35
##  4      : 48   2      : 30   1      :152   3      : 44     3      : 33
##  1      : 47   5      : 30   7      : 73   2      : 36     10     : 14
##  6      : 41   3      : 28   4      : 40   8      : 24     4      : 12
##  5      : 39   (Other): 61   5      : 34   6      : 22     7      :  9
##  (Other): 66   NA's   : 16   (Other): 69   (Other): 69     (Other): 17
##        Class
##  benign   :458
##  malignant:241
##
##
##
##
##
```

```
# You can check the details here
#       https://www.rdocumentation.org/packages/mlbench/versions/2.1-3/topics/BreastCancer
```

1. Convert your Class variable into a numerical one since you have two classes (benign malignant) you can make it one of them as 0 and the other one is 1
2. Fit a logistic regression model to classify **Class** using Mitoses (DO NOT FORGET TO PARTITION YOUR DATA INTO TRAINING AND TESTING DATA SETS, DO NOT FORGET THAT THIS DATA SET INCLUDES QUALITATIVE PREDICTORS !)
3. Make predictions and compare with the true observations (using TEST DATA SET). Calculate and intepret the Confusion Matrix results
4. Fit a multiple logistic regression to classify **Class** by using more than one predictor
5. Compare simple logistic and multiple logistic regression models using F1-score to make a decision on the best model. Why the overall accuracy is not enough as a performance measure ? Explain shortly

4 Part 2: Solution

Use the given R-code chunk below to make your calculations and summarize your result thereafter by adding comments on it,

- MAKE SURE THAT ALL NECESSARY PACKAGES ARE ALREADY INSTALLED and READY TO USE
- You can use as many as Rcode chunks you want. In the final output, both Rcodes and your ouputs including your comments should appear in an order

```
# You can check the details here
#       https://www.rdocumentation.org/packages/mlbench/versions/2.1-3/topics/BreastCancer

# numerical variable has comprised from class variable
# "malignant" = 1 , "benign" = 0

vectorIndex = c()
for(m in 1:length(BreastCancer$Class))
{
  if("malignant" == BreastCancer$Class[m])
  {
    vectorIndex[m] = 1;
  }
  else if ("benign" == BreastCancer$Class[m])
  {
    vectorIndex[m] = 0;
  }
}

# NewBreastCancer is the new fixed data
NewBreastCancer = BreastCancer
NewBreastCancer$Class = vectorIndex

# Data partitioning as %80.
traIndex <- sample(seq_len(nrow(NewBreastCancer)),
                   round(8/10*nrow(NewBreastCancer)))
length(traIndex)
```

```
## [1] 559
```

```
# BreastCancer's train data
TrainBreastCancer = NewBreastCancer[traIndex, ]
# BreastCancer's test data
TestBreastCancer <- NewBreastCancer[-traIndex, ]
# model fitting
FitBreastCancer = glm(Class ~ Mitoses, data = TrainBreastCancer,
                      family = "binomial")
summary(FitBreastCancer)
```

```
##
## Call:
## glm(formula = Class ~ Mitoses, family = "binomial", data = TrainBreastCancer)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2649  -0.7491  -0.7491   0.4001   1.6781
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.1274     0.1078 -10.462  < 2e-16 ***
## Mitoses2       2.4267     0.4730   5.130 2.89e-07 ***
## Mitoses3       3.6123     0.7438   4.856 1.20e-06 ***
## Mitoses4      18.6935  1398.7210   0.013    0.989
## Mitoses5      18.6935  1769.2576   0.011    0.992
## Mitoses6      18.6935  2284.1018   0.008    0.993
## Mitoses7      18.6935  1615.1039   0.012    0.991
## Mitoses8      18.6935  1615.1039   0.012    0.991
## Mitoses10     18.6935  1192.8333   0.016    0.987
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 727.90  on 558  degrees of freedom
## Residual deviance: 561.73  on 550  degrees of freedom
## AIC: 579.73
##
## Number of Fisher Scoring iterations: 16
```

```
# Fitting model's Prediction part
FitPreBreastCancer = ifelse(predict(FitBreastCancer, type = "response")
                            > 0.5, "1", "0")
# comparison
table(TrainBreastCancer$Class)
```

```
##
##   0   1
## 360 199
```

```r
table(FitPreBreastCancer)
```

```
## FitPreBreastCancer
##   0   1
## 466  93
```

```r
# error rate  of  training
# mean error rate
error = function(actual, predicted)
{
  mean(predicted != actual)
}
error(actual = TrainBreastCancer$Class, predicted = FitPreBreastCancer)
```

```
## [1] 0.2182469
```

```r
FitPreBreastCancer = ifelse(predict(FitBreastCancer, type = "response",
                                    newdata = TestBreastCancer)
                            > 0.5, "1", "0")
error(actual = TestBreastCancer$Class, predicted = FitPreBreastCancer)
```

```
## [1] 0.1785714
```

```r
# Calculate  the  Confusion  Matrix  on  training  data

library(caret)
```

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

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

```r
# Calculating  and  intepreting  the  Confusion  Matrix  results.
confusionMatrix(as.factor(FitPreBreastCancer),
                as.factor(TestBreastCancer$Class), mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  0  1
##          0 93 20
##          1  5 22
##
##                Accuracy : 0.8214
##                  95% CI : (0.7478, 0.881)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.0007454
##
##                   Kappa : 0.5265
##
##  Mcnemar's Test P-Value : 0.0051103
```

```
##
##             Sensitivity : 0.9490
##             Specificity : 0.5238
##          Pos Pred Value : 0.8230
##          Neg Pred Value : 0.8148
##               Precision : 0.8230
##                  Recall : 0.9490
##                      F1 : 0.8815
##              Prevalence : 0.7000
##          Detection Rate : 0.6643
##    Detection Prevalence : 0.8071
##       Balanced Accuracy : 0.7364
##
##        'Positive' Class : 0
##
```

5 References

https://www.r-project.org/ https://cran.rstudio.com/ https://www.reddit.com/r/programming/ https://www.w3schools.com/r/