# | | Homework 1: Simple Linear Regression an Logistic Regression

<Furkan ÖZELGE (14758028780)>

Last compiled on 31 Mart, 2022

## Table of Contents

## 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

## Part 1: 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

```
# FOR REPRODUCIBILITY
set.seed(28780)
# ALERT: YOU NEED TO USE YOUR STUDENT NUMBER LAST 5 DIGITS
# HERE instead of 442 MAKE SURE THAT YOU CHANGED
# BEFORE STARTING TO YOUR ANALYSIS
selfStudy <- 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)
examGrades <- c(63, 53, 52, 46, 34, 47, 43, 37, 40, 45, 53, 42, 32, 49)
df <- data.frame(selfStudy, examGrades)
# hist()
plot(df$selfStudy,df$examGrades)

# sample mean of x
xbar <- mean( df$selfStudy )

# sample mean of y
ybar <- mean( df$examGrades)

# denumerator for beta1 estimate
tempXX <- sum( (df$selfStudy- xbar)^2 )

# numerator for beta1 estimate
tempXY <- sum( (df$selfStudy- xbar) * (df$examGrades - ybar) )

# estimate of beta1, of slope
beta1_hat <- tempXY / tempXX
beta1_hat

## [1] 3.531271

# estimate of beta0, of intercept
beta0_hat <- ybar - beta1_hat * xbar
beta0_hat

## [1] -39.34715

# Our linear fit has the form
# mpg \approx beta0_hat + beta1_hat * EG
```

```r
# For EG = 25.7

EG = beta0_hat + beta1_hat * 25.7
EG
```

```
## [1] 51.40651
```

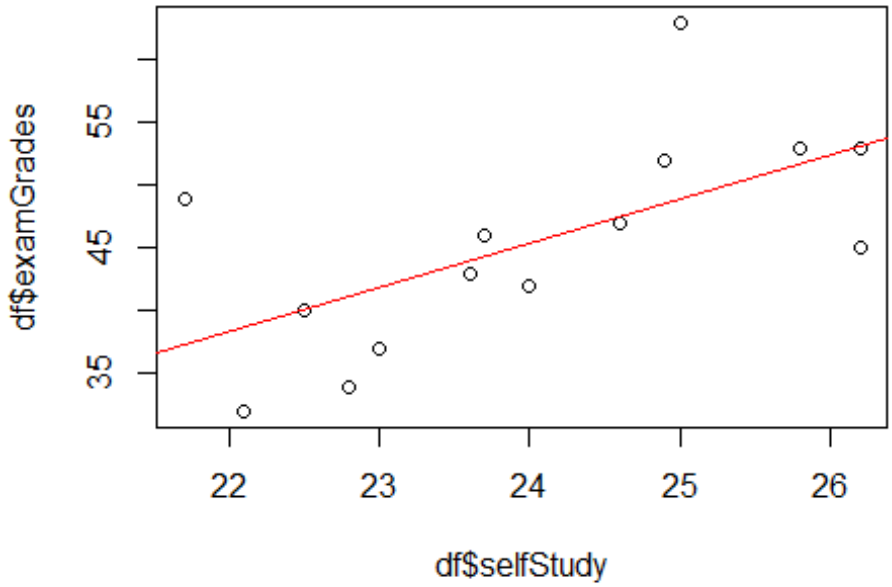
```r
# Use of lm function ----------------------------------------------------

# Use of lm function
EG_model <- lm(formula = examGrades ~ selfStudy, data = df)

EG_model
```

```
##
## Call:
## lm(formula = examGrades ~ selfStudy, data = df)
##
## Coefficients:
## (Intercept)    selfStudy
##     -39.347        3.531
```

```r
# For more information
summary(EG_model)
```

```
##
## Call:
## lm(formula = examGrades ~ selfStudy, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1721 -4.5049 -0.3471  1.5521 14.0654
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -39.347     30.291  -1.299   0.2184
## selfStudy      3.531      1.259   2.804   0.0159 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.774 on 12 degrees of freedom
## Multiple R-squared:  0.3958, Adjusted R-squared:  0.3455
## F-statistic: 7.861 on 1 and 12 DF,  p-value: 0.01593
```

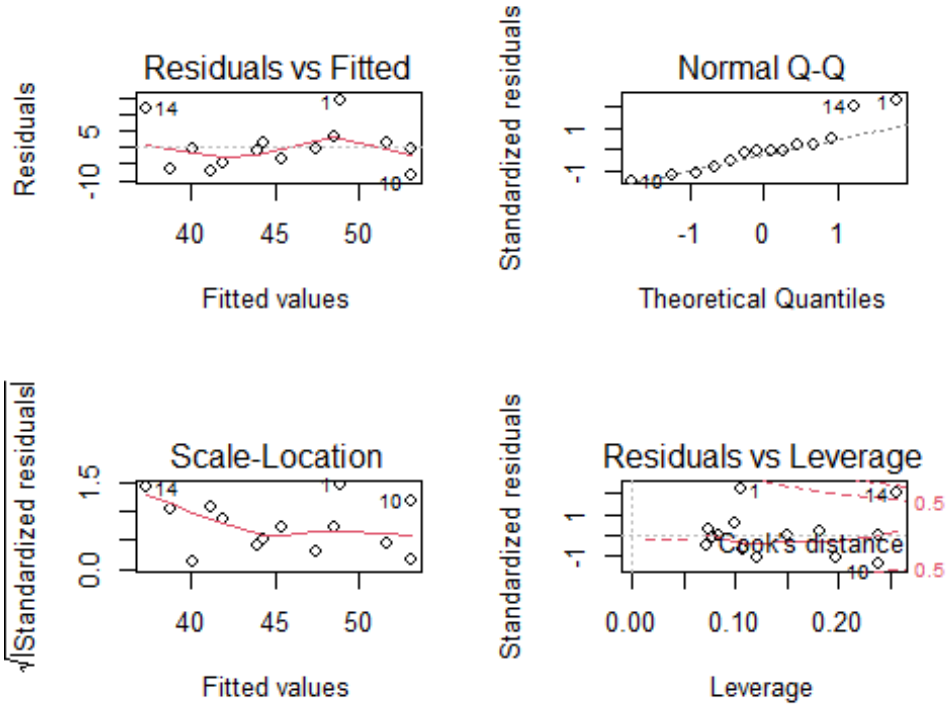
```r
abline(EG_model, col = "red")
```




```r
# ALWAYS START with DATA PARTITIONING
# %80 for training, %20 for testing
# MY STUDENT NUMBER 14758028780
set.seed(28780)
sample.size <- floor(0.80 * nrow(df))
train.index <- sample(seq_len(nrow(df)), size = sample.size)

# Partitioning on training and testing
train <- df[train.index, ]
test <- df[-train.index, ]

# MODEL BUILDING
# Simple linear regression
lm.fit = lm(formula = examGrades ~ selfStudy, data = df)
summary(lm.fit)

##
## Call:
## lm(formula = examGrades ~ selfStudy, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.1721 -4.5049 -0.3471  1.5521 14.0654
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -39.347     30.291  -1.299   0.2184
```

```
## selfStudy      3.531      1.259   2.804   0.0159 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.774 on 12 degrees of freedom
## Multiple R-squared:  0.3958, Adjusted R-squared:  0.3455
## F-statistic: 7.861 on 1 and 12 DF,  p-value: 0.01593
```

```
# Model diagnostic part
par(mfrow = c(2,2))
plot(lm.fit)
```




```
# PREDICTION : Use the testing data
class(lm.fit)
```

```
## [1] "lm"
```

```
predic_EG <- predict(lm.fit, newdata = test)
head(predic_EG)
```

```
##        2        6        8
## 53.17214 47.52211 41.87208
```

```
head(test$examGrades)
```

```
## [1] 53 47 37
```

```
cor(predic_EG, test$examGrades)^2
```

```
## [1] 0.9795918

confint(EG_model , level=0.95)

##                     2.5 %    97.5 %
## (Intercept) -105.3449302 26.650626
## selfStudy      0.7870966  6.275445
```

1.2) We got the least squares regression line with EG_model and abline method.. . Some variables are out of our line and messy, but we can still ignore that and still work with it. We can ignore vertical distance. Only the horizontal distance concerns us. 1.3) Our guesses aren't too bad in this. 1.4) Here, linearity is fine. In our changing variance, the residuals are well and homogeneously all over the place. 1.5)Here you may notice that our forecasts were not good.

## Part 2: Logistic Regression

Consider the available example data set below

```
# 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

## 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

```
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
##
##
##
##
##
```

```
View(BreastCancer)
BreastCancer$Class = factor(BreastCancer$Class, labels = c("0","1"))

# We should start with Data Partitioning
set.seed(28780)
default_idx <- sample(nrow(BreastCancer), 0.80 * nrow(BreastCancer))
# I am declaring training data set
trainingDataSet <- BreastCancer[default_idx, ]
dim(trainingDataSet)
```

```
## [1] 559  11
```

```
# table(trainingDataSet$default)
# I am declaring testing data set
default_tst <- BreastCancer[-default_idx, ]
dim(default_tst)
```

```
## [1] 140  11
```

```
model_glm = glm(formula = Class ~ Mitoses, data = trainingDataSet, family =
"binomial")
summary(model_glm)
```

```
##
## Call:
## glm(formula = Class ~ Mitoses, family = "binomial", data =
trainingDataSet)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2649  -0.6931  -0.6931   0.4001   1.7573
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.3039     0.1139 -11.447  < 2e-16 ***
## Mitoses2       2.4935     0.4464   5.585 2.33e-08 ***
## Mitoses3       3.7888     0.7447   5.087 3.63e-07 ***
## Mitoses4      18.8700  1251.0541   0.015  0.98797
```

```
## Mitoses5       2.6902     1.1238   2.394  0.01667 *
## Mitoses6      18.8700  2797.4419   0.007  0.99462
## Mitoses7       3.0957     1.0861   2.850  0.00437 **
## Mitoses8      18.8700  1495.2956   0.013  0.98993
## Mitoses10     18.8700  1097.2470   0.017  0.98628
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 712.54  on 558  degrees of freedom
## Residual deviance: 533.49  on 550  degrees of freedom
## AIC: 551.49
##
## Number of Fisher Scoring iterations: 16
```

```
coef(model_glm)
```

```
## (Intercept)    Mitoses2    Mitoses3    Mitoses4    Mitoses5    Mitoses6
##   -1.303910    2.493495    3.788817   18.869979    2.690205   18.869979
##    Mitoses7    Mitoses8   Mitoses10
##    3.095670   18.869979   18.869979
```

```
# Prediction part for the fitted model
# Result of logistic link function by default
head(predict(model_glm))
```

```
##         9       423       453       665       141       675
##  1.386294 -1.303910 -1.303910 -1.303910 -1.303910 -1.303910
```

```
# Now we can get probabilities by changing the type as response
head(predict(model_glm, type = "response"))
```

```
##         9       423       453       665       141       675
## 0.8000000 0.2135076 0.2135076 0.2135076 0.2135076 0.2135076
```

```
#  these are not predicted probabilities. To obtain the predicted
probabilities
head(predict(model_glm, type = "response", newdata = default_tst))
```

```
##         8        11        13        16        28        35
## 0.2135076 0.2135076 0.2135076 0.2135076 0.2135076 0.2135076
```

```
# Note that these are probabilities, not classifications.
# To obtain classifications, we will need to compare to the correct cutoff
value with an ifelse() statement.
# I set it to 0.5 because there are only 2 possibilities. 1 or 0 so we make
0.5.
predict(model_glm, type = "response") > 0.5
```

```
##     9   423   453   665   141   675   113   228   431   320   154   609
457
```

```
##  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
FALSE
##   584   318   195   183    79    80   451   103   655   632   518   225
544
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
FALSE
##    19   326   319   561   587   405   216   364   332   218   255   271
222
##  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE
FALSE
##   330   166   444   268   522   668    82   193   353   230    95   545
426
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##   243   275   179   158   526   534   652   461   487   148   598   558
285
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
TRUE
##   628   410   403   121   564   358   185   334   344   400   630   528
276
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##   479   560   106   499   439   421   547   264   118   641   624   360
658
## FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
FALSE
##   110   379   203   282    25   503   639   682   654   111    56   463
261
##  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
TRUE
##   168    81   324    17   623   554   670   540   478   309   322   186
181
##  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
FALSE
##     6   391   519   684   100   346   520   696   191   413   251   610
244
## FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##    41   486   156   601   325   163    84   176   138   160   142   177
428
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
FALSE
##   386    71   469    88   301   430    46   190   642   672     2   673
134
##  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
FALSE
##   475    74    42   664   196   143   622   667   357   226   685   202
449
## FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE
FALSE
```

```
##   207    75    43   406   349   583   393   124    51   383    54    72
155
## FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE
FALSE
##   514   637   678   589   376   659   328   676   687   369   594   350
491
## FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##   548   136   333   621   299   288   116     3   509   694   201    77
208
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
FALSE
##   112   620   397   532   303   399   568   151   605   231   235   422
382
## FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
TRUE
##   221   669   171   227   477   683   173   338   512   342   575    91
198
## FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##    89   556   445    49   210   490   602   305   651   538   108    66
618
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
FALSE
##    24   331   287   649   607   258   505   555    86   304   697   274
153
## FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
FALSE
##   390   539   640   470   128   573   135   511    92   674   259   644
359
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
TRUE
##   356   335   256   460   677   159   529   123   435   281   546     5
597
## FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
FALSE
##   559    55   570   293   498   147   115   280   109   585   355   626
365
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
FALSE
##   262   599   671   139    62   593   476   352   407    10   507    12
101
##  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
TRUE
##   290   698   648   164    20   563   456   686   117   446   557   566
577
##  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
FALSE
##   603   104   242   224   317   206     4   272   465   691   371   381
581
```

```
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##   310     7   199   582    34    15    40   270   125   336   157   471
370
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##   329   679    27    26   165   424   250   144   129   340   616   323
643
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
FALSE
##   517   214   377   302   692   152     1   283    65    52   634   314
388
## FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##    14   681   629   363   443   119    23   254   279   571   473   590
354
## FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
TRUE
##    93   467    78   523   591   680   150   579   447   562   361   306
316
## FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
FALSE
##   440   462    69    38    73   417   482   448   552    33   257   402
660
## FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
FALSE
##   688   411   137   666   481   483   592   161   619   219    47   172
31
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE
FALSE
##   543   693   617   245   398   657   615   297   184   102    96    37
416
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##   502   162   277   636   178   246   600   468    29   525    18   530
321
## FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
TRUE
##   595   437   515   432   367   273   345   631   394   269   542    30
663
## FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
FALSE
##   412   613   550   130   213    97   366   368   401   500   194   237
212
## FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
FALSE
##   107    32   551    87    59   414   114   253   395   633   278   374
611
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
TRUE
```

```
##   419   569   294   531   232   145   233   169   375   638   187   466
337
##  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
FALSE
##   484   492   373   420   438   252   536    63   362   127   596   348
662
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE
FALSE
##   408   533   298   389    67   576   442   170   209   341   635   513
236
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##    99   105    90   149    53   188   427   220   295   180   695   260
249
##  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
##   572   464   454   496   549   452    22   387    50   567    21   241
396
##  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
FALSE
```

```
# model_glm_pred = ifelse(predict(model_glm, type = "link") > 0, 1, 0)
model_glm_pred = ifelse(predict(model_glm, type = "response") > 0.5, 1, 0)

head(model_glm_pred)
```

```
##   9 423 453 665 141 675
##   1   0   0   0   0   0
```

```
table(model_glm_pred)
```

```
## model_glm_pred
##   0   1
## 459 100
```

```
# To make a small comparison
# Coming from the original values in training data
table(trainingDataSet$Class)
```

```
##
##   0   1
## 372 187
```

```
# Coming from the predicted values over the training data
table(model_glm_pred)
```

```
## model_glm_pred
##   0   1
## 459 100
```

```
# Training error rate
# For the calculation of basically mean error rate
```

```r
calc_class_err = function(actual, predicted)
{
  mean(actual != predicted)
}

calc_class_err(actual = trainingDataSet$Class, predicted = model_glm_pred)
```

```
## [1] 0.1949911
```

```r
# You can do a similar thing for testing, you must do indeed !
model_glm_pred = ifelse(predict(model_glm, type = "response",
                                newdata = default_tst) > 0.5, 1, 0)

head(model_glm_pred)
```

```
##  8 11 13 16 28 35
##  0  0  0  0  0  0
```

```r
length(model_glm_pred)
```

```
## [1] 140
```

```r
length(predict(model_glm, type = "response",
               newdata = default_tst))
```

```
## [1] 140
```

```r
calc_class_err(actual = default_tst$Class, predicted = model_glm_pred)
```

```
## [1] 0.2714286
```

```r
# Calculation of Confusion Matrix on training data
library(caret)
```

```
## Zorunlu paket yükleniyor: ggplot2
```

```
## Zorunlu paket yükleniyor: lattice
```

```r
# Training data set
# train_tab = table(predicted = model_glm_pred, actual =
trainingDataSet$default)
# train_tab
# train_con_mat = confusionMatrix(train_tab, positive = 1)
# train_con_mat

# Testing data set


test_tab = table(predicted = model_glm_pred, actual = default_tst$Class)
test_tab
```

```
##          actual
## predicted  0  1
```

```
##         0 84 36
##         1  2 18
```

```
help("confusionMatrix")
```

```
## starting httpd help server ...
```

```
##  done
```

```
test_con_mat = confusionMatrix(test_tab, mode = "everything", positive = "1")
test_con_mat
```

```
## Confusion Matrix and Statistics
##
##          actual
## predicted  0  1
##         0 84 36
##         1  2 18
##
##                Accuracy : 0.7286
##                  95% CI : (0.647, 0.8002)
##     No Information Rate : 0.6143
##     P-Value [Acc > NIR] : 0.003015
##
##                   Kappa : 0.3512
##
##  Mcnemar's Test P-Value : 8.636e-08
##
##             Sensitivity : 0.3333
##             Specificity : 0.9767
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.7000
##               Precision : 0.9000
##                  Recall : 0.3333
##                      F1 : 0.4865
##              Prevalence : 0.3857
##          Detection Rate : 0.1286
##    Detection Prevalence : 0.1429
##       Balanced Accuracy : 0.6550
##
##        'Positive' Class : 1
##
```

```
# Multiple Logistic Regression Case ------------------------------------
```

```
# Fitting the model on training data set, with balance and student as
predictors
model_glm = glm(formula = Class ~ Mitoses + Cell.shape , data =
trainingDataSet, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(model_glm)
```

```
##
## Call:
## glm(formula = Class ~ Mitoses + Cell.shape, family = "binomial",
##     data = trainingDataSet)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.33843   0.00000   0.00000   0.00003   2.87814
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.0969  2282.9258   0.000  0.99962
## Mitoses2         3.9181     1.2161   3.222  0.00127 **
## Mitoses3         2.4282     0.9757   2.489  0.01282 *
## Mitoses4        23.9853 11753.7154   0.002  0.99837
## Mitoses5        18.0737  2626.1776   0.007  0.99451
## Mitoses6        22.5062 27084.9518   0.001  0.99934
## Mitoses7        17.5853  3080.5888   0.006  0.99545
## Mitoses8        19.9724 16986.9920   0.001  0.99906
## Mitoses10       24.3183 10379.2669   0.002  0.99813
## Cell.shape.L    39.3413  9065.5600   0.004  0.99654
## Cell.shape.Q    -5.6649  5271.3983  -0.001  0.99914
## Cell.shape.C    19.8466  4576.8554   0.004  0.99654
## Cell.shape^4   -11.2691  9242.8838  -0.001  0.99903
## Cell.shape^5    -0.3857 10991.4865   0.000  0.99997
## Cell.shape^6    -9.2175  9335.5577  -0.001  0.99921
## Cell.shape^7    -2.5847  5987.9796   0.000  0.99966
## Cell.shape^8    -2.1317  2845.8457  -0.001  0.99940
## Cell.shape^9    -0.8215   887.0501  -0.001  0.99926
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 712.54  on 558  degrees of freedom
## Residual deviance: 146.79  on 541  degrees of freedom
## AIC: 182.79
##
## Number of Fisher Scoring iterations: 21
```

```
# prediction on testing data
model_glm_pred_mult = ifelse(predict(model_glm, newdata = default_tst,
                                     type = "response") > 0.5, 1, 0)

head(model_glm_pred_mult)
```

```
##  8 11 13 16 28 35
##  0  0  0  1  0  0
```

```r
# Confusion matrix on testing
testing_tab_mult = table(predicted = model_glm_pred_mult, actual =
default_tst$Class)
testing_con_mat_mult = confusionMatrix(testing_tab_mult, positive = "1")

print(testing_con_mat_mult)
```

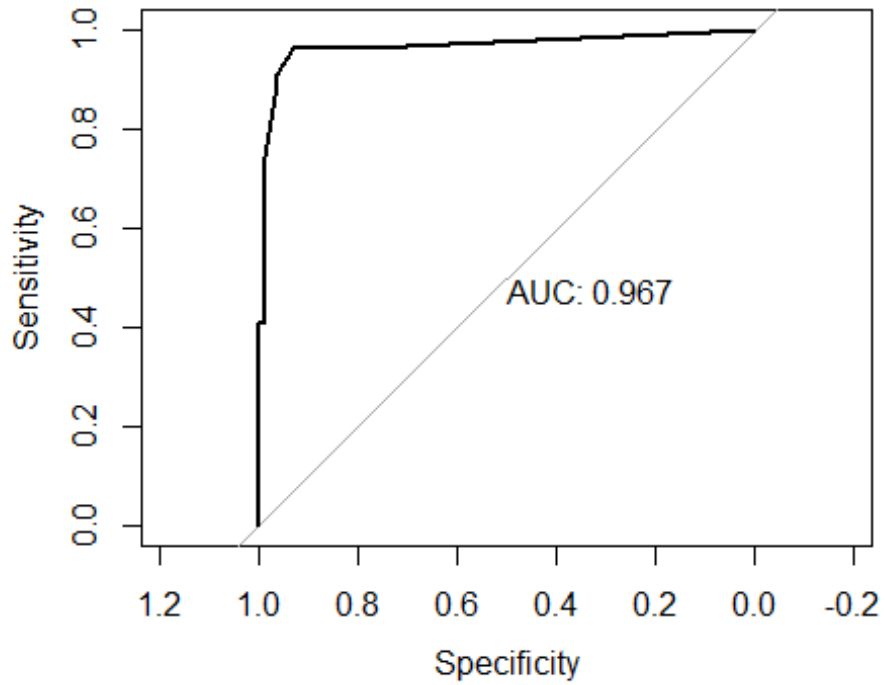
```
## Confusion Matrix and Statistics
##
##          actual
## predicted  0  1
##         0 83  6
##         1  3 48
##
##                Accuracy : 0.9357
##                  95% CI : (0.8815, 0.9702)
##     No Information Rate : 0.6143
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8629
##
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.8889
##             Specificity : 0.9651
##          Pos Pred Value : 0.9412
##          Neg Pred Value : 0.9326
##              Prevalence : 0.3857
##          Detection Rate : 0.3429
##    Detection Prevalence : 0.3643
##       Balanced Accuracy : 0.9270
##
##        'Positive' Class : 1
##
```

```r
testing_con_mat_mult$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8888889            0.9651163            0.9411765
##       Neg Pred Value            Precision               Recall
##            0.9325843            0.9411765            0.8888889
##                   F1           Prevalence       Detection Rate
##            0.9142857            0.3857143            0.3428571
## Detection Prevalence    Balanced Accuracy
##            0.3642857            0.9270026
```

```r
# For graphical interpretation one can use ROC curve
# For multiple logistic regression example

library(pROC)
```

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

##
## Attaching package: 'pROC'

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

# Predictions on testing data
test_prob_mult = predict(model_glm, newdata = default_tst, type = "response")
# Drawing ROC curve for the given model
roc

## function (...)
## {
##     UseMethod("roc")
## }
## <bytecode: 0x0000000028ee6ed8>
## <environment: namespace:pROC>

test_roc_mult = roc(default_tst$Class ~ test_prob_mult, plot = T,
                    print.auc = T)

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

## Setting direction: controls < cases
```



```
test_roc_mult

##
## Call:
## roc.formula(formula = default_tst$Class ~ test_prob_mult, plot = T,
print.auc = T)
##
## Data: test_prob_mult in 86 controls (default_tst$Class 0) < 54 cases
(default_tst$Class 1).
## Area under the curve: 0.9667

# The value of AUC (Area under the curve), high values are indicators of good
model !
as.numeric(test_roc_mult$auc)

## [1] 0.9667313

# 2-3) Our false negative (00) value is too high. Our positive true value is
less. For this reason, we could not get very accurate results.
# 2-5)
```

Here you can notice that our model is quite successful. 2.5) We should go with our first model to reduce the error rate. As we moved from the first model to the second model, our F1 value increased at a high rate. Or another option is we can go with our second model but we have to add more variables.

## References

Give a list of the available sources that you use while preparing your home-work (If you use other resources, you can make a list here for checking & reproducibility).

For instance;


- https://www.statlearning.com/
- https://lms.tedu.edu.tr/pluginfile.php/102130/mod_resource/content/1/LogisticRegression.R
- https://towardsdatascience.com/the-f1-score-bec2bbc38aa6
- https://stackoverflow.com/