Homework 1

Pattern Mining and Social Network Analysis

BOUYSSOU Gatien , de POURTALES Caroline, LAMBA Ankit

15 octobre, 2020

Classification

Overall

Supervised learning

Classification algorithms have categorical responses. In classification we build a function f(X) that takes a vector of input variables X and predicts its class membership, such that Y in C.

Possibilities of models

There are classifiers as logistic regression, Decision tree, Perceptron / Neural networks, K-nearest-neighbors, linear and quadratic logistic regression, Bayes …

Some indicators

Sensitivity and recall

The sensitivity (also named recall) is the percentage of true defaulters that are identified (True positive tests). For example, probability of predicting disease given true state is disease.

$$sensitivity = recall = \frac{TruePositiveTests}{PositivePopulation}$$

Specificity

The specificity is the percentage of non-defaulters that are correctly identified (True negative tests). 1 - specificity is the Type 1 error, it is the false positive rate. For example, probability of predicting non-disease given true state is non- disease.

$$specificity = \frac{TrueNegativeTests}{NegativePopulation}$$

Precision

The precision is the proportion of true positive tests among the positive tests.

$$precision = \frac{TruePositiveTests}{PositiveTests}$$

F-Mesure

The traditional F measure is calculated as follows:

$$F_Measure = \frac{(2 * Precision * Recall)}{(Precision + Recall)}$$

Rand index

The rand index is a mesure of similarity between two partitions from a single set.

Given two partitions π_1 and π_2 in E :

- a, the number of elements in π_1 and π_2
- b, the number of elements in π_1 and not in π_2
- c, the number of elements in π_2 and not in π_1
- d, the number of elements not in both π_1 and π_2

	in π_2	not in π_2
in π_1	a	b
not in π_1	c	d

$$RI(\pi_1, \pi_2) = \frac{a + d}{a + b + c + d}$$

Logistic Regression

How it works

In logistic regression, for covariates (X__1 , . . . , X__p), we want to estimate $p_i = P_r(Y_i = 1|X_1, ..., X_p)$

$$p_i = \frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}+...}}{1 + e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\beta_3 x_{i3}+\beta_4 x_{i4}+...}}$$

To come back to linear regression we define the logistic function as follow.

$$logit(p_i) = log(\frac{p_i}{1 - p_i}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \beta_4 x_{i4} + ...$$

We can define the odds :

$$\frac{odds(Y_i = 1|X1 = x_{i1} + 1)}{odds(Y_i = 1|X1 = x_{i1})} = e^{\beta_1}$$

Which indicator for validity ?

We use Maximum Likehood :

$$L(\beta) = \Pi_{i=1}^{n} p_i^{y_i} * (1 - p_i)^{y_i}$$

The goal is to maximise it by adjusting β vector.

An example in R

We use a dataset from the Wimbledon tennis tournament for Women in 2013. We will predict the result for player 1 (win=1 or loose=0) based on the number of aces won by each player and the number of unforced errors commited by both players. The data set is a subset of a data set from https://archive.ics.uci.edu/ml/datasets/Tennis+Major+Tournament+Match+Statistics.

```
id <- "1GNbIhjdhuwPOBr0Qz82JMkdjUVBuSoZd"
tennis <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",id), header = T)

# test and train set
n = dim(tennis)[1]
n2 = n*(3/4)
set.seed(1234)
train = sample(c(1:n), replace = F)[1:n2]

# reduction to two variables
tennis$ACEdiff = tennis$ACE.1 - tennis$ACE.2
tennis$UFEdiff = tennis$UFE.1 - tennis$UFE.2
head(tennis)
```

```
##         Player1      Player2 Result ACE.1 UFE.1 ACE.2 UFE.2 ACEdiff UFEdiff
## 1     M.Koehler   V.Azarenka      0     2    18     3    14      -1       4
## 2    E.Baltacha   F.Pennetta      0     0    10     4    14      -4      -4
## 3      S-W.Hsieh       T.Maria      1     1    13     2    29      -1     -16
## 4       A.Cornet        V.King      1     4    30     0    45       4     -15
## 5  Y.Putintseva    K.Flipkens      0     2    28     6    19      -4       9
## 6 A.Tomljanovic B.Jovanovski      0     6    42    11    40      -5       2
```

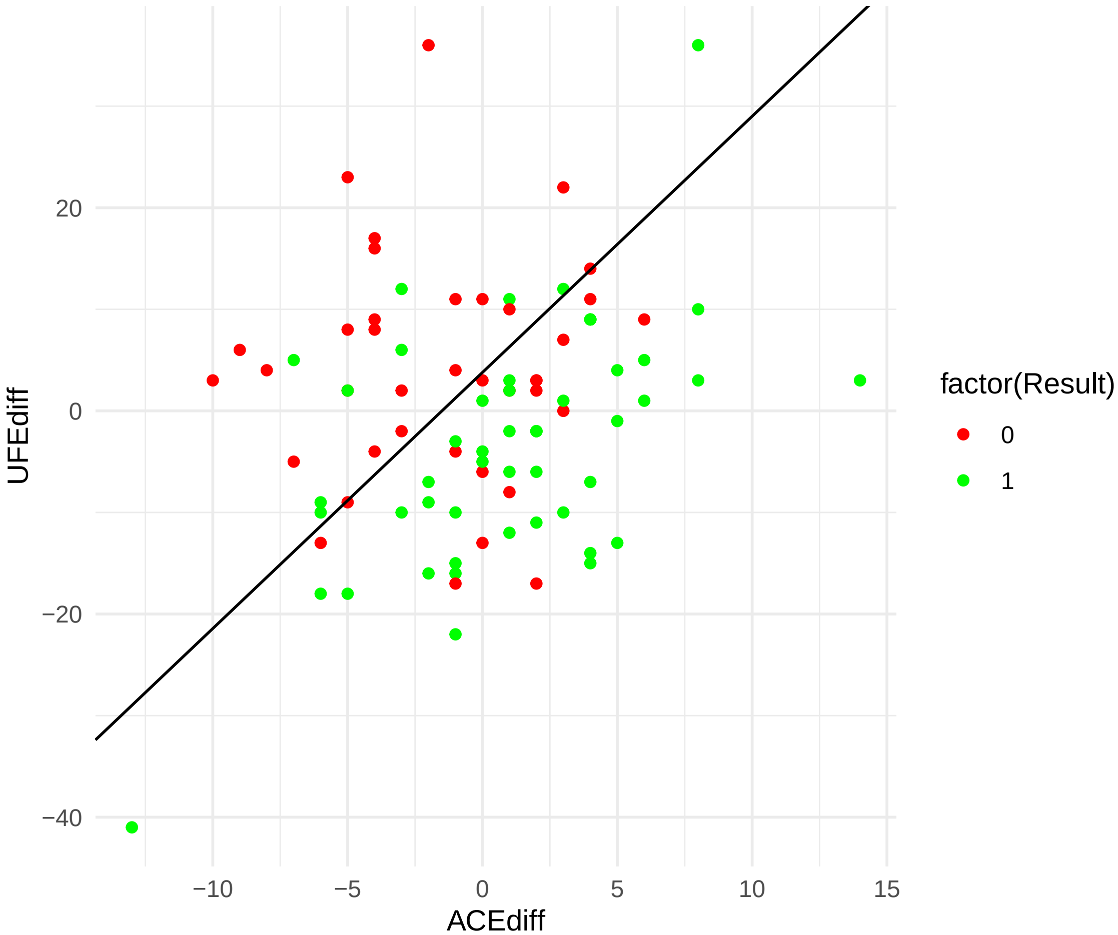
```
tennisTest = tennis[-train, ]
tennisTrain = tennis[train, ]
r.tennis2 = glm(Result ~ ACEdiff + UFEdiff, data = tennisTrain, family = "binomial")
summary(r.tennis2)
```

```
##
## Call:
## glm(formula = Result ~ ACEdiff + UFEdiff, family = "binomial",
##     data = tennisTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1204  -0.9994   0.5662   0.8918   1.8714
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.31318    0.24439   1.281  0.20004
## ACEdiff      0.20856    0.06575   3.172  0.00151 **
## UFEdiff     -0.08272    0.02454  -3.371  0.00075 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 120.352  on 87  degrees of freedom
## Residual deviance:  99.102  on 85  degrees of freedom
## AIC: 105.1
```

```
##
## Number of Fisher Scoring iterations: 4
```

```
#We calculate the slope
glm.b = -r.tennis2$coefficients[2]/r.tennis2$coefficients[3]
glm.a = -r.tennis2$coefficients[1]/r.tennis2$coefficients[3]

ggplot() + geom_point(aes(ACEdiff, UFEdiff, color = factor(Result)), data = tennisTrain, ) + scale_colo
  geom_abline(slope = glm.b, intercept = glm.a) +
  theme_minimal()
```



We can write :

$$logit(p_i) = log(\frac{p_i}{1-p_i}) = 0,31318 + 0,20856 * ACEDiff - 0,08272 * UFEDiff$$

We can observe AIC = 105.1

The confusion matrix is :

```
glm.Result_probs = predict(r.tennis2, newdata = tennisTest)
glm.Result_pred = ifelse(glm.Result_probs > 0.5, 1, 0)
glm.confusion_matrix = table(glm.Result_pred, tennisTest$Result)
glm.confusion_matrix
```

```
##
## glm.Result_pred  0  1
```

```
##                     0 15  5
##                     1  2  8
```

The accuracy rate is $\frac{17+25}{13+25+4+17} = 0.71$.

The sensitivity is the percentage of true output giving Player1-winner among the population of true Player1-winner :

```
glm.sensitivity = glm.confusion_matrix[2,2]/(glm.confusion_matrix[1,2] + glm.confusion_matrix[2,2])
glm.sensitivity
```

```
## [1] 0.6153846
```

The specificity is the percentage of true output giving Player2-winner (= Player1-looser) among the population of true Player2-winner:

```
glm.specificity = glm.confusion_matrix[1,1]/(glm.confusion_matrix[1,1] + glm.confusion_matrix[2,1])
glm.specificity
```

```
## [1] 0.8823529
```

The precision is the percentage of true output giving Player1-winner among all the outputs giving Player1-winner (even if not winner) :

```
glm.precision = glm.confusion_matrix[2,2]/(glm.confusion_matrix[2,1] + glm.confusion_matrix[2,2])
glm.precision
```

```
## [1] 0.8
```

So the F_Mesure is :

```
glm.fmesure = (2*glm.precision*glm.sensitivity)/(glm.sensitivity + glm.precision)
glm.fmesure
```

```
## [1] 0.6956522
```

Implémenter une ou deux classification(s) de plus entre :

An other example

Regression

Overall

Supervised learning

TO DO

Possibilities of models

TO DO

The accuracy of a model

The Mean Squarred error

The MSE mesures the mean accuracy of the predicted responses values for given observations. There are two MSE : the train MSE and the test MSE. \ The train MSE is use to fit a model while training. \ The test MSE is use to choose between models already trained. \

Let's define the mean squared error or MSE.

$$MSE = \frac{1}{n} \sum_{i} (y_i - \hat{f}(x_i))^2$$

Then the expected test MSE refers to the average test MSE that we would obtain if we repeatedly estimated f using a large number of training sets, and tested each at x_0. So that the expected test MSE is :

$$E(y_0 - \hat{f}(x_0))^2$$

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + (f(x_0) - E(\hat{f}(x_0)))^2 + Var(\varepsilon)$$

$Var(\varepsilon)$ represents the irreductible error. This term can not be reduced regardless how well our statstical model fits the data.

$(f(x_0) - E(\hat{f}(x_0))^2 = [Bias(\hat{f}(x_0))]^2$ is the squared Bias and refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. If the bias is low the model gives a prediction which is close to the true value.

$Var(\hat{f}(x_0))$ is the Variance of the prediction at $\hat{f}(x_0)$ and refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set. If the variance is high, there is a large uncertainty associated with the prediction.

RSS : residual sum of squares

We define the residual sum of squares (RSS) as :

$$RSS = \Sigma(y_i - \hat{y}_i)^2$$

We want to minimize the RSS.

RSE : residual standard error

$$RSE = \sqrt{\frac{1}{n-2} RSS}$$

R statistic

$$R^2 = 1 - \frac{RSS}{TSS}$$

$$TSS = \Sigma(y_i - \bar{y}_i)^2$$

is the total sum of squares. TSS measures the total variance in the response Y.

TSS − RSS measures the amount of variability in the response that is explained.

R^2 measures the proportion of variability in Y that can be explained using X.

F statistic

TO - DO

Simple Linear Regression

Definition

TO DO

DEFINITION

WHICH INDICATORS CAN WE USE

Simple linear regression lives up to its name: it is a very straightforward approach for predicting a quantitative response Y on the basis of a single predictor variable X. It assumes that there is approximately a linear relationship between X and Y. Mathematically, we can write this linear relationship as

$$Y \approx \beta_0 + \beta_1 * X$$

An example in R

The next dataset (source F. E. Harrell, Regression Modeling Strategies) contains the total hospital costs of 9105 patients with certain diseases in American hospitals between 1989 and 1991. The different variables are :

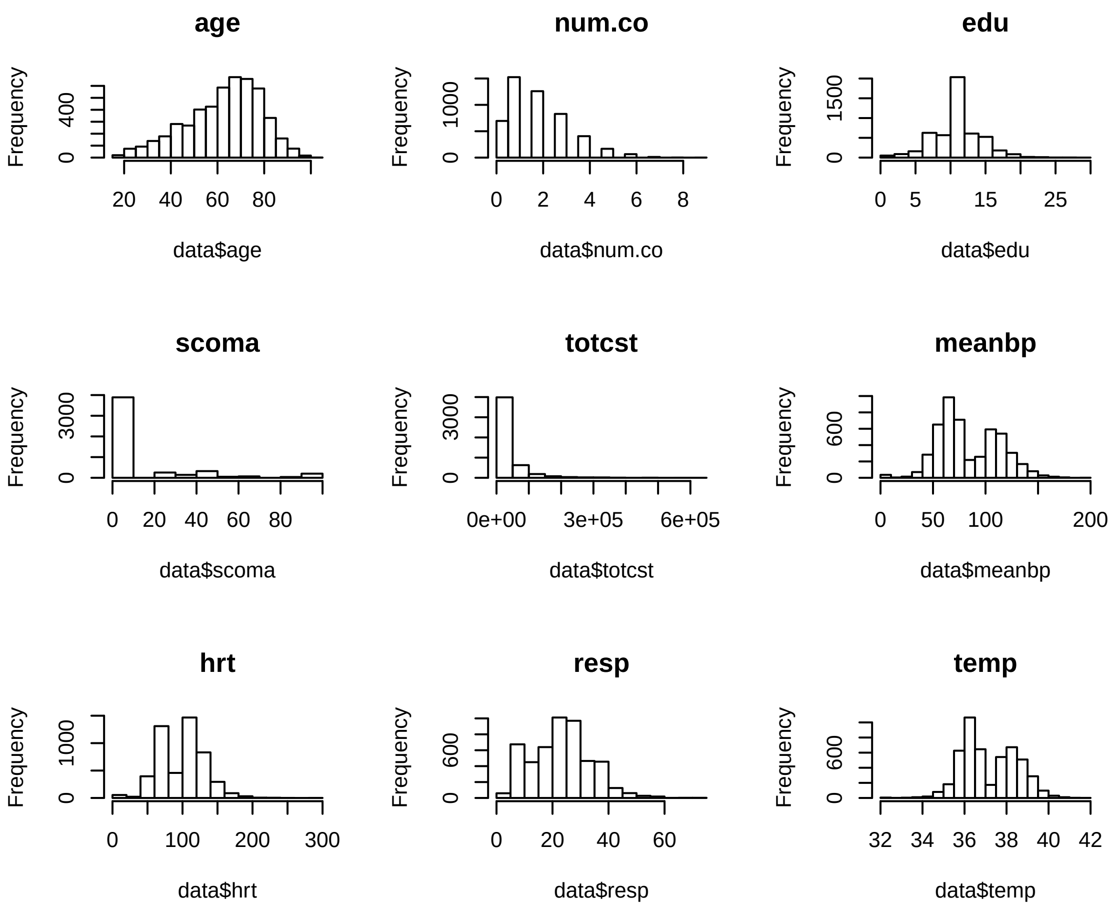
```r
id <- "1heRtzi8vBoBGMaM2-ivBQI5Ki3HgJTmO" # google file ID
data <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",  id), header = T)
head(data)
```

```
##     age           dzgroup num.co edu     income scoma totcst  race meanbp hrt
## 1 62.85       Lung Cancer      0  11   $11-$25k     0     NA other     97  69
## 2 60.34         Cirrhosis      2  12   $11-$25k    44     NA white     43 112
## 3 52.75         Cirrhosis      2  12 under $11k     0     NA white     70  88
## 4 42.38       Lung Cancer      2  11 under $11k     0     NA white     75  88
## 5 79.88 ARF/MOSF w/Sepsis      1  NA               26     NA white     59 112
## 6 93.02              Coma      1  14               55     NA white    110 101
##   resp  temp   pafi
## 1   22 36.00 388.00
## 2   34 34.59  98.00
## 3   28 37.40 231.66
## 4   32 35.00     NA
## 5   20 37.90 173.31
## 6   44 38.40 266.63
```

We would like to build models that help us to understand which predictors are mostly driving the total cost.

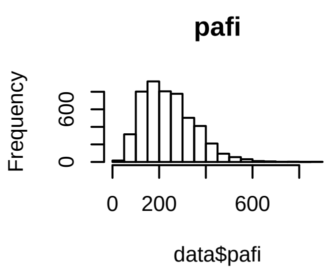
```r
# We only look at complete cases
data <- data[complete.cases(data), ]
data <- data[data$totcst > 0, ]

# histograms
par(mfrow = c(3, 3))
hist(data$age, main = 'age')
hist(data$num.co, main = 'num.co')
hist(data$edu, main = 'edu')
hist(data$scoma, main = 'scoma')
hist(data$totcst, main = 'totcst')
hist(data$meanbp, main = 'meanbp')
hist(data$hrt, main = 'hrt')
hist(data$resp, main = 'resp')
hist(data$temp, main = 'temp')
```

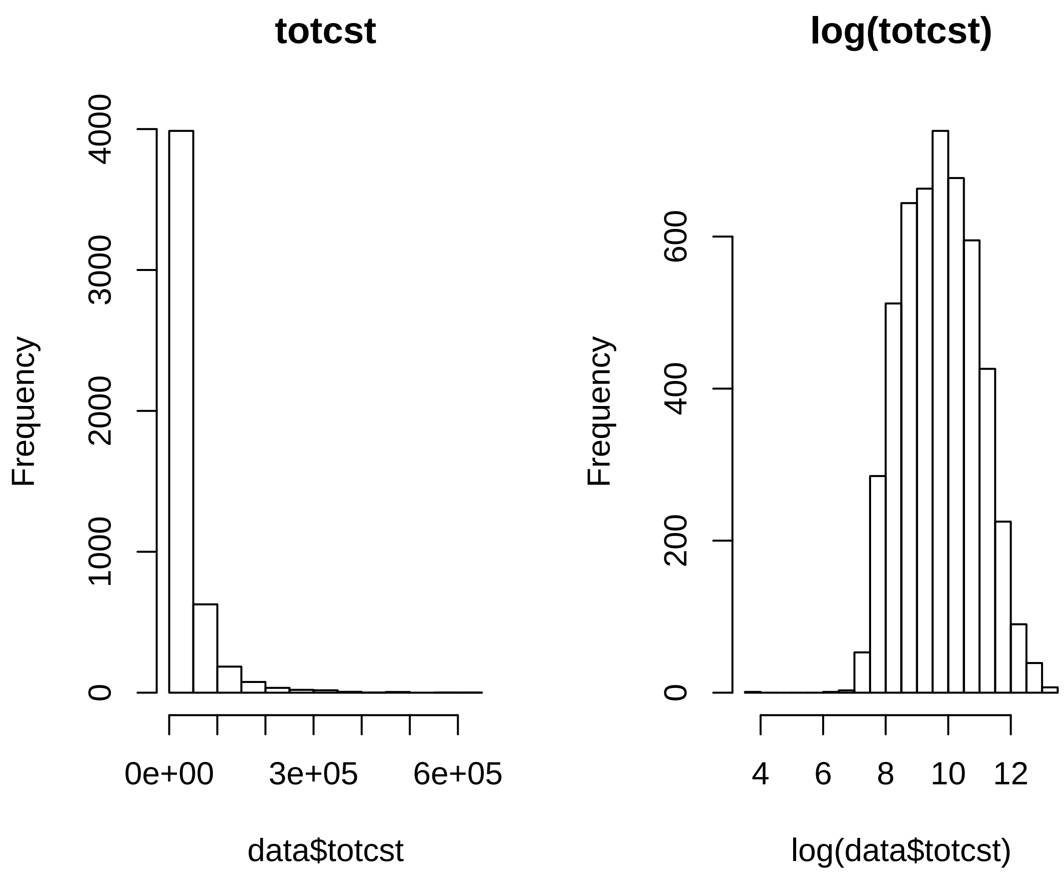



```r
hist(data$pafi, main = 'pafi')

#transformation
par(mfrow = c(1, 2))
```




```r
hist(data$totcst, main = 'totcst')
hist(log(data$totcst), main = 'log(totcst)')
```



Looking at the distribution of the cost we see we should apply a log transformation for a better distribution. Moreover it seems that only age and disease have an impact.

```
set.seed(12345)
train.proportion = 0.8
train.ind = sample(1:nrow(data), train.proportion* nrow(data))
data.train = data[train.ind, ]
data.test = data[-train.ind, ]

fit = lm(log(totcst)~ age + as.factor(dzgroup) , data = data.train)
summary(fit)
```

```
##
## Call:
## lm(formula = log(totcst) ~ age + as.factor(dzgroup), data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9537 -0.6718 -0.0441  0.6168  3.4989
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  10.8850663  0.0635606 171.255  < 2e-16 ***
## age                          -0.0078734  0.0009762  -8.065 9.59e-16 ***
## as.factor(dzgroup)CHF        -1.5430907  0.0456188 -33.826  < 2e-16 ***
## as.factor(dzgroup)Cirrhosis  -1.0132184  0.0717487 -14.122  < 2e-16 ***
## as.factor(dzgroup)Colon Cancer -1.4692225  0.0961233 -15.285  < 2e-16 ***
```

```
## as.factor(dzgroup)Coma         -0.4136454  0.0645263  -6.410 1.62e-10 ***
## as.factor(dzgroup)COPD         -1.3255009  0.0483302 -27.426  < 2e-16 ***
## as.factor(dzgroup)Lung Cancer  -1.6988078  0.0606564 -28.007  < 2e-16 ***
## as.factor(dzgroup)MOSF w/Malig -0.2353383  0.0571034  -4.121 3.85e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9428 on 3959 degrees of freedom
## Multiple R-squared:  0.3705, Adjusted R-squared:  0.3692
## F-statistic: 291.2 on 8 and 3959 DF,  p-value: < 2.2e-16
```

We can write :

$$log(totcost) = 8.0823597 - 0.0069950 * age + x_{ij} * \beta_j$$

where x_{ij} is 1 if patient i has disease j and β_j is the coefficient matchinf the disease in the previous tab.

We can calculate the MSE on the test set to evaluate the simple linear regression model.

```
y = predict(fit, newdata = data.test,
            newx=model.matrix(log(totcst)~. , data.test)[,-1])
mse = mean((y - log(data.test$totcst))^2)
mse
```

```
## [1] 0.8986823
```

Multiple linear regression

Definition

TO DO

DEFINITION

WHICH INDICATORS ?

An example in R

We use the same example than for simple linear regression.

```
fit_multiple = lm(log(totcst)~age*as.factor(dzgroup), data = data.train)
summary(fit_multiple)
```

```
##
## Call:
## lm(formula = log(totcst) ~ age * as.factor(dzgroup), data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9947 -0.6660 -0.0446  0.6165  3.5028
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     10.7209408  0.0814447 131.635  < 2e-16 ***
## age                             -0.0051751  0.0012862  -4.023 5.84e-05 ***
## as.factor(dzgroup)CHF           -1.1845264  0.2081502  -5.691 1.36e-08 ***
## as.factor(dzgroup)Cirrhosis     -0.6342383  0.2931372  -2.164 0.030553 *
## as.factor(dzgroup)Colon Cancer  -1.3738320  0.6193236  -2.218 0.026593 *
## as.factor(dzgroup)Coma           0.3113839  0.2516862   1.237 0.216090
```

```
## as.factor(dzgroup)COPD             -1.4103615  0.2865309  -4.922 8.91e-07 ***
## as.factor(dzgroup)Lung Cancer      -1.8670958  0.3369421  -5.541 3.20e-08 ***
## as.factor(dzgroup)MOSF w/Malig      0.5022105  0.2183406   2.300 0.021493 *
## age:as.factor(dzgroup)CHF          -0.0055886  0.0030703  -1.820 0.068799 .
## age:as.factor(dzgroup)Cirrhosis    -0.0067167  0.0052722  -1.274 0.202744
## age:as.factor(dzgroup)Colon Cancer -0.0016268  0.0095464  -0.170 0.864697
## age:as.factor(dzgroup)Coma         -0.0115444  0.0038546  -2.995 0.002762 **
## age:as.factor(dzgroup)COPD          0.0008436  0.0040713   0.207 0.835854
## age:as.factor(dzgroup)Lung Cancer   0.0026525  0.0053418   0.497 0.619531
## age:as.factor(dzgroup)MOSF w/Malig -0.0124110  0.0035559  -3.490 0.000488 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.941 on 3952 degrees of freedom
## Multiple R-squared:  0.374,  Adjusted R-squared:  0.3717
## F-statistic: 157.4 on 15 and 3952 DF,  p-value: < 2.2e-16
```

We can calculate the MSE on the test set to evaluate the multiple linear regression model.

```
y = predict(fit_multiple, newdata = data.test,
            newx=model.matrix(log(totcst)~age*as.factor(dzgroup) , data.test)[,-1])
mse = mean((y - log(data.test$totcst))^2)
mse
```

```
## [1] 0.8948407
```

The MSE-test for multiple linear regression is worst than for simple linear regression.

Simple linear regression is the best model so far for this problem.

Comparaison between R and sckit-learn in python

On classification

Logistic Regression

TO DO : comparaison between R and python

TO - DO : AN OTHER MODEL FOR THE SAME DATA SET

TO DO : comparaison between R and python

either knn, or decsion trees, or linear discriminant analysis or quadratic discriminant analysis

On Regression

Simple Linear Regression

TO DO : comparaison between R and python

Multiple Linear Regression

TO DO : comparaison between R and python