Régression Logistique et application avec R

gaspard PALAY

January 2021

1 Introduction

La regression logistique sert à analyser une variable binaire (O/1) (VRAI/FAUX) en fonction d'une variable quantitative. La regression logistique est typiquement utilisée dans des siutations de sciences humaines et sociales ou en médecine. Ce type de modèle est utilisé en machine learning pour des apprentissages supervisés. Cette algorithme va faire de la regression (non pas de la classification).

Dans cette étude, je vous présenterai dans un premier temps le modèle mathématique de la régression logistique. J'effectuerai dans un second temps un test de regression en machine learning sur R en prenant le jeu de données du Tintanic.

La premièere partie de ce document, c'est à dire l'explication du modèle mathématique à été écrite sur Overleaf, l'éditeur en ligne de Latex. La seconde partie de mon document, à savoir le test de regression logistique sur les naufragés du Titanic a été effectuée sur RStudio en RMarkdown puis exportée en PDF. Les résultats de ces deux documents ont été compilés ensuite sur un seul et même PDF.

2 Explication du modèle mathématique

Mon niveau actuel en mathématique et statistique étant mauvais, je tenterais d'expliquer dans cette partie au mieux, le concept mathématique de regression logistique. Il est fort probable que mon explication soit partielle, incomplète et imprécise. Vous lecteur étant avertit, je vous prierai donc de ne pas en tenir rigueur et de vous renseigner plus profondément sur le concept auprès d'experts du domaine.

La regression logistique est un prolongement de la regression linéaire.
Avec un modèle de regression linéaire classique, on aura le modèle mathématique suivant :
$= \alpha X + \beta$

L'espérance sera donc calculée avec la foction suivante :
$(Y) = \alpha X + \beta$

La fonction Y, dans un modèle de regression logistique étant distribuée de manière binaire, on considère une fonction généralisée de **lien** :
$(E(Y)) = \alpha X + \beta$

La fonction de lien, pour une regression logistique est exprimée comme telle :
$(p) = log(\frac{p}{1-p})$

Théorème de Bayes
Probabilités conditionnelles – On se place dans le cadre binaire $Y \in \{+, -\}$

Estimer la probabilité conditionnelle P(Y/X) $\begin{cases} P(Y = y_k) = \frac{P(Y=y_k) \times P(X/Y=y_k)}{P(X)} \\ \qquad\qquad = \frac{P(Y=y_k) \times P(X/Y=y_k)}{\sum_{l=1}^{k} P(Y=y_l \times P(X/Y=y_l)} \end{cases}$

Hypothèse fondamentale de la régression logistique

$$ln\left[\frac{P(X/Y=+)}{P(X/Y=-)}\right] = b_0 + b_1 + ... + b_j X_j$$

L'hypothèse précédente couvre les distributions suivantes :

- Loi gamma, Beta, Poisson
- Loi exponentielle
- Loi normale
- Lois discrètes
- Mélange de variables explicatives binaires (0/1) et numériques

Ses avantages sont les suivants :

1. Champ d'application théoriquement plus large que l'Analyse Discriminante

2. Sa capacité à traiter et proposer une interprétation des coefficients pour les variables explicatives binaires est très intéressante

Le modèle LOGIT, une autre écriture du rapport de probabilité

On écrit $\pi(X) = P(Y = +/X)$

Le **LOGIT** de P(Y=+/X) s'écrit de la manière suivante :

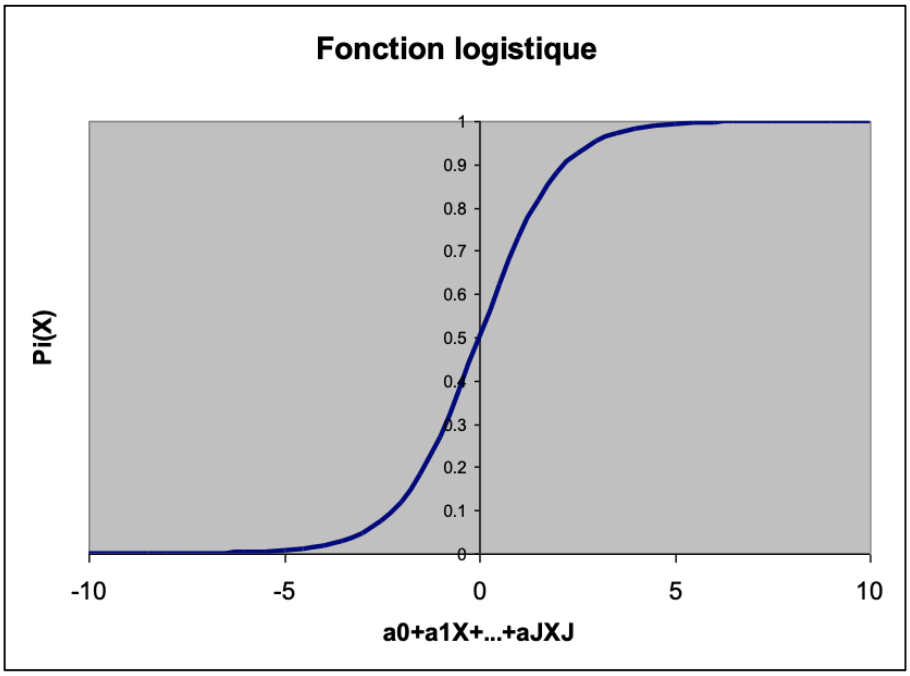
$$ln\left[\frac{\pi(X)}{1-\pi(X)}\right] = a_0 + a_1X_1 + ... + a_jX_j$$

$$\text{Avec } \pi(X) = \frac{e^{a0+a1+...+aj+Xj}}{1+e^{a0+a1+...+aj+Xj}}$$

Ou $\pi(X)$est la fonction de répartition de la loi Logistique

Les **odds**, c'est à dire le rapport de chance de la fonction logistique s'écrit de la manière suivante :

$$\frac{\pi(X)}{1-\pi(X)} = \frac{P(+/X}{P(-/X)}$$


Fonction logistique
Pi(X)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-10
-5
0
5
10
a0+a1X+...+aJXJ

3. Regression logistique avec R

Dans ce cas pratique, nous allons tenter de mettre en application un modèle algorithmique de regression logistique sur un jeu de données. Nous allons tenter de mesurer l'association entre la survenue d'un évenement, à savoir ici la survie ou non des passagers du Titanic, avec différentes variables explicatives à savoir la classe de croisière choisie par le passager : 'Pclass', son genre :'Sex', son age : 'Age', son nombre de frères / soeurs / conjoints : 'SibSp', son nombre de parents et d'enfants : 'Parch', le prix payé pour la croisièere : 'Fare', son quaie d'embarquement : 'Embarked'.

Une fois avoir évalué le pourcentage d'association entre ces deux variables, nous tenterons d'établir un modèle de prédiction en apprentissage supervisé, en testant notre jeu de donnée sur un fichier test.

```
install.packages("readr") #MImportation du dataset
install.packages("dplyr") #Manipulation du dataset
install.packages("tidyverse") #Manipulation du dataset
install.packages("caret") #package de machine learning
install.packages("caTools") #diviser son dataset en 75% train 25%test
install.packages("pastecs") #visualiser les stats de nos DataFrame
```

3.1. Importation des données du Titanic

Dans un premier temps, nous choisissons un jeu de données en ligne. Sur cet étude, je m'inspire d'un tutoriel et d'une compétition connue sur Kaggle à savoir : "Titanic - Machine Learning from Disaster" accessible sur ce lien : https://www.kaggle.com/c/titanic/data. Nous tléchargeons le jeu de données et le stockons en local sur notre machine. L'objectif de cet exercice est de prévoir si un pagé va survivre ou non au naufrage du titanic. Le jeu de données est composé de deux fichier "Train.csv" qui servira a entrainer notre algorithme de regression logitsique et "test.csv" qui nous servira a tester notre modèle.

```
test <- read_csv("/Users/gaspardpalay/Documents/PSB - Paris School of Business/Mathématique appliquée au
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   PassengerId = col_double(),
##   Pclass = col_double(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_double(),
##   Parch = col_double(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

```
train <- read_csv("/Users/gaspardpalay/Documents/PSB - Paris School of Business/Mathématique appliquée a
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   PassengerId = col_double(),
##   Survived = col_double(),
##   Pclass = col_double(),
##   Name = col_character(),
##   Sex = col_character(),
```

```
##   Age = col_double(),
##   SibSp = col_double(),
##   Parch = col_double(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

3.2. Manipulation et Nettoyage des données

Nous allons d'abbord visualiser nos deux jeux de données afin de connaitre la répartition des données sur ceux-ci

```
test
```

```
## # A tibble: 418 x 11
##    PassengerId Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin Embarked
##          <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr> <chr>
##  1         892      3 Kelly~ male   34.5     0     0 330911  7.83 <NA>  Q
##  2         893      3 Wilke~ fema~  47       1     0 363272  7    <NA>  S
##  3         894      2 Myles~ male   62       0     0 240276  9.69 <NA>  Q
##  4         895      3 Wirz,~ male   27       0     0 315154  8.66 <NA>  S
##  5         896      3 Hirvo~ fema~  22       1     1 31012~ 12.3  <NA>  S
##  6         897      3 Svens~ male   14       0     0 7538    9.22 <NA>  S
##  7         898      3 Conno~ fema~  30       0     0 330972  7.63 <NA>  Q
##  8         899      2 Caldw~ male   26       1     1 248738 29    <NA>  S
##  9         900      3 Abrah~ fema~  18       0     0 2657    7.23 <NA>  C
## 10         901      3 Davie~ male   21       2     0 A/4 4~ 24.2  <NA>  S
## # ... with 408 more rows
```

```
train
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name  Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr> <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Brau~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumi~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heik~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futr~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Alle~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Mora~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCa~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Pals~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 John~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nass~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

```
stat.desc(train)
```

```
##                PassengerId     Survived       Pclass Name Sex          Age
## nbr.val      8.910000e+02 891.00000000 8.910000e+02   NA  NA 7.140000e+02
## nbr.null     0.000000e+00 549.00000000 0.000000e+00   NA  NA 0.000000e+00
## nbr.na       0.000000e+00   0.00000000 0.000000e+00   NA  NA 1.770000e+02
## min          1.000000e+00   0.00000000 1.000000e+00   NA  NA 4.200000e-01
## max          8.910000e+02   1.00000000 3.000000e+00   NA  NA 8.000000e+01
```

```
## range        8.900000e+02   1.00000000 2.000000e+00   NA  NA 7.958000e+01
## sum          3.973860e+05 342.00000000 2.057000e+03   NA  NA 2.120517e+04
## median       4.460000e+02   0.00000000 3.000000e+00   NA  NA 2.800000e+01
## mean         4.460000e+02   0.38383838 2.308642e+00   NA  NA 2.969912e+01
## SE.mean      8.621678e+00   0.01630146 2.800944e-02   NA  NA 5.436405e-01
## CI.mean.0.95 1.692119e+01   0.03199378 5.497225e-02   NA  NA 1.067328e+00
## var          6.623100e+04   0.23677222 6.990151e-01   NA  NA 2.110191e+02
## std.dev      2.573538e+02   0.48659245 8.360712e-01   NA  NA 1.452650e+01
## coef.var     5.770266e-01   1.26770139 3.621485e-01   NA  NA 4.891222e-01
##                     SibSp        Parch Ticket         Fare Cabin Embarked
## nbr.val      891.00000000 891.00000000     NA   891.000000    NA       NA
## nbr.null     608.00000000 678.00000000     NA    15.000000    NA       NA
## nbr.na         0.00000000   0.00000000     NA     0.000000    NA       NA
## min            0.00000000   0.00000000     NA     0.000000    NA       NA
## max            8.00000000   6.00000000     NA   512.329200    NA       NA
## range          8.00000000   6.00000000     NA   512.329200    NA       NA
## sum          466.00000000 340.00000000     NA 28693.949300    NA       NA
## median         0.00000000   0.00000000     NA    14.454200    NA       NA
## mean           0.52300786   0.38159371     NA    32.204208    NA       NA
## SE.mean        0.03694329   0.02700393     NA     1.664792    NA       NA
## CI.mean.0.95   0.07250613   0.05299881     NA     3.267377    NA       NA
## var            1.21604308   0.64972824     NA  2469.436846    NA       NA
## std.dev        1.10274343   0.80605722     NA    49.693429    NA       NA
## coef.var       2.10846437   2.11234407     NA     1.543073    NA       NA
```

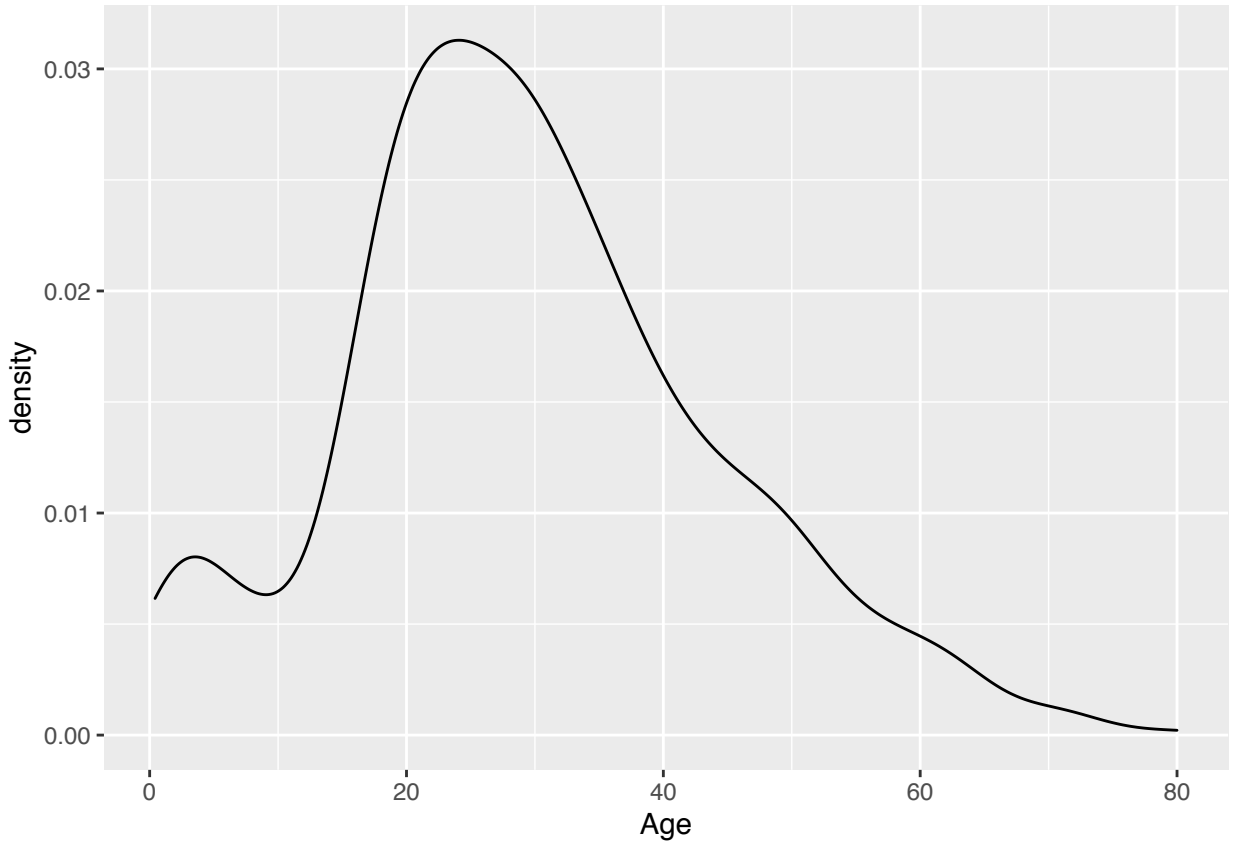
```
stat.desc(test)
```

```
##               PassengerId       Pclass Name Sex          Age        SibSp
## nbr.val      4.180000e+02 418.00000000   NA  NA 3.320000e+02 418.00000000
## nbr.null     0.000000e+00   0.00000000   NA  NA 0.000000e+00 283.00000000
## nbr.na       0.000000e+00   0.00000000   NA  NA 8.600000e+01   0.00000000
## min          8.920000e+02   1.00000000   NA  NA 1.700000e-01   0.00000000
## max          1.309000e+03   3.00000000   NA  NA 7.600000e+01   8.00000000
## range        4.170000e+02   2.00000000   NA  NA 7.583000e+01   8.00000000
## sum          4.600090e+05 947.00000000   NA  NA 1.005050e+04 187.00000000
## median       1.100500e+03   3.00000000   NA  NA 2.700000e+01   0.00000000
## mean         1.100500e+03   2.26555024   NA  NA 3.027259e+01   0.44736842
## SE.mean      5.909033e+00   0.04117562   NA  NA 7.782950e-01   0.04386194
## CI.mean.0.95 1.161520e+01   0.08093765   NA  NA 1.531028e+00   0.08621807
## var          1.459517e+04   0.70869046   NA  NA 2.011067e+02   0.80417771
## std.dev      1.208105e+02   0.84183755   NA  NA 1.418121e+01   0.89675956
## coef.var     1.097778e-01   0.37158194   NA  NA 4.684505e-01   2.00452137
##                     Parch Ticket         Fare Cabin Embarked
## nbr.val      418.00000000     NA   417.000000    NA       NA
## nbr.null     324.00000000     NA     2.000000    NA       NA
## nbr.na         0.00000000     NA     1.000000    NA       NA
## min            0.00000000     NA     0.000000    NA       NA
## max            9.00000000     NA   512.329200    NA       NA
## range          9.00000000     NA   512.329200    NA       NA
## sum          164.00000000     NA 14856.537600    NA       NA
## median         0.00000000     NA    14.454200    NA       NA
## mean           0.39234450     NA    35.627188    NA       NA
## SE.mean        0.04800326     NA     2.737806    NA       NA
## CI.mean.0.95   0.09435852     NA     5.381658    NA       NA
## var            0.96320264     NA  3125.657074    NA       NA
```

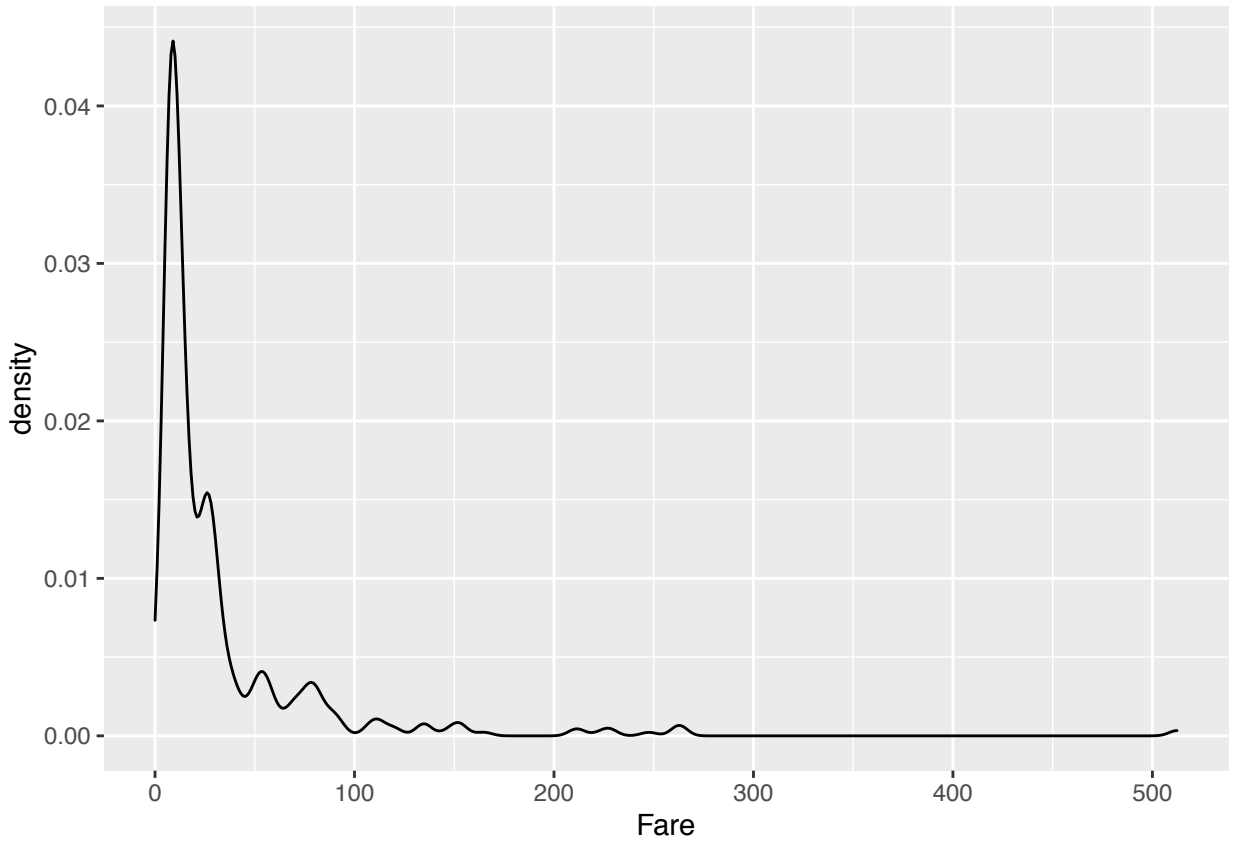
```
## std.dev        0.98142888     NA    55.907576    NA       NA
## coef.var       2.50144678     NA     1.569239    NA       NA
```

```
ggplot(train, aes(x=Age)) +
  geom_density()
```

```
## Warning: Removed 177 rows containing non-finite values (stat_density).
```




```
ggplot(train, aes(x=Fare)) +
  geom_density()
```



Nous commençons par regrouper les deux fichiers dans une seule et même dataframe. On appelera cette table “full”

```
full <- bind_rows(train,test)
class(full)
```

On visualise ensuite notre nouvelle table

```
head(full)
```

```
## # A tibble: 6 x 12
##   PassengerId Survived Pclass Name  Sex     Age SibSp Parch Ticket  Fare Cabin
##         <dbl>    <dbl>  <dbl> <chr> <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
## 1           1        0      3 Brau~ male     22     1     0 A/5 2~  7.25 <NA>
## 2           2        1      1 Cumi~ fema~    38     1     0 PC 17~ 71.3  C85
## 3           3        1      3 Heik~ fema~    26     0     0 STON/~  7.92 <NA>
## 4           4        1      1 Futr~ fema~    35     1     0 113803 53.1  C123
## 5           5        0      3 Alle~ male     35     0     0 373450  8.05 <NA>
## 6           6        0      3 Mora~ male     NA     0     0 330877  8.46 <NA>
## # ... with 1 more variable: Embarked <chr>
```

On remarque que la premiere colonne est constitué du passenger_id, la seconde nous indique si la personne a survécu (1 : elle a survécu, 0 elle n’a pas survécu), la troisième “Pclass” est la formule de croisère choisie. On a ensuite une colonne pour le Sexe du passager, son age.
SibSp nous donne le nombre de conjoints, frères et soeurs dudit passager, Parch nous donne le nombre d’enfants et de parents, fare est le prix qu’a payé le passager, Cabin son numéro de cabine et Embarked sa porte d’embarquement dans le navire.

On étudie ensuite le nombre de données manquantes sur la dataset

```
sum(is.na(full)) #nombre total de valeurs manquantes
```

```
## [1] 1698
```

```
colMeans(is.na(full)) #Pourcentage de valeurs manquantes par colonnes
```

```
##  PassengerId     Survived       Pclass         Name          Sex          Age
## 0.0000000000 0.3193277311 0.0000000000 0.0000000000 0.0000000000 0.2009167303
##        SibSp        Parch       Ticket         Fare        Cabin     Embarked
## 0.0000000000 0.0000000000 0.0000000000 0.0007639419 0.7746371276 0.0015278839
```

Afin d'obtenir un jeu de données propre on va supprimer les valeurs manquantes de chaques colonnes

```
full <- full[!is.na(full$Embarked),]
#supprime les valeurs manquantes de Embarked
full <- full[!is.na(full$Survived),]
#supprime les valeurs manquantes de Survived
full[is.na(full$Age),]$Age <- median(full$Age,na.rm = T)
#remplace les valeurs manquantes de l'Age par la médiane de Full
view(full)
```

On vérifie d'abbord que notre jeu de données ne comporte plus de NA.

```
sum(is.na(full))
```

```
## [1] 687
```

On va ensuite selectionner les données qui nous interessent pour tester notre regression logistique.

```
full <- full %>% select(c('Survived', 'Pclass', 'Sex', 'Age',
                          'SibSp', 'Parch', 'Fare', 'Embarked'))
```

```
sum(is.na(full)) #nombre total de valeurs manquantes
```

```
## [1] 0
```

```
colMeans(is.na(full)) #Pourcentage de valeurs manquantes par colonnes
```

```
## Survived   Pclass      Sex      Age    SibSp    Parch     Fare Embarked
##        0        0        0        0        0        0        0        0
```

Le jeu de données étant néttoyé, nous allons maintenant mettre un seed qui nous permettra d'avoir les même résultats à chaque boucle lorqu'on fera une requête random.

3.3. Modélisation et Statistiques descriptives

On ajoute un seed, la valeur à laquelle le modèle commencera On redivise ensuite notre jeu de données en Train (75%) et en Test (25%)‘

```
set.seed(222)
smp_size <- floor(0.75 * nrow(full))
train_ind <- sample(seq_len(nrow(full)), size = smp_size)
train <- full[train_ind, ]
test <- full[-train_ind, ]
```

3.3.1 Création du modèle

```
fitControl <- trainControl(method = "cv" , number = 10, savePredictions = TRUE)
#paramétrage du modèle
```

```
lr_model <- train(factor(Survived) ~ .,
                  data = train,
                  method = 'glm',
                  family = binomial(),
                  trControl = fitControl)
```

```
summary(lr_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5054  -0.6376  -0.4309   0.6624   2.3814
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.786919   0.620098   7.720 1.17e-14 ***
## Pclass      -0.997457   0.162381  -6.143 8.11e-10 ***
## Sexmale     -2.623389   0.228186 -11.497  < 2e-16 ***
## Age         -0.032075   0.008675  -3.698 0.000218 ***
## SibSp       -0.310112   0.121600  -2.550 0.010764 *
## Parch       -0.113488   0.131884  -0.861 0.389506
## Fare         0.002847   0.002737   1.040 0.298234
## EmbarkedQ   -0.245116   0.440631  -0.556 0.578016
## EmbarkedS   -0.525647   0.267917  -1.962 0.049765 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 886.39  on 665  degrees of freedom
## Residual deviance: 602.39  on 657  degrees of freedom
## AIC: 620.39
##
## Number of Fisher Scoring iterations: 5
```

3.3.2 Test et estimation de la qualité du modèle

```
prediction_lr <- predict(lr_model, test)
test$Prediction <- prediction_lr
```

```
xtab <- table(test$Prediction , test$Survived)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##
```

```
##       0   1
##   0 126  27
##   1  12  58
##
##                Accuracy : 0.8251
##                  95% CI : (0.7688, 0.8726)
##     No Information Rate : 0.6188
##     P-Value [Acc > NIR] : 1.705e-11
##
##                   Kappa : 0.6163
##
##  Mcnemar's Test P-Value : 0.02497
##
##             Sensitivity : 0.9130
##             Specificity : 0.6824
##          Pos Pred Value : 0.8235
##          Neg Pred Value : 0.8286
##              Prevalence : 0.6188
##          Detection Rate : 0.5650
##    Detection Prevalence : 0.6861
##       Balanced Accuracy : 0.7977
##
##        'Positive' Class : 0
##
```

3.3.3 Lecture des résultats :

Les colonnes représentent la réalité de survie des passagers du Titanic tandis que les lignes représentent la prédiction du modèle de régression logistique. On interprète les résultats de la manièere suivante : il y a 138 personnes qui n'ont pas survécu au naufrage du titanic. Parmis ces 138 personnes, le modèles de prédiction a prévu que 126 ne survivraient pas mais s'est trompé sur 12 passagers en prévoyant qu'ils survivraient. A l'inverse, il ya 85 personnes qui ont survécues au Titanic, le modèle a prévu que 59 personnes d'entre elles survivraient mais n'a pas pu prévoir la survie de 27 personnes parmis les survivantes.

L'indicateur de niveau de précision nous informe que le modèle construit est fiable à 82%. Même si ce niveau de prédiction est bon, nous pourrions l'améliorer grâce à du fitter engineering ou encore en modifiant notre processus de nettoyage de données.

! NOTE AU LECTEUR ! : points d'attention - doutes méthodologiques à lever par un lecteur attentif, il est possible que ce workflow augmente l'accuracy de manière artificielle. Le nettoyage du jeu de donnée a été executé sur le full, il aurait surement dû être fait sur le Train avant le Full. Des lignes comportants des NA ont été supprimées du Test, cela relève l'accurary.

4. Sources

http://eric.univ-lyon2.fr/~ricco/cours/cours_regression_logistique.html

https://www.datacamp.com/community/tutorials/logistic-regression-R

https://stats.idre.ucla.edu/r/dae/logit-regression/

https://stats.idre.ucla.edu/r/dae/logit-regression/

http://larmarange.github.io/analyse-R/regression-logistique.html

https://www.r-bloggers.com/2015/09/how-to-perform-a-logistic-regression-in-r/

https://www.kaggle.com/c/titanic

https://www.xlstat.com/fr/solutions/fonctionnalites/regression-logistique-pour-reponse-binaires-et-multinomiales-logit-probit#:~:text=Pour%20la%20r%C3%A9gression%20logistique%2C%20la,par%20exemple%20la%20m%C3%AAme%20dose

https://fr.wikipedia.org/wiki/R%C3%A9gression_logistique

https://datascientest.com/regression-logistique-quest-ce-que-cest

https://www.youtube.com/watch?v=fUmDPVHah1U&ab_channel=StatB.Falissard

https://www.youtube.com/watch?v=j6QQWfSxFnE

https://www.youtube.com/watch?v=JrpPd1iGaRY&ab_channel=HocineTedjani

https://www.youtube.com/watch?v=td_BrAq5Rog&ab_channel=claudeaboki

https://www.youtube.com/watch?v=C4N3_XJJ-jU&t=210s&ab_channel=StatQuestwithJoshStarmer

https://www.youtube.com/watch?v=iyU2CkHrfQk&ab_channel=TomSherratt