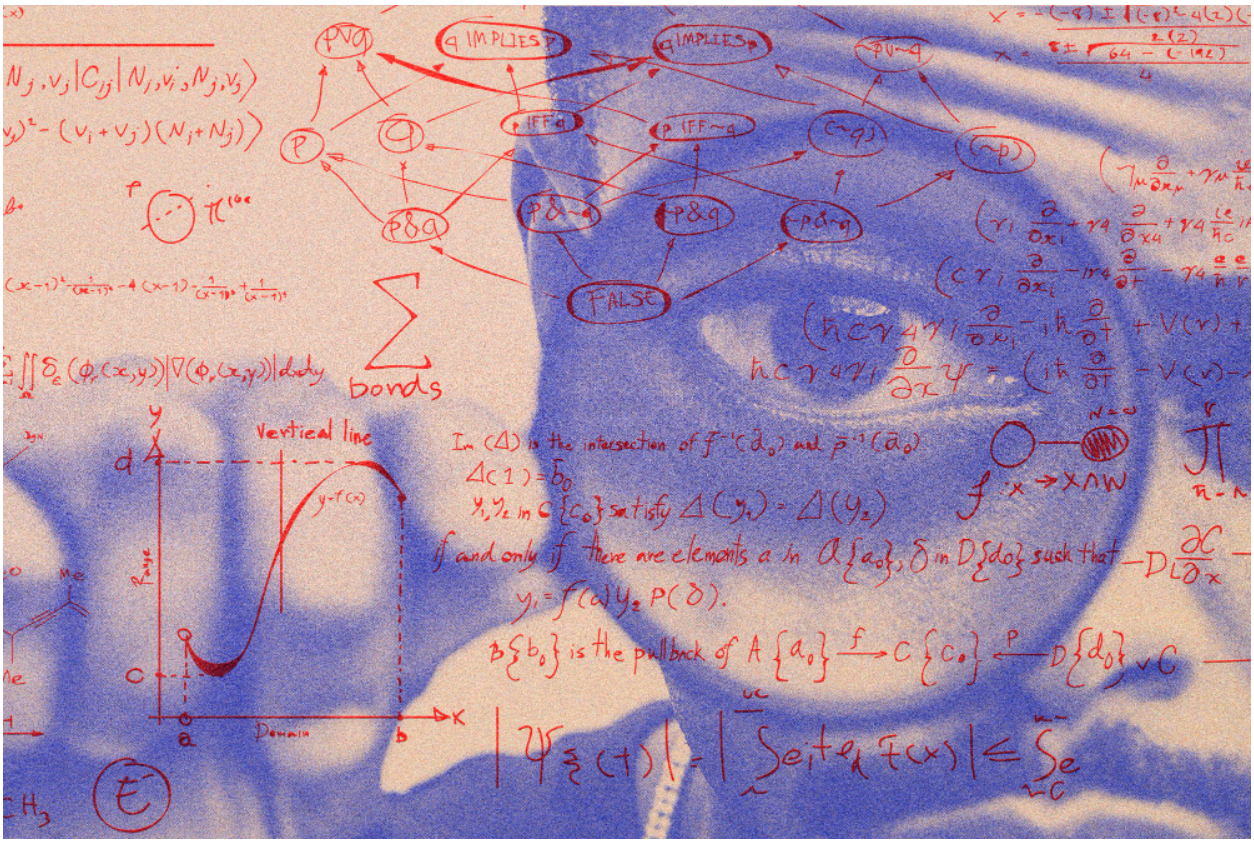
Modèle-Linéaires-Généralisés

Samuel Lauras

2024-07-01

Université Paris Dauphine (PSL) **- professeur :** ***Robin RYDER***

Support d'étude : Regression logistique sur les prédictiosn météorologiques



```
library(tidyverse)
library(corrplot)
library(pROC)
```

Préambule :

Lors de cette étude nous verrons la régression logistique dans le cadre des modèles linéaires généralisés.

Les modèles logistiques sont préférés aux modèles linéaires lorsqu'il s'agit de prédire des probabilités d'événements binaires, car ils capturent mieux les relations non linéaires et produisent des résultats réalistes bornés entre 0 et 1.

Nous possédons des données météorologiques, et chercherons à inférer sur chaque jour sur la probabilité de pluie pour le lendemain

1. Analyse du jeu de donnée

Le jeu de données concerne une quinzaine de données météorologiques (pressions atmosphériques, températures, vents etc...) agrégées en min,max et moyenne par jour.

Les données proviennent de *MeteoBlue*.

Nous cherchons à inférer la colonne *pluie.demain* qui est binaire.

```
df <- read.csv("./meteo.train.csv")

print(paste("Nombre de lignes:", nrow(df)))
```

```
## [1] "Nombre de lignes: 1180"
```

```
print(paste("Nombre de colonnes:", ncol(df)))
```

```
## [1] "Nombre de colonnes: 47"
```

```
types_de_colonnes <- sapply(df, class)
table(types_de_colonnes)
```

```
## types_de_colonnes
## integer logical numeric
##      14       1      32
```

On voit qu'on a beaucoup de colonnes. Sachant que pour chaque variable il existe sa déclinaison min, max et moy, on s'attend à rencontrer de la corrélation entre nos variables.

Essayons dans un premier temps de retirer celles qui ne nous intéressent pas, comme par exemple, celles relatives aux la date, temps, et l'ID.

```
table(df$Hour,df$Minute)
```

```
##
##        0
##   0 1180
```

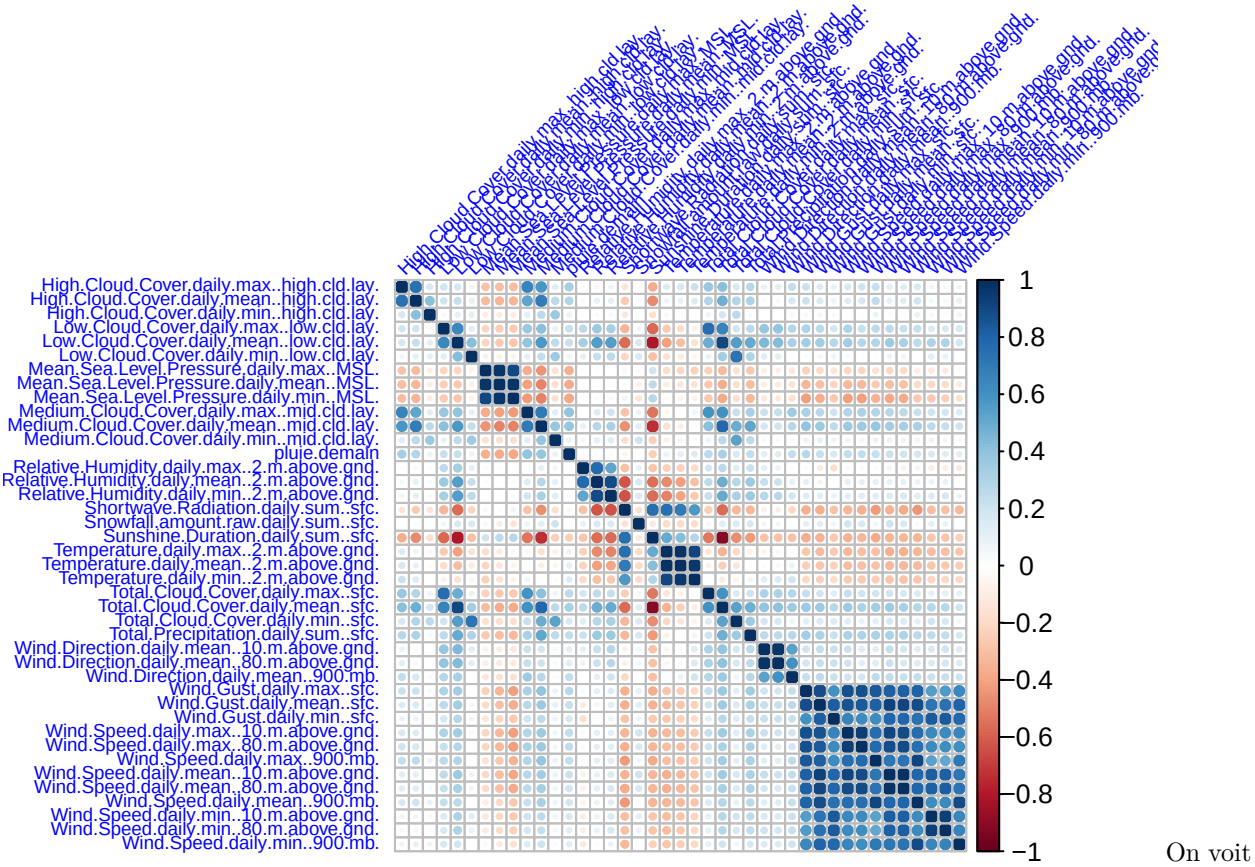
On voit que même les colonnes de temps (qui pourraient être utile en fonction) sont remplies à 0, et donc non pertinentes

Nous pouvons prendre en compte le mois de l'année qui nous renseigne sur la saison, et a des chances d'impacter la venue de pluie ou non. Nous devons le transformer en facteur, pour que le modèle le considère comme une variable catégorielle et non quantitative.

```
df$Month <- factor(df$Month)

# retirer les colonnes indésirables
df <- df %>% select(-id, -Year, -Day, - Hour, -Minute)
```

Regardons l'impact des variable les unes aux autres pour voir si il y a de la colinéarité.

```
# trier les colonnes par ordre alphabetique (question visuelle)
df <- df %>% select(order(names(df)))


corrplot(cor(df[,-13]), tl.cex = 0.6, tl.col = "blue", tl.srt = 45)
```



On voit qu'il y a des groupes de variables très fortement corrélées. Nous devrons travailler sur cela avant de pouvoir faire notre sélection de modèles.

2. Modélisation et selection de modèles

Sélection de variables, éliminer les corrélations

Lançons une première régression afin d'obtenir un premier modèle naïf...

```
options(width = 200)

# regression logistique simple
reg.naive <- glm(formula = pluie.demain ~ . ,
           family=binomial,
           data = df)
```

```r
summary(reg.naive, )
```

```
##
## Call:
## glm(formula = pluie.demain ~ ., family = binomial, data = df)
##
## Coefficients:
##                                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                               7.125e+01  1.289e+01   5.529 3.22e-08 ***
## High.Cloud.Cover.daily.max..high.cld.lay.  3.188e-03  2.932e-03   1.087 0.277006
## High.Cloud.Cover.daily.mean..high.cld.lay. -2.353e-03  6.922e-03  -0.340 0.733875
## High.Cloud.Cover.daily.min..high.cld.lay.  2.806e-03  2.095e-02   0.134 0.893445
## Low.Cloud.Cover.daily.max..low.cld.lay.    1.997e-03  3.458e-03   0.577 0.563719
## Low.Cloud.Cover.daily.mean..low.cld.lay.  -3.936e-03  8.299e-03  -0.474 0.635268
## Low.Cloud.Cover.daily.min..low.cld.lay.   -6.075e-04  7.138e-03  -0.085 0.932177
## Mean.Sea.Level.Pressure.daily.max..MSL.   -2.574e-01  7.658e-02  -3.361 0.000777 ***
## Mean.Sea.Level.Pressure.daily.mean..MSL.   5.147e-01  1.443e-01   3.568 0.000360 ***
## Mean.Sea.Level.Pressure.daily.min..MSL.   -3.309e-01  7.939e-02  -4.168 3.07e-05 ***
## Medium.Cloud.Cover.daily.max..mid.cld.lay. 5.220e-03  3.215e-03   1.624 0.104459
## Medium.Cloud.Cover.daily.mean..mid.cld.lay. 5.127e-03  6.843e-03   0.749 0.453678
## Medium.Cloud.Cover.daily.min..mid.cld.lay. -3.205e-03  9.610e-03  -0.334 0.738728
## Month2                                    -3.682e-01  3.851e-01  -0.956 0.339055
## Month3                                    -7.724e-01  4.135e-01  -1.868 0.061788 .
## Month4                                    -6.994e-01  4.888e-01  -1.431 0.152437
## Month5                                    -2.870e-01  5.309e-01  -0.541 0.588849
## Month6                                     2.997e-01  6.167e-01   0.486 0.626976
## Month7                                     3.212e-02  6.158e-01   0.052 0.958400
## Month8                                    -6.012e-01  5.624e-01  -1.069 0.285020
## Month9                                    -1.007e+00  4.875e-01  -2.066 0.038858 *
## Month10                                   -8.110e-01  4.096e-01  -1.980 0.047683 *
## Month11                                   -7.307e-01  3.755e-01  -1.946 0.051651 .
## Month12                                    2.483e-02  3.875e-01   0.064 0.948903
## Relative.Humidity.daily.max..2.m.above.gnd. -5.549e-03  2.073e-02  -0.268 0.788920
## Relative.Humidity.daily.mean..2.m.above.gnd. 1.615e-02  3.256e-02   0.496 0.619764
## Relative.Humidity.daily.min..2.m.above.gnd. -1.317e-02  1.855e-02  -0.710 0.477515
## Shortwave.Radiation.daily.sum..sfc.       -9.044e-05  1.262e-04  -0.717 0.473517
## Snowfall.amount.raw.daily.sum..sfc.       -4.161e-01  2.730e-01  -1.524 0.127519
## Sunshine.Duration.daily.sum..sfc.          4.978e-04  9.209e-04   0.541 0.588791
## Temperature.daily.max..2.m.above.gnd.      7.244e-02  9.969e-02   0.727 0.467438
## Temperature.daily.mean..2.m.above.gnd.     9.368e-02  1.693e-01   0.553 0.580009
## Temperature.daily.min..2.m.above.gnd.     -1.244e-01  8.815e-02  -1.412 0.158031
## Total.Cloud.Cover.daily.max..sfc.          5.303e-03  4.985e-03   1.064 0.287411
## Total.Cloud.Cover.daily.mean..sfc.         1.260e-02  1.223e-02   1.030 0.302837
## Total.Cloud.Cover.daily.min..sfc.          7.854e-03  6.426e-03   1.222 0.221622
## Total.Precipitation.daily.sum..sfc.        2.819e-02  2.874e-02   0.981 0.326549
## Wind.Direction.daily.mean..10.m.above.gnd. 5.487e-03  5.843e-03   0.939 0.347760
## Wind.Direction.daily.mean..80.m.above.gnd. -8.520e-03  6.023e-03  -1.415 0.157158
## Wind.Direction.daily.mean..900.mb.         5.238e-03  1.481e-03   3.536 0.000406 ***
## Wind.Gust.daily.max..sfc.                  8.086e-03  1.705e-02   0.474 0.635240
## Wind.Gust.daily.mean..sfc.                 3.591e-02  3.729e-02   0.963 0.335524
## Wind.Gust.daily.min..sfc.                  1.025e-02  2.794e-02   0.367 0.713613
## Wind.Speed.daily.max..10.m.above.gnd.      5.550e-02  3.522e-02   1.576 0.114998
## Wind.Speed.daily.max..80.m.above.gnd.      5.584e-03  2.908e-02   0.192 0.847714
```

```
## Wind.Speed.daily.max..900.mb.                -1.059e-02  1.222e-02  -0.867 0.386034
## Wind.Speed.daily.mean..10.m.above.gnd.       -5.178e-02  9.852e-02  -0.526 0.599203
## Wind.Speed.daily.mean..80.m.above.gnd.       -8.562e-02  7.082e-02  -1.209 0.226667
## Wind.Speed.daily.mean..900.mb.                7.624e-03  2.620e-02   0.291 0.771032
## Wind.Speed.daily.min..10.m.above.gnd.         1.701e-01  6.487e-02   2.623 0.008728 **
## Wind.Speed.daily.min..80.m.above.gnd.        -6.400e-02  4.309e-02  -1.485 0.137486
## Wind.Speed.daily.min..900.mb.                -3.539e-03  1.937e-02  -0.183 0.855029
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1635.4  on 1179  degrees of freedom
## Residual deviance: 1211.8  on 1128  degrees of freedom
## AIC: 1315.8
##
## Number of Fisher Scoring iterations: 5
```

Nous voyons que beaucoup de variables ne passent pas le test de student, celles-ci n'ont donc pas d'impact significatif sur le modèle.

Gardons en tête que ce premier modèle est juste "pour voir" car nous allons essayer de retirer les problèmes de multicolinéarité en retirant les variables trop corrélées entre elles.

Dans les hypothèses du modèle linéaire généralisé, tout comme en régression linéaire classique, nous avons besoin que les variables ne soient pas corrélées les unes aux autres. Ce qui peut poser plusieurs problèmes : instabilité des coefficients, interprétation difficile, et augmentation des variances.

On va partir des variables qui ont des p-valeurs les plus petites et regarder celles qui sont trop corrélées avec, pour les retirer.

On va décider de ne garder que les moyennes (et min ou max qui auront un impact plus significatif que leurs moyennes) des variables.

Quand à la variable Mois, étant donné que plusieures de ses modalités sont significatives, on peut la garder telle quelle, elle permettra des meilleurs prédictions.

```r
# Extraire les coéfficients et les p-valeurs
coefs <- summary(reg.naive)$coefficients

# Organiser les données dans un dataframe
coeff_df <- as.data.frame(coefs)
sorted_coeff_df <- coeff_df[order(coeff_df$`Pr(>|z|)`), c("Std. Error", "Pr(>|z|)")]
head(sorted_coeff_df,10)
```

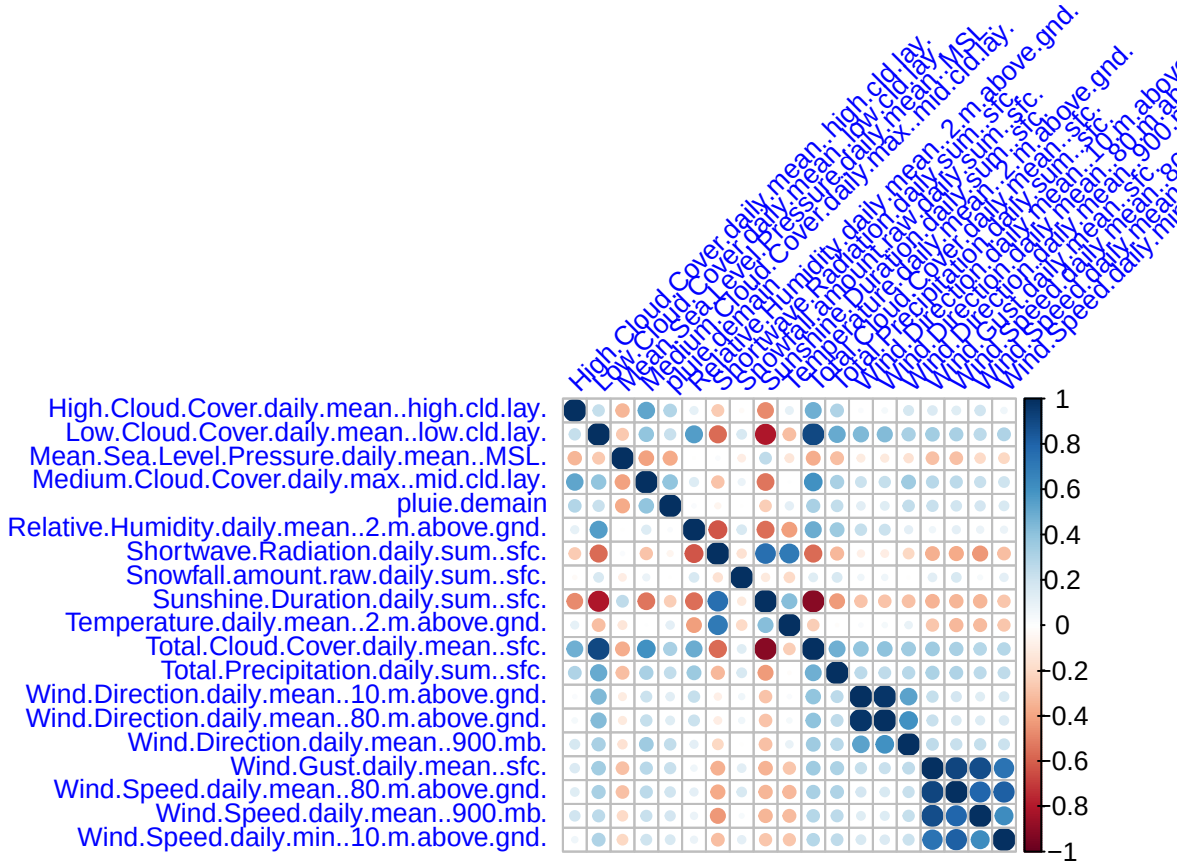
```
##                                        Std. Error     Pr(>|z|)
## (Intercept)                          12.886840102 3.223599e-08
## Mean.Sea.Level.Pressure.daily.min..MSL.   0.079386512 3.072366e-05
## Mean.Sea.Level.Pressure.daily.mean..MSL.  0.144264317 3.598404e-04
## Wind.Direction.daily.mean..900.mb.        0.001481375 4.061767e-04
## Mean.Sea.Level.Pressure.daily.max..MSL.   0.076577370 7.767535e-04
## Wind.Speed.daily.min..10.m.above.gnd.     0.064868998 8.727662e-03
## Month9                                    0.487503370 3.885840e-02
## Month10                                   0.409553852 4.768254e-02
## Month11                                   0.375483431 5.165097e-02
## Month3                                    0.413548284 6.178829e-02
```

Dans le cas des variables *Medium.Cloud.Cover.daily.max..mid.cld.lay* et *Wind.Speed.daily.min..10.m.above.gnd.*

on décide de les garder plutôt que leur equivalent moyennes car elles ont une p-valeur suffisament petite et distinctive.

```
# retirer du jeu de donnée les colonnes min/max ainsi que les deux spécifiées
df.no_corr <- df %>%  select_at(vars(-contains(c("min","max")),
                                Medium.Cloud.Cover.daily.max..mid.cld.lay.,
                                - Medium.Cloud.Cover.daily.mean..mid.cld.lay.,
                                Wind.Speed.daily.min..10.m.above.gnd.,
                                -Wind.Speed.daily.mean..10.m.above.gnd.))

# trier le dataframe
df.no_corr <- df.no_corr %>% select(order(names(df.no_corr)))
```

```
corrplot(cor(df.no_corr[-5]), tl.cex = 0.8, tl.col = "blue", tl.srt = 45)
```



On recommance ce processus de regarder la significativité des coeficients et leurs impacts en terme de corrélations...

```
reg_test2 <- glm(formula = pluie.demain ~ . ,
           family=binomial,
           data = df.no_corr)

summary(reg_test2)
```

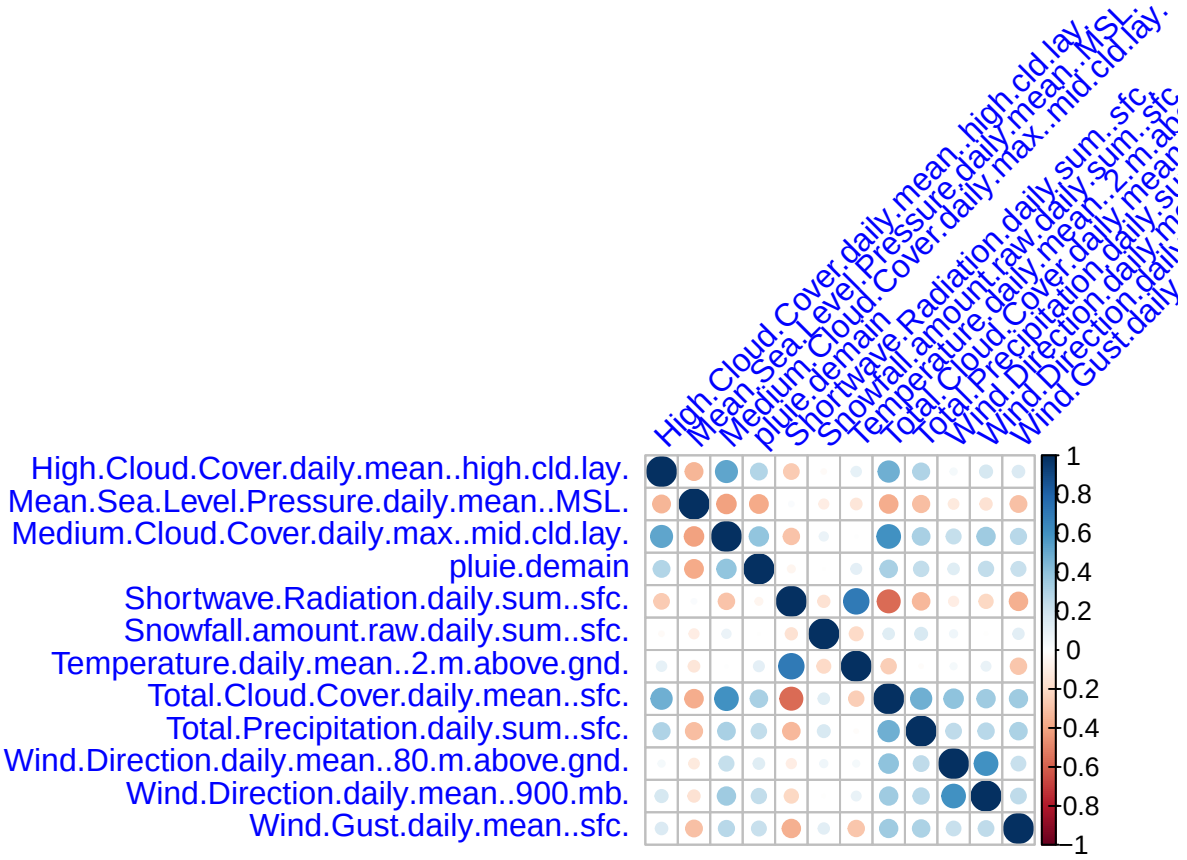
```
##
## Call:
## glm(formula = pluie.demain ~ ., family = binomial, data = df.no_corr)
##
```

```
## Coefficients:
##                                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                                   6.896e+01  1.175e+01   5.871 4.33e-09 ***
## High.Cloud.Cover.daily.mean..high.cld.lay.    5.217e-03  4.854e-03   1.075  0.28246
## Low.Cloud.Cover.daily.mean..low.cld.lay.     -3.445e-03  7.119e-03  -0.484  0.62845
## Mean.Sea.Level.Pressure.daily.mean..MSL.     -6.994e-02  1.139e-02  -6.140 8.25e-10 ***
## Medium.Cloud.Cover.daily.max..mid.cld.lay.    9.444e-03  2.383e-03   3.962 7.42e-05 ***
## Month2                                       -4.567e-01  3.633e-01  -1.257  0.20874
## Month3                                       -8.528e-01  3.903e-01  -2.185  0.02888 *
## Month4                                       -9.024e-01  4.586e-01  -1.967  0.04913 *
## Month5                                       -6.871e-01  5.002e-01  -1.374  0.16951
## Month6                                       -3.792e-01  5.732e-01  -0.662  0.50819
## Month7                                       -5.708e-01  5.766e-01  -0.990  0.32219
## Month8                                       -1.058e+00  5.277e-01  -2.006  0.04488 *
## Month9                                       -1.390e+00  4.584e-01  -3.032  0.00243 **
## Month10                                      -1.014e+00  3.876e-01  -2.616  0.00889 **
## Month11                                      -9.633e-01  3.565e-01  -2.702  0.00690 **
## Month12                                       1.388e-01  3.634e-01   0.382  0.70239
## Relative.Humidity.daily.mean..2.m.above.gnd.  1.298e-04  1.121e-02   0.012  0.99076
## Shortwave.Radiation.daily.sum..sfc.           9.238e-05  1.103e-04   0.838  0.40227
## Snowfall.amount.raw.daily.sum..sfc.          -2.515e-01  1.909e-01  -1.318  0.18766
## Sunshine.Duration.daily.sum..sfc.             2.777e-04  8.757e-04   0.317  0.75116
## Temperature.daily.mean..2.m.above.gnd.        4.704e-02  2.397e-02   1.963  0.04968 *
## Total.Cloud.Cover.daily.mean..sfc.            1.582e-02  1.003e-02   1.576  0.11497
## Total.Precipitation.daily.sum..sfc.           3.113e-02  2.466e-02   1.263  0.20674
## Wind.Direction.daily.mean..10.m.above.gnd.    5.468e-03  5.695e-03   0.960  0.33697
## Wind.Direction.daily.mean..80.m.above.gnd.   -9.017e-03  5.903e-03  -1.527  0.12664
## Wind.Direction.daily.mean..900.mb.            4.238e-03  1.387e-03   3.055  0.00225 **
## Wind.Gust.daily.mean..sfc.                    5.015e-02  2.617e-02   1.916  0.05533 .
## Wind.Speed.daily.mean..80.m.above.gnd.       -5.200e-02  2.853e-02  -1.823  0.06832 .
## Wind.Speed.daily.mean..900.mb.               -1.236e-03  1.101e-02  -0.112  0.91058
## Wind.Speed.daily.min..10.m.above.gnd.         3.697e-02  3.314e-02   1.116  0.26458
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1635.4  on 1179  degrees of freedom
## Residual deviance: 1266.9  on 1150  degrees of freedom
## AIC: 1326.9
##
## Number of Fisher Scoring iterations: 4
```

Meme chose on va retirer les variables qui ont des p-valeurs trop elevées et sont trop corellés à d'autres groupes de variable.

```
df.no_corr <- df.no_corr %>%  select(-Low.Cloud.Cover.daily.mean..low.cld.lay.,
                        -Relative.Humidity.daily.mean..2.m.above.gnd.,
                        -Sunshine.Duration.daily.sum..sfc.,
                        -Wind.Direction.daily.mean..10.m.above.gnd.,
                        -Wind.Speed.daily.min..10.m.above.gnd.,
                        -Wind.Speed.daily.mean..80.m.above.gnd.,
                        -Wind.Speed.daily.mean..900.mb.)
```

```
corrplot(cor(df.no_corr[,-4]), tl.cex = 1, tl.col = "blue", tl.srt = 45)
```



Nous avons un graphe très satisfaisant, nous pouvons considérer ce dataframe *df.no_corr* pour modélioser nos modèles

Modélisation et recherche du meilleur modèle

Commençons par chercher "à la main" un modèle logistique en sélectionnant les variables

A la main

```
reg.no_corr <- glm(formula = pluie.demain ~ . ,
           family=binomial,
           data = df.no_corr)

summary(reg.no_corr)
```

```
##
## Call:
## glm(formula = pluie.demain ~ ., family = binomial, data = df.no_corr)
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                             6.671e+01  1.153e+01   5.786 7.21e-09 ***
## High.Cloud.Cover.daily.mean..high.cld.lay.  5.624e-03  4.222e-03   1.332  0.18280
## Mean.Sea.Level.Pressure.daily.mean..MSL.   -6.785e-02  1.124e-02  -6.039 1.55e-09 ***
## Medium.Cloud.Cover.daily.max..mid.cld.lay.  9.499e-03  2.234e-03   4.251 2.13e-05 ***
```

```
## Month2                                      -4.333e-01  3.626e-01  -1.195  0.23203
## Month3                                      -8.768e-01  3.828e-01  -2.291  0.02198 *
## Month4                                      -9.003e-01  4.417e-01  -2.039  0.04150 *
## Month5                                      -6.951e-01  4.839e-01  -1.436  0.15089
## Month6                                      -3.989e-01  5.527e-01  -0.722  0.47044
## Month7                                      -5.880e-01  5.565e-01  -1.057  0.29071
## Month8                                      -1.080e+00  5.141e-01  -2.101  0.03566 *
## Month9                                      -1.393e+00  4.481e-01  -3.109  0.00188 **
## Month10                                     -1.015e+00  3.842e-01  -2.640  0.00828 **
## Month11                                     -9.664e-01  3.543e-01  -2.728  0.00638 **
## Month12                                      1.243e-01  3.598e-01   0.345  0.72977
## Shortwave.Radiation.daily.sum..sfc.          1.189e-04  8.821e-05   1.348  0.17754
## Snowfall.amount.raw.daily.sum..sfc.         -2.856e-01  1.873e-01  -1.524  0.12740
## Temperature.daily.mean..2.m.above.gnd.       4.977e-02  2.326e-02   2.139  0.03240 *
## Total.Cloud.Cover.daily.mean..sfc.           1.183e-02  4.111e-03   2.879  0.00400 **
## Total.Precipitation.daily.sum..sfc.          2.965e-02  2.344e-02   1.265  0.20588
## Wind.Direction.daily.mean..80.m.above.gnd. -3.411e-03  1.548e-03  -2.204  0.02755 *
## Wind.Direction.daily.mean..900.mb.           3.858e-03  1.306e-03   2.954  0.00314 **
## Wind.Gust.daily.mean..sfc.                   2.265e-02  8.458e-03   2.678  0.00740 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1635.4  on 1179  degrees of freedom
## Residual deviance: 1271.0  on 1157  degrees of freedom
## AIC: 1317
##
## Number of Fisher Scoring iterations: 4
```

Retirons les coefficients avec les p-valeurs les plus grandes (non significatives) et variables pour les groupes de variables correlées (les vents, les températures, les nuages...).

```
df.reg_manual <- df.no_corr  %>% select(-High.Cloud.Cover.daily.mean..high.cld.lay.,
                             -Snowfall.amount.raw.daily.sum..sfc.,
                             -Temperature.daily.mean..2.m.above.gnd.,
                             -Total.Precipitation.daily.sum..sfc.,
                             -Shortwave.Radiation.daily.sum..sfc.)

# regression avec selection manuelle
reg_manual <- glm(formula = pluie.demain ~ . ,
           family=binomial (link = "logit"),
           data = df.reg_manual)
summary(reg_manual)
```

```
##
## Call:
## glm(formula = pluie.demain ~ ., family = binomial(link = "logit"),
##     data = df.reg_manual)
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                             70.126158  11.244443   6.237 4.47e-10 ***
## Mean.Sea.Level.Pressure.daily.mean..MSL.  -0.070563   0.010965  -6.435 1.23e-10 ***
## Medium.Cloud.Cover.daily.max..mid.cld.lay.  0.011360   0.002110   5.383 7.31e-08 ***
```

```
## Month2                                       -0.414527   0.348356  -1.190  0.23406
## Month3                                       -0.433369   0.341593  -1.269  0.20456
## Month4                                       -0.129351   0.340819  -0.380  0.70429
## Month5                                        0.291171   0.341629   0.852  0.39405
## Month6                                        0.871942   0.354370   2.461  0.01387 *
## Month7                                        0.709592   0.354042   2.004  0.04504 *
## Month8                                        0.103888   0.343207   0.303  0.76212
## Month9                                       -0.499415   0.345539  -1.445  0.14837
## Month10                                      -0.494813   0.338557  -1.462  0.14387
## Month11                                      -0.713313   0.337619  -2.113  0.03462 *
## Month12                                      -0.009351   0.349763  -0.027  0.97867
## Total.Cloud.Cover.daily.mean..sfc.            0.008517   0.003011   2.829  0.00467 **
## Wind.Direction.daily.mean..80.m.above.gnd. -0.004288   0.001501  -2.856  0.00429 **
## Wind.Direction.daily.mean..900.mb.            0.004891   0.001247   3.921 8.81e-05 ***
## Wind.Gust.daily.mean..sfc.                    0.020243   0.008067   2.509  0.01210 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1635.4  on 1179  degrees of freedom
## Residual deviance: 1288.2  on 1162  degrees of freedom
## AIC: 1324.2
##
## Number of Fisher Scoring iterations: 4
```

Nous avons a présent un modèle sélectionne à la main qui convient.

Avec la procédure automatique

Nous pouvons lancer la procédure de selection automatique à partir du dataset non corrélé pour voir si il fait mieux.

```r
# regression avec selection optimisée
reg.opt.logit <- step(reg.no_corr, direction = "both",trace = 0)

summary(reg.opt.logit)
```

```
##
## Call:
## glm(formula = pluie.demain ~ Mean.Sea.Level.Pressure.daily.mean..MSL. +
##     Medium.Cloud.Cover.daily.max..mid.cld.lay. + Month + Temperature.daily.mean..2.m.above.gnd. +
##     Total.Cloud.Cover.daily.mean..sfc. + Wind.Direction.daily.mean..80.m.above.gnd. +
##     Wind.Direction.daily.mean..900.mb. + Wind.Gust.daily.mean..sfc.,
##     family = binomial, data = df.no_corr)
##
## Coefficients:
##                                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)                             67.580772  11.315255   5.973 2.34e-09 ***
## Mean.Sea.Level.Pressure.daily.mean..MSL.   -0.068453   0.011032  -6.205 5.47e-10 ***
## Medium.Cloud.Cover.daily.max..mid.cld.lay.  0.010279   0.002141   4.800 1.58e-06 ***
## Month2                                  -0.368859   0.355626  -1.037 0.299638
## Month3                                  -0.672181   0.352895  -1.905 0.056811 .
## Month4                                  -0.637812   0.377868  -1.688 0.091426 .
## Month5                                  -0.403521   0.405012  -0.996 0.319095
## Month6                                  -0.091485   0.459480  -0.199 0.842180
```

```
## Month7                                        -0.348093   0.477742  -0.729 0.466233
## Month8                                        -0.914789   0.463980  -1.972 0.048654 *
## Month9                                        -1.279648   0.422857  -3.026 0.002477 **
## Month10                                       -0.985176   0.374160  -2.633 0.008463 **
## Month11                                       -0.943839   0.350433  -2.693 0.007074 **
## Month12                                        0.033814   0.356201   0.095 0.924372
## Temperature.daily.mean..2.m.above.gnd.         0.069973   0.021304   3.284 0.001022 **
## Total.Cloud.Cover.daily.mean..sfc.             0.011379   0.003150   3.612 0.000304 ***
## Wind.Direction.daily.mean..80.m.above.gnd. -0.003633   0.001519  -2.393 0.016726 *
## Wind.Direction.daily.mean..900.mb.             0.003852   0.001285   2.996 0.002732 **
## Wind.Gust.daily.mean..sfc.                     0.023003   0.008261   2.784 0.005361 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1635.4  on 1179  degrees of freedom
## Residual deviance: 1277.2  on 1161  degrees of freedom
## AIC: 1315.2
##
## Number of Fisher Scoring iterations: 4
```

Le modèle trouvé par l'algorithme qui sélectionne le meilleur modèle itérativement selon son AIC (Ici AIC car nous cherchons la performance brute) trouve quelque chose qui ressemble à notre sélection manuelle. Son AIC est meilleur, on peut considérer ce modèle comme étant plus performant que le précedant.

Pour résoudre les problèmes de variables corrélées et perdre un minimum d'information. Nous pouvons choisir de faire une ACP sur nos données puis d'utiliser ces composantes principales dans le modèle. (Qui revient à faire une PLS moins élaborée).

L'avantage est que nous utilisons toute l'information contenue dans l'ensemble des variables et n'avons aucun problème de multicolinéarité.

Le désavantage est dans l'interprétation, et le fait qu'il faille transformer les prochaines données à inférer (ramener nos colonnes à des combinaisons linéaires pour obtenir nos composantes principales).

Avec une Analyse en Composantes Principales

```
library(FactoMineR)

acp <- PCA(df[, !(names(df) %in% c("pluie.demain","Month"))],scale.unit = TRUE, ncp = 10)
```

```
## Warning: ggrepel: 28 unlabeled data points (too many overlaps). Consider increasing max.overlaps
```

```
acp_data <- as.data.frame(acp$ind$coord)

acp_data$Y <- df$pluie.demain

reg_acp <- glm(Y ~ . ,
    family = binomial,
    data = acp_data)

summary(reg_acp)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial, data = acp_data)
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.019857   0.068415   0.290   0.7716
## Dim.1        0.232047   0.021024  11.038  < 2e-16 ***
## Dim.2        0.155854   0.030475   5.114 3.15e-07 ***
## Dim.3        0.411136   0.035770  11.494  < 2e-16 ***
## Dim.4       -0.048642   0.042112  -1.155   0.2481
## Dim.5       -0.116898   0.051315  -2.278   0.0227 *
## Dim.6       -0.019960   0.049657  -0.402   0.6877
## Dim.7        0.007544   0.059518   0.127   0.8991
## Dim.8        0.095974   0.067287   1.426   0.1538
## Dim.9        0.006182   0.068455   0.090   0.9280
## Dim.10       0.020129   0.070692   0.285   0.7758
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1635.4  on 1179  degrees of freedom
## Residual deviance: 1307.4  on 1169  degrees of freedom
## AIC: 1329.4
##
## Number of Fisher Scoring iterations: 4
```

Finalement on voit que l'AIC n'est pas significativement plus bas, ainsi on ne va pas considérer cette option.

Avec une regression Probit

Le modèle probit, par rapport au modèle logit, offre une gestion plus précise des observations extrêmes en utilisant une distribution normale cumulative, ce qui peut conduire à des prédictions légèrement plus réalistes dans certains contextes.

```
reg.no_corr <- glm(formula = pluie.demain ~ . ,
           family=binomial (link = "probit"),
           data = df.no_corr)
# regression avec selection optimisée
reg.opt.probit <- step(reg.no_corr, direction = "both",trace = 0)

summary(reg.opt.probit)
```

```
##
## Call:
## glm(formula = pluie.demain ~ Mean.Sea.Level.Pressure.daily.mean..MSL. +
##     Medium.Cloud.Cover.daily.max..mid.cld.lay. + Month + Snowfall.amount.raw.daily.sum..sfc. +
##     Temperature.daily.mean..2.m.above.gnd. + Total.Cloud.Cover.daily.mean..sfc. +
##     Wind.Direction.daily.mean..80.m.above.gnd. + Wind.Direction.daily.mean..900.mb. +
##     Wind.Gust.daily.mean..sfc., family = binomial(link = "probit"),
##     data = df.no_corr)
##
## Coefficients:
##                                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                              40.9857741  6.6560679   6.158 7.38e-10 ***
## Mean.Sea.Level.Pressure.daily.mean..MSL.  -0.0414241  0.0064824  -6.390 1.66e-10 ***
## Medium.Cloud.Cover.daily.max..mid.cld.lay.  0.0060962  0.0012665   4.814 1.48e-06 ***
```

```
## Month2                                       -0.2247884  0.2085406  -1.078  0.28107
## Month3                                       -0.4215599  0.2071856  -2.035  0.04188 *
## Month4                                       -0.3888599  0.2230114  -1.744  0.08122 .
## Month5                                       -0.2333106  0.2388178  -0.977  0.32860
## Month6                                       -0.0243496  0.2711043  -0.090  0.92843
## Month7                                       -0.1701704  0.2821115  -0.603  0.54637
## Month8                                       -0.5183019  0.2752024  -1.883  0.05965 .
## Month9                                       -0.7526042  0.2498976  -3.012  0.00260 **
## Month10                                      -0.5761881  0.2213654  -2.603  0.00924 **
## Month11                                      -0.5530749  0.2069951  -2.672  0.00754 **
## Month12                                       0.0167670  0.2087162   0.080  0.93597
## Snowfall.amount.raw.daily.sum..sfc.          -0.1558928  0.1107049  -1.408  0.15908
## Temperature.daily.mean..2.m.above.gnd.        0.0374623  0.0126884   2.952  0.00315 **
## Total.Cloud.Cover.daily.mean..sfc.            0.0067718  0.0018729   3.616  0.00030 ***
## Wind.Direction.daily.mean..80.m.above.gnd. -0.0021318  0.0008980  -2.374  0.01760 *
## Wind.Direction.daily.mean..900.mb.            0.0021740  0.0007592   2.863  0.00419 **
## Wind.Gust.daily.mean..sfc.                    0.0145179  0.0048502   2.993  0.00276 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1635.4  on 1179  degrees of freedom
## Residual deviance: 1276.7  on 1160  degrees of freedom
## AIC: 1316.7
##
## Number of Fisher Scoring iterations: 4
```

On voit que le modèle probit fait légèrement mieux que son équivalent logit. Nous allons le retenir pour la suite.

Evaluation du modèle sur les données (C)

Nous allons effectuer une cross-validation, pour tester les erreurs produites par le modèle en conditions réelles, c'est à dire entraîné avec une partie des données puis évalué sur le reste du jeu de données, le tout répété pour pouvoir comparer notre modèle logistique du probit.

Nous choisissons

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
# le nombre de folds à effectuer

k <- 10
n <- nrow(df)

results <- data.frame(
  MODEL = character(0),
```

```
  R2 = numeric(0),
  RMSE = numeric(0),
  MAE = numeric(0)
)

for (model.type in c("logit","probit")){
  for (i in 1:k) {
    # désordonner les données de façon aléatoires
    data <- df.no_corr[sample(n),]

    #répartir les données en 80% train / 20% test
    data.train  <- data[1:floor(n*0.8), ]
    data.test <- data[floor(n*0.8):n, ]

    # construire le modèle
    if (model.type=="logit"){
      model.trained <- glm(formula = pluie.demain ~ Mean.Sea.Level.Pressure.daily.mean..MSL. +
        Medium.Cloud.Cover.daily.max..mid.cld.lay. + Month + Temperature.daily.mean..2.m.above.gnd. +
        Total.Cloud.Cover.daily.mean..sfc. + Wind.Direction.daily.mean..80.m.above.gnd. +
        Wind.Direction.daily.mean..900.mb. + Wind.Gust.daily.mean..sfc.,
        family = binomial(link = "logit"),
        data = data.train)
    } else {
      model.trained <- glm(formula = pluie.demain ~ Mean.Sea.Level.Pressure.daily.mean..MSL. +
        Medium.Cloud.Cover.daily.max..mid.cld.lay. + Month + Snowfall.amount.raw.daily.sum..sfc. +
        Temperature.daily.mean..2.m.above.gnd. + Total.Cloud.Cover.daily.mean..sfc. +
        Wind.Direction.daily.mean..80.m.above.gnd. + Wind.Direction.daily.mean..900.mb. +
        Wind.Gust.daily.mean..sfc.,
        family = binomial(link = "probit"),
        data = data.train)
    }
    # prédire et les évaluer selon plusieurs critères
    predictions <- predict(model.trained, data.test)
    res <- data.frame( MODEL = model.type,
              R2 = R2(predictions, data.test$pluie.demain),
              RMSE = RMSE(predictions, data.test$pluie.demain),
              MAE = MAE(predictions, data.test$pluie.demain))

    results <- rbind(results, res)
  }
}

results %>% group_by(MODEL) %>% summarize_all(mean)
```

```
## # A tibble: 2 x 4
##   MODEL     R2  RMSE   MAE
##   <chr>  <dbl> <dbl> <dbl>
## 1 logit  0.228 1.37  1.07
## 2 probit 0.227 0.884 0.703
```

On peut voir que le modèle Probit est bien meilleur que sont concurrent logit, on s'y attendait avec un AIC plus faible, mais une cross-validation sur des données réelles nous permet de nous rendre compte à quel point.

Ce modèle reg.opt.probit semble être le meilleur trouvé jusqu'ici, nous allons à présent le considérer dans la suite de l'étude.

3. Prediction et test du modèle

Commençons les prédiction avec notre modèle. Dans un premier temps sur nos données pour se rendre compte de sa performance prédictive.

Le modèle probit (tout comme le logistique) nous fournis une inférence, nombre compris entre 0 et 1.

Pour interpréter ce résultat nous devons fixer un seuil de décision pour-lequel on considère le résultat positif ou négatif.

Le choix du seuil dépend du contexte et de la performance du modèle. Le modèle va toujours faire des erreures, faux positifs et négatifs, il s'agit de trouver le bon compromis.

En effet en fonction du contexte, il va être plus ou moins critique d'affirmer ou non un diagnostique.

- Test sensible: Prenons le cas d'un test covid, il est préférable que le test indique positif même si nous ne sommes pas certain que la personne est en réalité malade, elle peux etre saine (=faux positif). L'inverse est cependant critique, une personne malade non détectée est très grave (test négatif alors que la personne est malade = faux négatif). On va chercher à réduire le nombre de faux négatifs.
- Test spécifique: Prenons le dépistage du cancer du pancréas, les traitements peuvent être invasifs et ont des effets secondaires significatifs. Le diagnostic erroné peut entraîner des traitements inutiles et des interventions chirurgicales non souhaités. Ici on va chercher à réduire le nombre de faux-positifs et à valider le diagnostique qu'avec certitude.
- Compromis, meilleure prédictions: Dans notre cas où nous cherchons à être le plus fiable possible et d'avoir une balance correcte entre faux négatifs et faux positifs, nous cherchons à les minimiser. Ainsi nous pouvons chercher un seuil t idéal tel qui maximise le nombre prédictions correctes.

Nous verrons 2 méthodes pour cela: - le critère de youden : qui minimise *Sensibilité + Specificité - 1* respectivement sensibilité étant le taux de Vrais positifs, et Specificité le taux de Vrais négatifs. - le critère de minimisation de la distance au coin supérieur gauche.

Ici nous décidons le compris de se tromper le moins souvent, donc on cherchera un critère qui minimise les Faux négatifs et positifs.

Nous fixons un seuil à 0.5 pour l'instant.

Matrice de confusion

```
df.eval <- data.frame(
  pluie.demain = df.no_corr[,"pluie.demain"],
  regression = predict(reg.opt.probit, df.no_corr, type = "response")
  )

df.eval$prediction <- ifelse(df.eval$regression > 0.5, 1, 0)

head(df.eval,5)
```

```
##   pluie.demain regression prediction
## 1        FALSE  0.7469829          1
## 2        FALSE  0.2807289          0
## 3         TRUE  0.8249054          1
## 4         TRUE  0.8844516          1
## 5         TRUE  0.9086641          1
```

```
# Matrice de confusion
table(df.eval$pluie.demain, df.eval$prediction)
```

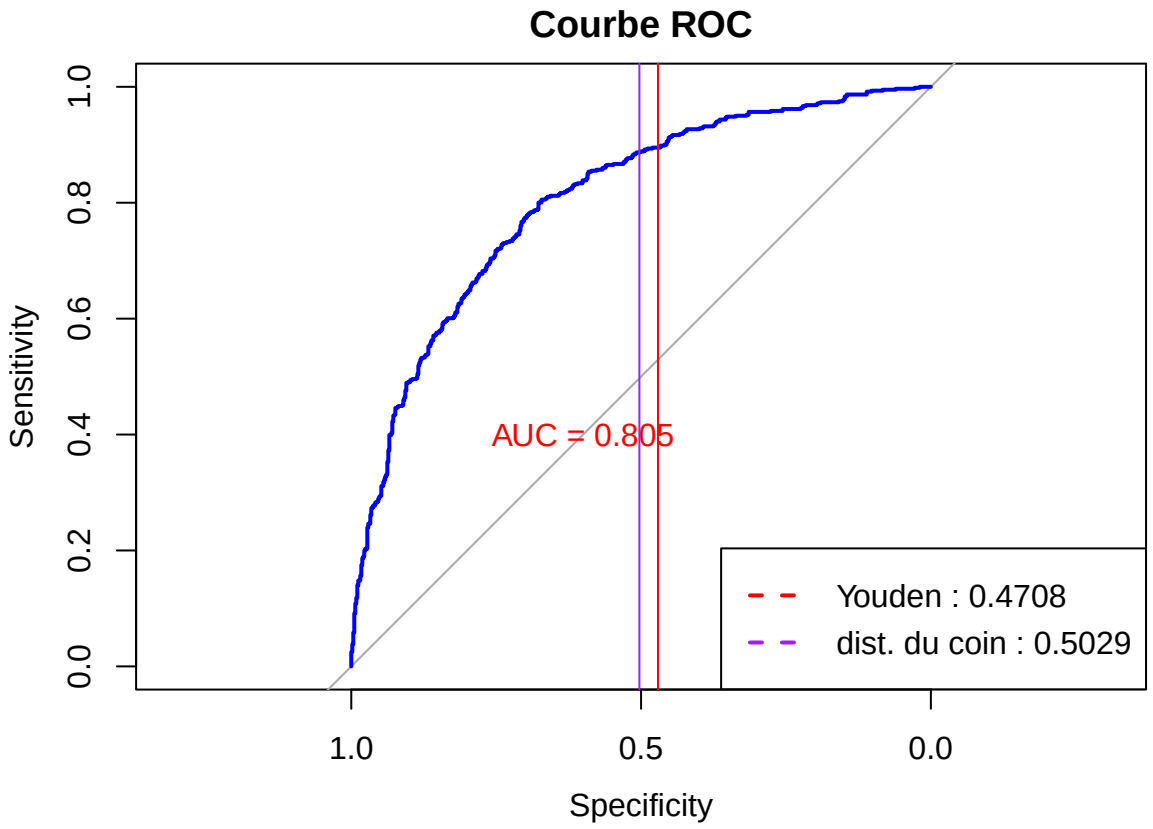
```
##
```

```
##           0   1
##   FALSE 406 173
##   TRUE  138 463
```

On voit ici les faux-positifs et négatifs, on constate à l'œil qu'un bon cinquième des données sont mal prédites, mais cela reste relativement satisfaisant.

courbe ROC

```r
# Créer l'objet ROC
pred <- predict(reg.opt.probit, type = "response")
roc_obj <- roc(df$pluie.demain, pred)

# Tracer la courbe ROC
plot(roc_obj, col = "blue", main = "Courbe ROC")

# Ajouter l'AUC au graphique
auc_value <- auc(roc_obj)
text(0.6, 0.4, paste("AUC =", round(auc_value, 3)), col = "red")

# Calcul des seuils optimaux
threshold_youden <- round(coords(roc_obj, "best", ret = "threshold", best.method = "youden"),4)
threshold_dist <- round(coords(roc_obj, "best", ret = "threshold", best.method = "closest.topleft"),4)

# légende et tracés
abline(v=threshold_youden, col="red")
abline(v=threshold_dist, col = "purple")

legend("bottomright", legend = c(paste("Youden :",threshold_youden),
                                 paste("dist. du coin :",threshold_dist)),
       col = c("red", "purple"), lty = 2, lwd = 2)
```



La courbe ROC test la matrice de confusion avec tout les seuils possibles. elle permet de voit la qualité de prédiction et trouver le meilleur seuil.

AUC ~ 0.8 , Bonne performance

Nous pouvons prendre une moyenne des deux seuils pour fixer une seuil optimum.

```
# seuil optimal est la moyenne des deux methodes précedants
threshold_opt = mean(c(threshold_dist$threshold, threshold_youden$threshold))
# Recalcul des prédictions avec le nouveau seuil
df.eval$prediction <- ifelse(df.eval$regression > threshold_opt, 1, 0)
# Matrice de confusion
table(df.eval$pluie.demain, df.eval$prediction)
```

```
##
##           0   1
##   FALSE 400 179
##   TRUE  130 471
```

Predictions sur le fichier météo

Utilisons notre modèle entraîné sur des nouvelles données afin de les prédire.

```
# importation et transformation du jeu de données
df.test <- read.csv("./meteo.test.csv")
df.test$Month <- factor(df.test$Month)

# appeler le modèle et inférer les données
df.test.pred <- data.frame(regression = predict(reg.opt.probit, df.test, type = "response"))
```

```r
# appliquer la prévision avec le seuil
df.test.pred$prediction <- ifelse(df.test.pred$regression > threshold_opt, T, F)

# concaténer les données de prévisions et le jeu de données
df.test.pred <- cbind(df.test[,1:4], df.test.pred )

# sauver les résultats dans un fichier de sortie
write.csv(df.test.pred, file = "previsions.csv", row.names = FALSE)

head(df.test.pred)
```

```
##   id Year Month Day regression prediction
## 1 36 2010     7   6  0.5593040       TRUE
## 2 44 2010     7  14  0.7767216       TRUE
## 3 54 2010     7  24  0.6930674       TRUE
## 4 58 2010     7  28  0.7431742       TRUE
## 5 74 2010     8  13  0.6179845       TRUE
## 6 86 2010     8  25  0.3075500      FALSE
```

** Les données sont disponible dans le fichier *previsions.csv*

Bien que nous ne puissions pas vérifier la qualité des données, les observations précédentes sur la cross-validation et la courbe ROC nous donnent une haute qualité d'ajustement des données avec le modèle.

Les nouvelles données à inférer se trouvent être de la même source que les premières, ainsi on peut s'attendre que les prévisions soient bonnes.

Conclusion

Prise de recul générale :

Cette étude nous illustre un cas d'utilisation concrète d'une régression logistique linéaire (et probit) sur des données structurées en apprentissage supervisé.

Il est très intéressant d'étudier les hypothèses des modèles pour valider mathématiquement leurs fondements et comprendre s'il a un sens face à nos données.

Les GLM nous offrent la possibilité d'adapter la fonction de lien (& distribution des erreurs) pour mieux correspondre aux caractéristiques des données, permettant ainsi de modéliser une large gamme de types de variables dépendantes au-delà des simples relations linéaires.

Critique et améliorations:

Des axes d'améliorations sont envisageables pour améliorer la qualité des prédictions:

- inclure la temporalité des des données dans le modèle (inclure la saisonnalité des données, c'est à dire le mois de l'année en tant que variable qualitative)
- proposer une plus grande granularité temporelle des données (3 par jour), avec une sortie pluie.demain en probabilité plutôt que binaire
- inclure des variable météorologiques (agrégées) qui correspondraient à un historique des jours précédents, pour avoir une notion de série temporelle dans l'impact de la pluie demain, pour ne pas seulement prendre en compte les phénomènes présents
- utiliser d'une fonction STEP basé sur le BIC qui prend plus en compte la parcimonie du modèle que sa performance brute