Modèle linéaire généralisé et Choix de modèles

An

08/01/2022

Introduction

L'objectif de ce devoir serait d'effectuer les analyses pertinentes sur les données de météo à Bâle. On cherche à prétendire s'il pleuvra le lendemain dans cette ville. Pour cette variable d'intérêt :

- proposer et valider un modèle ;
- proposer une prédiction binaire pour les lendemains des journées incluses dans le fichier meteo.test.csv ;

Analyse de données et Pré-traitement

Définir le répertoir de travail et lire le jeu de données

```
rm(list=ls());
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

library(readr)
d <- read_csv("meteo.train.csv")
```

On verra s'il y a des valeurs manquante et puis les enlever.

```
apply(apply(d,2,is.na),2,sum)
d <- d[complete.cases(d), ]
```

Dans notre jeu de données, j'ai constaté que les 6 premieres colones sont les informations précisant le moment d'observation. L'objectif de notre travail serait de prédire la pluie de lendemain en fonction des conditions météorologiques sans dépendement du moment d'observation. On va donc les enlever.

```
d.meteo = d[-c(1:6)]
```

Je crée une fonction qui permet de transformer le libellé des colonnes en raccourci

```
 clean_heads_name <- function(.data, unique = FALSE) {
    n <- if (is.data.frame(.data)) colnames(.data) else .data
    n <- gsub("Direction","dir",n)
    n <- gsub("^Temperature","te",n)
    n <- gsub("^Wind","wi",n)
    n <- gsub("Duration","dur",n)
    n <- gsub("^Relative","rel",n)
    n <- gsub("^relative","rel",n)
    n <- gsub("humidity","hu",n)
    n <- gsub("Humidity","hu",n)
    n <- gsub("Pressure","press",n)
    n <- gsub("pressure","press",n)
    n <- gsub("^Cloud","clo",n)
    n <- gsub("cloud","clo",n)
    n <- gsub("[^a-zA-Z0-9_]+", "_", n)
    n <- gsub("([A-Z][a-z])", "_\\1", n)
    n <- tolower(trimws(n))
    n <- gsub("(^_+|_+$)", "", n)

    if (unique) n <- make.unique(n, sep = "_")

    if (is.data.frame(.data)) {
      colnames(.data) <- n
      .data
    } else {
      n
    }
  }

library(dplyr)
d.meteo <- d.meteo %>%  clean_heads_name()
```

Méthodologie

Pour notre jeu de données, l'idée est de le diviser en deux parties : l'une (90%) pour construire notre modèle et l'autre (10%) pour valider sa prédiction (Cross-Validation)

```
index<-1:nrow(d.meteo)
testindex<-c(1:trunc(length(index)*0.9))
### il fautdrait utiliser  la fonction sample() afin d'extraire notre échantillon d'entraînement
### mais cela va impacter le résultat de l'analyse de modèle dans le rapport car à chaque fois le fichier rmd est rendu, l'échantillon va changer

d.trainset = d.meteo[testindex,]
#enlever la variable à expliquer "Pluie_demain"
d.trainset.NoRainResult = d.trainset[-c(41)]
d.testset  = d.meteo[-testindex,]
```

```
library(corrplot)
corrplot(cor(d.trainset.NoRainResult), type="upper", order="hclust", tl.col="black", tl.srt=45)
```

Il y a bien une forte corrélation d'entre certaines variables : on n'utilisera qu'une seule parmis les variables quasi colinéaires. Donc, le modèle qu'on va chercher ne contients pas certainement tous les variables

On cherche à expliquer la variable d'intérêt "Pluie_Demain". Dans mon observation, cette variable présente seulement 2 modalités : "TRUE" ou "FALSE". Dans ce cas-là, on utilise la régression logistique binaire (c'est-à-dire de type « oui / non » ou « vrai / faux ») pour construire notre modèle logistique.

Il y a plusieurs critères pour comparer 2 modèles : R^2, R^2 ajusté, Cp Mallow, AIC et BIC Pour savoir lequel qui nous intéresse, on a une formule
$(SCR(m0) - SCR(m1))/SCR(m1) \times (n - |m0| - 1) \leq q$
dont m0,m1 sont les degrée de liberté des modèle m0 et m1. n est le nombre des données
1. q = 4 pour le test de Fisher
2. $q = -\infty$ pour le R2
3. q = 1 pour le R2 ajusté
4. q = 2 pour le Cp de Mallows
5. $q = 2\ n \times (n - |m0| - 1)$ pour AIC
6. $q = \log n/n \times (n - |m0| - 1)$ pour BIC

Vu la taille de nos données est importante, AIC et BIC seraient préférable pour être considérés comme les critères pricipaux pour la sélection de modèle
Pour ce faire, Il y a plusieurs méthodes : Méthode pas à pas ou Sélection de modèles exhaustive. On va proposer 2 modèles différents à partir de la méthode Pas à Pas combinée avec un des 6 crtitères étudiées précédement et choisir lequel qui nous conviendrait mieux à l'aide de sa qualité de prédiction.

L'objectif est de choisir un modèle dont AIC et BIC sont les plus petis. On n'est donc pas sûr d'obtenir un minimum global. Pour la méthode "pas à pas", nous avons 3 manières réalisées :
1. Méthode ascendante (forward selection) : A chaque pas, une variable est ajoutée au modèle, celle qui a l'apport le plus important (que l'on peut mesurer pour un test par celle qui a la plus petite p-valeur).
2.Méthode descendante (backard selection) : A chaque pas, une variable est enlevée au modèle, celle qui le plus fort impact (que l'on peut mesurer pour un test par celle qui a la plus grande p-valeur).
3.Méthode progressive (stepwise selection) : C'est la même méthode que la méthode ascendante, à l'exception que'à chaque étape, on peut remettre en cause une variable présente dans le modèle selon la méthode descendante.

Sélection de modèle

On définit d'abord le modèle complet et le modèle minimum

```
model_complet<-glm(d.trainset$pluie_demain~.,data=d.trainset.NoRainResult,family=binomial())
model_min<-glm(d.trainset$pluie_demain~1,data=d.trainset.NoRainResult,family=binomial())
```

Définissons le modèle de manière ascendante et de manière descendante

```
model_step_bck <- step(model_complet,direction="backward",k=log(nrow(d.trainset)))
model_step_fwd <- step(model_min,direction="forward",scope=list(lower=formula(model_min),upper=formula(model_comp
let)))
```

```
summary(model_step_bck)
```

```
##
## Call:
## glm(formula = d.trainset$pluie_demain ~ te_daily_mean_2_m_above_gnd +
##     mean__sea__level_press_daily_mean_msl + medium__cloud__cover_daily_mean_mid_cld_lay +
##     wi__speed_daily_mean_80_m_above_gnd + wi_dir_daily_mean_900_mb +
##     te_daily_min_2_m_above_gnd + mean__sea__level_press_daily_max_msl +
##     mean__sea__level_press_daily_min_msl + total__cloud__cover_daily_max_sfc +
##     wi__speed_daily_max_10_m_above_gnd + wi__speed_daily_min_10_m_above_gnd,
##     family = binomial(), data = d.trainset.NoRainResult)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5755  -0.8410   0.1456   0.8683   2.9887
##
## Coefficients:
##                                          Estimate Std. Error z value
## (Intercept)                             70.850607  12.882412   5.500
## te_daily_mean_2_m_above_gnd              0.218716   0.050385   4.341
## mean__sea__level_press_daily_mean_msl    0.587794   0.143243   4.103
## medium__cloud__cover_daily_mean_mid_cld_lay  0.020439   0.003261   6.267
## wi__speed_daily_mean_80_m_above_gnd     -0.105637   0.031094  -3.397
## wi_dir_daily_mean_900_mb                 0.004313   0.001123   3.842
## te_daily_min_2_m_above_gnd              -0.180812   0.054554  -3.314
## mean__sea__level_press_daily_max_msl    -0.301335   0.075546  -3.989
## mean__sea__level_press_daily_min_msl    -0.360345   0.079479  -4.534
## total__cloud__cover_daily_max_sfc        0.009537   0.003194   2.986
## wi__speed_daily_max_10_m_above_gnd       0.083920   0.021016   3.993
## wi__speed_daily_min_10_m_above_gnd       0.137015   0.039467   3.472
##                                          Pr(>|z|)
## (Intercept)                             3.80e-08 ***
## te_daily_mean_2_m_above_gnd             1.42e-05 ***
## mean__sea__level_press_daily_mean_msl   4.07e-05 ***
## medium__cloud__cover_daily_mean_mid_cld_lay 3.67e-10 ***
## wi__speed_daily_mean_80_m_above_gnd     0.000680 ***
## wi_dir_daily_mean_900_mb                0.000122 ***
## te_daily_min_2_m_above_gnd              0.000918 ***
## mean__sea__level_press_daily_max_msl    6.64e-05 ***
## mean__sea__level_press_daily_min_msl    5.79e-06 ***
## total__cloud__cover_daily_max_sfc       0.002827 **
## wi__speed_daily_max_10_m_above_gnd      6.52e-05 ***
## wi__speed_daily_min_10_m_above_gnd      0.000517 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1472.2  on 1061  degrees of freedom
## Residual deviance: 1124.9  on 1050  degrees of freedom
## AIC: 1148.9
##
## Number of Fisher Scoring iterations: 4
```

```
summary(model_step_fwd)
```

```
##
## Call:
## glm(formula = d.trainset$pluie_demain ~ mean__sea__level_press_daily_min_msl +
##     medium__cloud__cover_daily_max_mid_cld_lay + wi_dir_daily_mean_900_mb +
##     te_daily_max_2_m_above_gnd + wi__gust_daily_max_sfc + medium__cloud__cover_daily_mean_mid_cld_lay +
##     snowfall_amount_raw_daily_sum_sfc + te_daily_min_2_m_above_gnd +
##     total__cloud__cover_daily_mean_sfc, family = binomial(),
##     data = d.trainset.NoRainResult)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4105  -0.8627   0.2476   0.8387   2.7965
##
## Coefficients:
##                                          Estimate Std. Error z value
## (Intercept)                             65.828019  11.935251   5.515
## mean__sea__level_press_daily_min_msl    -0.068917   0.011654  -5.914
## medium__cloud__cover_daily_max_mid_cld_lay   0.007767   0.002442   3.181
## wi_dir_daily_mean_900_mb                 0.003666   0.001139   3.219
## te_daily_max_2_m_above_gnd               0.128953   0.032295   3.993
## wi__gust_daily_max_sfc                   0.022152   0.006191   3.578
## medium__cloud__cover_daily_mean_mid_cld_lay  0.006372   0.004721   1.350
## snowfall_amount_raw_daily_sum_sfc       -0.370881   0.188202  -1.971
## te_daily_min_2_m_above_gnd              -0.090384   0.035824  -2.523
## total__cloud__cover_daily_mean_sfc       0.010769   0.004655   2.313
##                                         Pr(>|z|)
## (Intercept)                             3.48e-08 ***
## mean__sea__level_press_daily_min_msl    3.35e-09 ***
## medium__cloud__cover_daily_max_mid_cld_lay  0.001470 **
## wi_dir_daily_mean_900_mb                0.001285 **
## te_daily_max_2_m_above_gnd              6.53e-05 ***
## wi__gust_daily_max_sfc                  0.000346 ***
## medium__cloud__cover_daily_mean_mid_cld_lay 0.177063
## snowfall_amount_raw_daily_sum_sfc       0.048764 *
## te_daily_min_2_m_above_gnd              0.011636 *
## total__cloud__cover_daily_mean_sfc      0.020705 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1472.2  on 1061  degrees of freedom
## Residual deviance: 1145.5  on 1052  degrees of freedom
## AIC: 1165.5
##
## Number of Fisher Scoring iterations: 4
```

Vu l'AIC du model_step_bck est plus petit que celui du model_step_fwd. On va essayer d'analyser ce modèle On commence par comparer notre modèle au modèle sans covariable

```
pchisq(1472.2-1124.9, 1061     - 1050    , lower = F)
```

```
## [1] 9.031883e-68
```

On obtient une p-valeur très faible : on rejette le modèle sans covariable. Ce dernier exlique que notre modèle est donc utile.
Comparons maintenant notre modèle au modèle saturé

```
pchisq(1124.9, 1052  , lower = F)
```

```
## [1] 0.0583877
```

La p-valeur est > 5% : on rejette le modèle saturé. Autrement dit, notre modèle est suffisant.
Cette fois-ci, on va comparer les 2 modèles model_step_bck et model_step_fwd en faisant le test "likelihood ratio test".

```
anova(model_step_bck, model_step_fwd, test = "LRT")
```

1	
2	

2 rows | 1-1 of 6 columns

```
## on peut calculer manuellement sans passer la fonction ANOVA
## 1-pchisq(deviance(model_step_fwd)-deviance(model_step_bck),df.residual(model_step_fwd)-df.residual(model_step_bck))
## les 2 manières nous donnent le même résultat
```

On constate que la p-value est assez faible < 0,05, ce dernier nous explique le modèle plus compliqué est plus utile que celui plus simple. Ici, il s'agit le model_step_bck. Autremeent, la différence entre l'AIC de model_step_bck et model_step_fwd est statistiquement significative. On va regarder le critère BIC, on a la tendance de choisir un modèle dont le BIC est plus petit. De ce qu'on voit, le BIC du model_step_fwd est plus grand que celui model_step_bck et la marge entre les 2 valeurs est bien significative (>5)

```
BIC(model_step_bck)
```

```
## [1] 1208.52
```

```
BIC(model_step_fwd)
```

```
## [1] 1215.15
```

```
abs(BIC(model_step_bck) - BIC(model_step_fwd)
)
```

```
## [1] 6.629389
```

Pour décider définitvement lequel qui nous convient mieux, on va aller à l'étape prochaine

Prédiction

```
model_step_bck.predict<-predict(model_step_bck,d.testset,type="response")
model_step_fwd.predict<-predict(model_step_fwd,d.testset,type="response")
```

Avant de valider la prédiction, on va faire une fonction permettant de calculer le taux d'erreur entre la prédiction et la réalité.

```
calculate_error_rate <- function(y_obs,y_pred){
  mc <- table(y_obs,y_pred)
  print("Matrice de confusion :")
  print(mc)
  #nb mal classés
  wrong <- sum(mc) - sum(diag(mc))
  print(paste("Mal classés :",wrong))
  #taux d'erreur
  err <- wrong/sum(mc)
  print(paste("taux erreur (%) :",err*100))
  #intervalle de confiance
  #effectif
  n <- sum(mc)
  #écart-type
  et <- sqrt(err*(1-err)/n)
  #quantile loi normale 95%
  z <- qnorm(0.95)
  #borne basse
  bb <- err - z * et
  bh <- err + z * et
  # bornes intervalle
  print("Bornes intervalle de confiance")
  print(c(bb,bh))
}
```

On définit un seuil de prédiction à 50% pour que le lendemain soit considère comme un jour "pluvieux"

```
threeshold <- 0.5
calculate_error_rate(d.testset$pluie_demain,ifelse(model_step_bck.predict>threeshold,"TRUE","FALSE"))
```

```
## [1] "Matrice de confusion :"
##        y_pred
## y_obs   FALSE TRUE
##   FALSE    32   19
##   TRUE     16   51
## [1] "Mal classés : 35"
## [1] "taux erreur (%) : 29.6610169491525"
## [1] "Bornes intervalle de confiance"
## [1] 0.2274466 0.3657737
```

```
calculate_error_rate(d.testset$pluie_demain,ifelse(model_step_fwd.predict>threeshold,"TRUE","FALSE"))
```

```
## [1] "Matrice de confusion :"
##        y_pred
## y_obs   FALSE TRUE
##   FALSE    33   18
##   TRUE     15   52
## [1] "Mal classés : 33"
## [1] "taux erreur (%) : 27.9661016949153"
## [1] "Bornes intervalle de confiance"
## [1] 0.2116983 0.3476237
```

Pour model_step_fwd, nous avons obtenu un taux d'erreur à 27.966% par rapport à 29.66% de model_step_bck et la borne d'intervalle est mieux amélioré que celle de model_step_bck. Ce chiffre me semble acceptable. Donc, le model_step_fwd me convient plus.
Néanmoins, en réalité, pour pouvoir trouver le meuilleur modèle, cela dépend significativement du jeu d'entraînement et également du seuil de prédiction. En pratique, on devrait toujours améliorer notre modèle en enrichissant notre jeu d'entrainement et maintenir les test sur la validation de prédiction. Dans ce devoir, les variables dans le modèle pourraient varier en fonction de l'échantillon qu'on a définit au début.

Prédiction du jeu de test

```
library(readr)
meteo.test = read_csv('meteo.test.csv')

d.meteo.test<-meteo.test[-c(1:6)]
d.meteo.test <- d.meteo.test %>%  clean_heads_name()

model.predict.test<-predict(model_step_fwd,d.meteo.test,type="response")
predict.demain= (model.predict.test >= 0.5)
d.model.predict <- cbind(d.meteo.test,predict.demain)
```