TP3 Real Estate

Vhiny-Guilley

13/11/2020

```r
library(Metrics)
library(ggplot2)
library(corrplot)
```

loading packages

```
## corrplot 0.84 loaded
```

```r
library(bayestestR)
```

```
##
## Attaching package: 'bayestestR'

## The following object is masked from 'package:Metrics':
##
##     auc
```

```r
library(lars); library(MASS);library(glmnet)
```

```
## Loaded lars 1.2

## Loading required package: Matrix

## Loaded glmnet 4.0-2
```

Lecture des données On sépare les prix en 2 classes 0 et 1. Une transaction appartient à la classe 1 si son prix est supérieure à la médiane de la variable prix et 0 sinon.

```r
tab=read.table("RealEstate.csv",header=TRUE,sep=',');
medianHousePrice=median(tab$Y.house.price.of.unit.area);
medHousePriceBin=as.numeric(tab$Y.house.price.of.unit.area>medianHousePrice);

##
tabmed = tab
tabmed = tabmed[,-1]
tabmed$Y.house.price.of.unit.area = medHousePriceBin
head(tabmed)
```

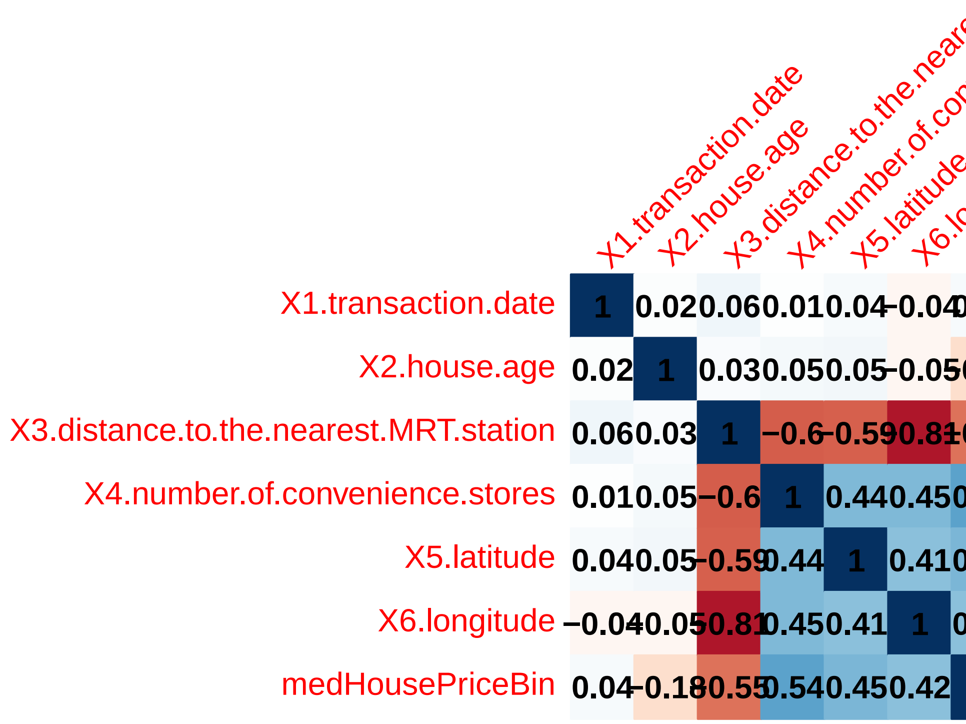
```
##   X1.transaction.date X2.house.age X3.distance.to.the.nearest.MRT.station
## 1            2012.917         32.0                               84.87882
## 2            2012.917         19.5                              306.59470
## 3            2013.583         13.3                              561.98450
## 4            2013.500         13.3                              561.98450
## 5            2012.833          5.0                              390.56840
## 6            2012.667          7.1                             2175.03000
##   X4.number.of.convenience.stores X5.latitude X6.longitude
## 1                              10    24.98298     121.5402
## 2                               9    24.98034     121.5395
## 3                               5    24.98746     121.5439
## 4                               5    24.98746     121.5439
## 5                               5    24.97937     121.5425
## 6                               3    24.96305     121.5125
##   Y.house.price.of.unit.area
## 1                          0
## 2                          1
## 3                          1
## 4                          1
## 5                          1
## 6                          0
```

```
colnames(tabmed)[dim(tabmed)[2]] <- "medHousePriceBin" # change la variable price en medHousePriceBin d
head(tabmed)
```

```
##   X1.transaction.date X2.house.age X3.distance.to.the.nearest.MRT.station
## 1            2012.917         32.0                               84.87882
## 2            2012.917         19.5                              306.59470
## 3            2013.583         13.3                              561.98450
## 4            2013.500         13.3                              561.98450
## 5            2012.833          5.0                              390.56840
## 6            2012.667          7.1                             2175.03000
##   X4.number.of.convenience.stores X5.latitude X6.longitude medHousePriceBin
## 1                              10    24.98298     121.5402                0
## 2                               9    24.98034     121.5395                1
## 3                               5    24.98746     121.5439                1
## 4                               5    24.98746     121.5439                1
## 5                               5    24.97937     121.5425                1
## 6                               3    24.96305     121.5125                0
```

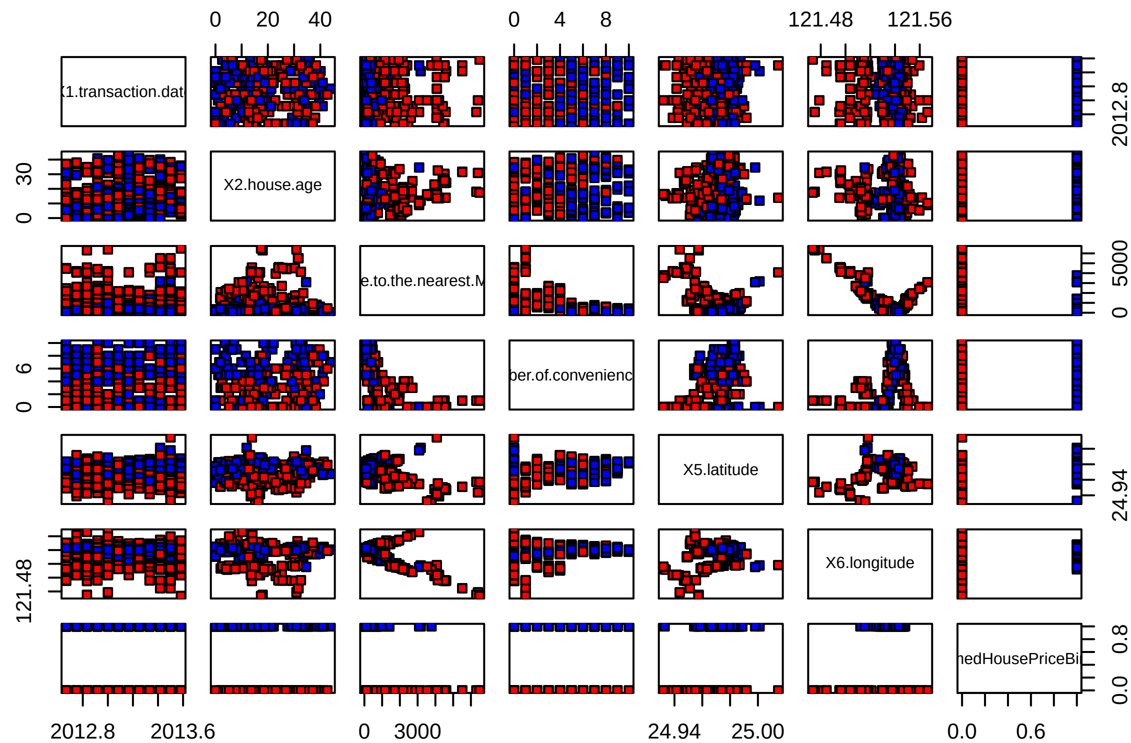
```
mcor = cor(tabmed) # correlation matrix

corrplot(mcor, method="color", addCoef.col= "black", tl.srt =
45, sig.level=0.01, insig="blank")
```



visualisation des données

```
pairs(tabmed,pch=22,bg=c("red","blue")[unclass(factor(tabmed[,"medHousePriceBin"]))])
```



Dans ce graphe les points bleus sont les transactions dont le prix est supérieur à la médiane et les rouges sont celles dont le prix est inférieur. A part, X1.transaction.date/X2.house.date, sur lequel on peut pas distinguer des clusters, sur les autres plots , on voit nettement des cluster se former dans chacun des covariables plots.

La proximité à la station la plus proche est une variables fortement corrélée avec la longitude, et moyennement corrélée aux autres variables.

Dans la suite on va essayer de generer un modèle de regression logistique jeu de données.

logistic model

```
#set.seed(1234)
p = 0.8
ind = sample(2, nrow(tabmed), replace = T, prob = c(p,1-p)) ## selection aleatoire 80 - 20 des indices
tab.train = as.data.frame(tabmed[ind == 1,])
tab.test = as.data.frame(tabmed[ind == 2,])
```

organisation du dataset

```
model.full = glm(medHousePriceBin ~ ., data = tab.train, family = 'binomial')
summary(model.full)
```

Etude du model

```
##
## Call:
## glm(formula = medHousePriceBin ~ ., family = "binomial", data = tab.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2802  -0.3929   0.2009   0.5665   4.4884
##
## Coefficients:
##                                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)                            -3.336e+03  3.400e+03  -0.981   0.3265
## X1.transaction.date                     1.472e+00  6.095e-01   2.415   0.0158
## X2.house.age                           -6.144e-02  1.381e-02  -4.448 8.66e-06
## X3.distance.to.the.nearest.MRT.station -2.422e-03  4.884e-04  -4.958 7.11e-07
## X4.number.of.convenience.stores         1.733e-01  7.288e-02   2.378   0.0174
## X5.latitude                             8.228e+01  1.879e+01   4.379 1.19e-05
## X6.longitude                           -1.381e+01  2.633e+01  -0.524   0.6000
##
## (Intercept)
## X1.transaction.date                    *
## X2.house.age                           ***
## X3.distance.to.the.nearest.MRT.station ***
## X4.number.of.convenience.stores        *
## X5.latitude                            ***
## X6.longitude
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 463.90  on 334  degrees of freedom
## Residual deviance: 243.41  on 328  degrees of freedom
## AIC: 257.41
##
## Number of Fisher Scoring iterations: 7
```

Avec un seuil de p-value à 0.01, les variables statistiquement significative sont X2,X3,X4 et X5. De plus le test rejete la varible X6.longitude avec une probabilité de 0.95! Ce qui est sans doute du à la corrélation de celle ci avec X3.

```
 ### prediction
prob = predict.glm(model.full, newdata = tab.test,type = "response") # give the predicted probability
OR = exp(model.full$coefficients) # odd ratio
summary(prob)
```

Prediction

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000009 0.0735826 0.7371952 0.5519591 0.8926892 0.9824158
```

```
OR
```

```
##                          (Intercept)                     X1.transaction.date
##                         0.000000e+00                            4.356043e+00
##                         X2.house.age X3.distance.to.the.nearest.MRT.station
##                         9.404115e-01                            9.975811e-01
##        X4.number.of.convenience.stores                            X5.latitude
##                         1.189212e+00                            5.400046e+35
##                         X6.longitude
##                         1.005525e-06
```

```


Threshold = 0.5
Y.pred.full =  as.integer(prob >= Threshold)
confusion_matrix.full = table(Y.pred.full,tab.test$medHousePriceBin)
confusion_matrix.full
```

Performance du model

```
##
## Y.pred.full  0  1
##           0 30  0
##           1 16 33
```

La matrice de confusion nous donne une performance

```
pred.accuracy.full = sum(diag(confusion_matrix.full))/sum(confusion_matrix.full)*100#   prediction accu
pred.recall.full = confusion_matrix.full[2,2]/sum(confusion_matrix.full[,2])*100 # probabilité de bien p
pred.specifity.full = confusion_matrix.full[1,1]/sum(confusion_matrix.full[,1])*100 # probabilité de bi
pred.precision.full = confusion_matrix.full[2,2]/sum(confusion_matrix.full[2,])*100
pred.error_rate.full = sum(diag(confusion_matrix.full[1:2,2:1]))/sum(confusion_matrix.full) *100 # prob

actual.accurary =  as.double(table(tab.test$medHousePriceBin)[1]/sum(table(tab.test$medHousePriceBin)))
pred.accuracy.full
```

```
## [1] 79.74684
```

```
pred.recall.full
```

```
## [1] 100
```

```
pred.error_rate.full
```

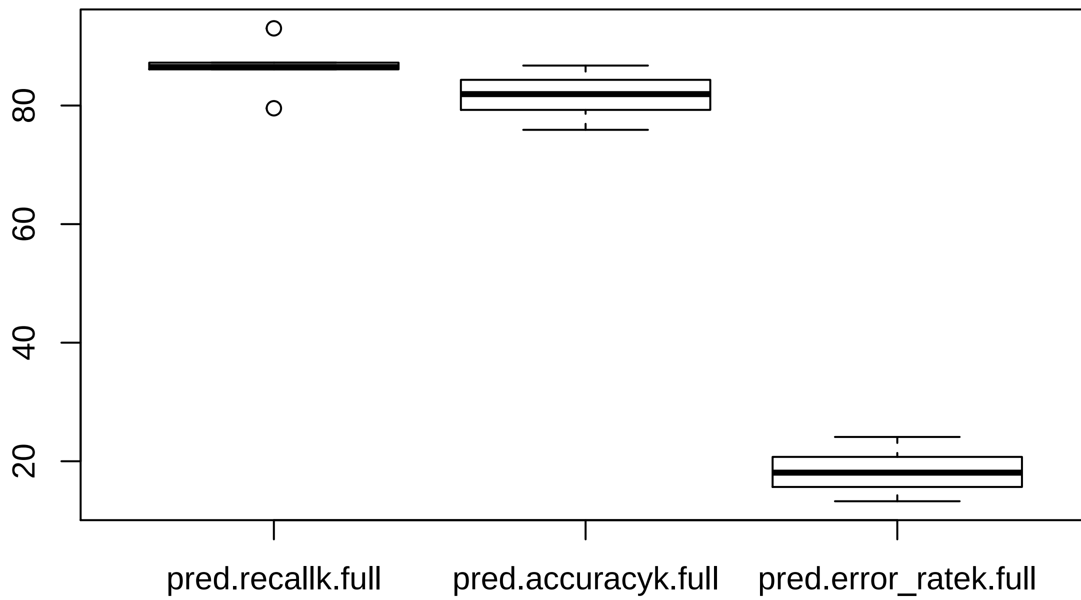
```
## [1] 20.25316
```

```
actual.accurary
```

```
## [1] 0.5822785
```

```r
##shuffling
rows <- sample(nrow(tabmed))
tabmed <- tabmed[rows, ]
## folds
k = 5 #as.integer(1/(1-r)) ## fold number
fold = cut(seq(1,nrow(tabmed)), breaks = k,labels = FALSE)
##
pred.accuracyk.full = c()
pred.recallk.full = c()
pred.error_ratek.full = c()

for (i in 1:k) {
  test_rows = which(fold == i,arr.ind = TRUE)
  tab.testk = tabmed[test_rows,]
  tab.traink = tabmed[-test_rows,]
  #Y.testk = Y[test_rows]
  ### regression logistic
  model.fullk=glm(medHousePriceBin~.,family=binomial,data = tab.traink)
  ### prediction
  prob = predict.glm(model.fullk, newdata = tab.testk,type = "response") # give prob
  Y.pred.full =  as.integer(prob >= Threshold)
  confusion_matrix = table(Y.pred.full,tab.testk$medHousePriceBin)
  pred.accuracyk.full[i] = sum(diag(confusion_matrix))/sum(confusion_matrix)*100#   prediction accuracy
  pred.recallk.full[i] = confusion_matrix[2,2]/sum(confusion_matrix[,2])*100 # the prediction of being
  pred.error_ratek.full[i] = sum(diag(confusion_matrix[1:2,2:1]))/sum(confusion_matrix) *100
}
boxplot(data.frame(pred.recallk.full,pred.accuracyk.full,pred.error_ratek.full))
```



k-folds le model full

```
mean(pred.recallk.full)
```

```
## [1] 86.48007
```

```
mean(pred.error_ratek.full)
```

```
## [1] 18.36321
```

```
mean(pred.accuracyk.full)
```

```
## [1] 81.63679
```

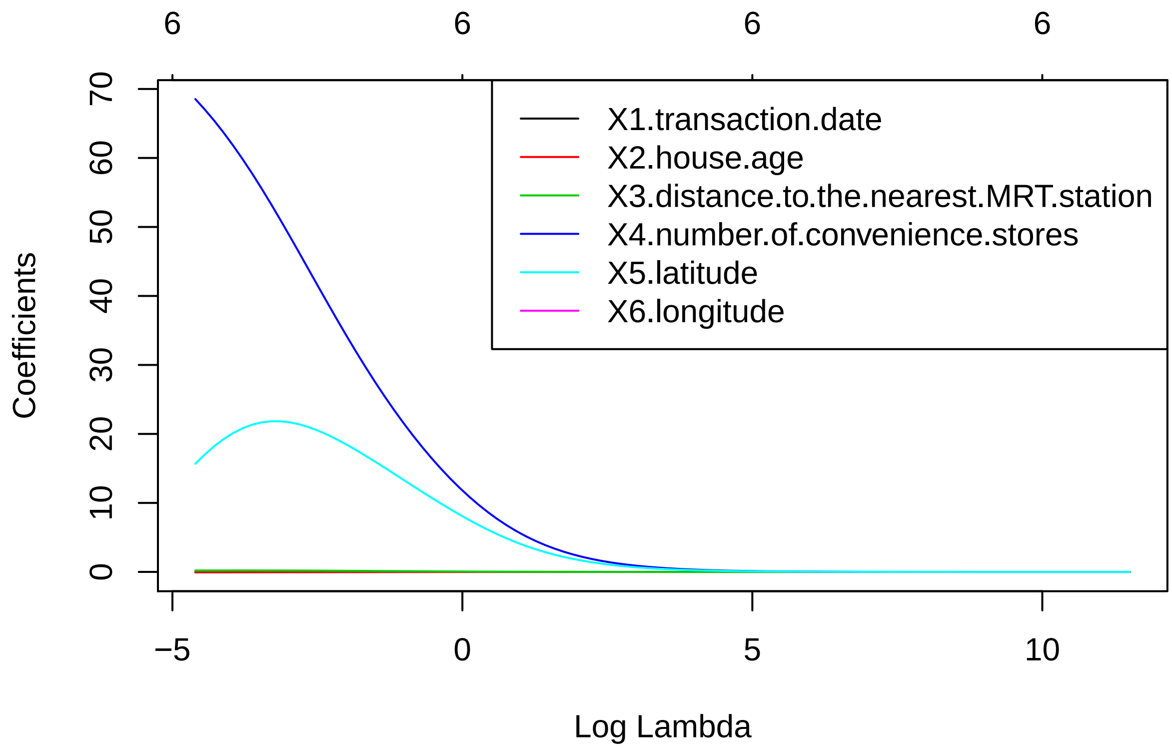
En utilisant le k-fold on évalue la precision du model. nous donne une bonne performance global. Avec une performance global de 80% et un taux d'erreur 20%.

Ridge regression

```
X.train = as.matrix(tab.train[,-dim(tab.train)[2]])
X.test = as.matrix(tab.test[,-dim(tab.test)[2]])
Y.test = tab.test$medHousePriceBin
Y.train = tab.train$medHousePriceBin
```

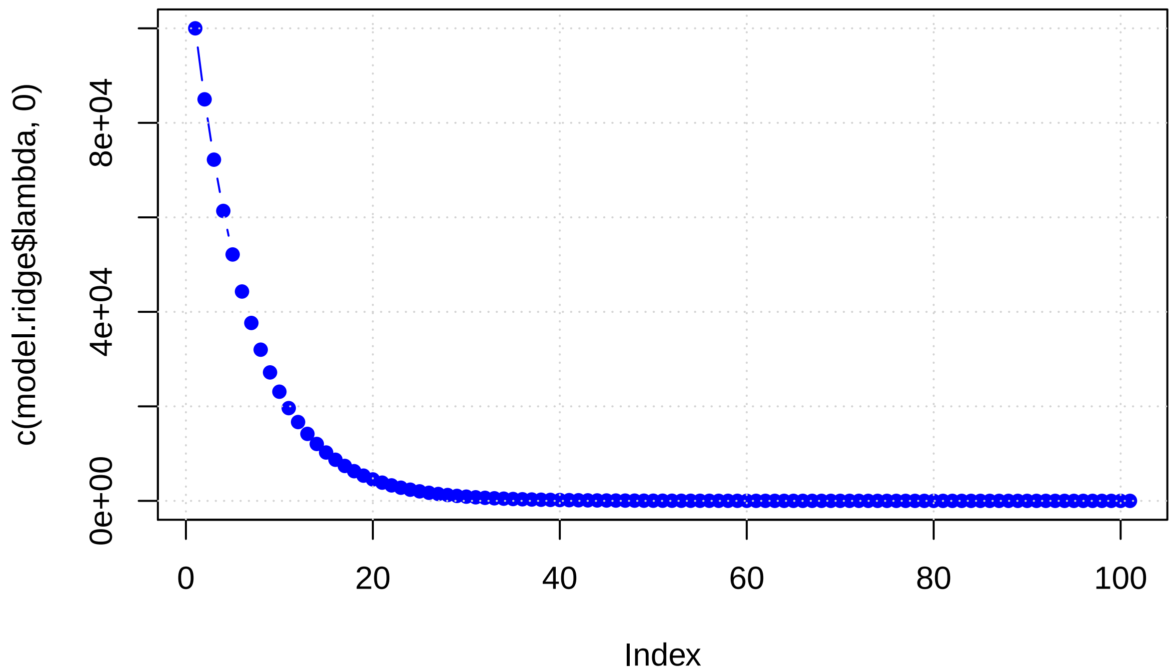
organisation du dataset

```r
grid = 10^seq(5,-2,length = 100) # sequence des lambda
model.ridge <- glmnet(X.train,Y.train,alpha=0,lambda = grid,family = "binomial")
plot(model.ridge,xvar="lambda",type="l",col=1:nrow(tab.train)-1);legend("topright",legend=colnames(tab.t
```



Ridge model

```r
plot(c(model.ridge$lambda,0),pch = 16,type = "b",col = "blue"); grid()
```




```
####################### cross validation
ridge.cv.out<-cv.glmnet(X.train, Y.train, alpha = 0,nfolds = 10,family = "binomial"); ridge.cv.out # on
```

Selection du λ par cross validation

```
##
## Call:  cv.glmnet(x = X.train, y = Y.train, nfolds = 10, alpha = 0, family = "binomial")
##
## Measure: Binomial Deviance
##
##      Lambda Measure      SE Nonzero
## min 0.02776   0.803 0.05283       6
## 1se 0.07724   0.850 0.04076       6
```

```
ridge.lamb.min<-ridge.cv.out$lambda.min # le meilleur lambda est celui qui produit the min MSE
```

On selectionne le modele le lambda qui minimise le MSE pour notre modèle. On effectue 10 folds.

```
ridge.predbest <- predict(model.ridge, s = ridge.lamb.min, newx = X.test,type = 'response')
ridge.predbest[1:20]
```

```
##  [1] 0.840605898 0.945790679 0.849631125 0.613398212 0.654063204 0.069814218
```

```
##  [7] 0.825548038 0.806337552 0.002784678 0.945354662 0.299231818 0.498473188
## [13] 0.029476396 0.772448626 0.892696769 0.003865942 0.650579236 0.750972131
## [19] 0.480372446 0.523866596
```

```r
Threshold = 0.5
Y.pred.ridge =  as.integer(ridge.predbest >= Threshold)
confusion_matrix.ridge = table(Y.pred.ridge,Y.test)
confusion_matrix.ridge
```

prediction

```
##             Y.test
## Y.pred.ridge  0  1
##            0 31  1
##            1 15 32
```

La matrice de confusion nous donne une performance

```r
pred.accuracy.ridge = sum(diag(confusion_matrix.ridge))/sum(confusion_matrix.ridge)*100#   prediction a
pred.recall.ridge = confusion_matrix.ridge[2,2]/sum(confusion_matrix.ridge[,2])*100 # probabilité de bi
pred.specifity.ridge = confusion_matrix.ridge[1,1]/sum(confusion_matrix.ridge[,1])*100 # probabilité de
pred.precision.ridge = confusion_matrix.ridge[2,2]/sum(confusion_matrix.ridge[2,])*100
pred.error_rate.ridge = sum(diag(confusion_matrix.ridge[1:2,2:1]))/sum(confusion_matrix.ridge) *100 # p

actual.accurary =  as.double(table(tab.test$medHousePriceBin)[1]/sum(table(tab.test$medHousePriceBin)))
pred.accuracy.ridge
```

```
## [1] 79.74684
```

```r
pred.recall.ridge
```

```
## [1] 96.9697
```

```r
pred.specifity.ridge
```

```
## [1] 67.3913
```

```r
pred.error_rate.ridge
```

```
## [1] 20.25316
```

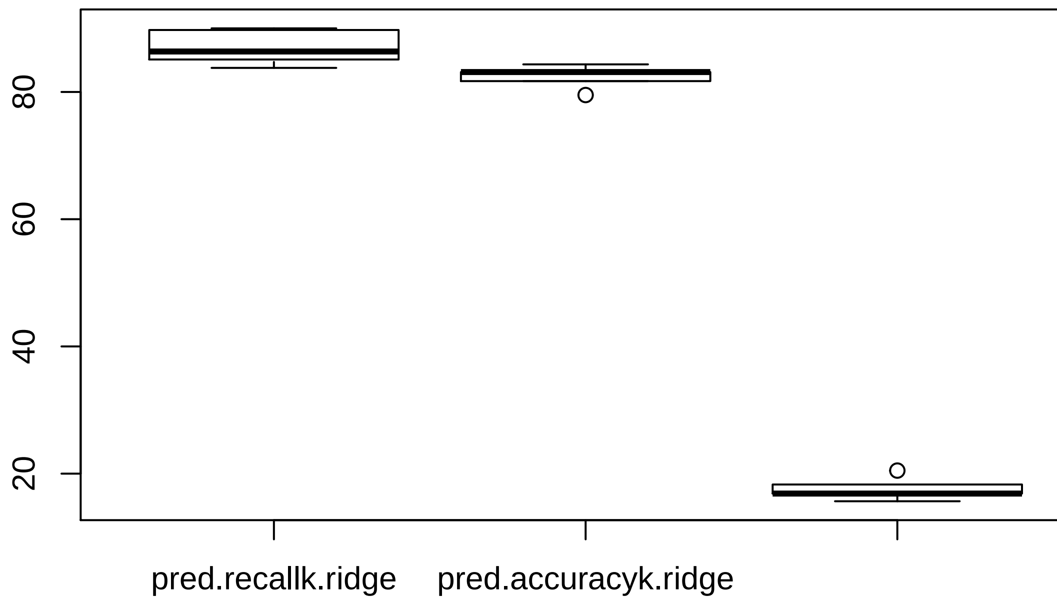
```r
actual.accurary
```

```
## [1] 0.5822785
```

On retrouve une performance de 76% sur le modèle. Le modèle predit bien la classe des hauts prix à 77%.

```r
##shuffling
rows <- sample(nrow(tabmed))
tabmed <- tabmed[rows, ]
## folds
k = 5 #as.integer(1/(1-r)) ## fold number
fold = cut(seq(1,nrow(tabmed)), breaks = k,labels = FALSE)
##
pred.accuracyk.ridge = c()
pred.recallk.ridge = c()
pred.error_ratek.ridge = c()

for (i in 1:k) {
  test_rows = which(fold == i,arr.ind = TRUE)
  tab.testk = tabmed[test_rows,]
  tab.traink = tabmed[-test_rows,]
  #Y.testk = Y[test_rows]
  ### regression logistic
  model.ridgek=glm(medHousePriceBin~.,family=binomial,data = tab.traink)
  ### prediction
  prob = predict.glm(model.ridgek, newdata = tab.testk,type = "response") # give prob
  Y.pred.ridge =  as.integer(prob >= Threshold)
  confusion_matrix = table(Y.pred.ridge,tab.testk$medHousePriceBin)
  pred.accuracyk.ridge[i] = sum(diag(confusion_matrix))/sum(confusion_matrix)*100#   prediction accurac
  pred.recallk.ridge[i] = confusion_matrix[2,2]/sum(confusion_matrix[,2])*100 # the prediction of being
  pred.error_ratek.ridge[i] = sum(diag(confusion_matrix[1:2,2:1]))/sum(confusion_matrix) *100
}
boxplot(data.frame(pred.recallk.ridge,pred.accuracyk.ridge,pred.error_ratek.ridge))
```



k-folds le model ridge

```
mean(pred.recallk.ridge)
```

```
## [1] 86.99948
```

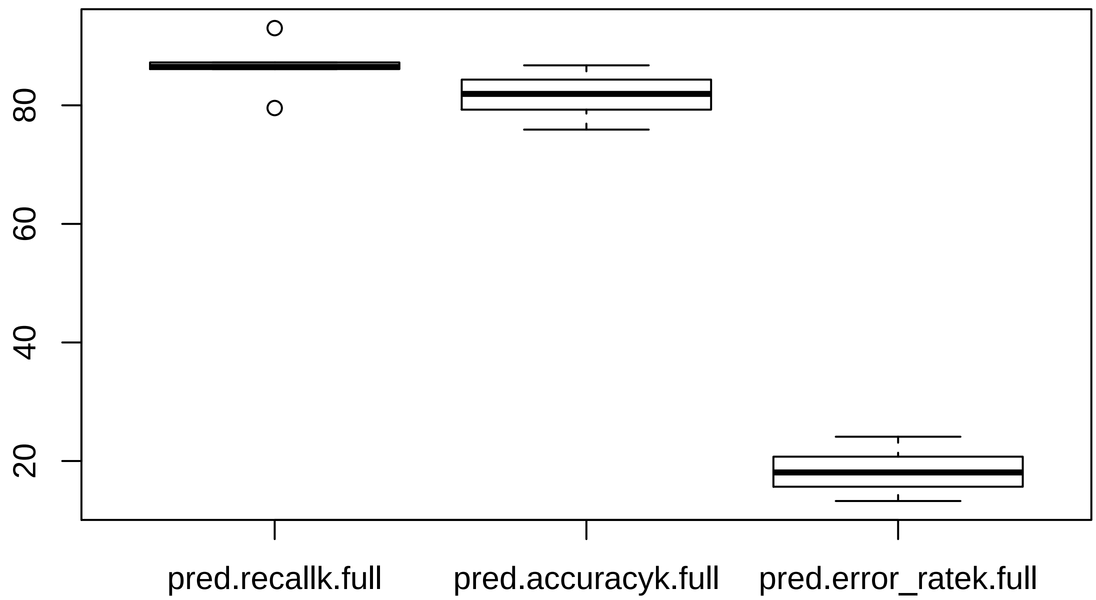
```
mean(pred.error_ratek.ridge)
```

```
## [1] 17.63444
```

```
mean(pred.accuracyk.ridge)
```

```
## [1] 82.36556
```

```r
boxplot(data.frame(pred.recallk.full,pred.accuracyk.full,pred.error_ratek.full))
```




```r
mean(pred.recallk.full)
```

```
## [1] 86.48007
```

```r
mean(pred.error_ratek.full)
```

```
## [1] 18.36321
```

```r
mean(pred.accuracyk.full)
```

```
## [1] 81.63679
```