TP3 Real Estate

Kaabi Linda & Vhiny-Guilley Mombo

13/11/2020

Dans cette étude on utilise le dataset “RealEstate.csv” pour expliquer les hauts et les bas prix en utilisants des regressions logistiques. ## 1) Chargemment des données et differents packages

## loading packages

library(Metrics)  
library(ggplot2)  
library(corrplot)

## corrplot 0.84 loaded

library(bayestestR)

##   
## Attaching package: 'bayestestR'

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

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. Préduire si une transaction appartient à une de ces 2 classes peut se faire par une régressions logistiques.

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

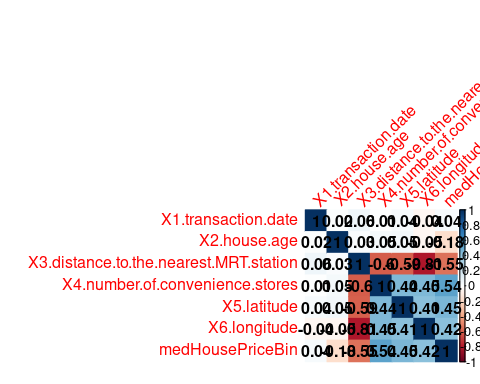
colnames(tabmed)[dim(tabmed)[2]] <- "medHousePriceBin" # change la variable price en medHousePriceBin dans le tableau.  
head(tabmed)

## 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

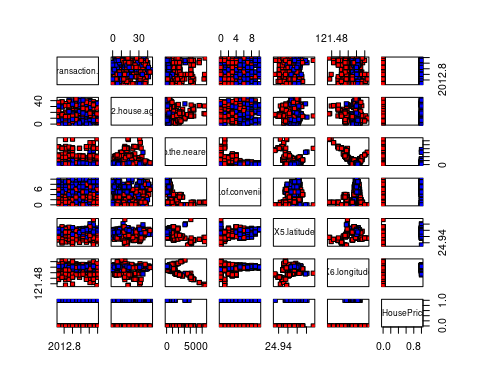
## visualisation des données

Le plot des différentes covariables nous permet de distinguer les dependances a priori entre elles.

mcor = cor(tabmed) # correlation matrix  
corrplot(mcor, method="color", addCoef.col= "black", tl.srt =  
45, sig.level=0.01, insig="blank")



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



#dans le plot les points rouges sont les transactions dont le prix sont  
# en dessous de la mediane [0], et le bleus celles au dessus [1]

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 clusterS se former dans chacun des covariables plots. La proximité à la station la plus proche X3 est une variables fortement corrélée avec la X6.longitude.

# I) Les modèles.

Dans la suite on va essayer de generer differents modèles à partir des regression logistique sur notre eu de données:

On commencera par un modèle dit “full”, qui utiliseras toutes les variables, ensuite on fera une selection de variables grace une méthode “stepwise”, puis on fera des regressions logistiques sous contraintes classiques en utilisant “ridge” et “lasso”.

### organisation du dataset

on va partitionner notre jeu de données en training et test avec des proportions respectives de 80% et 20%.

set.seed(1234) #assure la repetabilité des differents pocessus aleatoire  
p = 0.8 # represente 80% qui va etre pour le trainin set   
ind = sample(2, nrow(tabmed), replace = T, prob = c(p,1-p)) ## selectionne les indices  
#aleatoirent entre 2 groupe, le premiere groupe represente 80 % de l'ensemble   
#des indices et le 2e groupe le reste.  
tab.train = as.data.frame(tabmed[ind == 1,]) #training dataset  
tab.test = as.data.frame(tabmed[ind == 2,]) #test dataset  
  
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

## I.1 Modele full

### a) model full

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

##   
## Call:  
## glm(formula = medHousePriceBin ~ ., family = "binomial", data = tab.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.3966 -0.3026 0.1507 0.5368 3.1611   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -2.404e+03 3.908e+03 -0.615 0.53834  
## X1.transaction.date 1.381e+00 6.299e-01 2.193 0.02834  
## X2.house.age -7.016e-02 1.460e-02 -4.807 1.53e-06  
## X3.distance.to.the.nearest.MRT.station -2.596e-03 5.141e-04 -5.049 4.44e-07  
## X4.number.of.convenience.stores 2.619e-01 8.212e-02 3.189 0.00143  
## X5.latitude 9.477e+01 2.235e+01 4.241 2.22e-05  
## X6.longitude -2.255e+01 3.117e+01 -0.723 0.46946  
##   
## (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: 467.11 on 336 degrees of freedom  
## Residual deviance: 221.48 on 330 degrees of freedom  
## AIC: 235.48  
##   
## Number of Fisher Scoring iterations: 7

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

### b) Prediction avec le model full

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

## (Intercept) X1.transaction.date   
## 0.000000e+00 3.979501e+00   
## X2.house.age X3.distance.to.the.nearest.MRT.station   
## 9.322441e-01 9.974075e-01   
## X4.number.of.convenience.stores X5.latitude   
## 1.299354e+00 1.439174e+41   
## X6.longitude   
## 1.609538e-10

Une augmentation de X4 le nombre de magasins de proximité, et la latitute X5 ou la date de transaction X1 semble augmenter augmenter le prix de la transaction. Tandis que une augmententation des autres variables telles que le l’age de maison X2, la longitude X6 ou la distance a la station la plus proche X3 semble diminuer le prix de la transaction.

### c) Performance du model full

Threshold = 0.5 # for MAE estimation  
Y.pred.full = as.integer(prob >= Threshold)   
confusion\_matrix.full = table(Y.pred.full,tab.test$medHousePriceBin)  
confusion\_matrix.full

##   
## Y.pred.full 0 1  
## 0 33 9  
## 1 8 27

La matrice de confusion nous donne une performance

accuracy.full = sum(diag(confusion\_matrix.full))/sum(confusion\_matrix.full)\*100#   
 recall.full = confusion\_matrix.full[2,2]/sum(confusion\_matrix.full[,2])\*100 #  
 precision.full =confusion\_matrix.full[2,2]/sum(confusion\_matrix.full[2,])\*100  
 error\_rate.full = sum(diag(confusion\_matrix.full[1:2,2:1]))/sum(confusion\_matrix.full) \*100

Avec une specificité de 80% et un un recall de 75% le model full prédit assez bien les éléments de differentes classe. Globalement on fait une erreur de 22%, ce qui relativement bas au vu de la taille de l’echantillon.

Afin d’évaluer d’avoir une meilleur fiabilité à notre modèle on va effecteur une Validation croisée.

### d) k-folds pour le model full

##shuffling  
set.seed(1234)  
rows <- sample(nrow(tabmed)) # melange des rows du dataset  
tabmed <- tabmed[rows, ]  
## folds  
k = 15 #as.integer(1/(1-r)) ## fold number  
fold = cut(seq(1,nrow(tabmed)), breaks = k,labels = FALSE) #organisations des lignes du  
#dataset en plusieurs fold  
#initialisation des vectueurs qui va contenier les performances a chaque etape  
accuracyk.full = c()  
recallk.full = c()  
precisionk.full = c()  
error\_ratek.full = c()  
  
for (i in 1:k) {  
 test\_rows = which(fold == i,arr.ind = TRUE) #selection des lignes qui correspondent   
 # au fold i  
 tab.testk = tabmed[test\_rows,]  
 tab.traink = tabmed[-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 probability  
 Y.pred.full = as.integer(prob >= Threshold) #MAE estimation  
 confusion\_matrix = table(Y.pred.full,tab.testk$medHousePriceBin)  
 accuracyk.full[i] = sum(diag(confusion\_matrix))/sum(confusion\_matrix)\*100#   
 recallk.full[i] = confusion\_matrix[2,2]/sum(confusion\_matrix[,2])\*100 #  
 precisionk.full[i] =confusion\_matrix[2,2]/sum(confusion\_matrix[2,])\*100  
 error\_ratek.full[i] = sum(diag(confusion\_matrix[1:2,2:1]))/sum(confusion\_matrix) \*100   
}  
mean(recallk.full)

## [1] 86.26258

mean(error\_ratek.full)

## [1] 18.11287

mean(accuracyk.full)

## [1] 81.88713

mean(precisionk.full)

## [1] 78.74136

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 d’environ 20%.

## I.2) Regression Logistique avec selection de variable Stepwise

On a vu que certaines variables, comme X6.longitude était pas significative dans notre model full,dans cette partie, on va effectuer une selection de Variables, puis une regression logistique sur ces dernières.

### a) Resstep Model

Selection des variables à partir du model précédent.

resstep<-step(model.full,direction='both'); summary(resstep)

## Start: AIC=235.48  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 222.02 234.02  
## <none> 221.48 235.48  
## - X1.transaction.date 1 226.41 238.41  
## - X4.number.of.convenience.stores 1 232.81 244.81  
## - X5.latitude 1 242.44 254.44  
## - X2.house.age 1 248.18 260.18  
## - X3.distance.to.the.nearest.MRT.station 1 265.27 277.27  
##   
## Step: AIC=234.02  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 222.02 234.02  
## + X6.longitude 1 221.48 235.48  
## - X1.transaction.date 1 226.74 236.74  
## - X4.number.of.convenience.stores 1 232.94 242.94  
## - X5.latitude 1 243.59 253.59  
## - X2.house.age 1 249.18 259.18  
## - X3.distance.to.the.nearest.MRT.station 1 279.21 289.21

##   
## Call:  
## glm(formula = medHousePriceBin ~ X1.transaction.date + X2.house.age +   
## X3.distance.to.the.nearest.MRT.station + X4.number.of.convenience.stores +   
## X5.latitude, family = "binomial", data = tab.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.3391 -0.2916 0.1624 0.5223 3.1496   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -5.072e+03 1.404e+03 -3.612 0.000304  
## X1.transaction.date 1.348e+00 6.286e-01 2.145 0.031950  
## X2.house.age -7.077e-02 1.461e-02 -4.843 1.28e-06  
## X3.distance.to.the.nearest.MRT.station -2.417e-03 4.475e-04 -5.400 6.66e-08  
## X4.number.of.convenience.stores 2.392e-01 7.401e-02 3.232 0.001231  
## X5.latitude 9.448e+01 2.205e+01 4.284 1.83e-05  
##   
## (Intercept) \*\*\*  
## X1.transaction.date \*   
## X2.house.age \*\*\*  
## X3.distance.to.the.nearest.MRT.station \*\*\*  
## X4.number.of.convenience.stores \*\*   
## X5.latitude \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 467.11 on 336 degrees of freedom  
## Residual deviance: 222.02 on 331 degrees of freedom  
## AIC: 234.02  
##   
## Number of Fisher Scoring iterations: 7

Sans surprise, comme vu dans le TP2, la variable X6.longitude a été éliminée dans la sélection de variables, ce qui augmente la significaivité des autres variables, c’est ce qu’on remarque en comparant les test effectués sur une régression logistique de type full (toutes les variables) et les test sur une régression logistique de type stepwise.

OR\_step=exp(resstep$coefficients) # give the odd ratios on stepwise model  
OR\_step

## (Intercept) X1.transaction.date   
## 0.000000e+00 3.850789e+00   
## X2.house.age X3.distance.to.the.nearest.MRT.station   
## 9.316744e-01 9.975862e-01   
## X4.number.of.convenience.stores X5.latitude   
## 1.270203e+00 1.074748e+41

Comme precedemment, les variables X1.transaction date, X4.convenience.stores et X5.latitude, leur augmentation semble entrainer une augmentation de la probabilité que medHousePriceBin soit au-dessus de la médiane du prix de vente, au risque de 5%.

## b ) Prédiction sur le model stepwise

prob= predict.glm(resstep, newdata = tab.test,type = "response") # give the predicted prob on the stepwise model  
pred\_step=as.numeric(prob>Threshold) #

On calcule les prédictions contruites à partir de ce modèle pred\_step qui donne la valeur des pour tous les individus dans le data set test. pred\_step vaut 1 si pred\_step > 1/2.

confusion\_matrix.step = table(pred\_step,tab.test$medHousePriceBin)  
confusion\_matrix.step

##   
## pred\_step 0 1  
## 0 33 7  
## 1 8 29

La matrice de confusion nous montre qu’on predit mieux de vrai evenements qu’on en fait des erreurs.

pred.accuracy.step = sum(diag(confusion\_matrix.step))/sum(confusion\_matrix.step)\*100  
#prediction accuracy  
pred.recall.step = confusion\_matrix.step[2,2]/sum(confusion\_matrix.step[,2])\*100   
# probabilité que la prediction des hauts prix soit correctes   
pred.specifity.step = confusion\_matrix.step[1,1]/sum(confusion\_matrix.step[,1])\*100   
# probabilité que la prediction des bas prix soit correctes   
pred.precision.step = confusion\_matrix.step[2,2]/sum(confusion\_matrix.step[2,])\*100  
#   
pred.error\_rate.step = sum(diag(confusion\_matrix.step[1:2,2:1]))/sum(confusion\_matrix.step) \*100 # probabilité d'obtenir une erreur  
  
pred.accuracy.step

## [1] 80.51948

pred.recall.step

## [1] 80.55556

pred.specifity.step

## [1] 80.4878

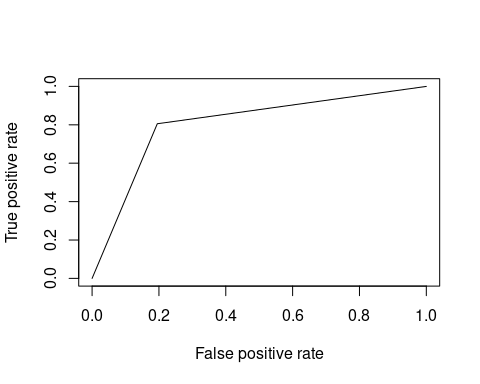
pred.error\_rate.step

## [1] 19.48052

Comme dit precedemment mais avec des chiffres, on effectue des bonnes prédictions en moyenne à 80.5%. Ce qui est pas un bon score au vu de la taille de l’échantillon.

### Courbe ROC et AUC

library(ROCR)  
  
pred=prediction(pred\_step,tab.test$medHousePriceBin)  
perf=performance(pred, "tpr" ,"fpr")  
plot(perf)



La courbe est assez proche du coin supérieur gauche, ainsi le modèle semble capturer le plus possible de vrais évènements avec le moins possible de faux évènements.

ROC\_auc=performance(pred,"auc")  
AUC=ROC\_auc@y.values[[1]]  
print(AUC)

## [1] 0.8052168

L’AUC est de 0.805.

### c) k-fold pour le model stepwise

On effectue 15 folds

rows=sample(nrow(tabmed))  
tabmedk=tabmed[rows,]  
  
## folds  
  
k=15  
fold=cut(seq(1,nrow(tabmed)),breaks=k, labels=FALSE)  
##  
accuracyk.step=c()  
recallk.step=c()  
error\_ratek.step=c()  
precisionk.step=c()  
  
for (i in 1:k){  
 test\_rows=which(fold==i, arr.ind=TRUE)  
 tab.testk=tabmedk[test\_rows,]  
 tab.traink=tabmedk[-test\_rows,]  
   
 # regression logistique  
 model.full=glm(medHousePriceBin~., family=binomial, data=tab.traink)  
 model.stepk=step(model.full,direction='both')  
   
 # prediction  
   
 pred\_stepk=predict.glm(model.stepk, newdata= tab.testk, type="response")  
 Y.pred.step=as.numeric(pred\_stepk>=Threshold)  
 confusion\_matrix=table(Y.pred.step, tab.testk$medHousePriceBin)  
 accuracyk.step[i]=sum(diag(confusion\_matrix))/sum(confusion\_matrix)\*100 # prediction accuracy   
 precisionk.step[i] =confusion\_matrix[2,2]/sum(confusion\_matrix[2,])\*100  
 recallk.step[i]=confusion\_matrix[2,2]/sum(confusion\_matrix[,2])\*100   
 error\_ratek.step[i]=sum(diag(confusion\_matrix[1:2,2:1]))/sum(confusion\_matrix)\*100  
}

## Start: AIC=290.05  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 276.11 288.11  
## <none> 276.05 290.05  
## - X1.transaction.date 1 278.44 290.44  
## - X4.number.of.convenience.stores 1 284.80 296.80  
## - X5.latitude 1 297.70 309.70  
## - X2.house.age 1 303.28 315.28  
## - X3.distance.to.the.nearest.MRT.station 1 318.97 330.97  
##   
## Step: AIC=288.11  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 276.11 288.11  
## - X1.transaction.date 1 278.46 288.46  
## + X6.longitude 1 276.05 290.05  
## - X4.number.of.convenience.stores 1 284.81 294.81  
## - X5.latitude 1 298.19 308.19  
## - X2.house.age 1 303.44 313.44  
## - X3.distance.to.the.nearest.MRT.station 1 337.69 347.69  
## Start: AIC=264.43  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 250.75 262.75  
## <none> 250.43 264.43  
## - X1.transaction.date 1 256.57 268.57  
## - X4.number.of.convenience.stores 1 257.29 269.29  
## - X5.latitude 1 276.42 288.42  
## - X2.house.age 1 281.40 293.40  
## - X3.distance.to.the.nearest.MRT.station 1 315.76 327.76  
##   
## Step: AIC=262.75  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 250.75 262.75  
## + X6.longitude 1 250.43 264.43  
## - X1.transaction.date 1 256.72 266.72  
## - X4.number.of.convenience.stores 1 257.33 267.33  
## - X5.latitude 1 277.19 287.19  
## - X2.house.age 1 281.74 291.74  
## - X3.distance.to.the.nearest.MRT.station 1 334.27 344.27  
## Start: AIC=298.95  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 285.27 297.27  
## <none> 284.95 298.95  
## - X1.transaction.date 1 289.52 301.52  
## - X4.number.of.convenience.stores 1 298.06 310.06  
## - X5.latitude 1 303.16 315.16  
## - X2.house.age 1 306.47 318.47  
## - X3.distance.to.the.nearest.MRT.station 1 327.37 339.37  
##   
## Step: AIC=297.27  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 285.27 297.27  
## + X6.longitude 1 284.95 298.95  
## - X1.transaction.date 1 289.68 299.68  
## - X4.number.of.convenience.stores 1 298.06 308.06  
## - X5.latitude 1 303.90 313.90  
## - X2.house.age 1 307.16 317.16  
## - X3.distance.to.the.nearest.MRT.station 1 344.57 354.57  
## Start: AIC=292.54  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 278.67 290.67  
## <none> 278.54 292.54  
## - X1.transaction.date 1 284.62 296.62  
## - X4.number.of.convenience.stores 1 287.68 299.68  
## - X5.latitude 1 298.74 310.74  
## - X2.house.age 1 310.37 322.37  
## - X3.distance.to.the.nearest.MRT.station 1 326.06 338.06  
##   
## Step: AIC=290.67  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 278.67 290.67  
## + X6.longitude 1 278.54 292.54  
## - X1.transaction.date 1 284.67 294.67  
## - X4.number.of.convenience.stores 1 287.68 297.68  
## - X5.latitude 1 299.42 309.42  
## - X2.house.age 1 310.73 320.73  
## - X3.distance.to.the.nearest.MRT.station 1 345.77 355.77  
## Start: AIC=299.9  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 286.07 298.07  
## <none> 285.90 299.90  
## - X1.transaction.date 1 288.64 300.64  
## - X4.number.of.convenience.stores 1 295.19 307.19  
## - X5.latitude 1 309.99 321.99  
## - X2.house.age 1 313.17 325.17  
## - X3.distance.to.the.nearest.MRT.station 1 329.96 341.96  
##   
## Step: AIC=298.07  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 286.07 298.07  
## - X1.transaction.date 1 288.75 298.75  
## + X6.longitude 1 285.90 299.90  
## - X4.number.of.convenience.stores 1 295.19 305.19  
## - X5.latitude 1 310.75 320.75  
## - X2.house.age 1 313.76 323.76  
## - X3.distance.to.the.nearest.MRT.station 1 347.30 357.30  
## Start: AIC=299.55  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 285.94 297.94  
## <none> 285.55 299.55  
## - X1.transaction.date 1 290.16 302.16  
## - X4.number.of.convenience.stores 1 296.08 308.08  
## - X5.latitude 1 305.64 317.64  
## - X2.house.age 1 310.43 322.43  
## - X3.distance.to.the.nearest.MRT.station 1 329.63 341.63  
##   
## Step: AIC=297.94  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 285.94 297.94  
## + X6.longitude 1 285.55 299.55  
## - X1.transaction.date 1 290.37 300.37  
## - X4.number.of.convenience.stores 1 296.14 306.14  
## - X5.latitude 1 306.49 316.49  
## - X2.house.age 1 311.14 321.14  
## - X3.distance.to.the.nearest.MRT.station 1 345.91 355.91  
## Start: AIC=301.03  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 287.16 299.16  
## <none> 287.03 301.03  
## - X1.transaction.date 1 290.23 302.23  
## - X4.number.of.convenience.stores 1 296.86 308.86  
## - X5.latitude 1 305.59 317.59  
## - X2.house.age 1 310.06 322.06  
## - X3.distance.to.the.nearest.MRT.station 1 334.42 346.42  
##   
## Step: AIC=299.16  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 287.16 299.16  
## - X1.transaction.date 1 290.30 300.30  
## + X6.longitude 1 287.03 301.03  
## - X4.number.of.convenience.stores 1 296.88 306.88  
## - X5.latitude 1 306.13 316.13  
## - X2.house.age 1 310.45 320.45  
## - X3.distance.to.the.nearest.MRT.station 1 352.85 362.85  
## Start: AIC=301.42  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 287.75 299.75  
## <none> 287.42 301.42  
## - X1.transaction.date 1 291.41 303.41  
## - X4.number.of.convenience.stores 1 297.69 309.69  
## - X5.latitude 1 306.90 318.90  
## - X2.house.age 1 311.93 323.93  
## - X3.distance.to.the.nearest.MRT.station 1 333.57 345.57  
##   
## Step: AIC=299.75  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 287.75 299.75  
## + X6.longitude 1 287.42 301.42  
## - X1.transaction.date 1 291.64 301.64  
## - X4.number.of.convenience.stores 1 297.69 307.69  
## - X5.latitude 1 307.83 317.83  
## - X2.house.age 1 312.58 322.58  
## - X3.distance.to.the.nearest.MRT.station 1 349.05 359.05  
## Start: AIC=302.77  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 288.80 300.80  
## <none> 288.77 302.77  
## - X1.transaction.date 1 293.70 305.70  
## - X4.number.of.convenience.stores 1 297.50 309.50  
## - X5.latitude 1 312.84 324.84  
## - X2.house.age 1 322.49 334.49  
## - X3.distance.to.the.nearest.MRT.station 1 332.82 344.82  
##   
## Step: AIC=300.8  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 288.80 300.80  
## + X6.longitude 1 288.77 302.77  
## - X1.transaction.date 1 293.70 303.70  
## - X4.number.of.convenience.stores 1 297.64 307.64  
## - X5.latitude 1 313.38 323.38  
## - X2.house.age 1 322.81 332.81  
## - X3.distance.to.the.nearest.MRT.station 1 350.97 360.97  
## Start: AIC=291.73  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 278.10 290.10  
## <none> 277.73 291.73  
## - X1.transaction.date 1 280.63 292.63  
## - X4.number.of.convenience.stores 1 290.68 302.68  
## - X5.latitude 1 300.24 312.24  
## - X2.house.age 1 308.75 320.75  
## - X3.distance.to.the.nearest.MRT.station 1 320.21 332.21  
##   
## Step: AIC=290.1  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 278.10 290.10  
## - X1.transaction.date 1 280.89 290.89  
## + X6.longitude 1 277.73 291.73  
## - X4.number.of.convenience.stores 1 290.68 300.68  
## - X5.latitude 1 301.05 311.05  
## - X2.house.age 1 309.45 319.45  
## - X3.distance.to.the.nearest.MRT.station 1 335.06 345.06  
## Start: AIC=302.77  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 288.84 300.84  
## <none> 288.77 302.77  
## - X1.transaction.date 1 291.89 303.89  
## - X4.number.of.convenience.stores 1 299.13 311.13  
## - X5.latitude 1 313.00 325.00  
## - X2.house.age 1 316.93 328.93  
## - X3.distance.to.the.nearest.MRT.station 1 332.48 344.48  
##   
## Step: AIC=300.84  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 288.84 300.84  
## - X1.transaction.date 1 291.91 301.91  
## + X6.longitude 1 288.77 302.77  
## - X4.number.of.convenience.stores 1 299.17 309.17  
## - X5.latitude 1 313.52 323.52  
## - X2.house.age 1 317.23 327.23  
## - X3.distance.to.the.nearest.MRT.station 1 350.16 360.16  
## Start: AIC=299.24  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 285.74 297.74  
## <none> 285.24 299.24  
## - X1.transaction.date 1 288.20 300.20  
## - X4.number.of.convenience.stores 1 290.33 302.33  
## - X5.latitude 1 305.97 317.97  
## - X2.house.age 1 312.34 324.34  
## - X3.distance.to.the.nearest.MRT.station 1 339.14 351.14  
##   
## Step: AIC=297.74  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 285.74 297.74  
## - X1.transaction.date 1 288.60 298.60  
## + X6.longitude 1 285.24 299.24  
## - X4.number.of.convenience.stores 1 290.50 300.50  
## - X5.latitude 1 307.29 317.29  
## - X2.house.age 1 313.13 323.13  
## - X3.distance.to.the.nearest.MRT.station 1 357.05 367.05  
## Start: AIC=300.93  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 287.28 299.28  
## <none> 286.93 300.93  
## - X1.transaction.date 1 291.18 303.18  
## - X4.number.of.convenience.stores 1 293.19 305.19  
## - X5.latitude 1 303.52 315.52  
## - X2.house.age 1 312.54 324.54  
## - X3.distance.to.the.nearest.MRT.station 1 342.55 354.55  
##   
## Step: AIC=299.28  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 287.28 299.28  
## + X6.longitude 1 286.93 300.93  
## - X1.transaction.date 1 291.40 301.40  
## - X4.number.of.convenience.stores 1 293.20 303.20  
## - X5.latitude 1 304.24 314.24  
## - X2.house.age 1 313.05 323.05  
## - X3.distance.to.the.nearest.MRT.station 1 361.60 371.60  
## Start: AIC=299.58  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 285.59 297.59  
## <none> 285.58 299.58  
## - X1.transaction.date 1 291.40 303.40  
## - X4.number.of.convenience.stores 1 294.74 306.74  
## - X5.latitude 1 309.99 321.99  
## - X2.house.age 1 315.86 327.86  
## - X3.distance.to.the.nearest.MRT.station 1 326.53 338.53  
##   
## Step: AIC=297.59  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 285.59 297.59  
## + X6.longitude 1 285.58 299.58  
## - X1.transaction.date 1 291.41 301.41  
## - X4.number.of.convenience.stores 1 294.90 304.90  
## - X5.latitude 1 310.27 320.27  
## - X2.house.age 1 316.04 326.04  
## - X3.distance.to.the.nearest.MRT.station 1 343.95 353.95  
## Start: AIC=294.49  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude + X6.longitude  
##   
## Df Deviance AIC  
## - X6.longitude 1 280.68 292.68  
## <none> 280.49 294.49  
## - X1.transaction.date 1 287.31 299.31  
## - X4.number.of.convenience.stores 1 291.83 303.83  
## - X5.latitude 1 303.12 315.12  
## - X2.house.age 1 307.53 319.53  
## - X3.distance.to.the.nearest.MRT.station 1 327.56 339.56  
##   
## Step: AIC=292.68  
## medHousePriceBin ~ X1.transaction.date + X2.house.age + X3.distance.to.the.nearest.MRT.station +   
## X4.number.of.convenience.stores + X5.latitude  
##   
## Df Deviance AIC  
## <none> 280.68 292.68  
## + X6.longitude 1 280.49 294.49  
## - X1.transaction.date 1 287.39 297.39  
## - X4.number.of.convenience.stores 1 291.84 301.84  
## - X5.latitude 1 303.87 313.87  
## - X2.house.age 1 308.05 318.05  
## - X3.distance.to.the.nearest.MRT.station 1 344.75 354.75

mean(recallk.step)

## [1] 87.42318

mean(error\_ratek.step)

## [1] 17.60141

mean(accuracyk.step)

## [1] 82.39859

mean(precisionk.step)

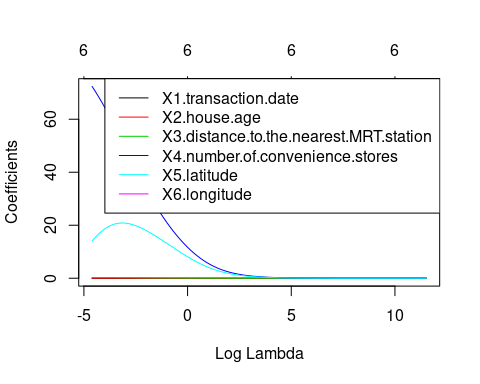
## [1] 80.56085

En utilisant la validation croisée, on remarque que nous avons une performance globale qui se situe autour de 80%, avec un taux d’erreur autour de 20%.

## I.3 Ridge regression

On effectue ici une regression ridge. ### a )Ridge model

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



### b) Selection du par cross validation

####################### cross validation  
ridge.cv.out<-cv.glmnet(X.train, Y.train, alpha = 0,nfolds = 10,family = "binomial"); ridge.cv.out # on sélectionne la meilleure valeur de lambda par validation croisée

##   
## Call: cv.glmnet(x = X.train, y = Y.train, nfolds = 10, alpha = 0, family = "binomial")   
##   
## Measure: Binomial Deviance   
##   
## Lambda Measure SE Nonzero  
## min 0.02884 0.7427 0.03697 6  
## 1se 0.05531 0.7756 0.03343 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.

### c) Prediction du model Ridge

ridge.pred <- predict(model.ridge, s = ridge.lamb.min, newx = X.test,type = 'response')  
Y.pred.ridge = as.integer(ridge.pred >= Threshold)  
confusion\_matrix.ridge = table(Y.pred.ridge,Y.test) # matrice de confusion  
confusion\_matrix.ridge

## Y.test  
## Y.pred.ridge 0 1  
## 0 31 8  
## 1 10 28

### d) Performance du model

pred.accuracy.ridge = sum(diag(confusion\_matrix.ridge))/sum(confusion\_matrix.ridge)\*100# prediction accuracy  
pred.recall.ridge = confusion\_matrix.ridge[2,2]/sum(confusion\_matrix.ridge[,2])\*100 # proportion des hauts prix bien predit   
pred.specifity.ridge = confusion\_matrix.ridge[1,1]/sum(confusion\_matrix.ridge[,1])\*100 # proportion des bas prix bien predits  
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 # proportion des mauvaises predictions  
  
pred.accuracy.ridge

## [1] 76.62338

pred.recall.ridge

## [1] 77.77778

pred.specifity.ridge

## [1] 75.60976

pred.error\_rate.ridge

## [1] 23.37662

On retrouve une performance de 76% sur le modèle. Le modèle predit bien la classe des hauts prix à 77% et celle de pas prix à 75%.

### e) k-folds le model ridge

##shuffling  
set.seed(1234)  
rows <- sample(nrow(tabmed)) # melange des rows du dataset  
tabmedk <- tabmed[rows, ]  
## folds  
k = 15 #as.integer(1/(1-r)) ## fold number  
fold = cut(seq(1,nrow(tabmed)), breaks = k,labels = FALSE) #organisations des lignes du  
#dataset en plusieurs fold  
#initialisation des vectueurs qui va contenier les performances a chaque etape  
accuracyk.ridge = c()  
recallk.ridge = c()  
precisionk.ridge = c()  
error\_ratek.ridge = c()  
for (i in 1:k) {  
 test\_rows = which(fold == i,arr.ind = TRUE)   
 tab.testk = tabmedk[test\_rows,]  
 tab.traink = tabmedk[-test\_rows,]  
 X.traink = as.matrix(tab.traink[,-dim(tab.traink)[2]])  
 X.testk = as.matrix(tab.testk[,-dim(tab.testk)[2]])  
 Y.testk = tab.testk$medHousePriceBin  
 Y.traink = tab.traink$medHousePriceBin  
 ### regression logistic   
 model.ridgek <- glmnet(X.traink,Y.traink,alpha=0,lambda = grid,family = "binomial")  
 ### prediction  
 ####################### cross validation  
 ridge.cv.out<-cv.glmnet(X.traink, Y.traink, alpha = 0,nfolds = 10,family = "binomial") # on sélectionne la meilleure valeur de lambda par validation croisée  
 ridge.lamb.mink<-ridge.cv.out$lambda.min # le meilleur lambda est celui qui produit the min MSE  
   
 prob <- predict(model.ridgek, s = ridge.lamb.mink, newx = X.testk,type = 'response')  
 Y.pred.ridge = as.integer(prob >= Threshold)   
 confusion\_matrix = table(Y.pred.ridge,tab.testk$medHousePriceBin)  
 accuracyk.ridge[i] = sum(diag(confusion\_matrix))/sum(confusion\_matrix)\*100  
 recallk.ridge[i] = confusion\_matrix[2,2]/sum(confusion\_matrix[,2])\*100  
 precisionk.ridge[i] =confusion\_matrix[2,2]/sum(confusion\_matrix[2,])\*100  
 error\_ratek.ridge[i] = sum(diag(confusion\_matrix[1:2,2:1]))/sum(confusion\_matrix) \*100   
}  
  
mean(recallk.ridge)

## [1] 86.68707

mean(error\_ratek.ridge)

## [1] 18.61552

mean(accuracyk.ridge)

## [1] 81.38448

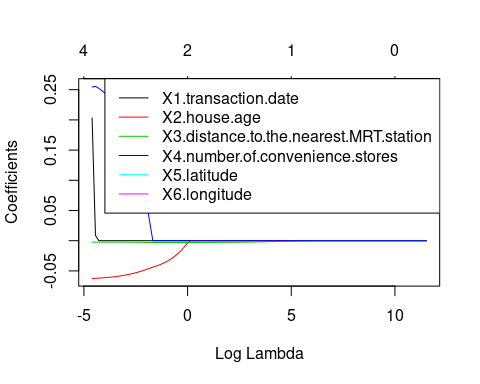
mean(precisionk.ridge)

## [1] 78.41565

## I.4) Modèle Lasso

### a) Model Lasso

grid = 10^seq(5,-2,length = 100) # sequence des lambda  
lasso=glmnet(X.train,Y.train,alpha=1,lambda=grid,family = "binomial", standardize=FALSE)  
plot(lasso,xvar="lambda",type="l",col=1:10);legend("topright",legend=colnames(X.train[,1:ncol(X.train)]), col=1:10, lty=1)

 On voit que plus lambda est grand, plus le nombre de variables sélectionnées est petit.

### b) Selection du par cross validation

lasso.cv.out=cv.glmnet(X.train, Y.train, alpha = 1,nfolds = 10,family = "binomial"); lasso.cv.out

##   
## Call: cv.glmnet(x = X.train, y = Y.train, nfolds = 10, alpha = 1, family = "binomial")   
##   
## Measure: Binomial Deviance   
##   
## Lambda Measure SE Nonzero  
## min 0.00923 0.7484 0.09288 5  
## 1se 0.04923 0.8361 0.05650 4

lasso.min=lasso.cv.out$lambda.min # on sélectionne le plus petit lambda qui minimise le MSE

On voit que plus lambda est grand, plus le nombre de variables sélectionnées est petit.

lasso.cv.out=cv.glmnet(X.train, Y.train, alpha = 1,nfolds = 10,family = "binomial"); lasso.cv.out

##   
## Call: cv.glmnet(x = X.train, y = Y.train, nfolds = 10, alpha = 1, family = "binomial")   
##   
## Measure: Binomial Deviance   
##   
## Lambda Measure SE Nonzero  
## min 0.00636 0.7230 0.08094 5  
## 1se 0.04088 0.8038 0.06328 4

lasso.min=lasso.cv.out$lambda.min # on sélectionne le plus petit lambda qui minimise le MSE

On effectue une prédiction du modèle lasso avec le plus petit lambda minimisant le MSE

### c) Prediction du model Lasso

prob= predict(lasso, s=lasso.min, newx=X.test, type="response")  
lasso.pred=as.numeric(prob>=Threshold) # transforme les données en données binaire avec un seuil de 1/2

### d) Performance du model

confusion\_matrix.lasso = table(lasso.pred,Y.test) # on crée la matrice de confusion   
confusion\_matrix.lasso

## Y.test  
## lasso.pred 0 1  
## 0 30 7  
## 1 11 29

pred.accuracy.lasso = sum(diag(confusion\_matrix.lasso))/sum(confusion\_matrix.lasso)\*100# prediction accuracy  
pred.recall.lasso = confusion\_matrix.lasso[2,2]/sum(confusion\_matrix.lasso[,2])\*100 # proportion des hauts prix bien predit   
pred.specifity.lasso = confusion\_matrix.lasso[1,1]/sum(confusion\_matrix.lasso[,1])\*100 # proportion des bas prix bien predits  
pred.precision.lasso = confusion\_matrix.lasso[2,2]/sum(confusion\_matrix.lasso[2,])\*100  
pred.error\_rate.lasso =  
sum(diag(confusion\_matrix.lasso[1:2,2:1]))/sum(confusion\_matrix.lasso) \*100 # proportion des mauvaises predictions  
  
pred.accuracy.lasso

## [1] 76.62338

pred.recall.lasso

## [1] 80.55556

pred.specifity.lasso

## [1] 73.17073

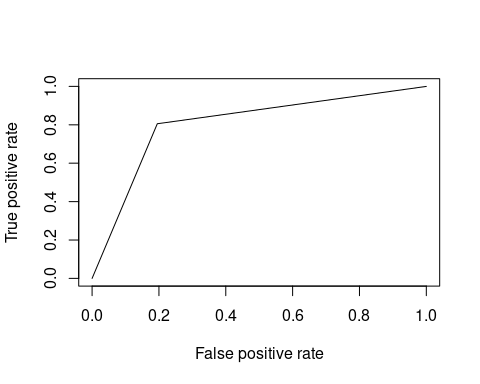
pred.error\_rate.lasso

## [1] 23.37662

Avec une performance globale de 76% on predit les hauts prix à 80% et les as prix à 76% Ce qui reste dans la gamme des performances des regressions precedentes.

## Courbe ROC et AUC

pred.lasso.min=prediction(lasso.pred,tab.test$medHousePriceBin)  
perf.lasso.min=performance(pred.lasso.min, "tpr" ,"fpr")  
plot(perf)

 La courbe est assez proche du coin supérieur gauche, ainsi le modèle semble capturer le plus possible de vrais évènements avec le moins possible de faux évènements.

ROC\_auc.lasso=performance(pred.lasso.min,"auc")  
AUC.lasso=ROC\_auc.lasso@y.values[[1]]  
print(AUC.lasso)

## [1] 0.7686314

l’AUC est de : 0.76

### e) K-folds le model lasso

##shuffling  
set.seed(1234)  
rows <- sample(nrow(tabmed)) # melange des rows du dataset  
tabmedk <- tabmed[rows, ]  
## folds  
k = 15 #as.integer(1/(1-r)) ## fold number  
fold = cut(seq(1,nrow(tabmed)), breaks = k,labels = FALSE) #organisations des lignes du  
#dataset en plusieurs fold  
#initialisation des vectueurs qui va contenier les performances a chaque etape  
accuracyk.lasso = c()  
recallk.lasso = c()  
precisionk.lasso = c()  
error\_ratek.lasso = c()  
for (i in 1:k) {  
 test\_rows = which(fold == i,arr.ind = TRUE)   
 tab.testk = tabmedk[test\_rows,]  
 tab.traink = tabmedk[-test\_rows,]  
 X.traink = as.matrix(tab.traink[,-dim(tab.traink)[2]])  
 X.testk = as.matrix(tab.testk[,-dim(tab.testk)[2]])  
 Y.testk = tab.testk$medHousePriceBin  
 Y.traink = tab.traink$medHousePriceBin  
 ### regression logistic   
 model.lassok <- glmnet(X.traink,Y.traink,alpha=1,lambda = grid,family = "binomial")  
 ### prediction  
 ####################### cross validation  
 lasso.cv.out<-cv.glmnet(X.traink, Y.traink, alpha = 1,nfolds = 10,family = "binomial")   
 # on sélectionne la meilleure valeur de lambda par validation croisée  
 lasso.lamb.mink<-lasso.cv.out$lambda.min # le meilleur lambda est celui qui produit the min MSE  
   
 prob <- predict(model.lassok, s = lasso.lamb.mink, newx = X.testk,type = 'response')  
 Y.pred.lasso = as.integer(prob >= Threshold)   
 confusion\_matrix = table(Y.pred.lasso,tab.testk$medHousePriceBin)  
 accuracyk.lasso[i] = sum(diag(confusion\_matrix))/sum(confusion\_matrix)\*100  
 recallk.lasso[i] = confusion\_matrix[2,2]/sum(confusion\_matrix[,2])\*100  
 precisionk.lasso[i] =confusion\_matrix[2,2]/sum(confusion\_matrix[2,])\*100  
 error\_ratek.lasso[i] = sum(diag(confusion\_matrix[1:2,2:1]))/sum(confusion\_matrix) \*100   
}  
mean(accuracyk.lasso)

## [1] 81.63139

mean(recallk.lasso)

## [1] 88.45316

mean(precisionk.lasso)

## [1] 78.3217

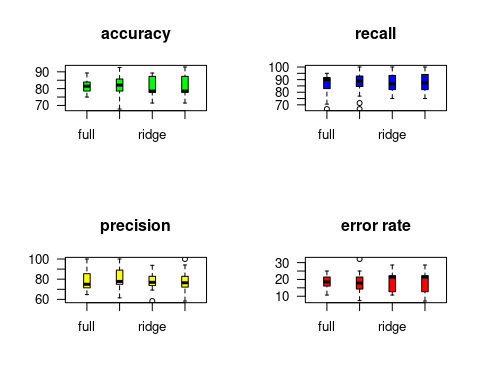
mean(error\_ratek.lasso)

## [1] 18.36861

# II )Conclusion

On va comparer les differents modeles entre eux.

par(mfrow = c(2,2))  
boxplot(data.frame(accuracyk.full,accuracyk.step,accuracyk.ridge,accuracyk.lasso),col = "green",boxwex = 0.2, las = 1,names = c("full","stepw","ridge","lasso"),main = "accuracy")  
boxplot(data.frame(recallk.full,recallk.step,recallk.ridge,recallk.lasso),col = "blue",boxwex = 0.2, las = 1,names = c("full","stepw","ridge","lasso"),main = "recall")  
boxplot(data.frame(precisionk.full,precisionk.step,precisionk.ridge,precisionk.lasso),col = "yellow",boxwex = 0.2, las = 1,names = c("full","stepw","ridge","lasso"),main = "precision")  
boxplot(data.frame(error\_ratek.full,error\_ratek.step,error\_ratek.ridge,error\_ratek.lasso),col = "red",boxwex = 0.2, las = 1,names = c("full","stepw","ridge","lasso"),main = "error rate")

 On remarque, qu’en terme d’accuracy, le stepwise est meilleur. Et aussi lorsqu’on regarde le recall la proportion des transactions à haut prix bien prédites. La moyenne regression stepwise reste légèrement au dessus des autres, avec une box bien plus courte, centrée autour de la moyenne. Donc sur la base de ces 2 indicateurs (voir 3 car error\_rate = 100-accuracy), si le but est de prédire les transactions de prix élévées(supérieur à la médiane), une régression stepwise est meilleur.