TP3 Real Estate

Vhiny-Guilley

13/11/2020

#### 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.

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)

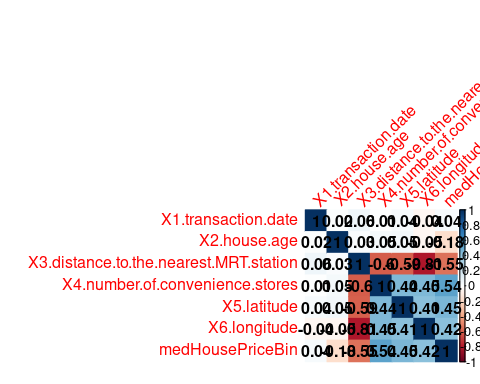
## X1.transaction.date X2.house.age X3.distance.to.the.nearest.MRT.station  
## 1 2012.917 32.0 84.87882  
## 2 2012.917 19.5 306.59470  
## 3 2013.583 13.3 561.98450  
## 4 2013.500 13.3 561.98450  
## 5 2012.833 5.0 390.56840  
## 6 2012.667 7.1 2175.03000  
## X4.number.of.convenience.stores X5.latitude 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)

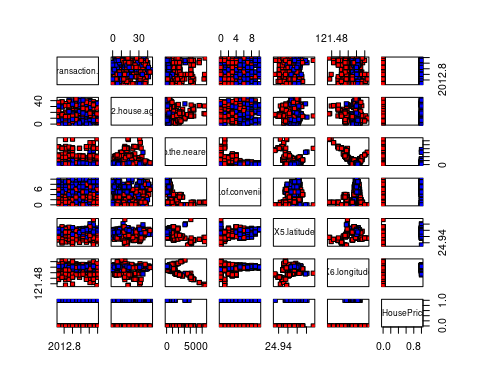
## X1.transaction.date X2.house.age X3.distance.to.the.nearest.MRT.station  
## 1 2012.917 32.0 84.87882  
## 2 2012.917 19.5 306.59470  
## 3 2013.583 13.3 561.98450  
## 4 2013.500 13.3 561.98450  
## 5 2012.833 5.0 390.56840  
## 6 2012.667 7.1 2175.03000  
## X4.number.of.convenience.stores X5.latitude 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

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 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

#### organisation du dataset

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

#### Etude du model

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   
## -2.2211 -0.4183 -0.0029 0.5745 4.3324   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) -4.697e+03 3.231e+03 -1.454 0.14602  
## X1.transaction.date 1.325e+00 5.994e-01 2.210 0.02710  
## X2.house.age -6.078e-02 1.394e-02 -4.360 1.30e-05  
## X3.distance.to.the.nearest.MRT.station -2.036e-03 4.315e-04 -4.719 2.37e-06  
## X4.number.of.convenience.stores 2.175e-01 7.067e-02 3.078 0.00208  
## X5.latitude 7.483e+01 1.823e+01 4.104 4.07e-05  
## X6.longitude 1.350e+00 2.439e+01 0.055 0.95587  
##   
## (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: 458.84 on 330 degrees of freedom  
## Residual deviance: 248.82 on 324 degrees of freedom  
## AIC: 262.82  
##   
## Number of Fisher Scoring iterations: 6

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

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

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000002 0.0829782 0.6946973 0.5309650 0.8463638 0.9807682

OR

## (Intercept) X1.transaction.date   
## 0.000000e+00 3.760857e+00   
## X2.house.age X3.distance.to.the.nearest.MRT.station   
## 9.410259e-01 9.979660e-01   
## X4.number.of.convenience.stores X5.latitude   
## 1.242986e+00 3.138718e+32   
## X6.longitude   
## 3.855627e+00

#### Performance du model

Threshold = 0.5  
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 29 5  
## 1 11 38

La matrice de confusion nous donne une performance

pred.accuracy.full = sum(diag(confusion\_matrix.full))/sum(confusion\_matrix.full)\*100# prediction accuracy  
pred.recall.full = confusion\_matrix.full[2,2]/sum(confusion\_matrix.full[,2])\*100 # probabilité de bien predire les hauts prix   
pred.specifity.full = confusion\_matrix.full[1,1]/sum(confusion\_matrix.full[,1])\*100 # probabilité de bien predire les bas prix   
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 # probabilité d'obtenir une erreur  
  
actual.accurary = as.double(table(tab.test$medHousePriceBin)[1]/sum(table(tab.test$medHousePriceBin))) # model accuracy  
pred.accuracy.full

## [1] 80.72289

pred.recall.full

## [1] 88.37209

pred.error\_rate.full

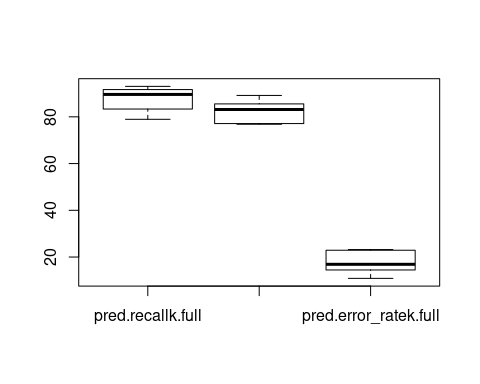
## [1] 19.27711

actual.accurary

## [1] 0.4819277

#### k-folds le model full

##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 ill ability  
 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))



mean(pred.recallk.full)

## [1] 87.31079

mean(pred.error\_ratek.full)

## [1] 17.64619

mean(pred.accuracyk.full)

## [1] 82.35381

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%.

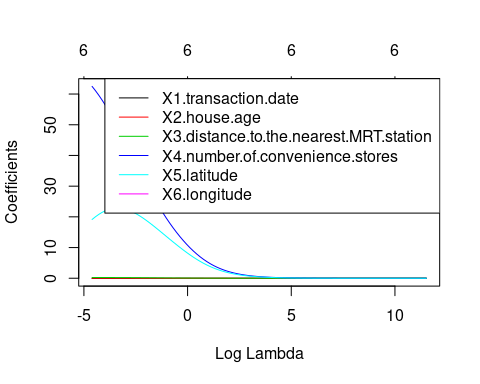
### Ridge regression

#### organisation du dataset

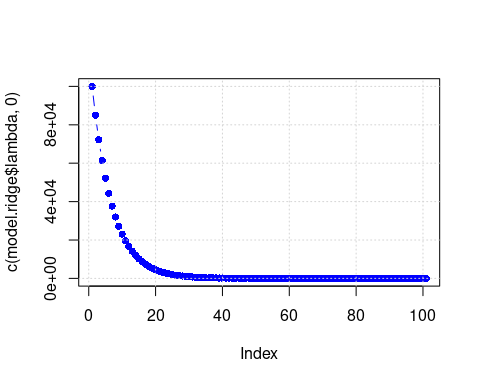
X.train = as.matrix(tab.train[,-dim(tab.train)[2]])  
X.test = as.matrix(tab.test[,-dim(tab.test)[2]])  
Y.test = tab.test$medHousePriceBin  
Y.train = tab.train$medHousePriceBin

#### 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")  
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)



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



#### 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 0.02781 0.8194 0.03575 6  
## 1se 0.06425 0.8528 0.02949 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] 0.8374513667 0.0006218433 0.0479224746 0.7618872544 0.9083927289  
## [6] 0.8914244090 0.0025448623 0.8387073736 0.6263700893 0.8379935287  
## [11] 0.4283122728 0.0014874124 0.7581875916 0.7687104581 0.8003930141  
## [16] 0.7132209826 0.7681921748 0.8936438616 0.0380245529 0.1824292464

#### prediction

Threshold = 0.5  
Y.pred.ridge = as.integer(ridge.predbest >= Threshold)  
confusion\_matrix.ridge = table(Y.pred.ridge,Y.test)  
confusion\_matrix.ridge

## Y.test  
## Y.pred.ridge 0 1  
## 0 29 6  
## 1 11 37

La matrice de confusion nous donne une performance

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 # probabilité de bien predire les hauts prix   
pred.specifity.ridge = confusion\_matrix.ridge[1,1]/sum(confusion\_matrix.ridge[,1])\*100 # probabilité de bien predire les bas prix   
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 # probabilité d'obtenir une erreur  
  
actual.accurary = as.double(table(tab.test$medHousePriceBin)[1]/sum(table(tab.test$medHousePriceBin))) # model accuracy  
pred.accuracy.ridge

## [1] 79.51807

pred.recall.ridge

## [1] 86.04651

pred.specifity.ridge

## [1] 72.5

pred.error\_rate.ridge

## [1] 20.48193

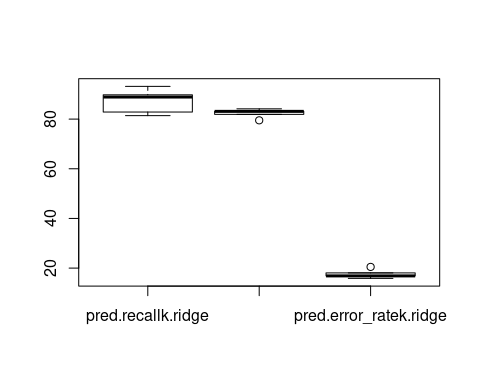
actual.accurary

## [1] 0.4819277

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

#### k-folds le model ridge

##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 accuracy  
 pred.recallk.ridge[i] = confusion\_matrix[2,2]/sum(confusion\_matrix[,2])\*100 # the prediction of being ill ability  
 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))



mean(pred.recallk.ridge)

## [1] 87.22382

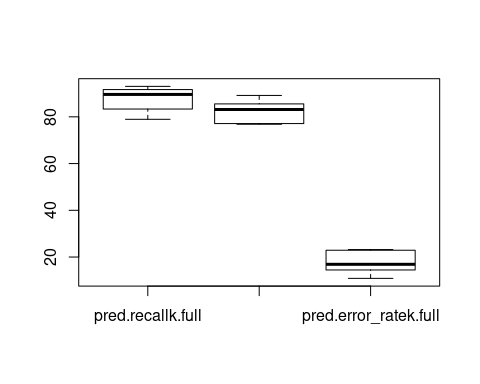
mean(pred.error\_ratek.ridge)

## [1] 17.62856

mean(pred.accuracyk.ridge)

## [1] 82.37144

boxplot(data.frame(pred.recallk.full,pred.accuracyk.full,pred.error\_ratek.full))



mean(pred.recallk.full)

## [1] 87.31079

mean(pred.error\_ratek.full)

## [1] 17.64619

mean(pred.accuracyk.full)

## [1] 82.35381