Résumé Sélection d'attributs
BDM : 2CS
Mme HAMDAD

<u>Libraires:</u>*library(mlbench),library(FactoMineR),library(factoextra),library(FSelector),library(cluster), library(MASS), library(glmnet).*

On charge et analyse notre dataset:

```
data(BostonHousing)
BostonHousing
dataset = BostonHousing

# Caractéristiques du dataset
summary(dataset)
dim(dataset)
names(dataset)

# Convertir chas (factor)=> (numeric)
dataset$chas <- as.numeric(as.character(dataset$chas))
cor(dataset)
str(dataset)
```

<u>Régression linéaire</u>: *y=medv*

```
reg = lm(medv~.,dataset)
summary(reg)

# Calculer l'erreur(MSE+RMSE)
mse <- mean(resid(reg)^2)
mse
rmse = sqrt(mse)
rmse
```

<u>résultat:</u>

```
> # Calculer l'erreur
> mse <- mean(resid(reg)^2)
> mse
[1] 21.89483
> rmse = sqrt(mse)
> rmse
[1] 4.679191
>
```

<u>Régression Effet non linéaire</u>: *y=medv*

D'après les résultats de la régression linéaire pure, on peut facilement voir les variables qui n'ont pas d'effet linéaire (ie ayant une p-value élevée).
Ces variables sont: Age et Indus.
<u>Remarque</u>: le nombre d'étoiles à droite indique si la variable a vraiment un effet ou pas.
`age` et `indus` ont 0 étoile et donc aucun effet linéaire contrairement à `nox` et `b`.

<u>Il faut qu'on les omis.</u>

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
crim        -1.080e-01  3.286e-02  -3.287 0.001087 ** 
zn           4.642e-02  1.373e-02   3.382 0.000778 ***
indus        2.056e-02  6.150e-02   0.334 0.738288    
chas         2.687e+00  8.616e-01   3.118 0.001925 ** 
nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
age          6.922e-04  1.321e-02   0.052 0.958229    
dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
tax         -1.233e-02  3.760e-03  -3.280 0.001112 ** 
ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
b            9.312e-03  2.686e-03   3.467 0.000573 ***
lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
---
```

```
# Suppression des variables qui n'ont pas d'effet linéaire
dataset$indus <- NULL
dataset$age <- NULL

# Refaire la régression linéaire
reg2 = lm(medv~.,dataset)
summary(reg2)

# Calculer l'erreur
mse <- mean(resid(reg2)^2)
mse
rmse = sqrt(mse)
rmse
```

<u>résultat:</u>

```
> # Calculer l'erreur
> mse <- mean(resid(reg2)^2)
> mse
[1] 21.89993
> rmse = sqrt(mse)
> rmse
[1] 4.679736
```

<u>Régression avec ACP</u>: *y=medv*

```
# ACP
dataset2 = BostonHousing
# Attention: l'ACP doit se faire sans inclure la variable explicative medv
dataset2 <- dataset2[0:13]
dataset2
dataset2$chas <- as.numeric(as.character(dataset2$chas))

# Effectuer l'ACP et spécifier le nombre de composantes principales à garder
resultats_acp <- PCA(dataset2, scale = TRUE)
summary(resultats_acp)
```

Eigenvalues

	Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10	Dim.11	Dim.12	Dim.13
Variance	6.127	1.433	1.243	0.858	0.835	0.657	0.535	0.396	0.277	0.220	0.186	0.169	0.064
% of var.	47.130	11.025	9.559	6.597	6.422	5.057	4.118	3.047	2.130	1.694	1.431	1.302	0.489
Cumulative % of var.	47.130	58.155	67.713	74.310	80.732	85.789	89.907	92.954	95.084	96.778	98.209	99.511	100.000

```
# Cumulative % of var -> DIM.5 = 80.732 > 80 (garder 5)
dataset2 = resultats_acp$ind$coord[,1:5]


# Ajouter medv pour faire la regl
medv = BostonHousing[14]
data = cbind(dataset2,medv)

# Reg
# Refaire la régression linéaire
reg3 = lm(medv~.,data)
summary(reg3)

# Calculer l'erreur
mse <- mean(resid(reg3)^2)
mse
rmse = sqrt(mse)
rmse
```

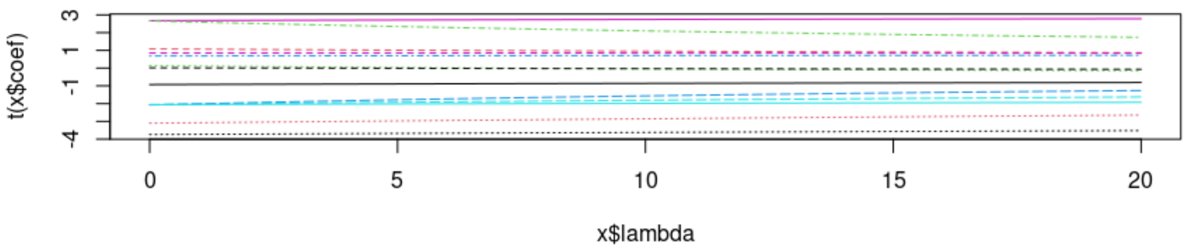
<u>résultat:</u>

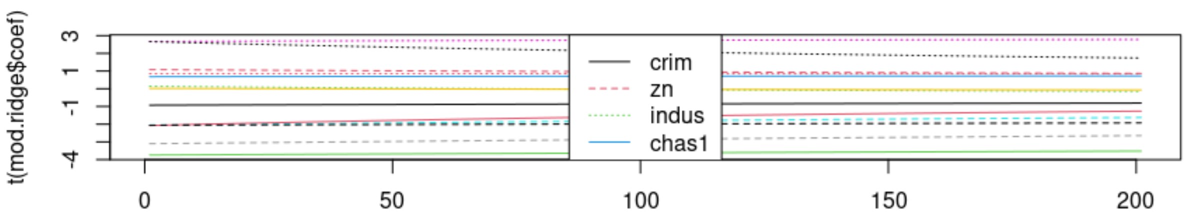
```
> # Calculer l'erreur
> mse <- mean(resid(reg3)^2)
> mse
[1] 25.58086
> rmse = sqrt(mse)
> rmse
[1] 5.057753
```

<u>Régression avec RIDGE</u>: *y=medv*

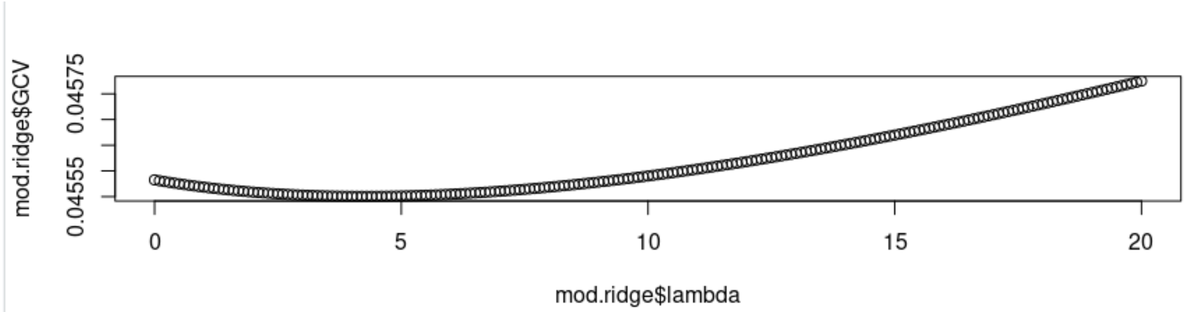
```
# Reg + Ridge
dataset4 = BostonHousing
# En quelque sorte: Grid search: Faire varier lambda
mod.ridge = lm.ridge(medv~.,dataset4,lambda=seq(0,20,0.1))
par(mfrow=c(2,1))

# schématiser les variations des coefficients des variables selon les lambdas
plot(mod.ridge)
matplot(t(mod.ridge$coef),lty=1:3,type='l',col=1:10)
legend("top",legend=rownames(mod.ridge$coef), col=1:10,lty=1:3)
```






```
# cross validation (ici pour chercher valeur optimale de lambda)
plot(mod.ridge$lambda,mod.ridge$GCV)
summary(mod.ridge$lambda,mod.ridge$GCV)
select(mod.ridge)
```



On voit que la valeur optimale est entre 0 et 5 $\approx 4\, ou\, 4.5$

```
> select(mod.ridge)
modified HKB estimator is 4.594163
modified L-W estimator is 3.961575
smallest value of GCV  at 4.3
```

```
# On prend donc lambda = 4.3

# refaire avec lambda optimal
mod.ridge$coef

# comparer avec mse
mod.ridge=lm.ridge(medv~crim+zn+chas+nox+rm+dis+rad+tax+ptratio+b+lstat,dataset4,la
mbda=4.3)

X.matrix
<-cbind(rep(1,length=length(dataset4$medv)),dataset4$crim,dataset4$zn,dataset4$chas
```

```
,dataset4$nox,dataset4$rm,dataset4$dis,dataset4$rad,dataset4$tax,dataset4$ptratio,d
ataset4$b,dataset4$lstat)
X.matrix1 <- as.matrix(dataset4[,-9])
head(X.matrix)

# Regularisation
fitted.vals<-X.matrix
%*%c(32.918471472,-0.104315656,0.043678727,2.747912270,-17.683126171,3.851764685,-1
.426638997,0.272261020,-0.010629921,-0.935344659,0.009278324,-0.516975577)

# Calculer l'erreur
mse.ridge=mean((dataset4$medv-fitted.vals)^2)
mse.ridge
rse.ridge = sqrt(mse.ridge)
rse.ridge
```

<u>résultat:</u>

```
> mse.ridge=mean((dataset4$medv-fitted.vals)^2)
> mse.ridge
[1] 21.96097
> rse.ridge = sqrt(mse.ridge)
> rse.ridge
[1] 4.686254
```

<u>Régression avec LASSO</u>: *y=medv*

```
# Reg + LASSO
dataset5 = BostonHousing

# Transformer les données en matrice
matrixx = model.matrix(medv~.,dataset5)

# var de sortie
Y = dataset5$medv

# Faire la régression Lasso (varier lambda)
lasso = cv.glmnet(matrixx,Y, alpha = 1, lambda = seq(0,1000, 10), grouped = FALSE,
nfolds = nrow(dataset5))

# Afficher le graphe
plot(lasso,main = "Choix de lambda")

# Calculer la valeur optimal de lambda
lambda_optimal = lasso$lambda.min
lambda_optimal

# Voir les coeff
m_lasso = glmnet(matrixx,Y, alpha = 1, lambda = lambda_optimal)
coef(m_lasso)

# Calculer l'erreur (MSE+RMSE)
mse.lasso = min(lasso$cvm)
mse.lasso
rmse.lasso = sqrt(mse.lasso)
rmse.lasso
```

<u>résultat:</u>

```
> # Afficher l'erreur (MSE)
> mse.lasso = min(lasso$cvm)
> mse.lasso
[1] 26.57231
> rmse.lasso = sqrt(mse.lasso)
> rmse.lasso
[1] 5.154833
```

Recapitulatif:

Methode	MSE	RMSE	Justification
Régression linéaire	21.89483	4.679191	Avec régression linéaire pure, l’erreur obtenu $MSE \approx 23$
Régression linéaire ENL	21.89993	4.679736	En éliminant les deux variables (age et indus), on a perdu une quantité d'informations. Car age et indus sont fortement corrélés à des variables ayant un effet linéaire considérable qui a été affaiblie => MSE augmente.
Régression linéaire avec ACP	25.58086	5.057753	L’ACP nous a fait perdre une quantité d’informations plus ou moins importante et donc la MSE augmente encore plus. (On a pris 5 composantes principales d'après 13 variables)
Régression linéaire avec RIDGE	21.96097	4.686254	RIDGE n'élimine pas les variables qui n’ont pas un effet linéaire mais par contre associe un coefficient à chacune des variables selon un lambda optimal et c’est pour cela que la MSE a diminuée par rapport à la régression avec ACP.
Régression linéaire avec LASSO	26.57231	5.154833	LASSO a fait perdre une quantité d’informations également en éliminant 4 variables de 13. Ce qui explique l’augmentation remarquable de la MSE.

Conclusion: Les résultats obtenus à partir de différentes méthodes de régression linéaire appliquées à un ensemble de données montrent que chaque méthode a ses avantages et ses limites. Mais, quand on est face a des Big DATAs, il devient très intéressant et même crucial d’opter pour les deux méthodes les plus utilisees dans le domaine du Deep Learning: RIDGE et LASSO.