SY19 - TP3

Regression et classification - Selection de modele

Theodore BOURGEON - Cyril LAY

### Classifieur

#### Recuperation des donnees

data <-read.table("tp3\_a18\_clas\_app.txt")

#### Observation des donnees

head(data)  
dim(data)

On sait ainsi que la colonne de validation est la derniere nommee “y”, qu’on a 200 observations et 36 predicteurs.

#### Graphique des donnees

On observe ainsi si une correlation particuliere s’observe graphiquement. Ce n’est pas le cas donc nous avons choisi de ne pas afficher ces graphes sans interet majeur.

pairs(data[,1:5])  
pairs(data[,6:10])

##### Selection de predicteurs

On recherche les colonnes ayant le plus gros impact pour affiner le modele: - On commence par effectuer une regression lineaire sur ces coefficiants

summary(lm(y~.,data=data))

On constate que les colonnes, x9 + x10 + x11 + x21 + x28 + x29 + x31 + x32 + x33 + x35 ont un poids important dans la classification. La p-value est tres petite, on rejette l’hypothese selon laquelle les colonnes ne sont pas correlees avec y.

* Ensuite on procede a differentes sous selections :

La premiere est **“la meilleur sous selection”** (qui est parfois impossible a cause d’un nombre de predicteurs trop grand, ici 2^36 combinaisons possibles, ce qui peut faire exploser le nombre de combinaisons). “Exhaustive” permet de tester toutes les combinaisons possibles des predicteurs parmis nvmax predicteurs et sors les meilleurs resultats.

library(leaps)  
regsubset <- regsubsets(y~., data=data, method = "exhaustive", nvmax = 36)  
plot(regsubset, scale = "r2")  
plot(regsubset, scale = "bic")  
  
res.sum <- summary(regsubset)  
data.frame(  
 Adj.R2 = which.max(res.sum$adjr2),  
 BIC = which.min(res.sum$bic)  
)

On constate que x8+x9+x10+x11+x13+x14+x15+x17+x19+x21+x23+x24+x25+x28+x29+x31+x32+x33+x34+x35 semble etre une solution pertinente.

D’apres les valeurs de sous selections on obtient le nombre optimal de predicteur d’apres BIC et r^2 ajustee.

* R^2 ajuste (18 predicteurs) :X9+X10+X11+X13+X14+X15+X17+X19+X21+X23+X25+X28+X29+X31+X32+X33+X34+X35
* BIC (8 predicteurs) : X9+X10+X21+X28+X29+X32+X33+X35

La seconde est **“forward stepwise selection”** (qui est permet d’etudier le poid des predicteurs quand p est trop grand). “Forward” commence avec aucun predicteur et ajoute au fur et a mesure celui qui a le plus d’impact parmis nvmax predicteurs.

On constate que x9+x10+x11+x13+x14+x15+x17+x19+x21+x23+x24+x25+x28+x29+x31+x32+x33+x34+x35 semble etre une solution pertinente.

La derniere est **“backward stepwise selection”** (qui est permet d’etudier le poid des predicteurs quand p est trop grand et que le nombre d’observations est plus grand que le mnombre de predicteurs). “Backward” commence avec tous les predicteurs et retire au fur et a mesure celui qui a le moins d’impact parmis nvmax predicteurs.

On constate que x9+x10+x11+x13+x14+x15+x17+x19+x21+x23+x28+x29+x31+x32+x33+x34+x35 semble etre une solution pertinente.

#### Modele possible

On choisit donc une serie de modele qu’on va pouvoir tester par la suite pour voir leurs reactions a travers differentes fonctions. On evaluera leur pertinance avec differentes methodes de validation.

Formula <- c(  
 y~.,  
 y~X9+X10+X11+X21+X28+X31+X32+X33+X35,  
 y~X8+X9+X10+X11+X13+X14+X15+X17+X19+X21+X23+X24+X25+X28+X29+X31+X32+X33+X34+X35,  
 y~X9+X10+X11+X13+X14+X15+X17+X19+X21+X23+X24+X25+X28+X29+X31+X32+X33+X34+X35,  
 y~X9+X10+X11+X13+X14+X15+X17+X19+X21+X23+X28+X29+X31+X32+X33+X34+X35,  
 y~X9+X10+X11+X13+X14+X15+X17+X19+X21+X23+X25+X28+X29+X31+X32+X33+X34+X35,  
 y~X9+X10+X21+X28+X29+X32+X33+X35)

On va desormais tester ces differentes selections.

#### Adjustement to the training error

Dans un premier temps on effectue cela pour LDA avec un ajustement de l’erreur d’entrainement. (on estime donc indirectement l’erreur test) Dans ce cas, on a pas besoin de separer notre jeu de donnee en un set de training et un set de test. On applique le modele sur les differentes selections et on estime l’erreur avec : R^2 (specific to regression), AIC(for a large number of class) et BIC (heavier penlaty for models with many variables)

library(MASS)  
for (i in (1:7)){  
 print(Formula[[i]])  
 lm.adj <- lm(Formula[[i]], data=data)  
 print(summary(lm.adj)$r.squared)  
 print(summary(lm.adj)$adj.r.squared)  
 print(AIC(lm.adj))  
 print(BIC(lm.adj))  
}

AIC n’est present qu’a titre pedagogique afin de se rendre compte de sa non pertinance. En effet, notre modele a seulement 2 classes. Il semble dans ce cas que la meilleur sous selection serait :

* Si on considere R squred adjusted : y ~ X9 + X10 + X11 + X13 + X14 + X15 + X17 + X19 + X21 + X23 + X25 + X28 + X29 + X31 + X32 + X33 + X34 + X35
* Si on considere BIC : y ~ X9 + X10 + X21 + X28 + X29 + X32 + X33 + X35

### LDA

LDA s’utilise principalement quand on considere que les classes ont une matrice de covariance identique.

data.classe1 <- data[data$y=="1",]  
data.classe2 <- data[data$y=="2",]  
(cov(data.classe1) - cov(data.classe2))

On constate ainsi qu’il s’agit d’une hypothese grossiere dans notre cas. LDA ne donnera surement pas le meilleur taux d’erreur.

#### Direct estimation of the test error

##### Validation-set approach

Pour la validation-set approach nous moyennerons cela sur 100 observations et nous ferons un boxplot pour se rendre compte de la variance de l’erreur. ###### Pour 2/3 train et 1/3 test

On constate ici que le taux d’erreur est minimal pour le modele possedant uniquement les colonnesX9 + X10 + X21 + X28 + X29 + X32 + X33 + X35 ou l’erreur est de 17,16418 % pour une set-validation a 2/3 pour le training et 1/3 pour les tests.

###### Pour 1/2 train et 1/2 test

Le taux d’erreur n’est pas meilleur.

###### Pour 3/4 train et 1/4 test

Ainsi, il semblerait qu’avec 150 lignes d’entrainements pour 50 lignes de tests on obtiendrait un modele y ~ X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35 qui donne de meilleurs resultats : 16,9% d’erreur.

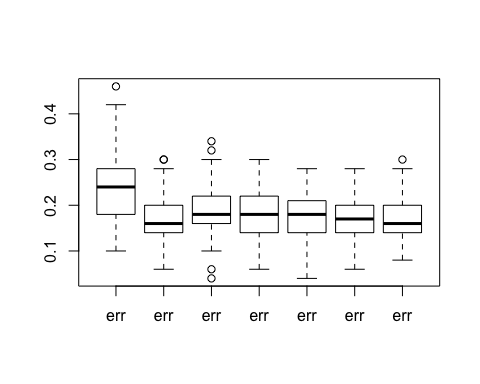
##### K-folds cross-validation

C’est la methode la plus utilise pour tester un modele. On divise de maniere aleatoire le jeu de donnee en k partie egales. On entraine le modele sur k-1 partie et on laisse la derniere pour le test et ce, pour toutes les parties. (K=5 ou K=10 est un bon compromis pour la balance biais variance.)

###### K=10

library(MASS)  
set.seed(5)  
err = rep(0,20)  
err\_mat = c()  
K=10   
  
for (f in (1:7)) {  
 for (l in (1:20)){  
 folds=sample(1:K,nrow(data),replace = TRUE)  
 CV <- rep(0,10)  
 for (k in (1:K)){  
 lda.cv <- lda(Formula[[f]], data = data[folds!=k,])  
 pred.cv <- predict(lda.cv, newdata = data[folds==k,])  
 confusion.cv <- table(data[folds==k,]$y, pred.cv$class)  
 CV[k] <- 1-sum(diag(confusion.cv))/nrow(data[folds==k,])  
 }  
 err[l] <- mean(CV)  
 }  
err\_mat <- cbind(err\_mat, err)  
print(Formula[[f]])  
print(mean(err))  
}

boxplot(err\_mat)

 On obtient donc une classification des modeles par cross-validation. Dans ce cas, pour K=10 on obtient un taux d’erreur moyen (pour 20 essais) minimal de 15,84613 % en selectionnant les donnees : X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35

###### K=5

Dans ce cas, pour K=5 on obtient un taux d’erreur moyen (pour 20 essais) minimal de 15,92783 % en selectionnant les donnees : X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35

#### Conclusion LDA

On obtient un taux d’erreur minimal de 15,84613 % en considerant les sous selections definies au prealable.

Apres avoir etudie les differentes methodes selection de modele nous nous focaliserons desormais sur de la cross-validation pour un K=10 dans les autres modeles.

### QDA

Lorsque les classes n’ont pas la meme matrice de covariance on utlise d’autres methodes, comme QDA. QDA est principalement utilise quand on a un grand nombre de parametre et que le nombre d’observation est grand. Ici nous procederons une nouvelle fois a la comparaison des modeles contenu dans Formula afin de les comparer avec QDA.

On constate que le modele y ~ X9 + X10 + X21 + X28 + X29 + X32 + X33 + X35 donne les meilleurs resultats par cross-validation avec un taux d’erreur de 14,81404%. Cependant d’apres les intervalles de confiances y ~ X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35 qui donne un taux d’erreur de 15,47252% est plus precis.

### Naive bayes classifier

La classification naive de bayes s’utilise principalement quand la matrice de covariance est diagonale or la matrice de covariance a des valeurs plus importante sur la diagonale. On pourrait donc supposer qu’il s’agit d’une bonne prediction.

library(e1071)  
data$y <- factor(data$y)  
set.seed(5)  
err = rep(0,20)  
err\_mat = c()  
K=10   
  
for (f in (1:7)) {  
 for (l in (1:20)){  
 folds=sample(1:K,nrow(data),replace = TRUE)  
 CV <- rep(0,10)  
 for (k in (1:K)){  
 naiveBaye <- naiveBayes(Formula[[f]], data = data[folds!=k,])  
 naiveBaye.pred <- predict(naiveBaye, newdata = data[folds==k,])  
 naiveBaye.confusion <- table(data[folds==k,]$y, naiveBaye.pred)  
 CV[k] <- 1-sum(diag(naiveBaye.confusion))/nrow(data[folds==k,])  
 }  
 err[l] <- mean(CV)  
 }  
err\_mat <- cbind(err\_mat, err)  
print(Formula[[f]])  
print(mean(err))  
}  
boxplot(err\_mat)  
save(naiveBaye, file = "env.Rdata")  
data$y = as.numeric(data$y)

### Conclusion NaiveBayes

Le classifieur de Baye nous donne, comme espere, un taux d’erreur interressant sur nos modeles. On obtient pour le modele **y ~ X9 + X10 + X11 + X13 + X14 + X15 + X17 + X19 + X21 + X23 + X24 + X25 + X28 + X29 + X31 + X32 + X33 + X34 + X35 un taux d’erreur de 10,3%.** Cependant le modele **y ~ X9 + X10 + X11 + X13 + X14 + X15 + X17 + X19 + X21 + X23 + X28 + X29 + X31 + X32 + X33 + X34 + X35 donne un taux d’erreur de 10,99917%** avec 2 predicteurs en moins, on gagne en interpretabilite.

### Logistic Regression

library("nnet")  
library("MASS")  
set.seed(5)  
err = rep(0,20)  
err\_mat = c()  
K=10   
  
for (f in (1:7)) {  
 for (l in (1:20)){  
 folds=sample(1:K,nrow(data),replace = TRUE)  
 CV <- rep(0,10)  
 for (k in (1:K)){  
 glm.cv <- multinom(Formula[[f]], data = data[folds!=k,], trace = FALSE)  
 glm.pred <- predict(glm.cv, newdata = data[folds==k,])  
 confusion.cv <- table(data[folds==k,]$y, glm.pred)  
 CV[k] <- 1-sum(diag(confusion.cv))/nrow(data[folds==k,])  
 }  
 err[l] <- mean(CV)  
 }  
err\_mat <- cbind(err\_mat, err)  
print(Formula[[f]])  
print(mean(err))  
}  
boxplot(err\_mat)

### Conclusion Logistic Regression

On obtient pour le modele y ~ X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35 un taux d’erreur de 15,29652% par le biais de la regression logistique

### Ridge regression

On sait que Ridge est plus adapte pour des problemes de regression, cependant a des fin pedagogique nous avons egalement essaye.

### Lasso regression

De meme pour lasso qui ne convient pas a notre probleme. (voir le code si besoin).

### Conclusion Ridge et Lasso

On constate en effet par le taux d’erreur respectivement 25,37313% et 52,3881% que ces methodes ne sont pas performante pour la classification.

## Conclusion Classification

L’etude des donnees nous a conduit a diverse subdivision du modele. Une fois teste, nous constatons que le modele le plus utile est celui provenant du classifieur naif de Bayes avec un taux d’erreur le plus faible sur les 20 predicteurs : X9 + X10 + X11 + X13 + X14 + X15 + X17 + X19 + X21 + X23 + X24 + X25 + X28 + X29 + X31 + X32 + X33 + X34 + X35

## Linear regression

On constate ainsi que nous avons 200 observations avec 50 predicteurs. Summary() nous donne une indication sur les valeurs significatives.

Nous pouvons donc choisir une première sélection de prédicteurs qui serait : X1+X2+X3+X10+X14+X16+X18+X19+X24+X31+X32+X34+X35+X37+X38+X39+X40+X41+X43+X46+X49

Comparons avec la selection que nous donne regsubset. Nous avons 50 predicteurs, nous ne pouvons donc pas utiliser la methode exhaustive (temps de calcul trop long). Par contre, nous pouvons comparer les methodes backward et forward selection.

D’apres les valeurs de sous selections on obtient le nombre optimal de predicteur d’apres BIC et r^2 ajustee.

* R^2 ajuste (33 predicteurs):X1+X2+X3+X9+X10+X11+X14+X15+X16+X17+X18+X19+X21+X24+X26+X27+X28+X31+X32+ X34+X35+X37+X38+X39+X40+X41+X42+X43+X44+X45+X46+X47+X49
* BIC (20 predicteurs) : X1+X2+X3+X10+X14+X16+X18+X19+X24+X32+X34+X35+X37+X38+X39+X40+X41+X43+X46+X49

Formula <- c(  
 y~.,  
 y~X1+X2+X3+X10+X14+X16+X18+X19+X24+X31+X32+X34+X35+X37+X38+X39+X40+X41+X43+X46+X49,  
 y~X1+X2+X3+X10+X14+X16+X18+X19+X24+X32+X34+X35+X37+X38+X39+X40+X41+X43+X46+X49,  
 y~X1+X2+X3+X9+X10+X11+X14+X15+X16+X17+X18+X19+X21+X24+X26+X27+X28+X31+X32+X34+X35+X37+X38+X39+X40+X41+X42+X43+X44+X45+X46+X47+X49  
)

Ici, nous allons effectuer une régression linéaire sans cross-validation, afin d’avoir un premier aperçu. De plus, cela nous permet de vérifier que les bruits sont bien décoréllés.

## Moindres carrés

Cela nous permet, dans le cas d’une simple regression lineaire d’etudier la repartition des residus et d’en conclure que, n’ayant pas de structure particuliere, ils sont bien decoreles.

Ici, nous effectuons à nouveau une régression linéaire, cette fois avec cross-validation.

## Moindres carrés avec cross validation

K<-10  
reps <- 30  
MSE<-rep(0,reps)  
  
for( f in (1:4)){  
 for (i in (1:reps)) {  
 folds=sample(1:K,nrow(data2),replace=TRUE)  
 CV<-rep(0,K)  
 for(k in (1:K)){  
 reg<-lm(Formula[[f]],data=data2[folds!=k,])  
 pred<-predict(reg,newdata=data2[folds==k,])  
 CV[k]<-sum((data2[folds==k,'y']-pred)^2)/nrow(data2[folds==k,])  
 }  
 MSE[i] <- mean(CV)  
}  
print(MSE.lm <- mean(MSE))  
boxplot(summary(MSE))  
}

On obtient donc respectivement pour les selections enoncees precedemment : [1] 1628.413 [1] 1264.353 [1] 1292.118 [1] 1310.301 L’etude des intervalles de confiance nous permets de dire que pour la regression lineaire, le modele qui semble minimiser l’erreur quadratique moyenne est **X1+X2+X3+X10+X14+X16+X18+X19+X24+X31+X32+ X34+X35+X37+X38+X39+X40+X41+X43+X46+X49 avec MSE = 1264.353**

## K plus proches voisins

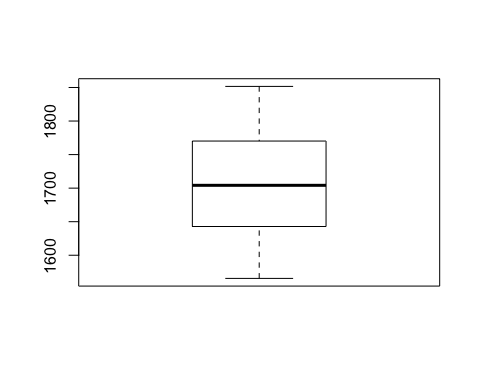
La cross validation est automatique avec la fonction knn.reg, si l’on ne fournit pas de données de test.

Avec un MSE = 12382.14, les k plus proches voisins ne sont pas un modele fiable dans ces conditions.

## Ridge regression, avec cross-validation

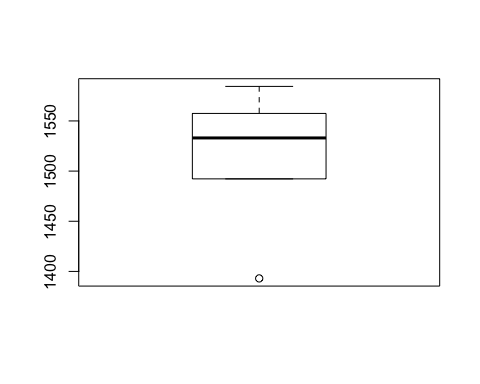
On effectue alors la regression de ridge.

boxplot(summary(MSE))

 Ainsi avec le regression de ridge, on obtient un MSE = 1705.302 ainsi que l’intervalle de confiance du boxplot. Cette solution est moins performante que la regression lineaire.

## Lasso, avec cross validation

boxplot(summary(MSE))

 Ainsi avec le regression de Lasso, on obtient un MSE = 1508.071 avec compris dans l’intervalle de confiance du boxplot.

## Conclusion Regression

Lasso performe mieux que Ridge mais la regression lineaire applique sur X1+X2+X3+X10+X14+X16+X18+X19+X24+X31+X32+ X34+X35+X37+X38+X39+X40+X41+X43+X46+X49 donne le plus faible MSE = 1264.353. Nous choisissons donc ce modele pour la prediction de ces donnees.