SY19 - TP3

Régression et classification - Sélection de modèle

Theodore BOURGEON - Cyril LAY

I. Problème de classification

Modèle

a) Récupération des données

```
data <-read.table("tp3_a18_clas_app.txt")
```

b) Observation des données

```
head(data)
dim(data)
```

Cette rapide observation nous apprend que la colonne de validation est la dernière nommée “y”, qu’il y a 200 observations et 36 prédicteurs.

c) Graphique des données

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

Ces lignes nous permettent d’observer une éventuelle corrélation entre les prédicteurs, ce qui n’est pas le cas ici.

d) Sélection de prédicteurs

A ce stade, nous recherchons les prédicteurs ayant le plus d’influence sur la colonne de validation, afin de bien choisir notre modèle, nous effectuons une régression linéaire en prenant en compte l’ensemble des prédicteurs.

```
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 très petite, on rejette l’hypothèse selon laquelle les colonnes ne sont pas corrélées avec y.

Ensuite, nous pouvons procéder à différentes sous sélections.

- La première méthode est **“la meilleure sous sélection”** (qui est parfois impossible à déterminer de cette manière, à cause d’un nombre de prédicteurs trop grand qui peut faire exploser le nombre de combinaisons). “**Exhaustive**” permet de tester toutes les combinaisons possibles des prédicteurs parmi nvmax prédicteurs et retourne les meilleurs résultats.

```
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* semblent être une solution pertinente.

Selon les valeurs BIC et R^2 des sous sélections, nous obtenons le nombre optimal de prédicteurs :

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

- Une seconde méthode est **"forward stepwise selection"**

Elle peut être utilisée même avec un grand nombre de prédicteurs. "Forward" commence avec une sélection de prédicteurs vide et ajoute pas à pas celui qui a le plus d'impact, parmi nvmax prédicteurs. On constate que *x9+x10+x11+x13+x14+x15+x17+x19+x21+x23+x24+x25+x28+x29+x31+x32+x33+x34+x35* semble être une solution pertinente.

- La dernière est **"backward stepwise selection"**.

"Backward" commence avec l'ensemble des prédicteurs et retire pas à pas celui qui a le moins d'impact, parmi nvmax prédicteurs. On constate que *x9+x10+x11+x13+x14+x15+x17+x19+x21+x23+x28+x29+x31 +x32+x33+x34+x35* semble être une solution pertinente.

e) Bilan

Nous avons donc à ce stade, une série de sous sélection de prédicteurs. Nous allons pouvoir tester ces différentes sélections selon nos différents modèles de classification et régression.

```
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)
```

Sélection

a) Estimation indirecte de l'erreur de test

Dans un premier temps nous appliquons une régression linéaire avec un ajustement de l'erreur d'entrainement (on estime donc indirectement l'erreur test). Dans ce cas, nous n'avons pas besoin de séparer notre jeu de donnée en un set d'apprentissage et de test. On applique le modèle sur les différentes sélections de prédicteurs et on calcule différents indicateurs : R^2, AIC, BIC.

```
library(MASS)
for (i in (1:7)){
 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 sera peu important car nous avons seulement 2 classes. Il semble dans ce cas que les meilleures sous sélection seraient :

- *X9 + X10 + X11 + X13 + X14 + X15 + X17 + X19 + X21 + X23 + X25 + X28 + X29 + X31 + X32 + X33 + X34 + X35*, si l'on met en priorité le R^2 ajusté.
- *X9 + X10 + X21 + X28 + X29 + X32 + X33 + X35*, si l'on met en priorité le critère BIC.

b) Estimation directe de l'erreur de test

LDA

LDA s'utilise principalement quand les classes ont une matrice de covariance semblable.

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

On constate que les valeurs des matrices de covariance ne sont pas vraiment semblables, LDA ne donnera probablement pas les meilleures performances.

1) Approche « Validation-set »

Pour cette approche, nous effectuons 100 fois une LDA avec des répartitions apprentissage/test aléatoires mais respectant les proportions 2/3, 1/3. A chaque itération, nous testons les différentes sélections de prédicteurs. La fonction boxplot nous permettra d'observer la variance de l'erreur.

On constate ici que le taux d'erreur est minimal pour le modèle possédant uniquement les colonnes X9 + X10 + X21 + X28 + X29 + X32 + X33 + X35 : **17,16%.**

Nous avons répété ces opérations pour différentes répartitions, et avons obtenu de meilleurs résultats avec ¾ des observations en apprentissage, et ¼ en test, où le modèle contenant les prédicteurs X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35 donne en moyenne **16,9%** d'erreur.

2) K-folds cross-validation

C'est une des méthodes les plus utilisées pour tester efficacement un modèle. On divise de manière aléatoire le dataset en K parties égales. On entraine le modèle sur K-1 parties et on laisse la dernière pour le test, puis on réitère l'opération en changeant la partie de test. K=5 ou K=10 est un bon compromis pour la balance biais variance.

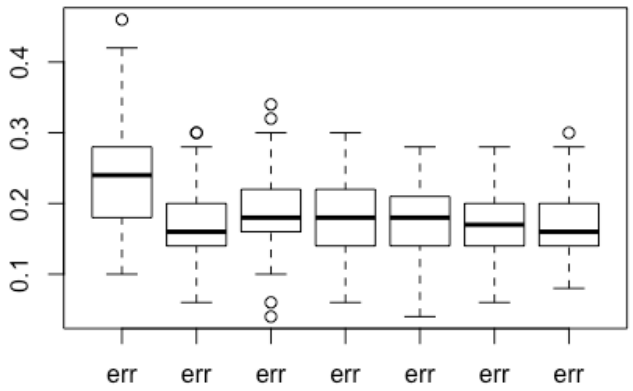
K=10

```
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 modèles par cross-validation. Avec K=10 nous obtenons un taux d'erreur moyen (pour 100 itérations) de 15,85 % avec la sélection de prédicteurs : X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35

Avec K=5 nous obtenons un taux d'erreur moyen (pour 100 itérations) de **15,93%** avec la même sélection de prédicteurs.

3) Conclusion LDA

Le taux d'erreur minimal obtenu est de **15,85%** en considérant toutes les sous sélections définies précédemment.

Pour la suite, nous allons privilégier la K-folds cross-validation avec K=10. Les autres méthodes de validation ont permis de mettre en application ce qui a pu être vu en cours.

QDA

Lorsque les classes n'ont pas la même matrice de covariance, nous pouvons utiliser QDA. QDA est principalement utilisé quand on a un grand nombre de paramètres et que le nombre d'observation est grand. Ici nous procèderons une nouvelle fois à la comparaison des sélections de prédicteurs afin de les comparer selon la méthode QDA.

On constate que le modèle contenant les prédicteurs X9 + X10 + X21 + X28 + X29 + X32 + X33 + X35 donne les meilleurs résultats par cross-validation avec **un taux d'erreur de 14,81%**. Cependant nous préférons la sélection X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35 qui est légèrement moins performante (**15,47%**), mais a une variance bien plus faible.

Naive bayes classifier

La classification naïve de Bayes s'utilise principalement quand la matrice de covariance est diagonale, or notre matrice de covariance a des coefficients plus importants sur la diagonale, ce qui est encourageant.

```
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 naive Bayes

Le classifieur de Bayes nous donne un taux d'erreur intéressant sur nos modèles. On obtient pour le modèle avec la sélection *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 modèle avec *X9 + X10 + X11 + X13 + X14 + X15 + X17 + X19 + X21 + X23 + X28 + X29 + X31 + X32 + X33 + X34 + X35* donne un taux d'erreur de **11,00%** mais possède 2 prédicteurs de moins, les résultats sont donc plus facilement interprétables.

Régression logistique

```
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 régression logistique

On obtient pour le modèle contenant la sélection de prédicteurs *X9 + X10 + X11 + X21 + X28 + X31 + X32 + X33 + X35* un taux d'erreur de **15,29652%** avec une régression logistique.

Régressions Ridge et Lasso

Même si les méthodes Ridge et Lasso sont plus adaptés à un problème de régression, nous les avons essayé à des fins pédagogiques, et avons obtenus respectivement **25,37%** et **52,39%** de taux d'erreur pour une set validation approach (2/3 entrainement, 1/3 test), ce qui confirme leur manque de performance pour la classification.

Conclusion Classification

En rassemblant l'ensemble des tests sur les différentes sélection de prédicteurs, selon les différentes méthodes de classification, le modèle le plus performant avec nos données est celui du classifieur naif de Bayes avec la sélection de 19 prédicteurs : X9 + X10 + X11 + X13 + X14 + X15 + X17 + X19 + X21 + X23 + X24 + X25 + X28 + X29 + X31 + X32 + X33 + X34 + X35. Le taux d'erreur obtenu est de 10,3%.

II. Problème de Régression

a) Régression linéaire

De la même façon que pour la classification, nous pouvons déterminer différentes sélections de prédicteurs en étudiant les p_values des prédicteurs, après standardisation. Une première sélection 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 sélection que nous donne regsubset. Nous avons 50 prédicteurs, nous ne pouvons donc pas utiliser la méthode exhaustive (temps de calcul trop long). Par contre, nous pouvons comparer les méthodes backward et forward sélection.

Selon les valeurs BIC et R^2 des sous sélections, nous obtenons le nombre optimal de prédicteurs.

- R^2 (33 prédicteurs) : *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 prédicteurs) : *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+X
40+X41+X42+X43+X44+X45+X46+X47+X49)
```

Dans un premier temps, nous effectuons une régression linéaire sans cross-validation, afin d'avoir un premier aperçu. L'absence de structure particulière dans la répartition des résidus nous permet de vérifier que les bruits sont bien décorellés.

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

b) 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))
}
```

Le modele qui possède l'erreur quadratique moyenne minimale est **X1+X2+X3+X10+X14+X16+X18+X19+X24+X31+X32+ X34+X35+X37+X38+X39+X40+X41+X43+X46+X49** avec une erreur de **1264.**

c) K plus proches voisins

```
data2[, -c(ncol(data2))] <- scale(data2[, -c(ncol(data2))])
MSE <- rep(1:20)
for (i in 1:20) {
  CV<-rep(0,15)
  for(k in 1:15){
    reg <- knn.reg(train = data2, y = data2['y'], k = k)
    CV[k] <- mean((data2$y - reg$pred)^2)
  }
    MSE[i] <- mean(CV)
}
```

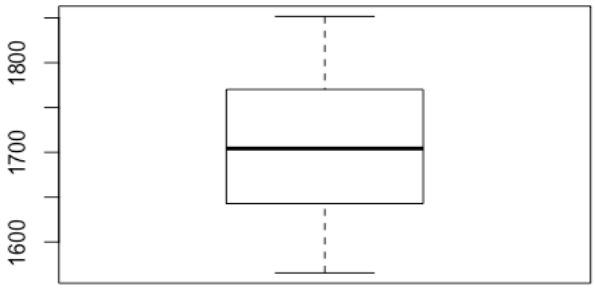
Il est intéressant de noter que la cross validation est automatique avec la fonction knn.reg, si l'on ne fournit pas de données de test.

Avec une erreur MSE = 12382, la méthode des k plus proches voisins n'est pas performante dans notre cas, même après avoir normalisé nos données.

d) Ridge régression avec cross-validation

```
K<-10
MSE <- rep(0,15)
set.seed(5)
for (i in (1:15)) {
  folds=sample(1:K,nrow(data2),replace=TRUE)
  CV<-rep(0,K)
  for(k in (1:K)){
    cv.out <- cv.glmnet(as.matrix(data2[folds!=k,1:50]), data2[folds!=k,]$y, alpha = 0)
    fit <- glmnet(as.matrix(data2[folds!=k,1:50]), data2[folds!=k,]$y, lambda = cv.out$la
```

```
mbda.min, alpha = 0)
    ridge.pred<-predict(fit, s=cv.out$lambda.min, newx = as.matrix(data2[folds==k,1:50]))
    CV[k] <- mean((data2[folds==k,'y']-ridge.pred)^2)
  }
  MSE[i] <- mean(CV)
}
MSE.rr <- mean(MSE)
boxplot(summary(MSE))
```

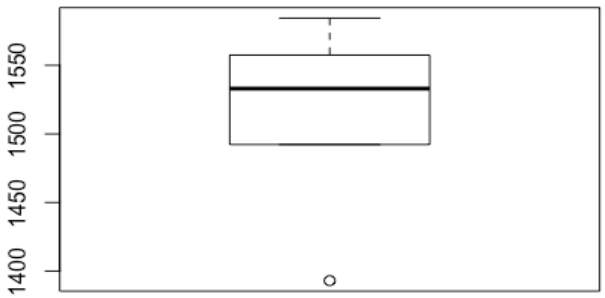


Avec cette méthode, on obtient un MSE = 1705.302 et un intervalle de confiance graphique. Cette solution est moins performante que la régression linéaire.

e) Lasso avec cross validation

```
…
    cv.out <- cv.glmnet(as.matrix(data2[folds!=k,1:50]), data2[folds!=k,]$y, alpha = 1)
    fit.lasso <- glmnet(as.matrix(data2[folds!=k,1:50]), data2[folds!=k,]$y, lambda = cv.
out$lambda.min, alpha = 1)
    lasso.pred<-predict(fit.lasso, s=cv.out$lambda.min, newx = as.matrix(data2[folds==k,1
:50]))
…
boxplot(summary(MSE))
```

Ainsi avec la régression Lasso, on obtient un MSE = 1508.071 avec compris dans l'intervalle de confiance du boxplot. C'est une meilleure performance que la régression ridge.



Conclusion sur le problème de régression

Cependant, une simple régression linéaire avec la sélection X1+X2+X3+X10+X14+X16+X18+X19+X24+X31+X32+X34+X35+X37+X38+X39+X40+X41+X43+X46+X49 donne l'erreur quadratique moyenne la plus faible, MSE = 1264.353. Nous choisissons donc ce modèle pour la prédiction des nouvelles données.