Master 2 MAS
Classification automatique

TP9 : Forêts aléatoires

Exercice 1. Récupérer le jeu de données d'apprentissage habituel synth_train.txt. On a $Y \in \{1, 2\}$ et $X \in \mathbb{R}^2$. On dispose de 100 données d'apprentissage.

1. Charger le jeu de données d'apprentissage dans R.

```
data <- read.table(file="../data/synth_train.txt", header=TRUE)
dim(data)
```

```
## [1] 100   3
```

2. Charger le package `randomForest`. Consulter l'aide de la fonction `randomForest` notez bien tous les paramètres par défaut (pour la classification) afin de savoir exactement quel algorithme est appliqué.

```
library(randomForest)
help(randomForest)
# http://gradientdescending.com/unsupervised-random-forest-example/

# y = variable de sortie.
# si y de class factor => classification
# si y de class numeric => régression

# mtry = nombre de variables tirées aléatoirement à chaque division.
# mtry = sqrt(p) par défaut.
# Paramètre le plus important à tuner !

# ntree = nombre d'arbres de la forêt.
# ntree = 500 par défaut. A regarder (cf. suite exercice)
# mais souvent "moins important".

# Paramètres des arbres de la forêt :
# - nodesize = 1 (si classification)
# - maxnodes : rien par défaut
```

3. Construire un forêt aléatoire (pour la classification) avec ces données d'apprentissage en gardant les paramètres par défaut. Vérifiez que vous comprenez les résultats du `print`. Vérifiez ensuite que vous comprenez les sorties suivantes : `predicted`, `confusion`.

```
data$y <- as.factor(data$y) # sinon régression !
rf <- randomForest(x=data[,-1], y=data$y)
rf # idem print(rf)
```

```
##
## Call:
##  randomForest(x = data[, -1], y = data$y)
##                Type of random forest: classification
```

```
##                       Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 6%
## Confusion matrix:
##    1  2 class.error
## 1 19  3  0.13636364
## 2  3 75  0.03846154
```

```
rf$predicted[1:10]
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  2  2  2  1  2  2  1  2  2
## Levels: 1 2
```

```
# predicted : prediction OOB des données d'apprentissage

rf$confusion # sur les prediction OOB.
```

```
##    1  2 class.error
## 1 19  3  0.13636364
## 2  3 75  0.03846154
```

```
table(data[,1],rf$predicted)
```

```
##
##      1  2
##   1 19  3
##   2  3 75
```

```
# colonne class.error : 1-TVP et 1-TVN
```

4. Pour avoir l'importance des variables, ajouter l'argument `importance=TRUE` et vérifiez que vous comprenez les sorties `importance` et `importanceSD`.

```
rf <- randomForest(x=data[,-1], y=data$y,importance=TRUE)
round(rf$importance, digit=3)
```

```
##        1     2 MeanDecreaseAccuracy MeanDecreaseGini
## x1 0.392 0.116                0.174           21.049
## x2 0.116 0.040                0.054           12.915
```

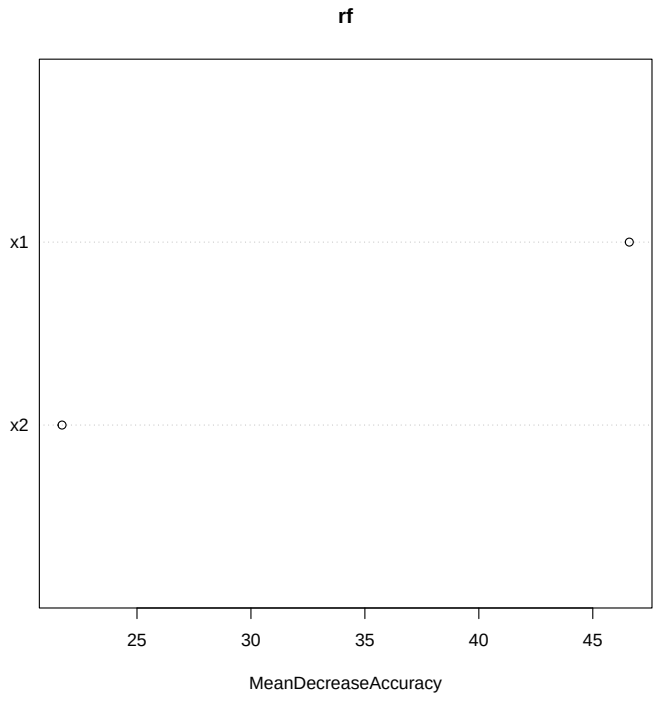
```
# Trois permières colonnes : diminution (resp. augmentation)
# moyenne sur les ntree=500 arbres, du taux de bien classés
# (resp. du taux d'erreur) OOB résultant des permutations.
# Les deux premières colonnes donnent la diminution moyenne
# dans les classes et la troisième (MeanDeacreaseAccuracy) donne
# la diminution moyenne globale. C'est une mesure d'importance
# des variables x1 et x2. Ici, x1 est plus importante (discriminante)
# que x2.

rf$importanceSD
```

```
##              1           2 MeanDecreaseAccuracy
## x1 0.008612243 0.003021670          0.003727684
```

```
## x2 0.007257847 0.002143571          0.002508713

# Ecart-type des 500 diminutions du taux de bien classés OOB
```

5. Testez les fonctions `importance` et `varImpPlot`.

```
?importance
round(importance(rf, scale=FALSE),digit=3) # on retrouve la sortie précédence

##        1     2 MeanDecreaseAccuracy MeanDecreaseGini
## x1 0.392 0.116                0.174           21.049
## x2 0.116 0.040                0.054           12.915

importance(rf, scale=FALSE, type=1)

##    MeanDecreaseAccuracy
## x1           0.17371860
## x2           0.05446875

round(importance(rf, , type=1),digit=3)

##    MeanDecreaseAccuracy
## x1               46.602
## x2               21.712

# scale=TRUE par defaut
# Par défaut, la fonction affiche les mesures d'importance "normalisées"
# (divisées par leur écart-type)

?varImpPlot
varImpPlot(rf, type=1)
```



```
# par defaut, scale=TRUE dans varImpPlot
```

6. Regardez ensuite les éléments `votes`, `oob.times` et leur lien avec l'argument `norm.votes`.

```
#votes
dim(rf$votes)
```

```
## [1] 100   2
```

```
#ligne i : une des 100 observations

head(rf$votes)
```

```
##           1          2
## 1 0.9578947 0.04210526
## 2 0.1139896 0.88601036
## 3 0.1348315 0.86516854
## 4 0.0000000 1.00000000
## 5 0.8315217 0.16847826
## 6 0.1538462 0.84615385
```

```
# ligne i : proportion d'arbres (pour lesquels i est OOB) qui
# classent i dans les classes 1 et 2 => estimation OOB des
# proba à posteriori

head(rf$oob.times)
```

```
## [1] 190 193 178 207 184 169
```

```
#ligne i : nombre d'arbres OOB de i.

head(sweep(rf$votes,1,rf$oob.times,'*'))
```

```
##     1   2
## 1 182   8
## 2  22 171
## 3  24 154
## 4   0 207
## 5 153  31
## 6  26 143
```

```
# ligne i : nombre d'abres  (pour lesquels i est OOB)
# qui classent i dans les classent 1 et 2.

# Pour avoir directement ces effectifs : norm.votes=FLASE
rf2 <- randomForest(x=data[,-1], y=data$y,importance=TRUE,
                    norm.votes=FALSE)
head(rf2$votes)
```

```
##     1   2
## 1 177   5
## 2  14 148
## 3  23 162
## 4   0 178
## 5 157  27
## 6  18 169
```

```r
# somme des deux colonnes : nombre d'arbres OOB de i.

head(rf2$oob.times)
```

```
## [1] 182 162 185 178 184 187
```

7. Vérifiez ensuite que vous comprenez les résultats de la sortie **err.rate** de la forêt. Où se trouve dans cette sortie le taux d'erreur OOB global et par classe de la fôret ?

```r
dim(rf$err.rate)
```

```
## [1] 500   3
```

```r
# 500 lignes (ntree=500 arbres)

head(rf$err.rate)
```

```
##             OOB         1          2
## [1,] 0.08571429 0.1666667 0.06896552
## [2,] 0.09677419 0.3000000 0.05769231
## [3,] 0.09210526 0.1538462 0.07936508
## [4,] 0.07058824 0.1764706 0.04411765
## [5,] 0.12222222 0.2500000 0.08571429
## [6,] 0.09574468 0.2000000 0.06756757
```

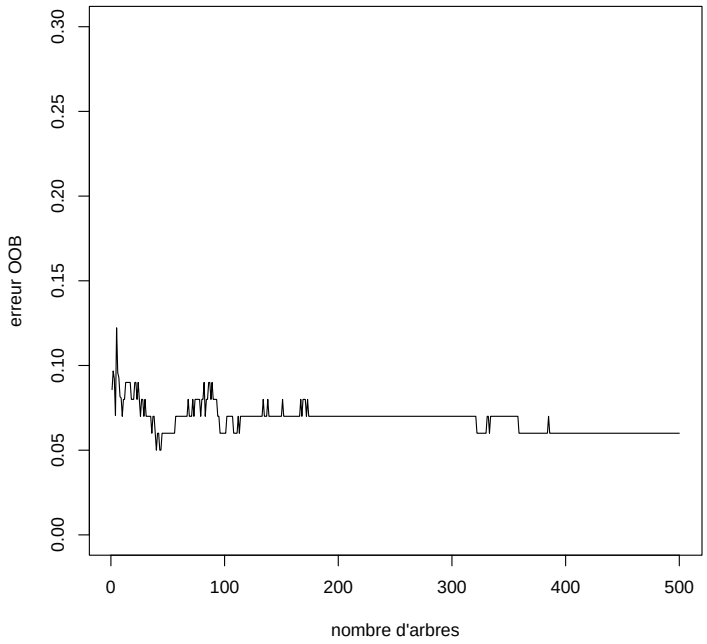
```r
# ligne i : taux derreur (dans les classes et global) trouvé
# avec les i premiers arbres !

rf$err.rate[500,]
```

```
##        OOB          1          2
## 0.06000000 0.13636364 0.03846154
```

```r
# les taux d'erreur OOB de la fôret sont donc dans la dernière ligne !
```

8. Utiliser **err.rate** pour représenter graphiquement le taux d'erreur OOB en fonction du paramètre **ntree** (nombre d'arbres de la forêt). Que pouvez-vous en déduire concernant le choix de ce paramètre ?

```r
plot(rf$err.rate[,"OOB"], type="l", ylim=c(0, 0.3),
     xlab="nombre d'arbres", ylab="erreur OOB")
```




```
# ntree=500 arbres semble être suffisant.
```

9. Calculer le taux d'erreur d'apprentissage.

```
pred <- predict(rf, newdata=data, type="class")
sum(pred!=data$y)/length(pred)
```

```
## [1] 0
```

10. Charger le jeu de données test synth_test.txt puis calculer le taux d'erreur test de la forêt paramétrée par défaut.

```
data.test <- read.table(file="../data/synth_test.txt", header=TRUE)
dim(data.test)
```

```
## [1] 200   3
```

```
pred.test <- predict(rf, newdata=data.test, type="class")
sum(pred.test!=data.test$y)/length(pred.test)
```

```
## [1] 0.05
```

11. Modifiez le paramètrage de la fôret pour que la méthode d'ensemble utilisée soit le `bagging` avec des arbres CART (de longueur max). Calculer alors le taux d'erreur des données test.

```
#ntree=2000
#mtry=p donc pas d'aléa et par CART-RI donc bagging.
bag <- randomForest(x=data[,-1], y=data$y, mtry=2, ntree=2000)
bag
```

```
##
## Call:
```

```
##  randomForest(x = data[, -1], y = data$y, ntree = 2000, mtry = 2)
##                Type of random forest: classification
##                      Number of trees: 2000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8%
## Confusion matrix:
##    1  2 class.error
## 1 18  4  0.18181818
## 2  4 74  0.05128205
```

```
pred.test <- predict(bag, newdata=data.test, type="class")
sum(pred.test!=data.test$y)/length(pred.test)
```

```
## [1] 0.06
```

Exercice 2. On reprend les données concernant $n = 1260$ exploitations agricoles. Les variables explicatives sont $p = 30$ critères économiques et financiers et la variable qualitative à expliquer est la variable difficulté de paiement (0=saine, 1=défaillant).

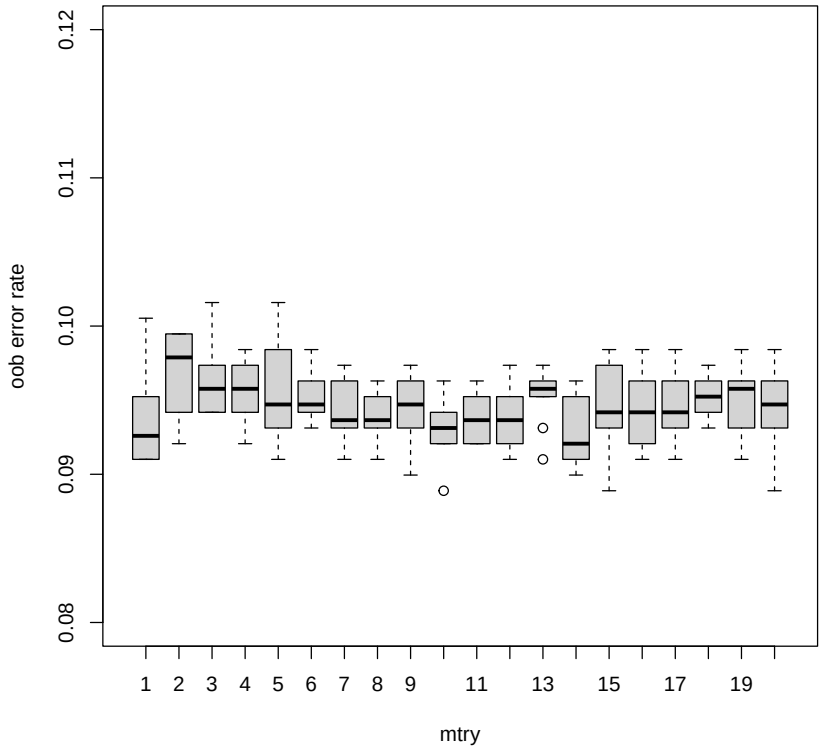
1. Charger le jeu de données Desbois_complet.rda dans R.

```
load("../data/Desbois_complet.rda")
table(data$DIFF)
```

```
##
##   0   1
## 653 607
```

```
data$DIFF <- as.factor(data$DIFF)
```

2. Créez un découpage aléatoire des données en 945 observations d'apprentissage et 315 observations test.
3. Quelle est l'erreur OOB de la forêt constuite sur les données d'apprentissage avec les paramètres `mtry` et `ntree` par défaut ? Quelles sont les valeurs de ces paramètres ici ? Proposez trois manières différentes de retrouver ce taux d'erreur OOB.
4. Le nombre d'arbres par défaut vous semble-il suffisant ?
5. Afin d'avoir une première idée du choix du paramètre `mtry` reproduire le graphique ci-dessous.



6. Implémenter une procédure de choix automatique du paramètre `mtry` (utilisant l'erreur OOB).
7. Prédire les données test avec la valeur optimale de `mtry` obtenue à la question précédente et calculer le taux d'erreur test.
8. Comparez ensuite la performance des forêts aléatoires avec celles de la régression logistique pour ces données. Attention de bien inclure le calibrage automatique du paramètre `mtry` dans la procédure de comparaison.