Machine Learning Project : Digits prediction

Guedj O. and Marcoux Pépin T.

1. Import des données

```
data(zip.train)
data(zip.test)
dim(zip.train)
```

```
## [1] 7291  257
```

```
dim(zip.test)
```

```
## [1] 2007  257
```

La première colonne correspond aux labels des données: les digits. Les 256 autres colonnes correspondent aux valeurs des pixels (image 16x16 = 256 pixels).

2. Préparation des data frames

```
train = as.data.frame(zip.train)
colnames(train)[1] = "Digit"
test = as.data.frame(zip.test)
colnames(test)[1] = "Digit"
train$Digit = as.factor(train$Digit)
test$Digit = as.factor(test$Digit)
```

2.1 Choix aléatoire des deux digits à prédire

```
set.seed(123)
digits_chosen  = sample(seq(from=0, to= 9, by=1), 2, replace = F)
cat("Les deux chiffres choisis sont",digits_chosen[1],"et",digits_chosen[2])
```

```
## Les deux chiffres choisis sont 2 et 9
```

On sélectionne les deux digits à prédire aléatoirement afin de ne pas faire d'hypothèse à priori sur la capacité des différents modèles à séparer les deux classes. Par exemple, la distinction entre 1 et 7 est complexe alors que celle entre 1 et 0 est relativement simple.

2.2 Data frame final pour entraîner les modèles

On crée un `train` et un `test` contennant uniquement les observations des digits choisis.

```
train.2d = subset(train, (Digit %in% c(digits_chosen)))
test.2d = subset(test, (Digit %in% c(digits_chosen)))
train.2d$Digit = as.factor(as.character(train.2d$Digit))
test.2d$Digit = as.factor(as.character(test.2d$Digit))
```

On vérifie que les observations des deux digits choisis sont équitablement représentés dans le `train` et dans le `test`.

```
par(mfrow = c(1,2))
barplot(table(train.2d$Digit), col = c("orange", "steelblue"), xlab = "train")
barplot(table(test.2d$Digit), col = c("orange", "steelblue"), xlab = "test")
title(main="Répartition des classes choisies\n",outer=TRUE,line=-2)
```

3. Classifiers

3.1 Naive Bayes

```
mod.NB = naiveBayes(Digit ~ ., data = train.2d)
pred.NB = predict(mod.NB, subset(test.2d, select = -Digit))
cm.NB = confusionMatrix(pred.NB, test.2d$Digit)
cm.NB$table
```

```
##           Reference
## Prediction   2   9
##          2 195   2
##          9   3 175
```

```
acc.NB = cm.NB$overall[1]
acc.NB
```

```
##  Accuracy
## 0.9866667
```

3.2 Linear Discriminant Analysis

```
mod.lda = lda(Digit ~ ., data = train.2d)
pred.lda = predict(mod.lda, subset(test.2d, select = -Digit))
cm.lda = confusionMatrix(pred.lda$class, test.2d$Digit)
cm.lda$table
```

```
##           Reference
## Prediction   2   9
##          2 192   3
##          9   6 174
```

```
acc.LDA = cm.lda$overall[1]
acc.LDA
```

```
## Accuracy
##    0.976
```

3.3 Quadratic Discriminant Analysis

```
#mod.qda = qda(Digit ~ ., data=train.2d)
#pred.qda = predict(mod.qda, subset(test.2d, select = -Digit))
#confusionMatrix(pred.qda$class, test.2d$Digit)
```

3.4 k NN

Cross-validation sur l'hyperparamètre k

```
trControl <- trainControl(method  = "cv",
                          number  = 5)
fit <- train(Digit ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:10),
             trControl  = trControl,
             metric     = "Accuracy",
             data       = train.2d)
fit
```

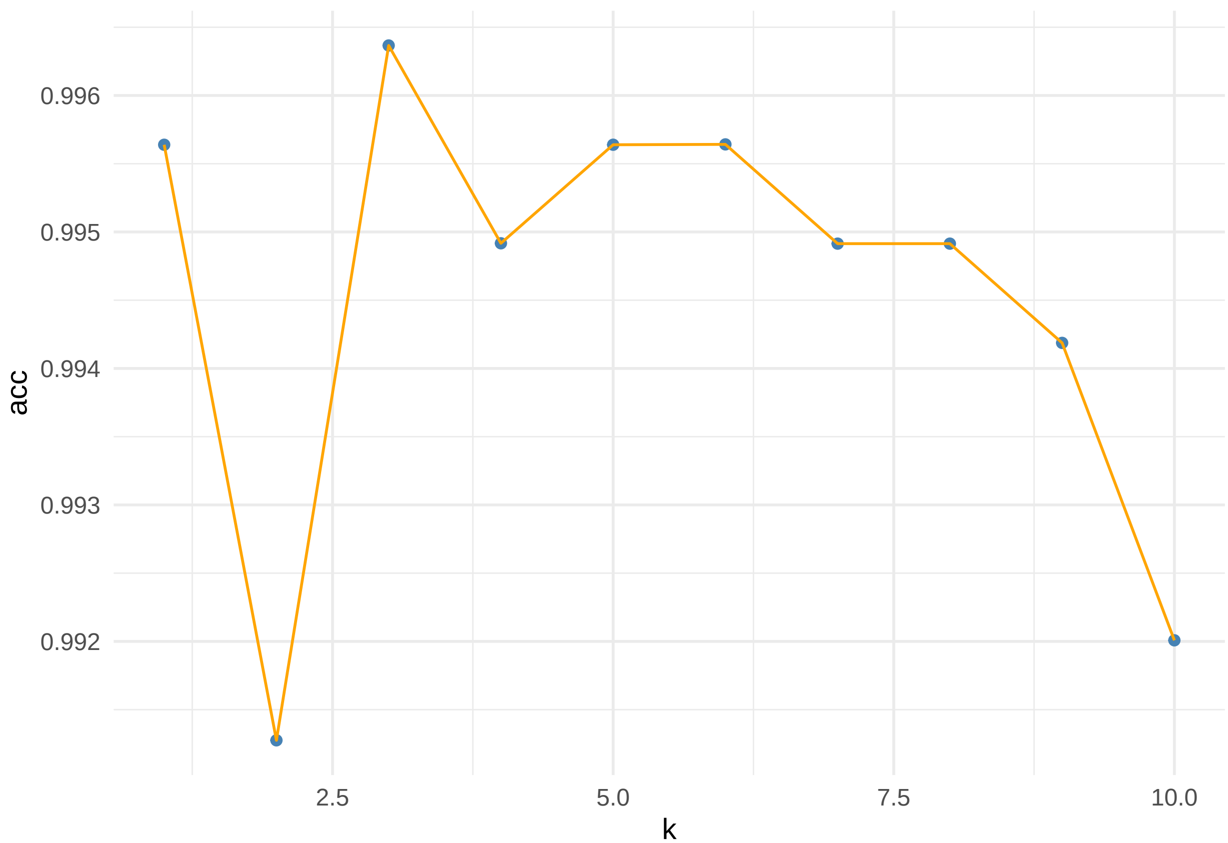
```
## k-Nearest Neighbors
##
## 1375 samples
##  256 predictor
##    2 classes: '2', '9'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 1100, 1101, 1099, 1100, 1100
## Resampling results across tuning parameters:
```

```
##
##    k   Accuracy   Kappa
##     1  0.9956390  0.9912452
##     2  0.9912753  0.9824989
##     3  0.9963663  0.9927087
##     4  0.9949170  0.9898033
##     5  0.9956390  0.9912518
##     6  0.9956416  0.9912553
##     7  0.9949144  0.9897984
##     8  0.9949144  0.9897984
##     9  0.9941871  0.9883403
##    10  0.9920079  0.9839759
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
acc.knn = max(fit$results$Accuracy)
acc.knn
```

```
## [1] 0.9963663
```

```
plot.knn = data.frame(k = fit$results$k, acc = fit$results$Accuracy)
ggplot(plot.knn, aes(x = k, y = acc)) +geom_point(col = "steelblue") + geom_line(col = "orange") +theme_
```



Remarque : A scores de précisions égaux, la fonction train a tendance à retenir le modèle le plus complexe, ce qui n'est pas forcement le choix optimal.

3.5 Decision Tree

```
mod.tree = tree(Digit~. , data = train.2d)
pred.tree = as.factor(predict(mod.tree, newdata=test.2d[-c(1)], type="class"))
cm.tree = confusionMatrix(pred.tree, test.2d$Digit)
cm.tree$table
```

```
##           Reference
## Prediction   2   9
##          2 191   5
##          9   7 172
```

```
acc.tree = cm.tree$overall[1]
acc.tree
```

```
## Accuracy
##    0.968
```

3.6 Bagging

```
mod.bag = bagging(Digit~., data=train.2d, coob=T)
pred.bag = as.factor(predict(mod.bag, newdata=test.2d[,-c(1)], type="class"))
cm.bag = confusionMatrix(pred.bag, test.2d$Digit)
cm.bag$table
```

```
##           Reference
## Prediction   2   9
##          2 197   4
##          9   1 173
```

```
acc.bag = cm.bag$overall[1]
acc.bag
```

```
##  Accuracy
## 0.9866667
```

3.7 Random Forest

3.7.1 Modèle

```
mod.rf = randomForest(train.2d[,-c(1)], train.2d$Digit)
pred.rf = as.factor(predict(mod.rf, newdata=test.2d[,-c(1)], type="class"))
cm.rf = confusionMatrix(pred.rf, test.2d$Digit)
cm.rf$table
```

```
##           Reference
## Prediction   2   9
##          2 197   3
##          9   1 174
```

```
acc.rf = cm.rf$overall[1]
acc.rf
```

```
##  Accuracy
## 0.9893333
```

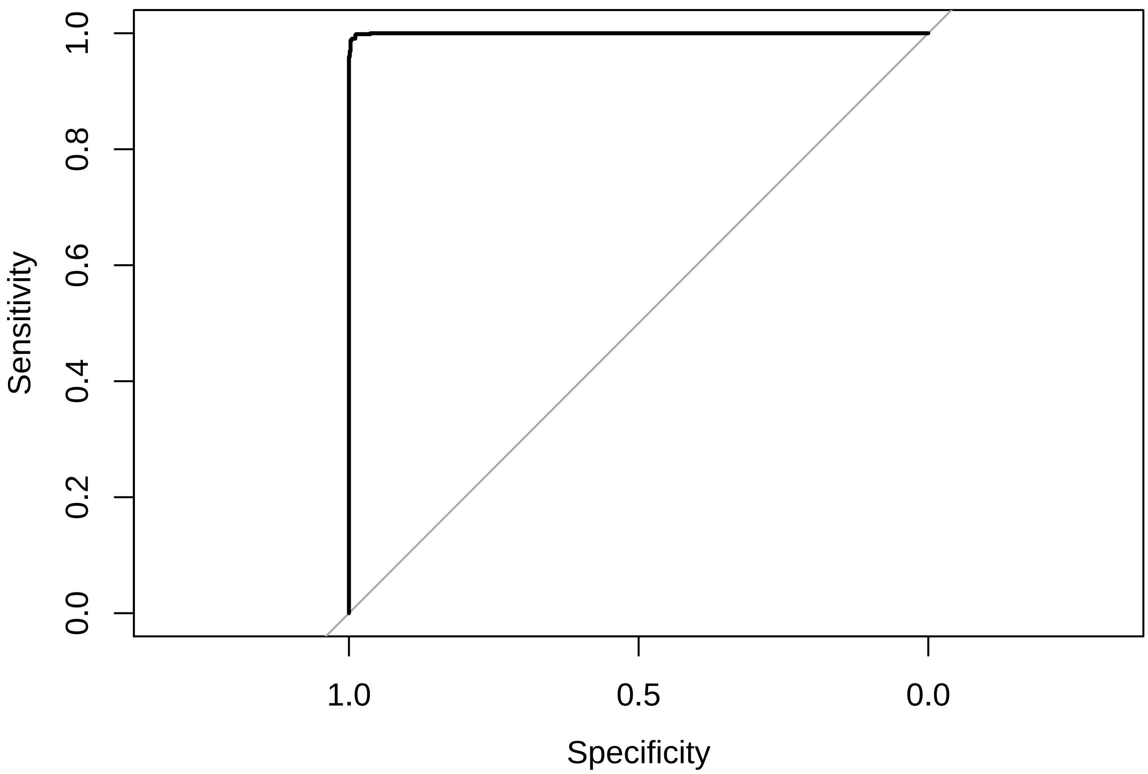
3.7.2 ROC et AUC

```r
roc.rf = roc(train.2d$Digit,mod.rf$votes[,2] )
```

```
## Setting levels: control = 2, case = 9
```

```
## Setting direction: controls < cases
```

```r
plot(roc.rf)
```



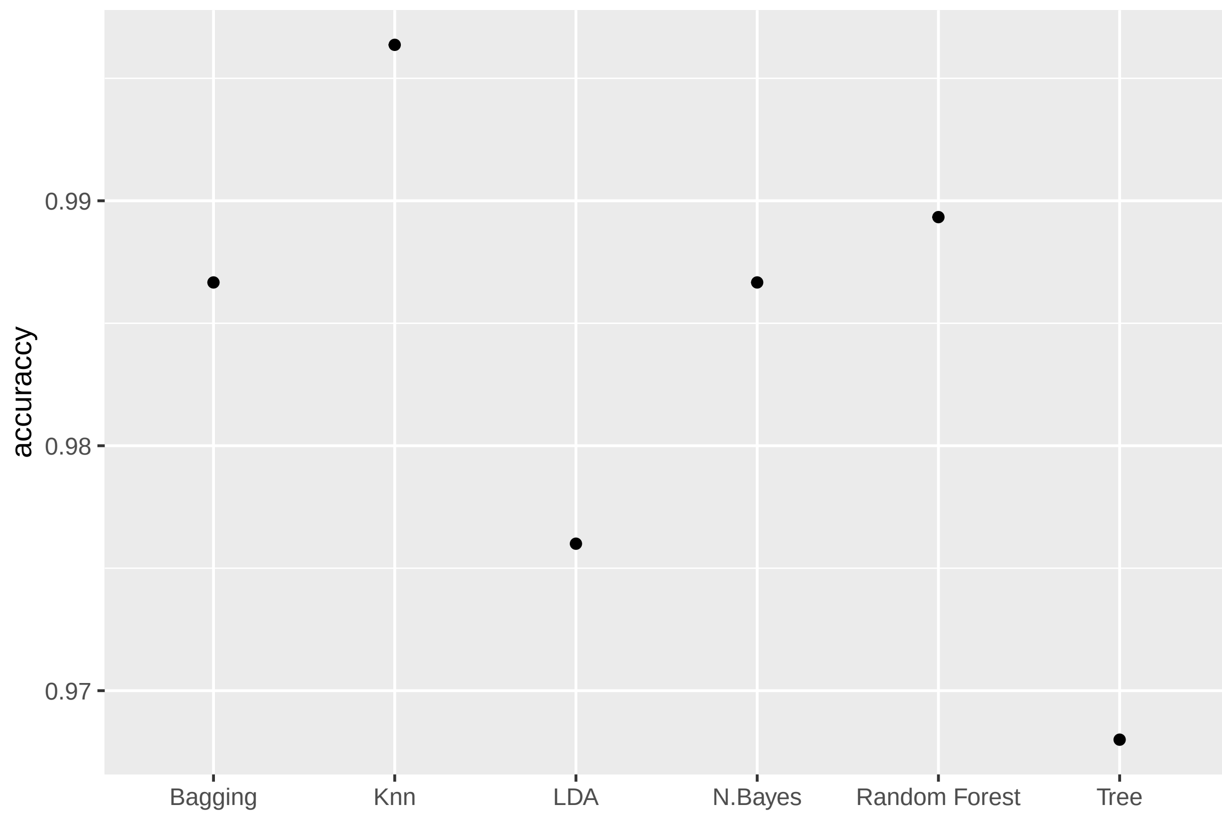

```r
auc(roc.rf)
```

```
## Area under the curve: 0.9998
```

3.6

4 Résumé des performances

```r
acc.df = data.frame(models = c("N.Bayes", "LDA", "Knn", "Tree","Bagging","Random Forest"),
accuraccy = c(acc.NB,acc.LDA,acc.knn,acc.tree,acc.bag,acc.rf))
ggplot(acc.df, aes(models, accuraccy))+ geom_point() +xlab("")
```


accuraccy
0.99
0.98
0.97
Bagging
Knn
LDA
N.Bayes
Random Forest
Tree