POLYTECHNIQUE MONTRÉAL

WORLD-CLASS ENGINEERING

MTH6312-Méthodes statistiques d'apprentissage

Rapport du Projet

Classification des données marketing bancaire

Remis par :

Mariem Jelassi – 1947407
Lyes Heythem BETTACHE – 1923715

Remis à : Luc Adjengue

Remis le : 20 Décembre 2018

1 Table des matières

Mise en contexte

Ce projet présente un ensemble de techniques d'apprentissage automatiques de données que nous avons exploré dans le cours MTH6312. Ce document utilise la Classification des données pour examiner un ensemble de données liées aux campagnes de marketing direct d'une institution bancaire portugaise. Le but de la classification est de prédire si le client souscrira à un dépôt à terme. Notre étude poursuit les principaux objectifs suivants : (1) Utiliser différentes méthodes de classification pour prédire si le client souscrira à un dépôt à terme ; (2) Analyser les sorties de modèles des méthodes utilisées.

La suite de ce rapport est organisée comme suit. La présentation de la problématique dans la partie 1. Ensuite, une présentation des données, variables et des méthodes d'apprentissage utilisées. Par la suite, La partie 3 présente la préparation et la visualisation des données. Après, une exportation des méthodes de classification dans la partie 4. Finalement, la dernière section présente la conclusion.

1. Problématique

L'ensemble des données analysées dans ce document provient d'une campagne de télémarketing menée par une institution bancaire portugaise. La campagne marketing contacte ces clients par téléphone pour tenter de vendre des abonnements de dépôts à terme. L'objectif de la classification est de prédire si le client souscrira à un dépôt à terme.

Le secteur de la gestion bancaire et financière a profité de l'analyse par 'data Science' de ces données. Ce document examine l'ensemble des données de 21 catégories de métadonnées de la campagne marketing afin prédire si le client souscrira à un dépôt à terme. La prédiction se fait suivant la valeur de la variable produit (dépôt à terme) qui peut être 'oui' ou 'non'.

La classification des données désigne l'utilisation de techniques d'apprentissage automatique pour organiser des jeux de données. Cela peut révéler des caractéristiques et des catégories cachées des données. En fait, l'analyse par classification permet l'exploration des données Bank Marketing, afin de déterminer si un client bancaire donné aura tendance à choisir un compte de dépôt à terme qui est un compte qu'ils utilise pour économiser son argent.

2. Données et méthode utilisées

2.1. Données du projet

Le tableau suivant présente la liste des variables que nous avons utilisé pour faire la classification. (Tableau 1)

Tableau 1. Tableau des variables

Variables	***Définition des variables***
age	Age du client - (numérique)
job	Métier du client - (catégorique) (admin, bluecollar, entrepreneur, housemaid, management, retired, selfemployed, services, student, technician, unemployed, unknown)
marital	État matrimonial du client - (catégorique) (divorced, married, single, unknown, note: divorced means divorced or widowed)
education	Niveau d'éducation du client - (catégorique) (basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown)
default	Indique si le client a du crédit en défaut - (catégorique) (no, yes, unknown)
loan	Le client est-il prêté au logement? - (catégorique) (no, yes, unknown)
housing	Le client est-il prêt personnel? - (catégorique) (no, yes, unknown)
contact	Type de contact de communication - (catégorique) (cellular, telephone)
month	Mois du dernier contact avec le client - (catégorique) (January - December)
day_of_week	Jour du dernier contact avec le client - (catégorique) (Monday - Friday)
duration	Durée du dernier contact avec le client, en secondes - (numérique) À des fins de référence uniquement, et non fiable pour la modélisation prédictive
compaign	Nombre de contacts clients au cours de cette campagne - (numérique) (comprennent le dernier contact)
pdays	Nombre de jours depuis la dernière campagne contactée d'une campagne précédente - (numérique) (999 signifie que le client n'a pas encore été contacté)
previous	Nombre de contacts clients réalisés avant cette campagne - (numérique)
poutcome	Résultats de campagne marketing antérieurs - (catégorique) (failure, nonexistent , success)

emp.var.rate	Taux de variation trimestrielle de l'emploi - (numérique)
cons.price.idx	Indice mensuel des prix à la consommation - (numérique)
cons.conf.idx	Indice de confiance mensuel des consommateurs - (numérique)
euribor3m	Taux journalier euribor à 3 mois - (numérique)
nr.employed	Nombre trimestriel d'employés - (numérique)
Y- (Term Deposit)	Abonnement vérifié (binaire: "oui", "non") Variable de sortie (cible souhaitée) -

2.2. Méthodes utilisés :

Dans le cadre de notre cours, nous avons eu l'occasion de travailler avec plusieurs méthodes d'apprentissage plus spécifiquement les méthodes utilisées pour la classification. En fait, nous avons choisi d'utiliser des méthodes paramétrique et non paramétrique. Les techniques de science des données utilisées dans le cadre de cette recherche sont :

- KNN ;
- Arbre de décision ;
- Random forest ;
- Réseau d neurones ;
- Naïves de bayes ;
- LDA ;
- QDA ;
- SVM ;
- Régression logistique.

3. Préparation et Visualisation des données

3.1. Préparation des données

L'ensemble de données examiné par ce document proviennent de la campagne de télémarketing menée par une institution bancaire. En fait, les clients ont été contactés plus d'une dans le but de leur vendre des abonnements de dépôts à terme. L'ensemble de données Bank Marketing comprend 4119 enregistrements, avec 21 observations par enregistrement. Chaque enregistrement comprend 20 observations explicatives sur le client contacté et une observation de réponse

indiquant si le client a souscrit à un dépôt à terme. Les 20 observations explicatives contiennent 4 types de données :

1) Données clients: âge, emploi, état civil, éducation, défaut, logement et emprunt.
2) Données de télémarketing: contact, mois, jour de la semaine et durée.
3) Données socio-économiques: taux de variation de l'emploi, indice des prix à la consommation, indice de confiance des consommateurs, taux Euribor à 3 mois et nombre d'employés.
4) Autres données: campagne, derniers jours, précédents et résultats passés.

Le jeu de données Bank Marketing contient des variables numériques (utiles pour l'analyse prédictive et l'apprentissage automatique), des variables catégorielles, des variables discrètes et continues. Au départ, les 20 variables explicatives semblent utiles pour prédire les futures souscriptions de dépôts à terme. Nous ne tiendrons compte dans ce qui suit que des données du client et télémarketing (12 premières variables dans le tableau 1). Ce choix est dû au fait que les autres variables avaient été utilisée par la compagnie de marketing et ne sont donc pas un facteur de prédiction en temps réel de la probabilité d'obtenir un dépôt à terme. En fait, les données que nous avons enlevées présentent une information par rapport à la compagnie de marketing. Ainsi, nous avons utilisé la fonction 'table' pour avoir le pourcentage des 'yes ' et 'no' pour le dépôt à terme de chaque catégorie de variables (voir l'annexe).

Étapes de préparation de données

1- Chercher s'il existe des valeurs NaN, nous remarquons qu'il n'y a pas des données manquantes.
2- Nous avons classé les variables 'age' et 'duration', Les variables 'age' vont être classées dans des intervalles de 18 à 100 pour avoir des données catégoriques comme les autres données (voir tableau 2). Pour la variable duration qui est la durée d'appel avec le client, cette dernière est enregistrée en secondes, nous l'avons changé en minutes.
3- Nous avons changé les variables catégoriques à des sorties numériques, puisqu'il y a des méthodes pour lesquelles on doit faire ce changement.
4- Nous avons enregistré la sortie Y comme as.factor.

5- Nous avons enregistré deux fichiers Excel. Le premier est un fichier ne contenant que des données numériques (sauf l'output Y) et le deuxième ne contient que des données en catégorie de type caractère.

Tableau 2. Variable 'age'

Ages	Catégories
1	$18 \leq x < 25$
2	$25 \leq x < 35$
3	$35 \leq x < 45$
4	$45 \leq x < 55$
5	$55 \leq x < 65$
6	$65 \leq x < 75$
7	$75 \leq x < 100$

3.2. Visualisation de données

Cette partie englobe la visualisation de nos données par rapport à la Variable de sortie Y qui est une variable binaire (yes, no). D'abord, nous avons 4119 observations divisées en 3668 'no' et 451 'yes'. Nous avons présenté l'ensemble des 12 variables que nous avons gardées dans des graphiques de box-plot par rapport à la sortie Y.

On peut conclure que les données présentent des sorties y de valeurs 'no', c.-à-d. que les clients dans la majorité des cas n'acceptent pas l'offre. Nous avons aussi utilisé la fonction 'table' pour avoir la fréquence de y par rapport à chaque variable. La fréquence d'output y par rapport aux variables d'entrées permet de connaitre la réponse (yes ou no) de chaque catégorie. Enfin, nous avons présenté des histogrammes des catégories de chaque variable. Nous représentons quelques graphiques de box-plot et d'histogrammes ainsi que des tableaux de fréquences, le reste des graphiques sont inclus dans l'annexe.

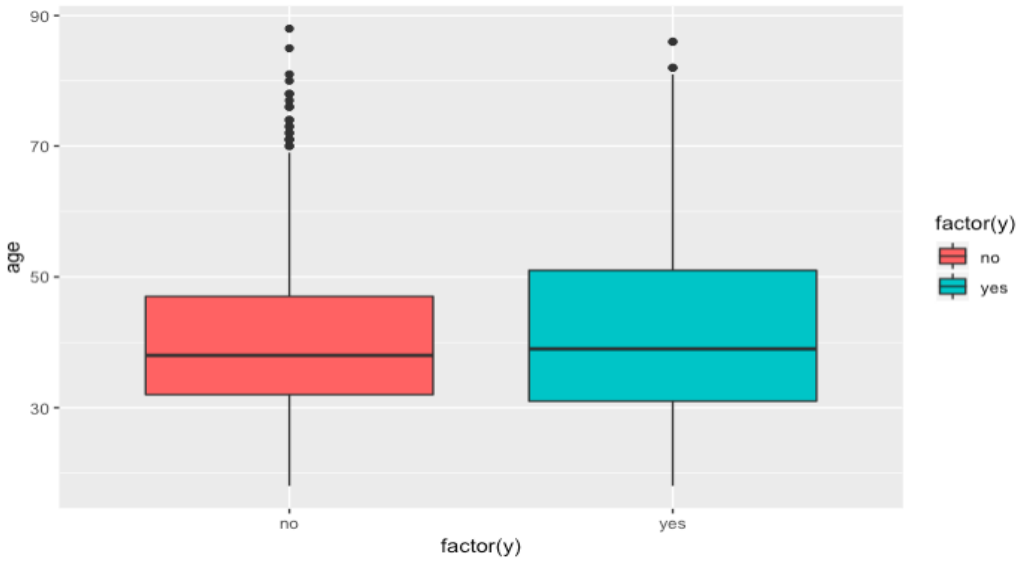


Figure 1. Variable 'age' par rapport la sortie Y

Tableau 3. La fréquence de y par rapport éducation

Frequency of Term Deposits by education	no	yes
basic.4y	391	38
basic.6y	211	17
basic.9y	531	43
high.school	824	97
illiterate	1	0
professional.course	470	65
university.degree	1099	165
unknown	141	26

Tableau 4. Fréquence de y par rapport job

Frequency of Term Deposits by Job	no	yes
admin.	879	133
blue-collar	823	61
entrepreneur	140	8
housemaid	99	11
management	294	30
retired	128	38
self-employed	146	13
services	358	35
student	63	19
technician	611	80
unemployed	92	19
unknown	35	4

4. Classification

Dans la classification, le jeu de données doit être divisé en ensembles d'entraînement et de test. L'entraînement sur modèle est effectué à l'aide de l'ensemble d'apprentissage et l'ensemble d'essai (test) sert à évaluer les performances du modèle de classification. Dans notre cas, nous avons coupé les données avec la méthode 'échantillonnage stratifié' qui divise les ensembles de données en sous-groupes. Par la suite, elle crée l'ensemble d'entrainement en sélectionnant ces sous-groupes avec la même distribution des résultats de l'ensemble des données. Nous avons échantillonné 75% de l'ensemble de données sous forme d'apprentissage et 25% sous forme d'ensemble de test. Nous avons utilisé une validation croisée k=10 fois avec un échantillonnage stratifié est utilisée pour diviser les données en ensembles d'entrainement et de test. Dans la modélisation de la classification, la validation croisée est utilisée avec 10 répétitions. Pour la plupart des méthodes utilisées, nous avons utilisé la fonction 'train' qui incluse dans la bibliothèque 'caret'. Nous avons inclus dans l'annexe le code R que nous avons utilisé pour générer les graphiques ci-dessous.

4.1. KNN

Pour la méthodes KNN la valeur de précision (Accuracy) a été utilisée pour sélectionner le modèle optimal. La meilleure valeur optimale pour le modèle est présentée en k = 10. (Voir figure ci-dessous)

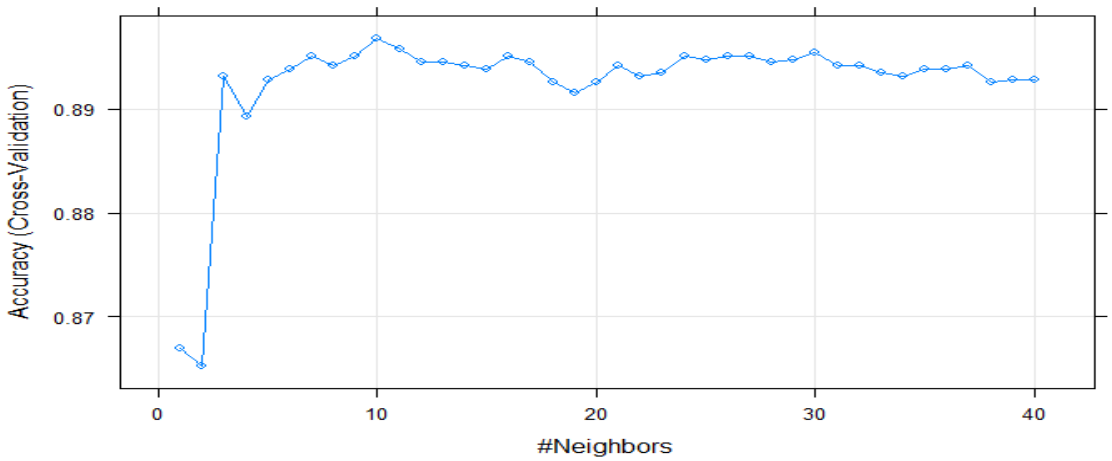


Figure 2: Accuracy en fonction de K

D'après la matrice de confusion, nous avons obtenu une précision de 0.8913, une sensitivité a 0.9902 ainsi qu'une spécificité de 0.0885. Nous pouvons remarquer que le modèle knn a relativement bien classé les clients qui n'ont pas accepté l'offre (99.02% qui n'acceptent pas: Non). Par conte, le modèle n'a pas bien classé les clients qui ont accepté l'offre (8.85% qui accepte: Yes). La matrice de confusion et l'erreur sont présentés ci-dessous (Tableau 5).

Tableau 5. . Matrice de confusion et taux d'erreur test (KNN)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  908 103
       yes   9  10

               Accuracy : 0.8913
                 95% CI : (0.8706, 0.9096)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.4853

                  Kappa : 0.1238
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9902
            Specificity : 0.0885
         Pos Pred Value : 0.8981
         Neg Pred Value : 0.5263
             Prevalence : 0.8903
         Detection Rate : 0.8816
   Detection Prevalence : 0.9816
      Balanced Accuracy : 0.5393

       'Positive' Class : no

Taux d'erreur test est: 0.10873786407767
```

4.2. Logistic Classification

Avec la Classification logistique, nous avons obtenu une précision de 0.899, une sensitivité a 0.9891 ainsi qu'une spécificité 01681. Ces résultats sont légèrement meilleurs comparé à KNN

puisque le pourcentage de la spécificité a augmenté à 16.81%. L'erreur obtenue par les données test est 0.100097.

Tableau 6. Matrice de confusion et taux d'erreur test (R.logistic)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  907  94
       yes  10  19

               Accuracy : 0.899
                 95% CI : (0.879, 0.9168)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.1994

                  Kappa : 0.2332
 Mcnemar's Test P-Value : 3.992e-16

            Sensitivity : 0.9891
            Specificity : 0.1681
         Pos Pred Value : 0.9061
         Neg Pred Value : 0.6552
             Prevalence : 0.8903
         Detection Rate : 0.8806
   Detection Prevalence : 0.9718
      Balanced Accuracy : 0.5786

       'Positive' Class : no

Taux d'erreur test est: 0.100970873786408
```

4.3. LDA

Avec LDA nous avons obtenu une précision de 0.8971, une sensitivité a 0.977 ainsi qu'une spécificité 0.2478. Ces résultats sont légèrement meilleurs comparé à KNN et LDA puisque le pourcentage de la spécificité est augmenté à 24,78%. L'erreur obtenue par les données test est 0.10291.

Tableau 7. Matrice de confusion et taux d'erreur test (LDA)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  896  85
       yes  21  28

               Accuracy : 0.8971
                 95% CI : (0.8769, 0.915)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.2609

                  Kappa : 0.2992
 Mcnemar's Test P-Value : 9.41e-10

            Sensitivity : 0.9771
            Specificity : 0.2478
         Pos Pred Value : 0.9134
         Neg Pred Value : 0.5714
             Prevalence : 0.8903
         Detection Rate : 0.8699
   Detection Prevalence : 0.9524
      Balanced Accuracy : 0.6124

       'Positive' Class : no

Taux d'erreur test est: 0.102912621359223
```

4.4. QDA

Pour la méthode QDA, la spécificité a augmenté à 30% ce qui est mieux que LDA, par contre la précision et la sensitivité sont restés les mêmes. Nous pouvons remarquer qu'une méthode plus flexible comme QDA peut nous donner de meilleurs résultats.

Tableau 8. Matrice de confusion et taux d'erreur test (QDA)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  890  79
       yes  27  34

               Accuracy : 0.8971
                 95% CI : (0.8769, 0.915)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.2609

                  Kappa : 0.34
 Mcnemar's Test P-Value : 7.287e-07

            Sensitivity : 0.9706
            Specificity : 0.3009
         Pos Pred Value : 0.9185
         Neg Pred Value : 0.5574
             Prevalence : 0.8903
         Detection Rate : 0.8641
   Detection Prevalence : 0.9408
      Balanced Accuracy : 0.6357

       'Positive' Class : no

Taux d'erreur test est: 0.102912621359223
```

4.5. Arbre de décision

Dans la classification des arbres de décision, une série de questions de test est organisée dans une arborescence et les données sont étiquetées dans les nœuds racine. Nous avons utilisé la classification de l'arbre de décision rpart qui est un algorithme d'extension de l'arbre de décision (Voir Figure 3).

D'après la matrice de confusion, nous avons obtenu une précision de 0.9029, une sensitivité a 0.9662 ainsi qu'une spécificité 0.3894. Nous pouvons remarquer que le modèle d'arbre de décision a relativement bien classé les clients qui n'ont pas accepté l'offre (96.62% qui n'accepte pas: Non). Par contre, le modèle n'a pas très bien classé les clients qui ont accepté l'offre (38.94% qui accepte : Yes). La matrice de confusion et l'erreur sont présenté ci-dessous (voir tableau 9).

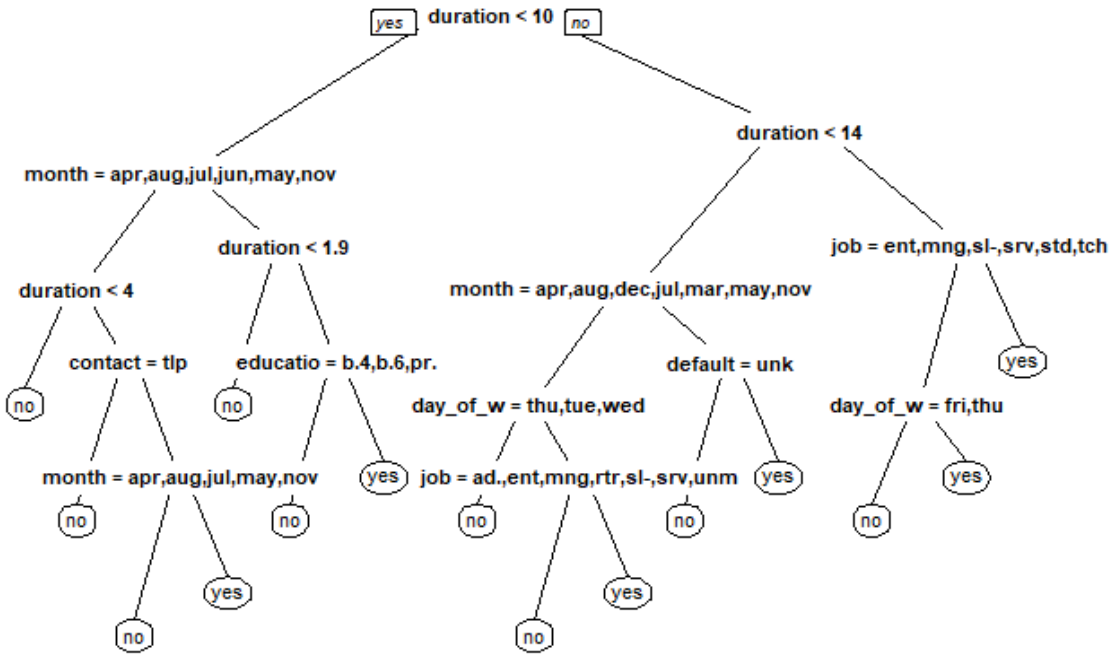


Figure 3:Arbre Optimale

Tableau 9. Matrice de confusion et taux d'erreur test (Arbre de décision)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  886  69
       yes  31  44

               Accuracy : 0.9029
                 95% CI : (0.8832, 0.9203)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.1049001

                  Kappa : 0.4171
 Mcnemar's Test P-Value : 0.0002156

            Sensitivity : 0.9662
            Specificity : 0.3894
         Pos Pred Value : 0.9277
         Neg Pred Value : 0.5867
             Prevalence : 0.8903
         Detection Rate : 0.8602
   Detection Prevalence : 0.9272
      Balanced Accuracy : 0.6778

       'Positive' Class : no

Taux d'erreur test est: 0.0970873786407767
```

4.6. Random forest

Random Forest est un algorithme d'apprentissage automatique flexible et facile à utiliser qui produit, même sans réglage hyper-paramètre, un bon résultat la plupart du temps. Il est également l'un des algorithmes les plus utilisés à cause de sa simplicité et le fait qu'il peut être utilisé pour les tâches de classification et de régression.

Pour la méthode Random forest, la valeur de précision (Accuracy) a été utilisé pour sélectionner le modèle optimal.

D'après la matrice de confusion, nous avons obtenus une précision de 0.8932, une sensitivité a 0.9684 ainsi qu'une spécificité 0.2832. Nous pouvons remarquer que le modèle d'arbre de décision a relativement bien classé les clients qui n'ont pas accepté l'offre (96.84% qui n'accepte pas: Non). Par conte, le modèle n'a pas très bien classé les clients qui ont accepté l'offre (28.32% qui accepte : Yes). (Voir Tableau 10)

Tableau 10. Matrice de confusion et taux d'erreur test (Random forest)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  888  81
       yes  29  32

               Accuracy : 0.8932
                 95% CI : (0.8727, 0.9114)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.4063

                  Kappa : 0.3151
 Mcnemar's Test P-Value : 1.158e-06

            Sensitivity : 0.9684
            Specificity : 0.2832
         Pos Pred Value : 0.9164
         Neg Pred Value : 0.5246
             Prevalence : 0.8903
         Detection Rate : 0.8621
   Detection Prevalence : 0.9408
      Balanced Accuracy : 0.6258

       'Positive' Class : no

Taux d'erreur test est: 0.106796116504854
```

4.7. Naïves de bayes

L'algorithme Naïve Bayes est basé sur le calcul des probabilités postérieures de différentes hypothèses et sur le choix de la probabilité la plus élevée. D'après le matrice de confusion, nous avons obtenue une précision de 0.8942, une sensitivité a 0.9695 ainsi qu'une spécificité 0.2832. Nous pouvons remarquer que le modèle Naïve de bayes a bien classé les clients qui n'ont pas accepté l'offre (96.95% qui n'accepte pas: Non). Par contre, comme les autres modèles précédents, il n'a pas bien classé les clients qui ont accepté l'offre (28.32% qui accepte : Yes). (Voir Tableau 11)

Tableau 11. Matrice de confusion et taux d'erreur test (Naïves de bayes)

```
          Reference
Prediction  no yes
       no  889  81
       yes  28  32

               Accuracy : 0.8942
                 95% CI : (0.8738, 0.9123)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.3679

                  Kappa : 0.318
 Mcnemar's Test P-Value : 6.336e-07

            Sensitivity : 0.9695
            Specificity : 0.2832
         Pos Pred Value : 0.9165
         Neg Pred Value : 0.5333
             Prevalence : 0.8903
         Detection Rate : 0.8631
   Detection Prevalence : 0.9417
      Balanced Accuracy : 0.6263

       'Positive' Class : no

Taux d'erreur test est: 0.105825242718447
```

4.8. SVM

Pour la classification avec Support Vector Machine, L'algorithme trouve un hyperplan optimal qui sépare les points de données appartenant à différentes classes. Différentes fonctions du noyau pouvant être utilisées avec SVM sont les suivantes: base radiale, hyperbolique, linéaire. Nous avons utilisé la fonction de noyau polynomiale (svmPoly) pour cet ensemble de données.

Nous avons obtenu une précision de 0.8971, une sensitivité a 0.9891 ainsi qu'une spécificité 0.1504. Cela n'est pas vraiment mieux que les autres modèles puisque le pourcentage de la spécificité est 15.04 %. (Voir tableau 12).

Tableau 12. Matrice de confusion et taux d'erreur test (SVM)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  907  96
       yes  10  17

               Accuracy : 0.8971
                 95% CI : (0.8769, 0.915)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.2609

                  Kappa : 0.2094
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9891
            Specificity : 0.1504
         Pos Pred Value : 0.9043
         Neg Pred Value : 0.6296
             Prevalence : 0.8903
         Detection Rate : 0.8806
   Detection Prevalence : 0.9738
      Balanced Accuracy : 0.5698

       'Positive' Class : no

Taux d'erreur test est: 0.102912621359223
```

4.9. Réseau de neurones

Le dernier algorithme que nous avons utilisé dans l'ensemble de données est le réseau de neurones. Les réseaux de neurones imitent la façon dont le cerveau fonctionne et résolvent les problèmes posés par les unités neuronales. Cette méthode a une fonction qui combine toutes les entrées et se propage aux autres neurones. (Voir figure.4)

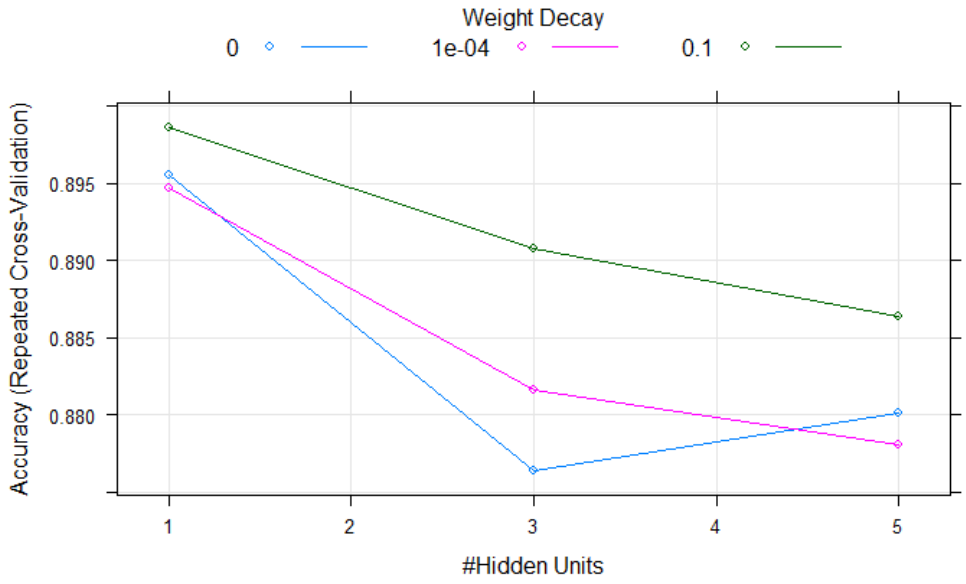


Figure 4. Accuracy en fonction des couches cachées

En utilisant RN, nous avons obtenu une précision de 0.9, une sensitivité a 0.9586 ainsi qu'une spécificité 0.4248. Cela n'est pas mieux que les autres modèles puisque le pourcentage de la spécificité a augmenté à 42.48 %. (Voir tableau 13).

Tableau 13. Matrice de confusion et taux d'erreur test (Réseau de neurones)

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  879  65
       yes  38  48

               Accuracy : 0.9
                 95% CI : (0.88, 0.9176)
    No Information Rate : 0.8903
    P-Value [Acc > NIR] : 0.17207

                  Kappa : 0.4282
 Mcnemar's Test P-Value : 0.01041

            Sensitivity : 0.9586
            Specificity : 0.4248
         Pos Pred Value : 0.9311
         Neg Pred Value : 0.5581
             Prevalence : 0.8903
         Detection Rate : 0.8534
   Detection Prevalence : 0.9165
      Balanced Accuracy : 0.6917

       'Positive' Class : no

Taux d'erreur test est: 0.1
```

4.10. Résumé graphique et comparaison des méthodes

D'après le graphe du taux d'erreur, nous remarquons que globalement, tous les modèles ont des erreurs relativement faibles de l'ordre de 10%, et le meilleur modèle est l'Arbre de décision (erreur minimum: 9.7%). Par contre, le meilleur modèle qui a bien classé les clients de la banque qui ont accepté l'offre est le modèle des réseaux de neurone (RN). En fait, d'après les graphes ci-dessous figures 5-6-7, nous pouvons remarquer que la meilleure spécificité est RN qui a le maximum de spécificité: 42.48%), ce qui nous ramené à choisir RN.

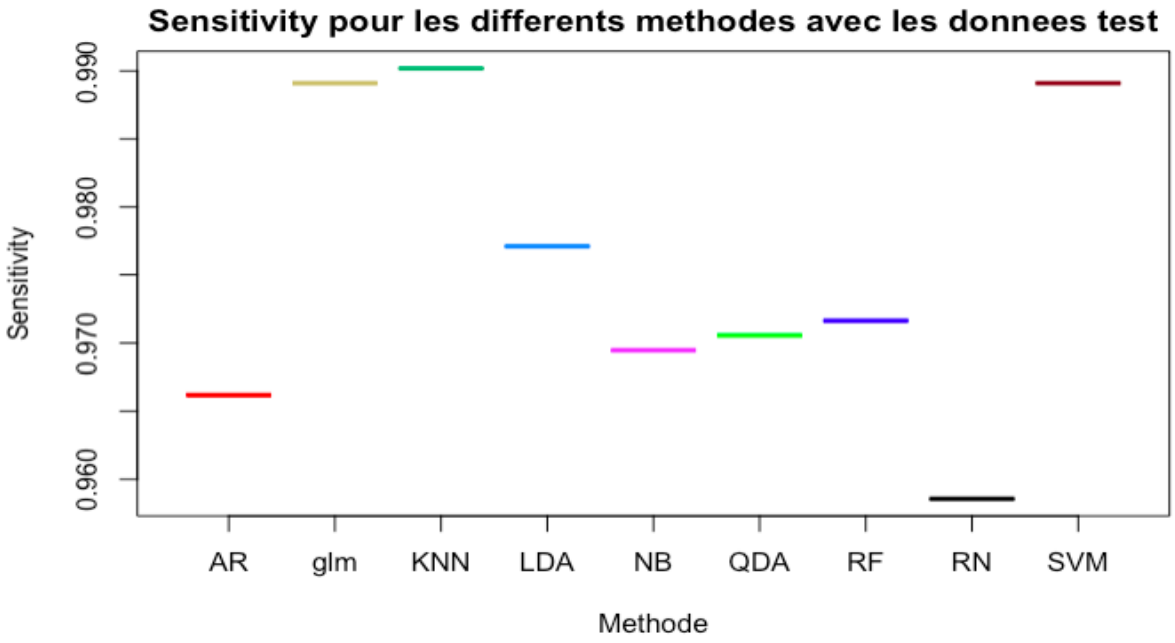


Figure 5. la sensitivité pour les differentes méthodes utilisées

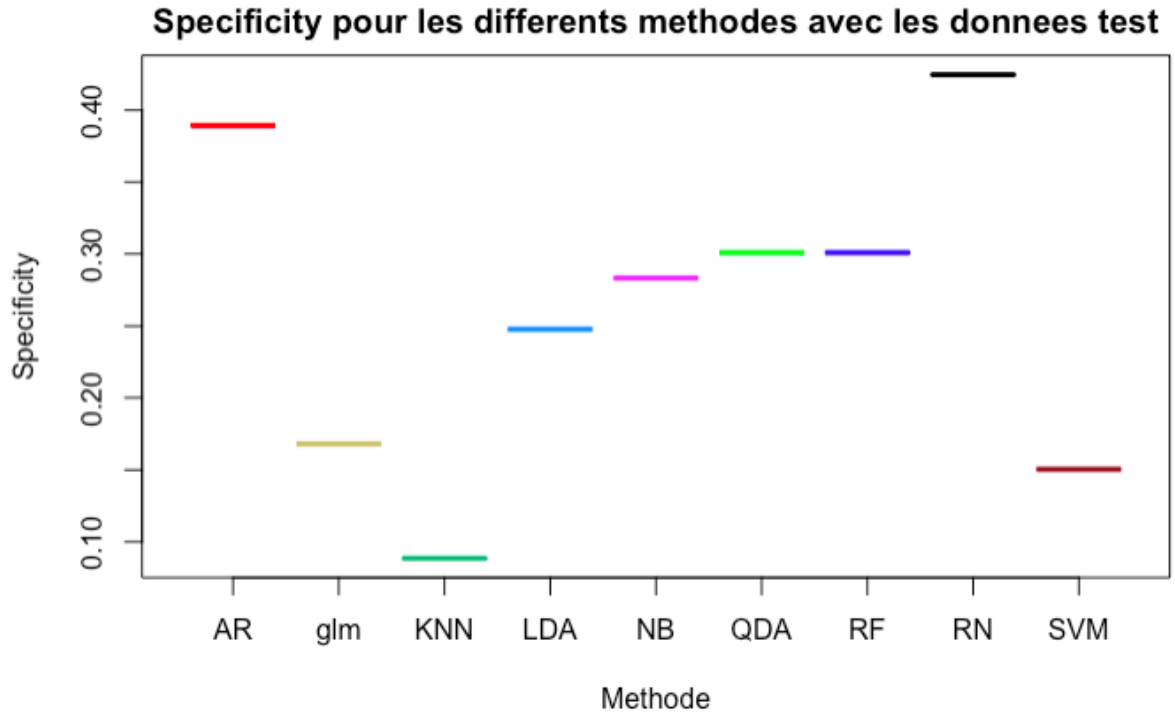


Figure 6. La sepécifité pour les differentes méthodes utilisées

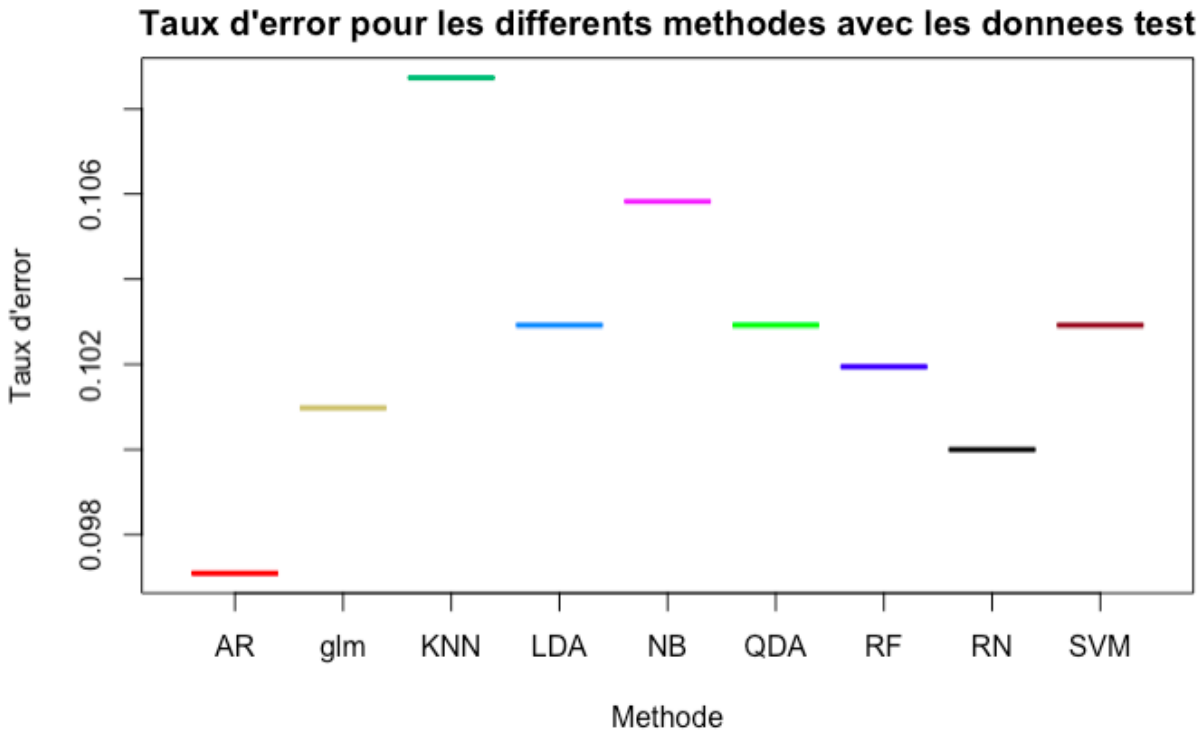


Figure 7. l'erreur pour les differentes méthodes utilisées

5. Conclusion

La méthode la plus efficace pour déterminer si un client choisira ou non un plan de dépôt à terme, après l'analyse exploratoire des données est la méthode de réseau de neurone.

Les résultats montrent que les réseaux de neurones constituent le meilleur modèle de classification permettant de prédire si le client va s'abonner au dépôt de banque. Cet algorithme devrait être utilisé pour établir les prévisions futures sur le processus d'apprentissage de la banque pour améliorer leurs secteurs financiers.

Le taux de prédiction relativement faible obtenu par tous les algorithmes que nous avons utilisés est dû à la nature de l'ensemble de données utilisées. En effet, le pourcentage des gens ayant répondu 'oui' au dépôt à terme ne constitue que 4,75% du nombre total de personnes interrogées.

ANNEXE

R Markdown

```
remove(list=ls())


library(readr)
library(ggplot2)
library(lattice)
library(plyr)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(caret)
library(mlbench)
library(foreign)
library(ggplot2)
library(reshape)

##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename

## The following objects are masked from 'package:plyr':
##
##     rename, round_any

library(scales)

##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(e1071)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(rpart)# Pour l'arbre de décision
library(rpart.plot) # Pour la représentation de lâ□□arbre de décision
```

Visualisation des données

```
mybank <- read_delim("bank-additional.csv",";",escape_double = FALSE, trim_ws
= TRUE)
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   age = col_double(),
##   duration = col_double(),
##   campaign = col_double(),
##   pdays = col_double(),
##   previous = col_double(),
##   emp.var.rate = col_double(),
##   cons.price.idx = col_double(),
##   cons.conf.idx = col_double(),
##   euribor3m = col_double(),
##   nr.employed = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
str(mybank)
```

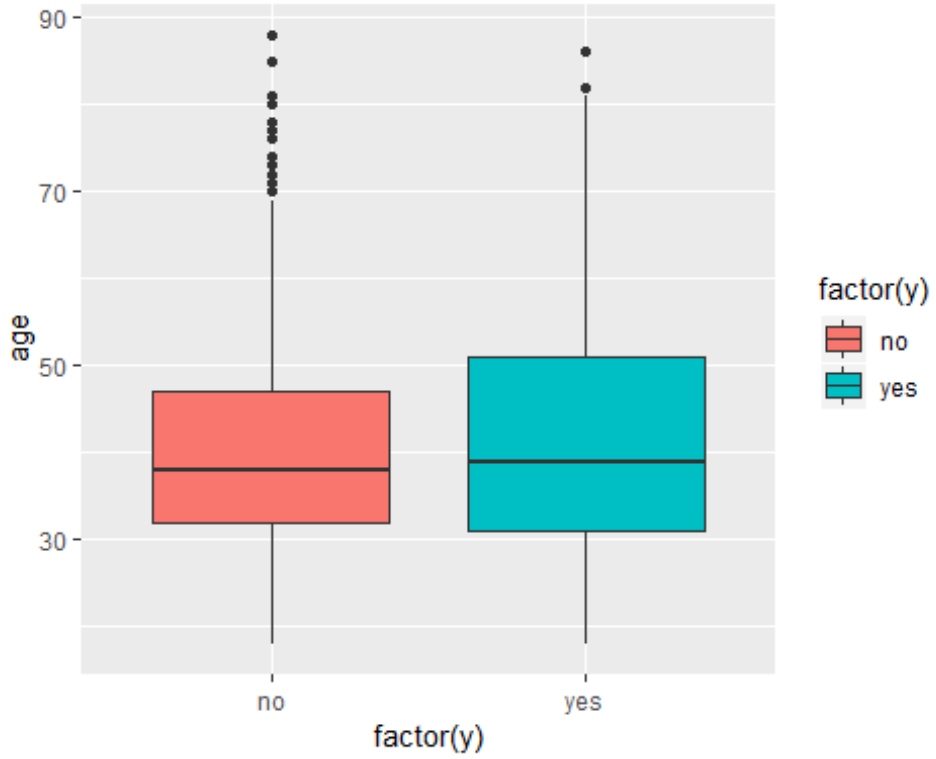
```
## Classes 'tbl_df', 'tbl' and 'data.frame':    4119 obs. of  21 variables:
##  $ age           : num  30 39 25 38 47 32 32 41 31 35 ...
##  $ job           : chr  "blue-collar" "services" "services" "services" ...
##  $ marital       : chr  "married" "single" "married" "married" ...
##  $ education     : chr  "basic.9y" "high.school" "high.school" "basic.9y"
...
##  $ default       : chr  "no" "no" "no" "no" ...
##  $ housing       : chr  "yes" "no" "yes" "unknown" ...
##  $ loan          : chr  "no" "no" "no" "unknown" ...
```

```
##  $ contact       : chr  "cellular" "telephone" "telephone" "telephone" ...
##  $ month         : chr  "may" "may" "jun" "jun" ...
##  $ day_of_week   : chr  "fri" "fri" "wed" "fri" ...
##  $ duration      : num  487 346 227 17 58 128 290 44 68 170 ...
##  $ campaign      : num  2 4 1 3 1 3 4 2 1 1 ...
##  $ pdays         : num  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous      : num  0 0 0 0 0 2 0 0 1 0 ...
##  $ poutcome      : chr  "nonexistent" "nonexistent" "nonexistent" "nonexis
tent" ...
##  $ emp.var.rate  : num  -1.8 1.1 1.4 1.4 -0.1 -1.1 -1.1 -0.1 -0.1 1.1 ...
##  $ cons.price.idx: num  92.9 94 94.5 94.5 93.2 ...
##  $ cons.conf.idx : num  -46.2 -36.4 -41.8 -41.8 -42 -37.5 -37.5 -42 -42 -3
6.4 ...
##  $ euribor3m     : num  1.31 4.86 4.96 4.96 4.19 ...
##  $ nr.employed   : num  5099 5191 5228 5228 5196 ...
##  $ y             : chr  "no" "no" "no" "no" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   job = col_character(),
##   ..   marital = col_character(),
##   ..   education = col_character(),
##   ..   default = col_character(),
##   ..   housing = col_character(),
##   ..   loan = col_character(),
##   ..   contact = col_character(),
##   ..   month = col_character(),
##   ..   day_of_week = col_character(),
##   ..   duration = col_double(),
##   ..   campaign = col_double(),
##   ..   pdays = col_double(),
##   ..   previous = col_double(),
##   ..   poutcome = col_character(),
##   ..   emp.var.rate = col_double(),
##   ..   cons.price.idx = col_double(),
##   ..   cons.conf.idx = col_double(),
##   ..   euribor3m = col_double(),
##   ..   nr.employed = col_double(),
##   ..   y = col_character()
##   .. )
```

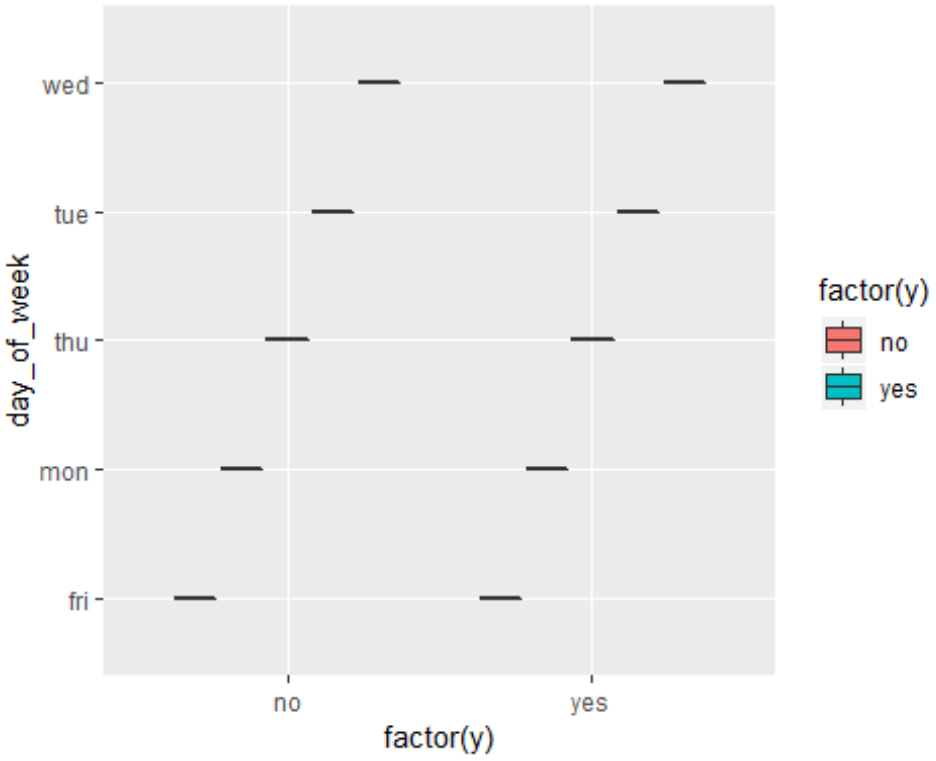
```
table(mybank$y)
```

```
##
##   no  yes
## 3668  451
```

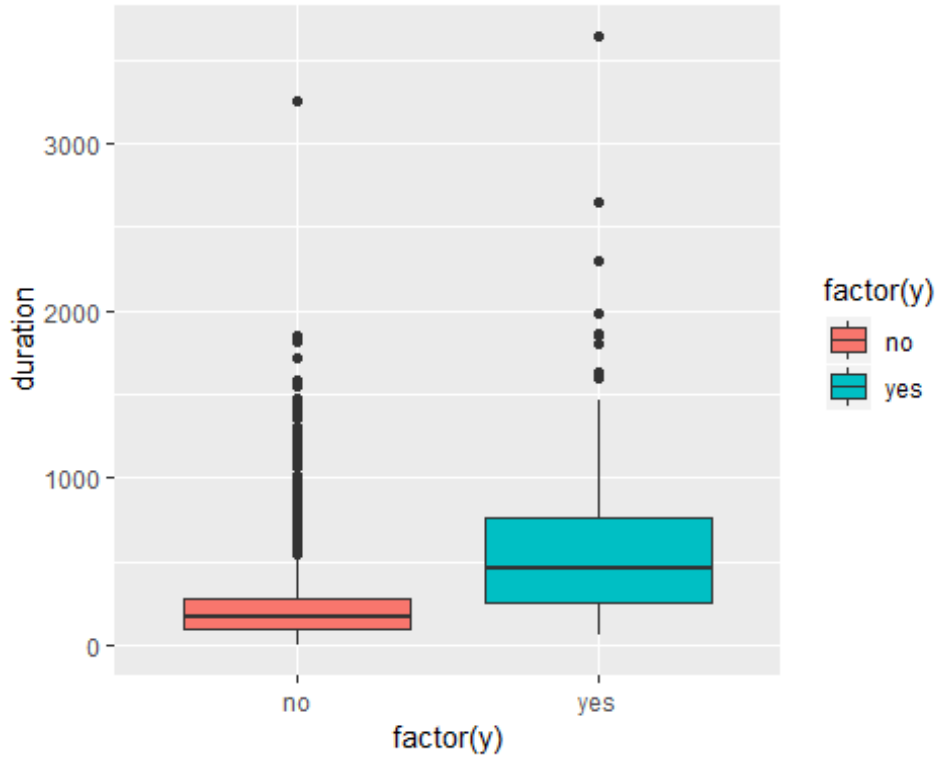
```
p_age <- ggplot(mybank, aes(factor(y), age)) + geom_boxplot(aes(fill = factor
(y)))
p_age
```



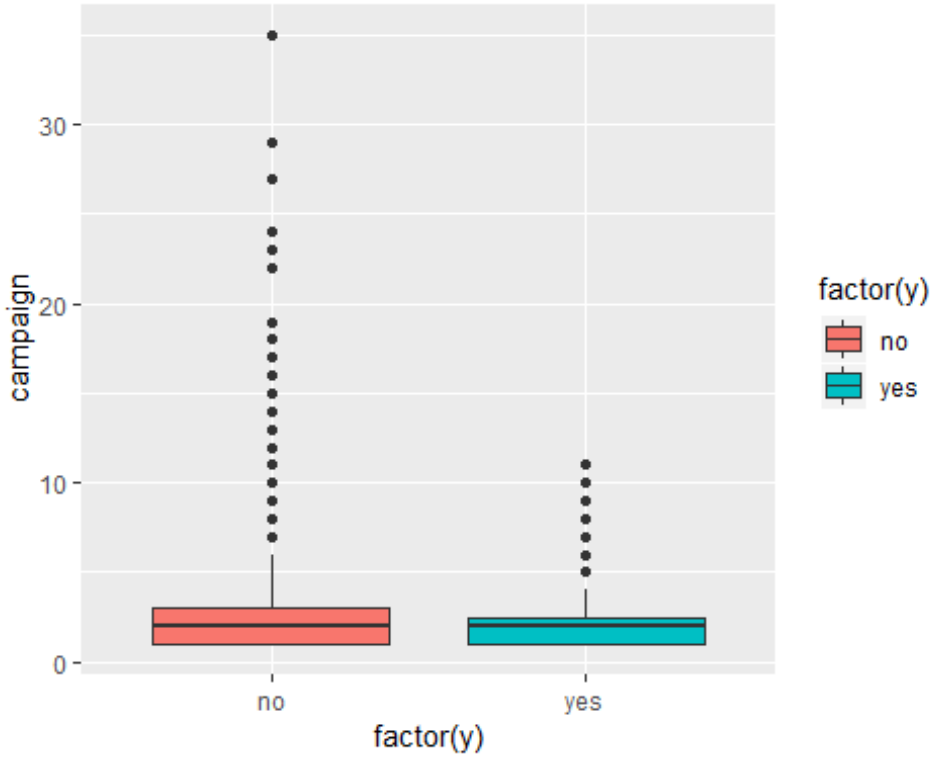

```
p_day_of_week <- ggplot(mybank, aes(factor(y), day_of_week)) + geom_boxplot(a
es(fill = factor(y)))
p_day_of_week
```



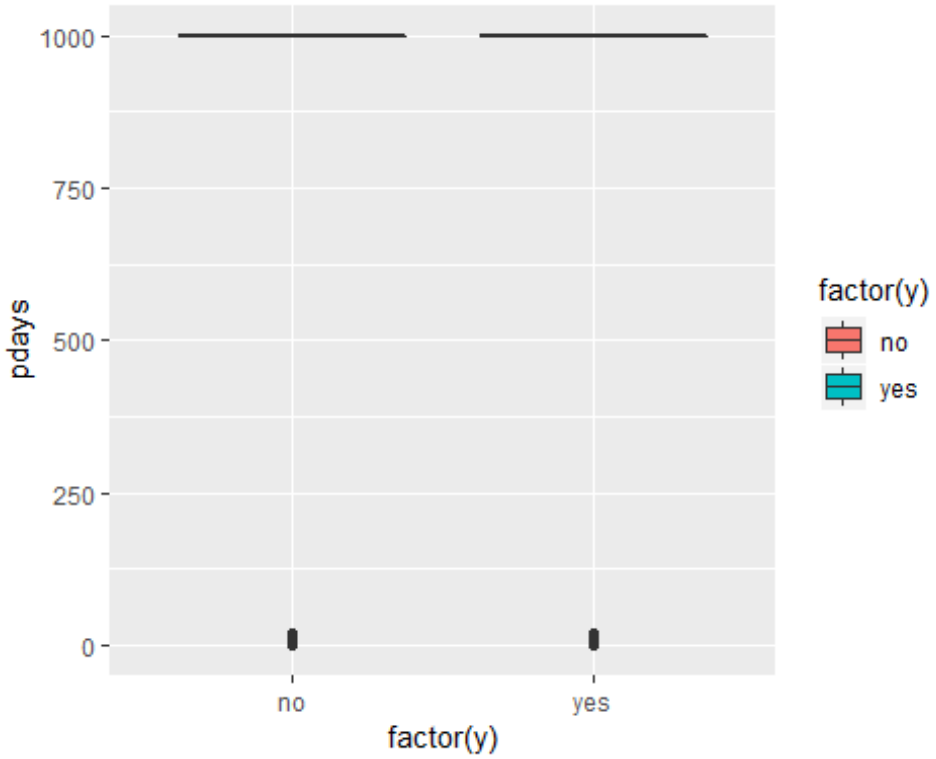
```
p_duration <- ggplot(mybank, aes(factor(y), duration)) + geom_boxplot(aes(fil
l = factor(y)))
p_duration
```



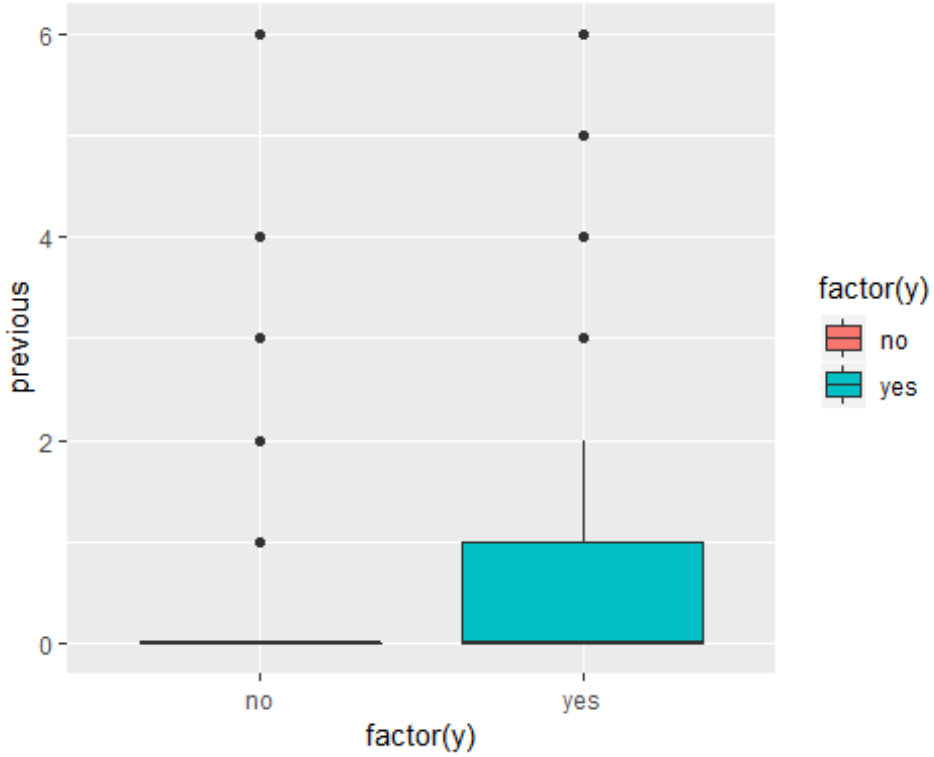

```
p_campaign <- ggplot(mybank, aes(factor(y), campaign)) + geom_boxplot(aes(fil
l = factor(y)))
p_campaign
```




```
p_pdays <- ggplot(mybank, aes(factor(y), pdays)) + geom_boxplot(aes(fill = fa
ctor(y)))
p_pdays
```



```
p_previous <- ggplot(mybank, aes(factor(y), previous)) + geom_boxplot(aes(fil
l = factor(y)))
p_previous
```




```
table(mybank$y)
```

```
##
##   no  yes
## 3668  451
```

```
print("Frequency of Term Deposits by Job")
```

```
## [1] "Frequency of Term Deposits by Job"
```

```
table(mybank$job,mybank$y , dnn = "Frequency of Term Deposits by Job" )
```

```
##                                   NA
## Frequency of Term Deposits by Job  no yes
##                       admin.      879 133
##                       blue-collar 823  61
##                       entrepreneur 140   8
##                       housemaid    99  11
##                       management  294  30
##                       retired     128  38
##                       self-employed 146  13
##                       services    358  35
##                       student      63  19
##                       technician  611  80
```

```
##                       unemployed     92  19
##                       unknown        35   4
```

```
age <- as.numeric((cut(mybank$age, c(16,25,35,45,55,65,75,100),labels=c(1,2,3
,4,5,6,7))))
table(age,mybank$y, dnn = "Frequency of Term Deposits by Age")
```

```
##                                    NA
## Frequency of Term Deposits by Age   no  yes
##                                  1  136   19
##                                  2 1334  162
##                                  3 1146  113
##                                  4  763   87
##                                  5  260   44
##                                  6   20   13
##                                  7    9   13
```

```
table(mybank$education,mybank$y , dnn = "Frequency of Term Deposits by educat
ion")
```

```
##                                         NA
## Frequency of Term Deposits by education   no  yes
##                       basic.4y             391   38
##                       basic.6y             211   17
##                       basic.9y             531   43
##                       high.school          824   97
##                       illiterate             1    0
##                       professional.course  470   65
##                       university.degree   1099  165
##                       unknown              141   26
```

```
table(mybank$default,mybank$y, dnn = "Frequency of Term Deposits by default")
```

```
##                                      NA
## Frequency of Term Deposits by default   no  yes
##                                 no      2913  402
##                                 unknown  754   49
##                                 yes        1    0
```

```
table(mybank$housing,mybank$y , dnn = "Frequency of Term Deposits by housing"
)
```

```
##                                      NA
## Frequency of Term Deposits by housing   no  yes
##                                 no      1637  202
##                                 unknown   96    9
##                                 yes     1935  240
```

```
table(mybank$loan,mybank$y, dnn = "Frequency of Term Deposits by loan")
```

```
##                                    NA
## Frequency of Term Deposits by loan   no  yes
```

```
##                             no      2975  374
##                             unknown   96    9
##                             yes      597   68
```

```
table(mybank$marital,mybank$y , dnn = "Frequency of Term Deposits by marital"
)
```

```
##                                     NA
## Frequency of Term Deposits by marital   no  yes
##                               divorced  403   43
##                               married  2257  252
##                               single    998  155
##                               unknown    10    1
```

```
table(mybank$contact,mybank$y, dnn = "Frequency of Term Deposits by contact")
```

```
##                                     NA
## Frequency of Term Deposits by contact   no  yes
##                              cellular  2277  375
##                              telephone 1391   76
```

```
table(mybank$month,mybank$y , dnn = "Frequency of Term Deposits by month")
```

```
##                                    NA
## Frequency of Term Deposits by month   no  yes
##                                  apr  179   36
##                                  aug  572   64
##                                  dec   10   12
##                                  jul  652   59
##                                  jun  462   68
##                                  mar   20   28
##                                  may 1288   90
##                                  nov  403   43
##                                  oct   44   25
##                                  sep   38   26
```

```
table(mybank$day_of_week,mybank$y, dnn = "Frequency of Term Deposits by day_o
f_week ")
```

```
##                                          NA
## Frequency of Term Deposits by day_of_week   no yes
##                                         fri 685  83
##                                         mon 757  98
##                                         thu 764  96
##                                         tue 750  91
##                                         wed 712  83
```

```
dura=mybank$duration/60
dura <- as.numeric(cut(dura, c(0,5,10,50),labels=c(1,2,3)))
table(dura,mybank$y, dnn = "Frequency of Term Deposits by duration ")
```

```
##                                          NA
## Frequency of Term Deposits by duration    no  yes
##                                        1 2867  144
##                                        2  625  138
##                                        3  174  168
```

```
table(mybank$campaign,mybank$y, dnn = "Frequency of Term Deposits by campaign
")
```

```
##                                          NA
## Frequency of Term Deposits by campaign    no  yes
##                                        1  1545  219
##                                        2   920  119
##                                        3   487   62
##                                        4   259   32
##                                        5   133    9
##                                        6    95    4
##                                        7    59    1
##                                        8    34    2
##                                        9    31    1
##                                        10   19    1
##                                        11   18    1
##                                        12   16    0
##                                        13   11    0
##                                        14    6    0
##                                        15    2    0
##                                        16    7    0
##                                        17   14    0
##                                        18    1    0
##                                        19    2    0
##                                        22    2    0
##                                        23    2    0
##                                        24    1    0
##                                        27    1    0
##                                        29    2    0
##                                        35    1    0
```

```
table(mybank$pdays,mybank$y, dnn = "Frequency of Term Deposits by pday ")
```

```
##                                      NA
## Frequency of Term Deposits by pday    no  yes
##                                    0      0    2
##                                    1      3    0
##                                    2      3    1
##                                    3     13   39
##                                    4      9    5
##                                    5      0    4
##                                    6     15   27
##                                    7      2    8
##                                    9      2    1
##                                    10     2    6
```

```
##                                    11     1    0
##                                    12     2    3
##                                    13     2    0
##                                    14     1    0
##                                    15     1    1
##                                    16     2    0
##                                    17     1    0
##                                    18     1    1
##                                    19     0    1
##                                    21     0    1
##                                   999 3608  351
```

```
table(mybank$previous,mybank$y, dnn = "Frequency of Term Deposits by previous
")
```

```
##                                          NA
## Frequency of Term Deposits by previous    no  yes
##                                        0 3231  292
##                                        1  376   99
##                                        2   46   32
##                                        3   10   15
##                                        4    4   10
##                                        5    0    2
##                                        6    1    1
```

```
table(mybank$poutcome,mybank$y, dnn = "Frequency of Term Deposits by poutcome
")
```

```
##                                          NA
## Frequency of Term Deposits by poutcome    no  yes
##                               failure    387   67
##                               nonexistent 3231  292
##                               success     50   92
```

```
table(mybank$emp.var.rate,mybank$y, dnn = "Frequency of Term Deposits by var
rate ")
```

```
##                                          NA
## Frequency of Term Deposits by var rate    no  yes
##                                       -3.4   65   39
##                                       -3      9   12
##                                       -2.9  107   57
##                                       -1.8  754  129
##                                       -1.7   42   45
##                                       -1.1   48   35
##                                       -0.2    1    0
##                                       -0.1  372   20
##                                       1.1   733   25
##                                       1.4  1537   89
```

```
table(mybank$cons.conf.idx,mybank$y, dnn = "Frequency of Term Deposits by idx
")
```

```
##                                    NA
## Frequency of Term Deposits by idx   no yes
##                                -50.8  14  10
##                                -50    12  13
##                                -49.5  13   7
##                                -47.1 174  27
##                                -46.2 546  51
##                                -45.9   1   0
##                                -42.7 624  43
##                                -42   369  17
##                                -41.8 408  23
##                                -40.8  45  30
##                                -40.4   3   3
##                                -40.3  17  13
##                                -40     9  14
##                                -39.8   9  15
##                                -38.3  16  17
##                                -37.5  21  18
##                                -36.4 733  25
##                                -36.1 505  23
##                                -34.8   8  15
##                                -34.6   5   9
##                                -33.6  11   3
##                                -33     9  12
##                                -31.4  51  24
##                                -30.1  20  16
##                                -29.8  17   8
##                                -26.9  28  15
```

```
table(mybank$euribor3m,mybank$y, dnn = "Frequency of Term Deposits by euri ")
```

```
##                                      NA
## Frequency of Term Deposits by euri   no yes
##                                  0.635   2   1
##                                  0.636   1   0
##                                  0.637   0   1
##                                  0.639   0   2
##                                  0.64    0   1
##                                  0.642   0   1
##                                  0.643   1   1
##                                  0.644   1   4
##                                  0.645   1   1
##                                  0.646   2   2
##                                  0.649   1   1
##                                  0.65    1   0
##                                  0.652   0   5
##                                  0.654   1   1
##                                  0.655   0   2
##                                  0.659   2   1
##                                  0.668   1   1
```

```
##                                0.672   1   0
##                                0.677   1   3
##                                0.682   0   2
##                                0.683   0   2
##                                0.692   2   0
##                                0.695   1   1
##                                0.697   1   3
##                                0.699   2   2
##                                0.7     2   0
##                                0.702   1   1
##                                0.704   0   2
##                                0.706   1   4
##                                0.707   1   0
##                                0.709   0   1
##                                0.71    1   0
##                                0.714   5   8
##                                0.715   5  11
##                                0.716   5   1
##                                0.717   1   1
##                                0.718   1   0
##                                0.719   4   1
##                                0.72    4   7
##                                0.721   0   1
##                                0.722   2   4
##                                0.723   0   1
##                                0.724   1   0
##                                0.728   2   1
##                                0.729   1   0
##                                0.73    2   1
##                                0.731   3   0
##                                0.735   0   1
##                                0.737   1   0
##                                0.739   7   3
##                                0.74    3   2
##                                0.741   1   0
##                                0.742   6   2
##                                0.748   2   2
##                                0.749   1   0
##                                0.752   1   0
##                                0.754   3   0
##                                0.755   1   0
##                                0.761   1   1
##                                0.762   1   0
##                                0.767   0   2
##                                0.768   1   0
##                                0.77    1   1
##                                0.771   1   1
##                                0.773   2   2
##                                0.778   1   0
##                                0.782   0   1
```

```
##                                0.788   1   1
##                                0.79    0   1
##                                0.797   0   2
##                                0.803   2   0
##                                0.809   1   1
##                                0.81    1   1
##                                0.819   3   0
##                                0.825   1   0
##                                0.827   2   1
##                                0.829   0   1
##                                0.834   3   0
##                                0.835   1   1
##                                0.838   5   1
##                                0.843   0   1
##                                0.846   2   2
##                                0.849   0   3
##                                0.851   1   3
##                                0.854   0   1
##                                0.859   1   1
##                                0.861   6   2
##                                0.869   3   0
##                                0.87    0   1
##                                0.873   7   2
##                                0.876   2   1
##                                0.877   1   4
##                                0.878   2   4
##                                0.879  14   6
##                                0.881   7   1
##                                0.882   1   1
##                                0.883  12   6
##                                0.884   7   5
##                                0.885   0   1
##                                0.886   2   1
##                                0.889   0   1
##                                0.893   3   0
##                                0.896   5   1
##                                0.898   7   2
##                                0.899   1   4
##                                0.9     3   3
##                                0.904   5   5
##                                0.905   0   3
##                                0.914   1   0
##                                0.921   0   1
##                                0.937   1   0
##                                0.942   1   0
##                                0.944   1   0
##                                0.953   2   0
##                                0.959   1   0
##                                0.965   1   0
##                                0.977   5   1
```

```
##                                0.982   1   1
##                                0.987   1   0
##                                0.996   1   0
##                                1       1   1
##                                1.016   0   1
##                                1.028   1   0
##                                1.029   4   1
##                                1.03    1   1
##                                1.031   1   0
##                                1.032   0   1
##                                1.035   1   1
##                                1.039   1   2
##                                1.04    1   1
##                                1.041   0   2
##                                1.043   0   1
##                                1.044   1   0
##                                1.046   0   1
##                                1.048   4   0
##                                1.049   1   1
##                                1.05    3   1
##                                1.059   0   1
##                                1.072   1   1
##                                1.085   1   0
##                                1.099   2   0
##                                1.215   1   0
##                                1.224   1   0
##                                1.244  36   5
##                                1.25   48   6
##                                1.252   2   1
##                                1.259   4   1
##                                1.26   13  13
##                                1.262   9   6
##                                1.264   8   3
##                                1.266  65   6
##                                1.268   8   5
##                                1.27   11   4
##                                1.281  59   9
##                                1.291  45   4
##                                1.299  57   2
##                                1.313  47   2
##                                1.327  55   4
##                                1.334  56   2
##                                1.344  52   1
##                                1.354  12   7
##                                1.365  10   5
##                                1.372   2   0
##                                1.384   2   0
##                                1.392   2   1
##                                1.4     0   1
##                                1.405  99   7
```

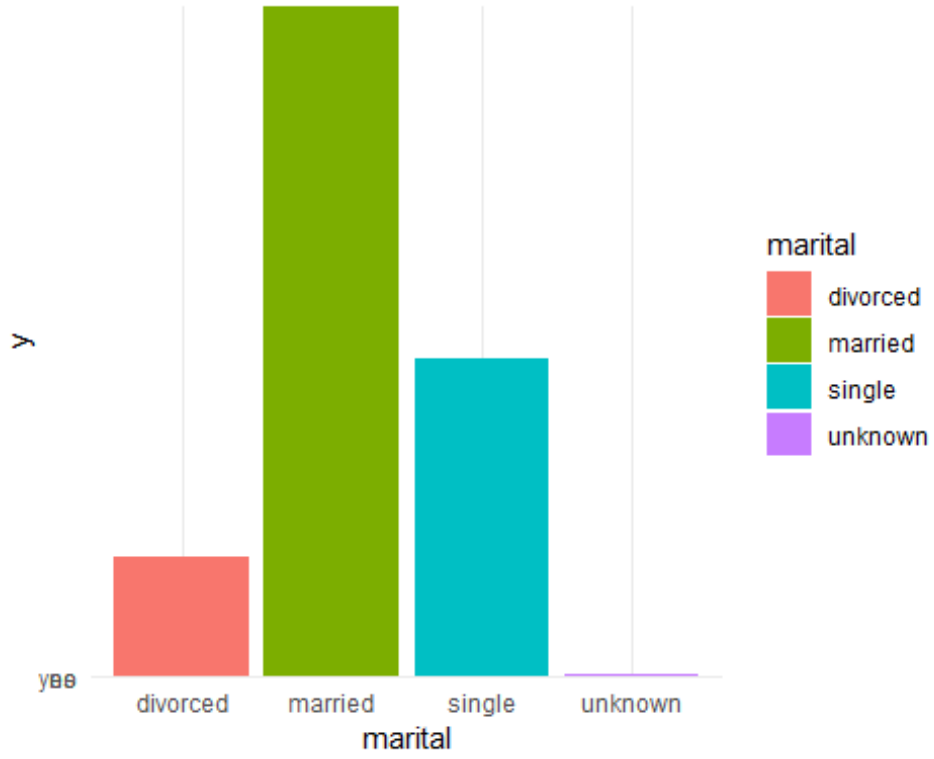
```
##                          1.406   1   2
##                          1.41   22   2
##                          1.415   4   1
##                          1.423   6   2
##                          1.435   4   0
##                          1.445   3   0
##                          1.453   3   2
##                          1.466   6   0
##                          1.479   5   1
##                          1.483   5   1
##                          1.498   0   2
##                          1.51    2   0
##                          1.52    0   1
##                          1.531   0   1
##                          1.538   1   0
##                          1.602   0   1
##                          1.614   1   1
##                          1.629   1   0
##                          1.64    1   1
##                          1.65    0   1
##                          1.663   1   0
##                          1.687   1   0
##                          1.703   1   0
##                          1.726   1   1
##                          1.757   2   1
##                          1.778   0   1
##                          1.799   0   2
##                          1.811   0   2
##                          3.329   1   0
##                          3.853   1   0
##                          4.021  56   5
##                          4.076  98   2
##                          4.12   78   4
##                          4.153  63   2
##                          4.191  70   3
##                          4.245   1   0
##                          4.343   0   1
##                          4.592   1   0
##                          4.663   1   0
##                          4.794   1   1
##                          4.855  78   4
##                          4.856 132   6
##                          4.857 269   5
##                          4.858  61   5
##                          4.859  74   2
##                          4.86   80   0
##                          4.864  90   7
##                          4.865  39   2
##                          4.866  32   3
##                          4.921   0   1
```

```
##                                 4.947   6   0
##                                 4.955   5   1
##                                 4.956   2   0
##                                 4.957  46   5
##                                 4.958  63   3
##                                 4.959  71   5
##                                 4.96   99   6
##                                 4.961 204   8
##                                 4.962 218  19
##                                 4.963 245  11
##                                 4.964 108   2
##                                 4.965 107   7
##                                 4.966  67   5
##                                 4.967  60   2
##                                 4.968  95   6
##                                 4.97   20   1
##                                 5.045   1   0
```

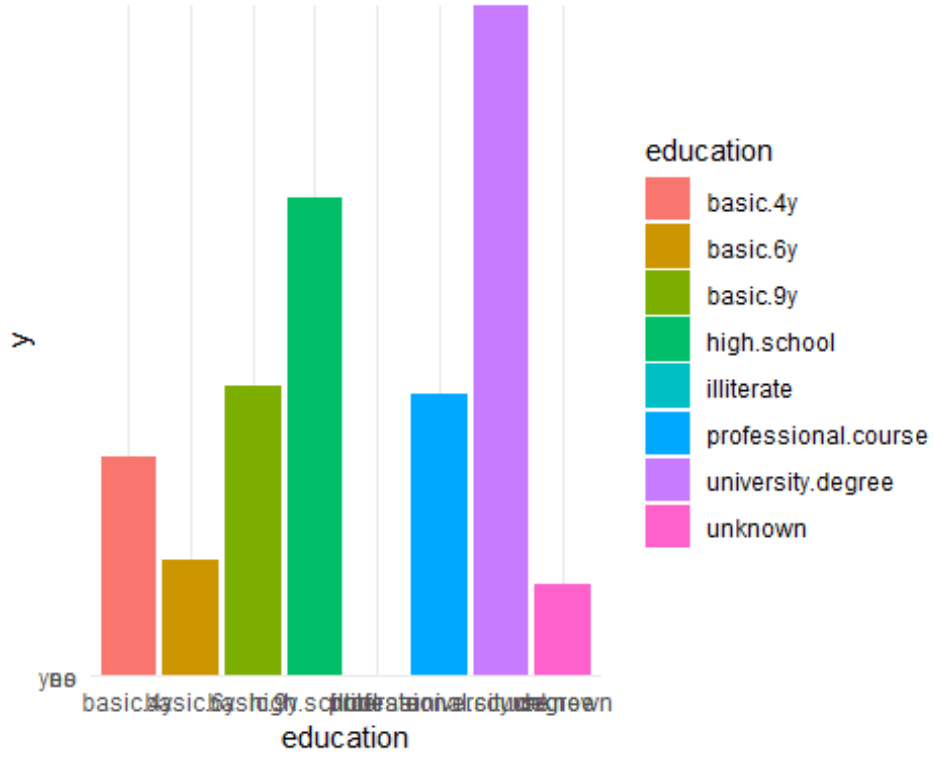
```
table(mybank$nr.employed,mybank$y, dnn = "Frequency of Term Deposits by emplo
yed nb ")
```

```
##                                          NA
## Frequency of Term Deposits by employed nb    no  yes
##                                       4963.6   48   35
##                                       4991.6   42   45
##                                       5008.7   22   38
##                                       5017.5   65   39
##                                       5023.5    9   12
##                                       5076.2  107   57
##                                       5099.1  732   91
##                                       5176.3    1    0
##                                       5191    733   25
##                                       5195.8  372   20
##                                       5228.1 1537   89
```

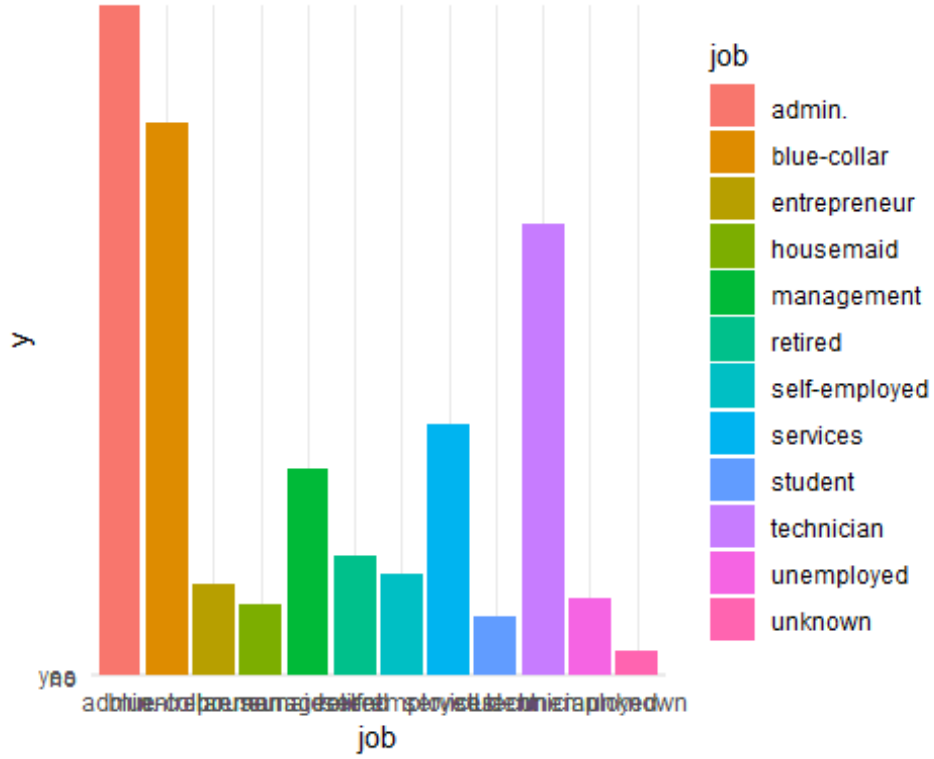
```
p<-ggplot(mybank, aes(x=marital, y=y, fill=marital)) +
  geom_bar(stat="identity")+theme_minimal()
p
```



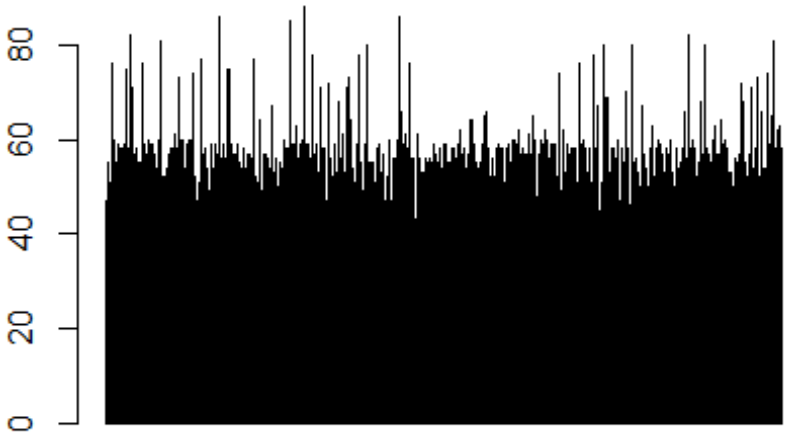

```
p<-ggplot(mybank, aes(x=education, y=y, fill=education)) +
  geom_bar(stat="identity")+theme_minimal()
p
```



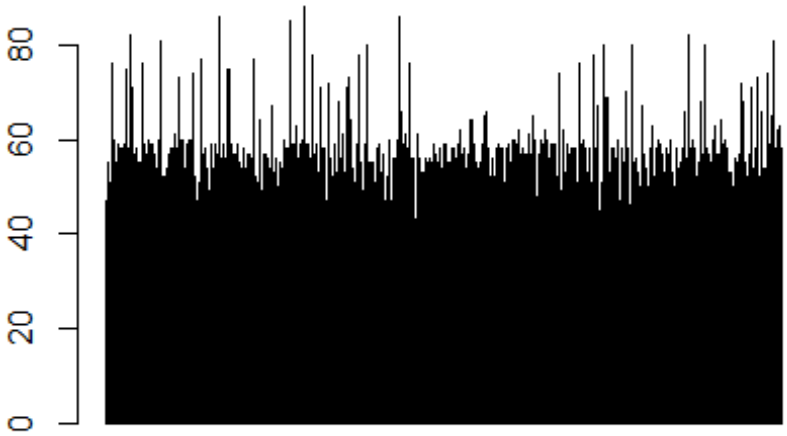
```
p<-ggplot(mybank, aes(x=job, y=y, fill=job )) +
  geom_bar(stat="identity")+theme_minimal()
p
```




```
barplot(mybank$age)
```




```r
barplot(mybank$age)
```



Préparation des données

```
#on voit si nos donnes contiens des valeures null
dim(mybank)

## [1] 4119   21

sum(is.na(mybank))

## [1] 0

#on supprime les donnÃ©es qui n'influence pas notre probleme

mybank <- subset(mybank, select = -c(campaign,pdays,previous,poutcome,emp.var
.rate,cons.price.idx,cons.conf.idx,euribor3m,nr.employed) )

#modifie l'Age de mybank
mybank$age <- as.numeric((cut(mybank$age, c(16,25,35,45,55,65,75,100),labels=
c(1,2,3,4,5,6,7))))

#modifie duration de mybank
mybank$duration=mybank$duration/60

mybank <- na.omit(mybank)
dim(mybank)

## [1] 4119   12

sum(is.na(mybank))

## [1] 0

#ici on change nos donnees a des donnees numerique pour l'utilise avec les me
thodes qui besoin des valeurs numerique (exemple:KNN)

#mybank2: numerique
mybank2=mybank
mybank2$y=as.factor(mybank2$y)
mybank2$default=as.numeric(as.factor(mybank2$default))
mybank2$housing=as.numeric(as.factor(mybank2$housing))
mybank2$loan=as.numeric(as.factor(mybank2$loan))

mybank2$contact=as.numeric(as.character(factor(mybank2$contact,levels=c('tele
phone','cellular','unknown'),labels=c(1,2,3))))

mybank2$month=as.numeric(as.character(factor(mybank2$month, levels=c('jun','f
eb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec','unknown'), l
abels=c(1,2,3,4,5,6,7,8,9,10,11,12,13))))

mybank2$day_of_week=as.numeric(as.factor(mybank2$day_of_week))

mybank2$marital=as.numeric(as.character(factor(mybank2$marital,levels=c('marr
```

```
ied','single','divorced','unknown'),labels=c(1,2,3,4))))

mybank2$education=as.numeric(factor(mybank2$education))
mybank2$job=as.numeric(as.factor(mybank2$job))

dim(mybank2)

## [1] 4119   12

sum(is.na(mybank2))

## [1] 0

mybank2 <- na.omit(mybank2)
dim(mybank2)

## [1] 4119   12

sum(is.na(mybank2))

## [1] 0

# On selectionne un ensemble de donnees train et test.
# Par default est sans remise (replace=FALSE)
set.seed(1)

list <- createDataPartition(mybank$y, p=0.25, list = FALSE)
train=mybank[-list, ]
test=mybank[list, ]

# On selectionne un ensemble de donnees train2 et test2 pour les donnÃ©es num
eriques
# Par default est sans remise (replace=FALSE)
set.seed(1)

list2 <- createDataPartition(mybank2$y, p=0.25, list = FALSE)
train2=mybank2[-list2, ]
test2=mybank2[list2, ]
```

Classification

KNN

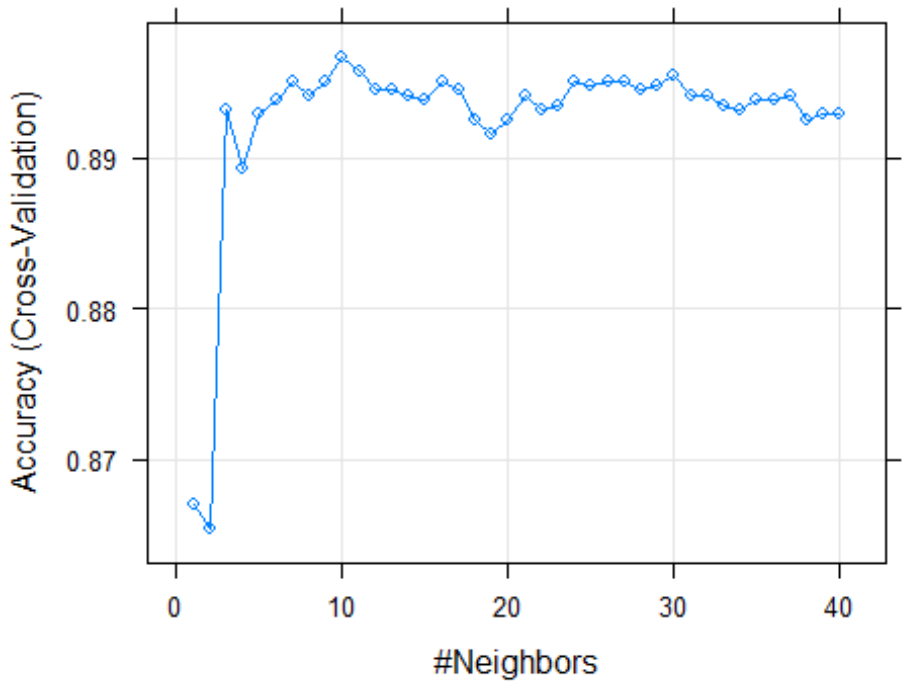
```
set.seed(1)



TrainingParameters <- trainControl(method  = "cv",
                           number  = 10,
                           classProbs=TRUE
```

```
                          )

fit.KNN <- train(y ~ .,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:40),
             trControl  = TrainingParameters,
             preProcess = c("center","scale"),
             data       = train2)

fit.KNN
```

```
## k-Nearest Neighbors
##
## 3089 samples
##   11 predictor
##    2 classes: 'no', 'yes'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2780, 2780, 2780, 2779, 2780, 2781, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.8669480  0.26418133
##    2  0.8653309  0.24412524
##    3  0.8931763  0.27444173
##    4  0.8892876  0.24113456
##    5  0.8928527  0.21788827
##    6  0.8938204  0.21945231
##    7  0.8951191  0.19802977
##    8  0.8941504  0.18334215
##    9  0.8951181  0.17598542
##   10  0.8967362  0.18316062
##   11  0.8957643  0.17128347
##   12  0.8944687  0.16381212
##   13  0.8944729  0.15912548
##   14  0.8941472  0.15050475
##   15  0.8938194  0.14679363
##   16  0.8951160  0.15344930
##   17  0.8944698  0.14487370
##   18  0.8925259  0.12248617
##   19  0.8915551  0.10887409
##   20  0.8925291  0.12281443
##   21  0.8941472  0.13790502
##   22  0.8931763  0.12401895
##   23  0.8934989  0.13292377
##   24  0.8951191  0.13959043
##   25  0.8947934  0.13164890
##   26  0.8951170  0.13207936
##   27  0.8951191  0.13098419
```

```
##    28  0.8944708  0.12332634
##    29  0.8947955  0.12248765
##    30  0.8954417  0.12448142
##    31  0.8941472  0.11091047
##    32  0.8941451  0.11021782
##    33  0.8934989  0.10581289
##    34  0.8931753  0.10072788
##    35  0.8938225  0.10619913
##    36  0.8938215  0.10650554
##    37  0.8941451  0.11091815
##    38  0.8925280  0.08778779
##    39  0.8928496  0.09302040
##    40  0.8928506  0.09301904
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 10.
```

```
plot(fit.KNN)
```




```
KNNPredictions <-predict(fit.KNN, test2)
cmKNN=confusionMatrix(KNNPredictions, test2$y)
cmKNN
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
```

```
##        no  908 103
##        yes   9  10
##
##                Accuracy : 0.8913
##                  95% CI : (0.8706, 0.9096)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.4853
##
##                   Kappa : 0.1238
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9902
##             Specificity : 0.0885
##          Pos Pred Value : 0.8981
##          Neg Pred Value : 0.5263
##              Prevalence : 0.8903
##          Detection Rate : 0.8816
##    Detection Prevalence : 0.9816
##       Balanced Accuracy : 0.5393
##
##        'Positive' Class : no
##
```

```
errorknn = mean(KNNPredictions != test2$y)

cat(sprintf("Taux d'erreur test est: %s",errorknn))
```

```
## Taux d'erreur test est: 0.10873786407767
```

Logistic Classification

```
library(boot)

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

model1<-glm(y~.,data = train2,family = binomial)
a=model1$model[1]
model1$model<- subset(model1$model, select = -c(y))
model1$model=c(model1$model,a)
cv.fit=cv.glm(train2, model1, K=10)
summary(model1)

##
## Call:
## glm(formula = y ~ ., family = binomial, data = train2)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.0941  -0.4178  -0.3120  -0.2074   2.8564
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.5693544  0.5777428  -9.640  < 2e-16 ***
## age          0.2179651  0.0595760   3.659 0.000254 ***
## job          0.0228331  0.0183224   1.246 0.212696
## marital     -0.0140080  0.0942078  -0.149 0.881796
## education    0.0908327  0.0332895   2.729 0.006361 **
## default     -0.6562196  0.2065497  -3.177 0.001488 **
## housing     -0.0157769  0.0676742  -0.233 0.815661
## loan         0.0008027  0.0899639   0.009 0.992881
## contact      1.3325336  0.1849586   7.204 5.83e-13 ***
## month       -0.1054679  0.0264572  -3.986 6.71e-05 ***
## day_of_week  0.0017994  0.0479010   0.038 0.970035
## duration     0.2426784  0.0137597  17.637  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2133.3  on 3088  degrees of freedom
## Residual deviance: 1629.2  on 3077  degrees of freedom
## AIC: 1653.2
##
## Number of Fisher Scoring iterations: 6
```

```
## Prediction for glm model
prediction1<-predict(model1,test2,type="response")
prediction3<-round(prediction1,0)
prediction3<-as.factor(prediction3)
levels(prediction3)<-c("no","yes")
actual1<-test2[,12]
levels(actual1)<-c("no","yes")
confglm<-confusionMatrix(prediction3, test2$y)
confglm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  907  94
##        yes  10  19
##
##                Accuracy : 0.899
##                  95% CI : (0.879, 0.9168)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.1994
```

```
##
##                   Kappa : 0.2332
##  Mcnemar's Test P-Value : 3.992e-16
##
##             Sensitivity : 0.9891
##             Specificity : 0.1681
##          Pos Pred Value : 0.9061
##          Neg Pred Value : 0.6552
##              Prevalence : 0.8903
##          Detection Rate : 0.8806
##    Detection Prevalence : 0.9718
##       Balanced Accuracy : 0.5786
##
##        'Positive' Class : no
##
```

```
errorLOG = mean(prediction3 != test2$y)
cat(sprintf("Taux dâ□□erreur test est: %s",errorLOG))
```

```
## Taux dâ□□erreur test est: 0.100970873786408
```

LDA

```
set.seed(1)
my.fld = createFolds(train2$y, k=5)
error.lda <- sapply(seq_along(my.fld), function(i) {
    model.lda = lda(y ~., data=train2[-my.fld[[i]], ])
    model.lda$model=c(model.lda$model,a)
    pred.lda = predict(model.lda, train2[my.fld[[i]], ])
    mean(pred.lda$class != train2$y[my.fld[[i]]] )
})
cat(sprintf("\nCV Mean Error Rate: %s\n",mean(error.lda)))
```

```
##
## CV Mean Error Rate: 0.0987412651088695
```

```
model.lda = lda(y ~ ., data=train2)
pred.lda = predict(model.lda, test2)
error.test.lda=mean(pred.lda$class != test2$y)
cat(sprintf("\n\n The classification error for test data is : %s\n\n",
error.test.lda))
```

```
##
##
##  The classification error for test data is : 0.102912621359223
```

```
cmLDA=confusionMatrix(pred.lda$class,test2$y)
cmLDA
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  no yes
##        no  896  85
##        yes  21  28
##
##                Accuracy : 0.8971
##                  95% CI : (0.8769, 0.915)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.2609
##
##                   Kappa : 0.2992
##  Mcnemar's Test P-Value : 9.41e-10
##
##             Sensitivity : 0.9771
##             Specificity : 0.2478
##          Pos Pred Value : 0.9134
##          Neg Pred Value : 0.5714
##              Prevalence : 0.8903
##          Detection Rate : 0.8699
##    Detection Prevalence : 0.9524
##       Balanced Accuracy : 0.6124
##
##        'Positive' Class : no
##
```

```
cat(sprintf("Taux dâ□□erreur test est: %s",error.test.lda))
```

```
## Taux dâ□□erreur test est: 0.102912621359223
```

QDA

```
set.seed(1)
my.fld = createFolds(train2$y, k=5)
error.qda <- sapply(seq_along(my.fld), function(i) {
    model.qda = qda(y ~., data=train2[-my.fld[[i]], ])
    model.qda$model=c(model.qda$model,a)
    pred.qda = predict(model.qda, train2[my.fld[[i]], ])
    mean(pred.qda$class != train2$y[my.fld[[i]]] )
})

cat(sprintf("\nCV Mean Error Rate: %s\n",mean(error.qda)))
```

```
##
## CV Mean Error Rate: 0.104241321555463
```

```
model.QDA = qda(y ~ ., data=train2)
pred.QDA = predict(model.QDA, test2)
error.test.QDA=mean(pred.QDA$class != test2$y)
cat(sprintf("\n\n The classification error for test data is : %s\n\n",
error.test.QDA))
```

```
##
##
##  The classification error for test data is : 0.102912621359223
```

```
cmQDA=confusionMatrix(pred.QDA$class,test2$y)
cmQDA
```

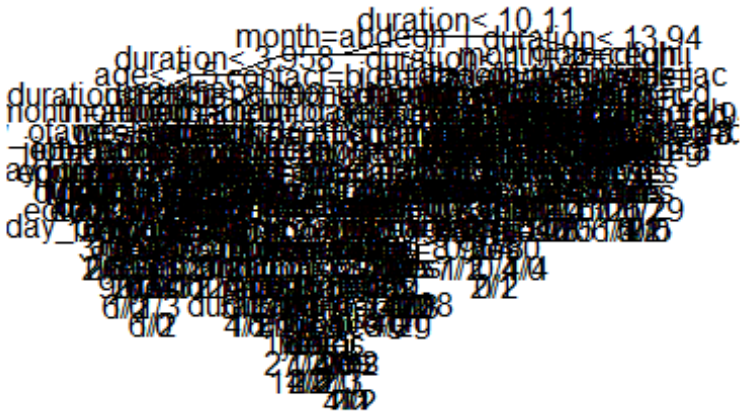
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  890  79
##        yes  27  34
##
##                Accuracy : 0.8971
##                  95% CI : (0.8769, 0.915)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.2609
##
##                   Kappa : 0.34
##  Mcnemar's Test P-Value : 7.287e-07
##
##             Sensitivity : 0.9706
##             Specificity : 0.3009
##          Pos Pred Value : 0.9185
##          Neg Pred Value : 0.5574
##              Prevalence : 0.8903
##          Detection Rate : 0.8641
##    Detection Prevalence : 0.9408
##       Balanced Accuracy : 0.6357
##
##        'Positive' Class : no
##
```

```
cat(sprintf("Taux dâ□□erreur test est: %s",error.test.QDA))
```

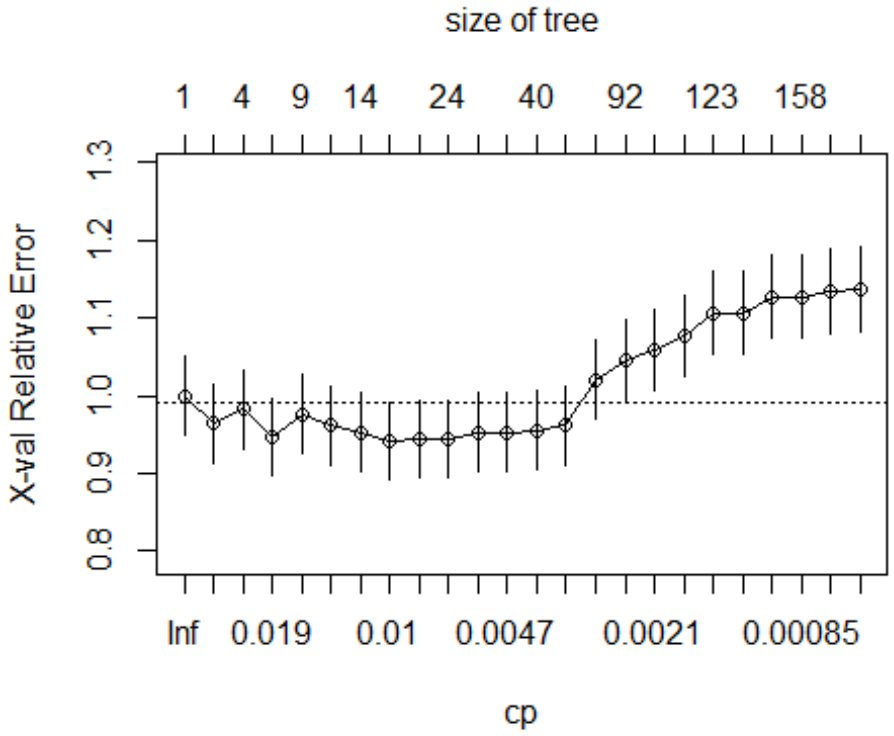
```
## Taux dâ□□erreur test est: 0.102912621359223
```

Arbre de décision

```
set.seed(1)

#Construction de l'arbre
ptitanic.Tree <- rpart(train$y~.,data=train,method="class", control=rpart.con
trol(minsplit=5,cp=0))

#Affichage du résultat
plot(ptitanic.Tree, uniform=TRUE, branch=0.5, margin=0.1)
text(ptitanic.Tree, all=FALSE, use.n=TRUE)
```

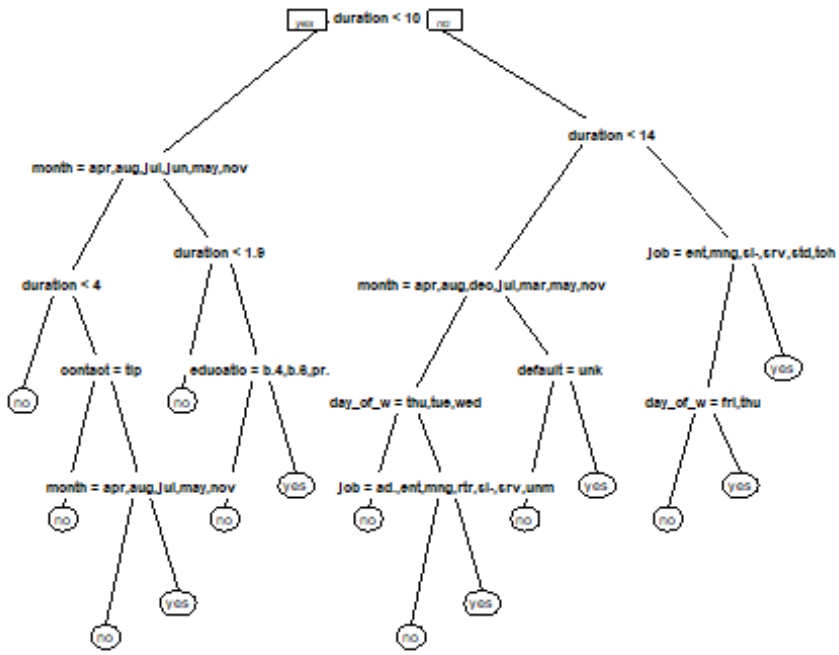



```
#On cherche a minimiser l'erreur pour définir le niveau d'élagage
plotcp(ptitanic.Tree)
```



Le graphique ci-dessus affiche le taux de mauvais classement en fonction de la taille de l'arbre. On cherche a minimiser l'erreur.

```
#Affichage du cp optimal
print(ptitanic.Tree$cptable[which.min(ptitanic.Tree$cptable[,4]),1])

## [1] 0.00887574

ptitanic.Tree_Opt <- prune(ptitanic.Tree,cp=ptitanic.Tree$cptable[which.min(p
titanic.Tree$cptable[,4]),1])

#Représentation graphique de l'arbre optimal
prp(ptitanic.Tree_Opt)
```




```
pred.tree = predict(ptitanic.Tree_Opt,test,type="class")


cmAR=confusionMatrix(pred.tree, as.factor(test$y))
cmAR

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  886  69
##        yes  31  44
##
```

```
##                Accuracy : 0.9029
##                  95% CI : (0.8832, 0.9203)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.1049001
##
##                   Kappa : 0.4171
##  Mcnemar's Test P-Value : 0.0002156
##
##             Sensitivity : 0.9662
##             Specificity : 0.3894
##          Pos Pred Value : 0.9277
##          Neg Pred Value : 0.5867
##              Prevalence : 0.8903
##          Detection Rate : 0.8602
##    Detection Prevalence : 0.9272
##       Balanced Accuracy : 0.6778
##
##        'Positive' Class : no
##
```

```
errortree = mean(pred.tree != test$y)
cat(sprintf("Taux d'erreur test est: %s",errortree))
```

```
## Taux d'erreur test est: 0.0970873786407767
```

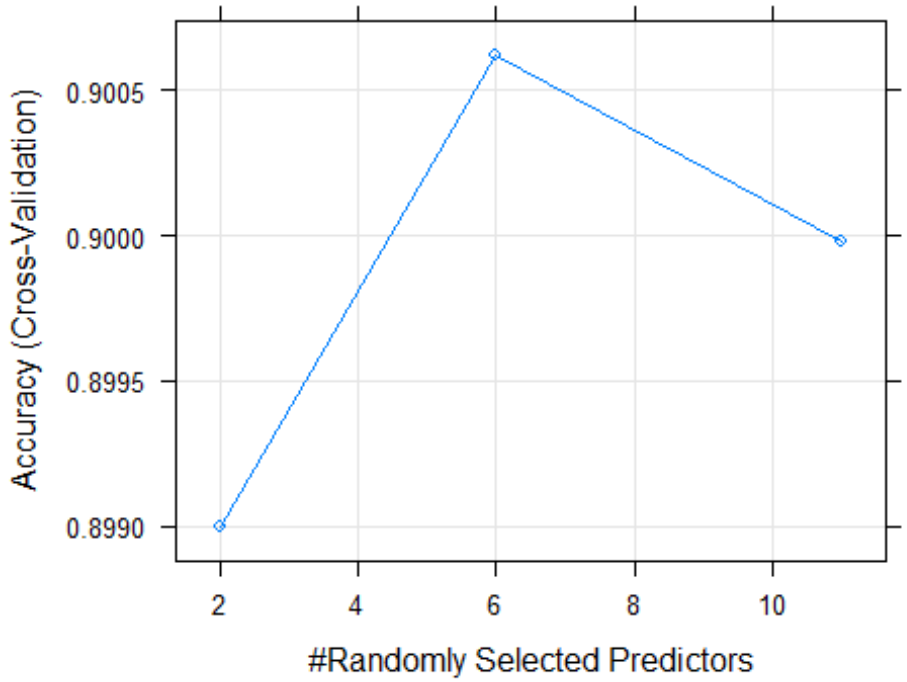
Random forest

```
set.seed(1)

TrainingParameters <- trainControl(method  = "cv",
                           number  = 10)

fitRF <- train(as.data.frame(train2[,-12]),train2$y,
             method     = "rf",
             trControl  = TrainingParameters,
             preProcess = c("center","scale")
    )


fitRF
```

```
## Random Forest
##
## 3089 samples
##   11 predictor
##    2 classes: 'no', 'yes'
##
## Pre-processing: centered (11), scaled (11)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2780, 2780, 2780, 2779, 2780, 2781, ...
```

```
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8989963  0.2420698
##    6    0.9006218  0.3893935
##   11    0.8999777  0.3848339
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

```
plot(fitRF)
```




```
RFPredictions <-predict(fitRF, test2)


cmRF=confusionMatrix(as.factor(RFPredictions), as.factor(test2$y))
cmRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  888  81
##        yes  29  32
##
##                Accuracy : 0.8932
##                  95% CI : (0.8727, 0.9114)
```

```
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.4063
##
##                   Kappa : 0.3151
##  Mcnemar's Test P-Value : 1.158e-06
##
##             Sensitivity : 0.9684
##             Specificity : 0.2832
##          Pos Pred Value : 0.9164
##          Neg Pred Value : 0.5246
##              Prevalence : 0.8903
##          Detection Rate : 0.8621
##    Detection Prevalence : 0.9408
##       Balanced Accuracy : 0.6258
##
##        'Positive' Class : no
##
```

```
errorRF = mean(RFPredictions != test2$y)
cat(sprintf("Taux d'erreur test est: %s",errorRF))
```

```
## Taux d'erreur test est: 0.106796116504854
```

Naïves de bayes

```
#Naive algorithm
set.seed(1)

TrainingParameters <- trainControl(method = "repeatedcv", number = 10, repeat
s=10)
NaiveModel <- train(as.data.frame(train[,-12]), train$y,
                    method = "nb",
                    preProcess=c("scale","center"),
                    trControl= TrainingParameters

                   )

NaivePredictions <-predict(NaiveModel, test)



cmNaive <-confusionMatrix(NaivePredictions, as.factor(test$y))

cmNaive
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  889  81
##        yes  28  32
```

```
##
##                Accuracy : 0.8942
##                  95% CI : (0.8738, 0.9123)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.3679
##
##                   Kappa : 0.318
##  Mcnemar's Test P-Value : 6.336e-07
##
##             Sensitivity : 0.9695
##             Specificity : 0.2832
##          Pos Pred Value : 0.9165
##          Neg Pred Value : 0.5333
##              Prevalence : 0.8903
##          Detection Rate : 0.8631
##    Detection Prevalence : 0.9417
##       Balanced Accuracy : 0.6263
##
##        'Positive' Class : no
##
```

```
errorNB = (mean(NaivePredictions != test$y))
cat(sprintf("Taux d'erreur test est: %s",errorNB ))
```

```
## Taux d'erreur test est: 0.105825242718447
```

SVM

```
set.seed(1)


# Create Training Data


TrainingParameters <- trainControl(method = "repeatedcv", number = 10, repeat
s=10)
SVModel <- train(y ~ ., data = train,
                 method = "svmPoly",
                 trControl= TrainingParameters,
                 tuneGrid = data.frame(degree = 1,
                                       scale = 1,
                                       C = 1),
                 preProcess = c("pca","scale","center"),
                 na.action = na.omit
)

SVMPredictions <-predict(SVModel, test)
# Create confusion matrix
cmSVM <-confusionMatrix(SVMPredictions,as.factor(test$y))
cmSVM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  907  96
##        yes  10  17
##
##                Accuracy : 0.8971
##                  95% CI : (0.8769, 0.915)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.2609
##
##                   Kappa : 0.2094
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9891
##             Specificity : 0.1504
##          Pos Pred Value : 0.9043
##          Neg Pred Value : 0.6296
##              Prevalence : 0.8903
##          Detection Rate : 0.8806
##    Detection Prevalence : 0.9738
##       Balanced Accuracy : 0.5698
##
##        'Positive' Class : no
##
```

```
errorSVM = (mean(SVMPredictions != test$y))
cat(sprintf("Taux d'erreur test est: %s",errorSVM))
```

```
## Taux d'erreur test est: 0.102912621359223
```

Réseau de neurons(RN)

```
set.seed(1)


TrainingParameters <- trainControl(method = "repeatedcv", number = 10, repeat
s=10)
# train model with neural networks
NNModel <- train(train[,-12], train$y,
                 method = "nnet",
                 trControl= TrainingParameters,
                 preProcess=c("scale","center"),
                 na.action = na.omit
)
```

```
## # weights:  46
## initial  value 2156.633985
## iter  10 value 693.483389
```

```
## iter  20 value 611.794364
## iter  30 value 591.821663
## iter  40 value 580.539974
## iter  50 value 576.221486
## iter  60 value 568.118893
## iter  70 value 554.428434
## iter  80 value 553.346621
## iter  90 value 553.116892
## iter 100 value 553.111512
## final  value 553.111512
## stopped after 100 iterations
.

.

.
## # weights:  46
## initial  value 1659.290251
## iter  10 value 726.930934
## iter  20 value 693.006046
## iter  30 value 668.939225
## iter  40 value 662.601860
## iter  50 value 658.288423
## iter  60 value 657.871795
## final  value 657.870709
## converged
```

```
NNPredictions <-predict(NNModel, test)
# Create confusion matrix
cmNN <-confusionMatrix(NNPredictions, as.factor(test$y))
cmNN
```

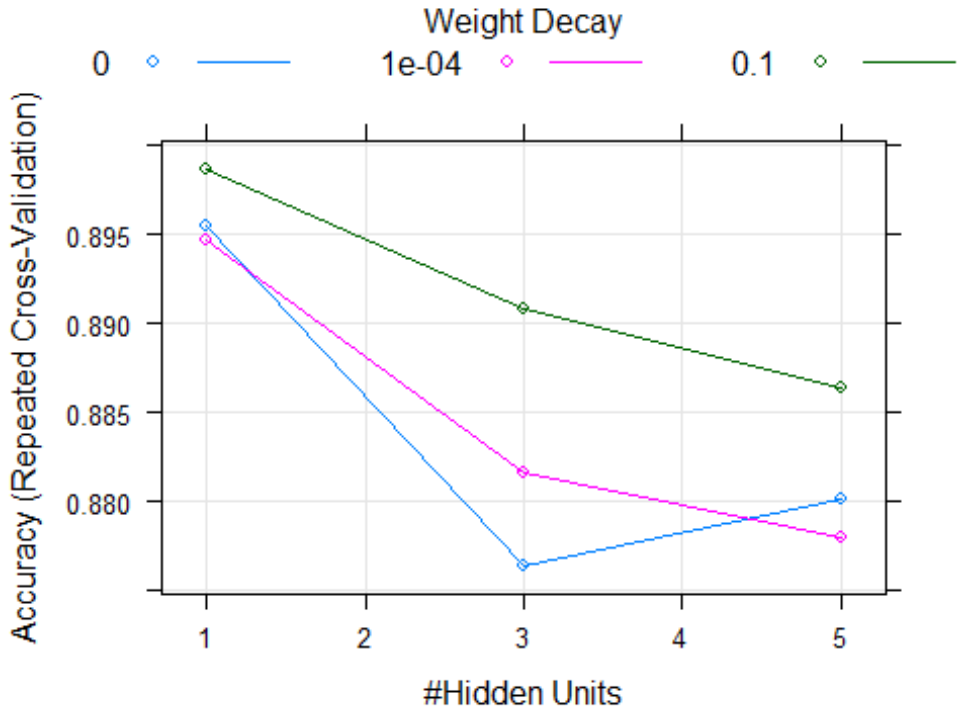
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  879  65
##        yes  38  48
##
##                Accuracy : 0.9
##                  95% CI : (0.88, 0.9176)
##     No Information Rate : 0.8903
##     P-Value [Acc > NIR] : 0.17207
##
##                   Kappa : 0.4282
##  Mcnemar's Test P-Value : 0.01041
##
##             Sensitivity : 0.9586
##             Specificity : 0.4248
##          Pos Pred Value : 0.9311
##          Neg Pred Value : 0.5581
```

```
##              Prevalence : 0.8903
##          Detection Rate : 0.8534
##    Detection Prevalence : 0.9165
##       Balanced Accuracy : 0.6917
##
##        'Positive' Class : no
##
```

```
errorRN = mean(NNPredictions != test$y)
cat(sprintf("Taux dâ erreur test est: %s",errorRN))
```

```
## Taux dâ erreur test est: 0.1
```

```
plot(NNModel)
```

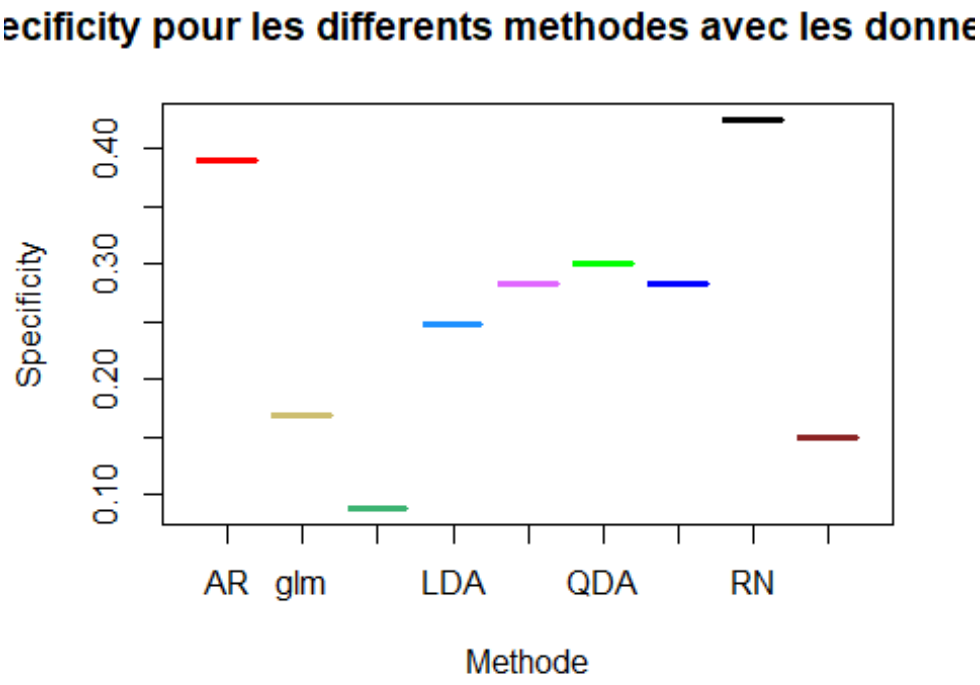


`

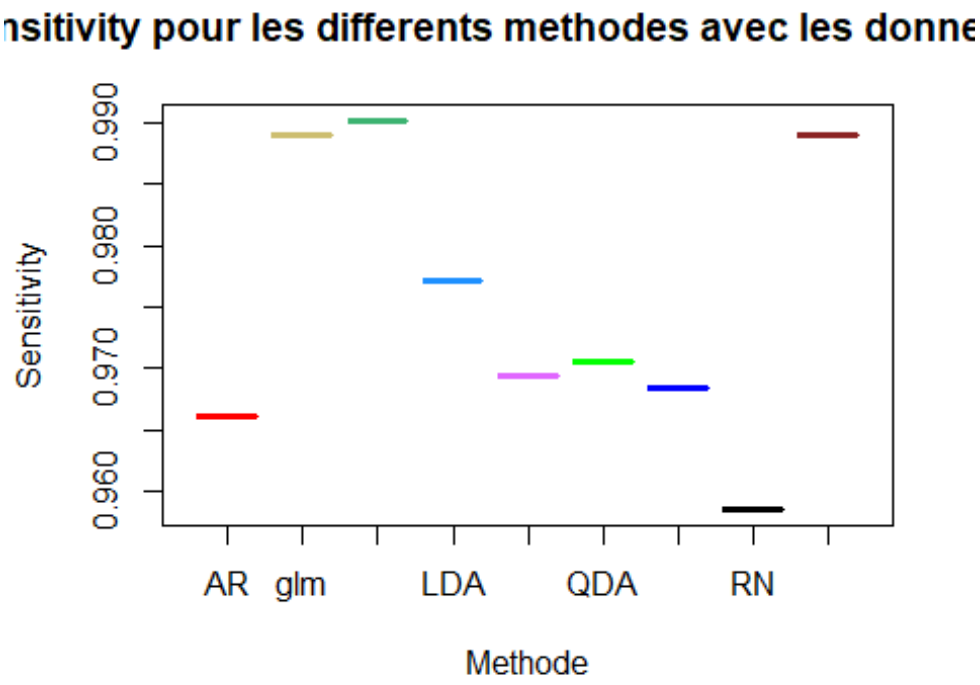
Résumé graphique et comparaison des méthodes

```
# plot(Specificity)
Specificity=c(cmSVM$byClass['Specificity'],cmNN$byClass['Specificity'] ,cmNaive$byClass['Specificity'],cmRF$byClass['Specificity'] ,cmAR$byClass['Specificity'],cmQDA$byClass['Specificity'],cmLDA$byClass['Specificity'],confglm$byClass['Specificity'],cmKNN$byClass['Specificity'])

axex<-as.factor(c("SVM","RN","NB","RF","AR","QDA","LDA","glm","KNN")) # Donnees test
```

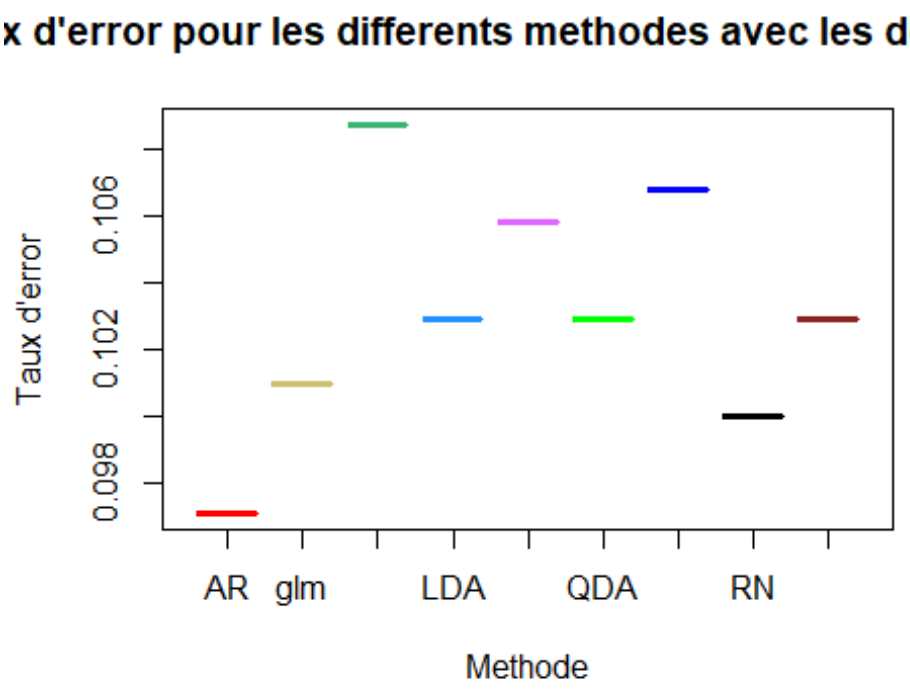
```
plot(axex,Specificity,col="blue",xlab="Methode", ylab="Specificity", main="Sp
ecificity pour les differents methodes avec les donnees test",border=c ("red"
,"lightgoldenrod3", "mediumseagreen",
"dodgerblue", "mediumorchid1", "green", "blue",
 "black","brown4"))
```




```
# plot(Sensitivity)
Sensitivity=c(cmSVM$byClass['Sensitivity'],cmNN$byClass['Sensitivity'] ,cmNai
ve$byClass['Sensitivity'],cmRF$byClass['Sensitivity'] ,cmAR$byClass['Sensitiv
ity'],cmQDA$byClass['Sensitivity'],cmLDA$byClass['Sensitivity'],confglm$byCla
ss['Sensitivity'],cmKNN$byClass['Sensitivity'])

axex<-as.factor(c("SVM","RN","NB","RF","AR","QDA","LDA","glm","KNN")) # Donne
es test

plot(axex,Sensitivity,col="blue",xlab="Methode", ylab="Sensitivity", main="Se
nsitivity pour les differents methodes avec les donnees test",border=c ("red"
,"lightgoldenrod3", "mediumseagreen",
"dodgerblue", "mediumorchid1", "green", "blue",
 "black","brown4"))
```




```
# plot(Sensitivity)
MSE=c(errorSVM ,errorRN,errorNB ,errorRF,errortree,error.test.QDA,error.test.
lda,errorLOG,errorknn )


axex<-as.factor(c("SVM","RN","NB","RF","AR","QDA","LDA","glm","KNN")) # Donne
es test

plot(axex,MSE,col="blue",xlab="Methode", ylab="Taux d'error", main="Taux d'er
ror pour les differents methodes avec les donnees test",border=c ("red","ligh
tgoldenrod3", "mediumseagreen",
"dodgerblue", "mediumorchid1", "green", "blue","black","brown4" ) )
```




```
 y=as.numeric(train2$y)
lda=as.factor(pred.lda$class)
qda=as.factor(pred.QDA$class)
knn <- as.numeric(KNNPredictions)
log <- as.numeric(prediction3)
lda <- lda
qda<- qda
tree <- as.numeric(pred.tree)
RN <-as.numeric( NNPredictions)
Nb  <-as.numeric( NaivePredictions)
SVM <- as.numeric(KNNPredictions)
RF <-as.numeric(RFPredictions)
mat <- matrix(c(knn),nrow=length(knn))
(mat.app <- cbind(mat,log,lda,qda,tree,RF,Nb,SVM,RN))

cor(mat.app)

##                      log       lda       qda      tree        RF        Nb
##      1.0000000 0.5873226 0.4778062 0.4241148 0.3503415 0.3935474 0.4279823
## log  0.5873226 1.0000000 0.7615851 0.6783892 0.4040724 0.4546012 0.6843720
## lda  0.4778062 0.7615851 1.0000000 0.8907596 0.4288763 0.5236198 0.8596683
## qda  0.4241148 0.6783892 0.8907596 1.0000000 0.4520742 0.5643641 0.7980872
## tree 0.3503415 0.4040724 0.4288763 0.4520742 1.0000000 0.6420330 0.4407995
## RF   0.3935474 0.4546012 0.5236198 0.5643641 0.6420330 1.0000000 0.5171114
## Nb   0.4279823 0.6843720 0.8596683 0.7980872 0.4407995 0.5171114 1.0000000
## SVM  1.0000000 0.5873226 0.4778062 0.4241148 0.3503415 0.3935474 0.4279823
## RN   0.3759429 0.5639216 0.6745056 0.7123154 0.6177967 0.6082340 0.6741543
```

```
##            SVM        RN
##      1.0000000 0.3759429
## log  0.5873226 0.5639216
## lda  0.4778062 0.6745056
## qda  0.4241148 0.7123154
## tree 0.3503415 0.6177967
## RF   0.3935474 0.6082340
## Nb   0.4279823 0.6741543
## SVM  1.0000000 0.3759429
## RN   0.3759429 1.0000000
```