Analyse des Ventes Automobiles avec le Boosting

Kenson FAVEUR

2025-03-25

Introduction

Ce rapport présente une analyse des ventes de sièges pour enfants dans 400 magasins différents aux Etats-Unis, en utilisant des méthodes de Boosting et de Forets Aléatoires. Le but est de prédire si un magasin aura de fortes ventes en fonction de plusieurs variables explicatives.

Charger les librairies nécessaires

Dans la première partie, nous commencons par le téléchargement des librairies nécessaires.

```
# Charger les librairies
library(ISLR2)
library(gbm)
library(randomForest)
library(caret)
```

Chargement des données et définition de la variable cible.

Dans cette section nous commencons par charger les données et définir la variable 'High' qui indiquera si les ventes ('Sales') d'un magasin sont supérieures à un seuil de 8 milliers d'unités.

Charger les données et définir la variable à expliquer

Effectuer une analyse préliminaire des données

Une analyse préliminaire des données permet de comprendre la structure du jeu de données et d'explorer la distribution de la variable Sales.

```
# Analyse préliminaire des données.
summary(Carseats)
```

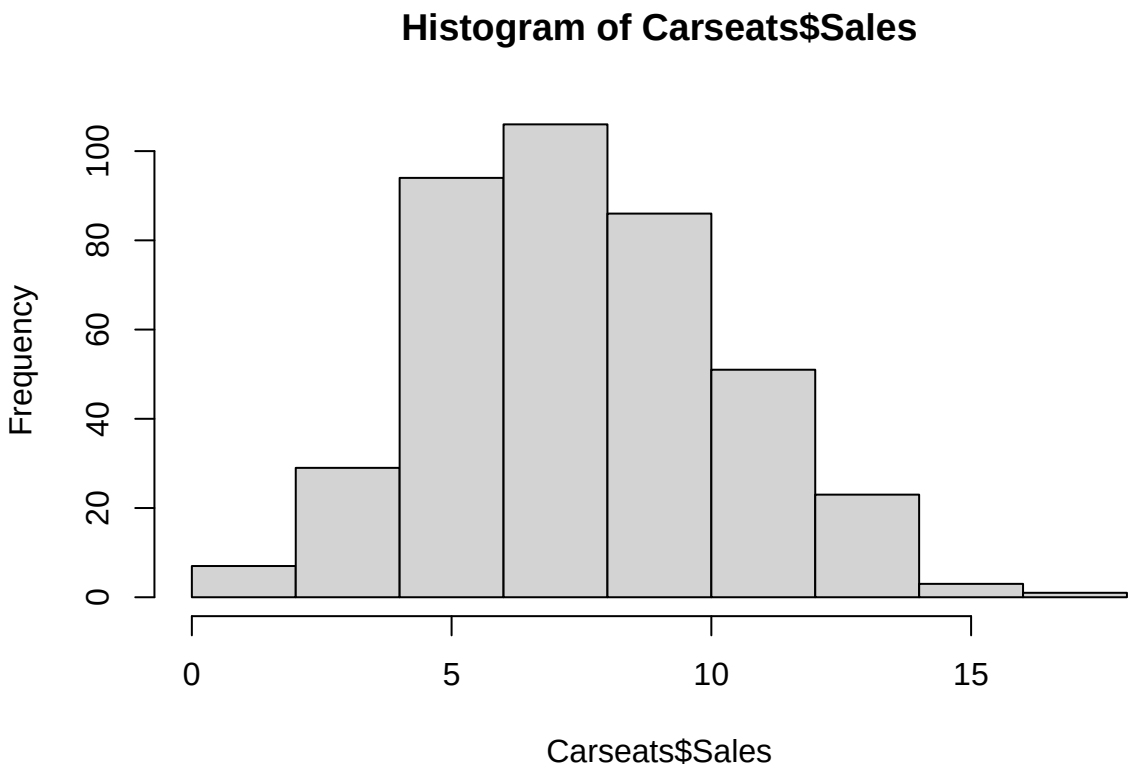
```
##      Sales          CompPrice       Income        Advertising
##  Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000
##  1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000
##  Median : 7.490   Median :125   Median : 69.00   Median : 5.000
##  Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635
```

```
##  3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000
##  Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000
##    Population        Price        ShelveLoc        Age          Education
##  Min.   : 10.0   Min.   : 24.0   Bad   : 96   Min.   :25.00   Min.   :10.0
##  1st Qu.:139.0   1st Qu.:100.0   Good  : 85   1st Qu.:39.75   1st Qu.:12.0
##  Median :272.0   Median :117.0   Medium:219   Median :54.50   Median :14.0
##  Mean   :264.8   Mean   :115.8                Mean   :53.32   Mean   :13.9
##  3rd Qu.:398.5   3rd Qu.:131.0                3rd Qu.:66.00   3rd Qu.:16.0
##  Max.   :509.0   Max.   :191.0                Max.   :80.00   Max.   :18.0
##  Urban       US       High
##  No :118   No :142   No :236
##  Yes:282   Yes:258   Yes:164
##
##
##
##
```

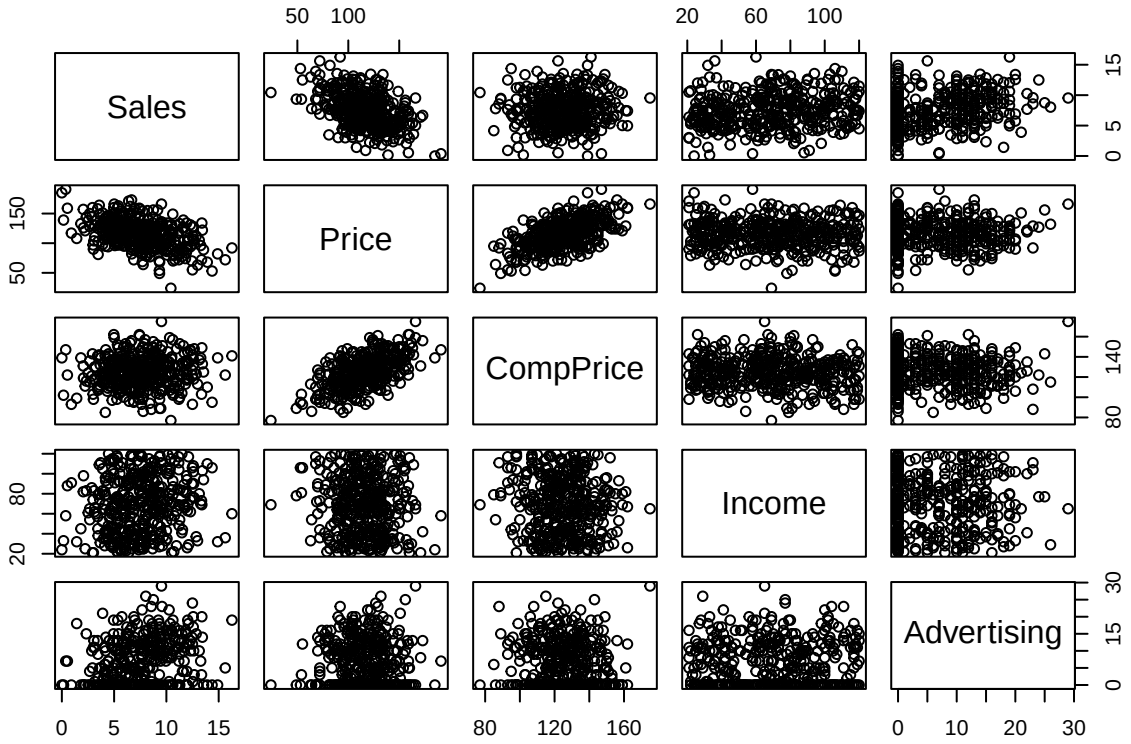
```
str(Carseats)
```

```
## 'data.frame':    400 obs. of  12 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
##  $ High       : Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 1 2 1 1 ...
```

```
hist(Carseats$Sales)
```




```
pairs(Carseats[, c("Sales", "Price", "CompPrice", "Income", "Advertising")])
```



Interprétation des résultats

Analyse des variables numériques

Ventes (Sales) Min = 0, Max = 16.27, Médiane = 7.49, Moyenne = 7.496

Les ventes varient considérablement entre 0 et 16,27 unités vendues.

La médiane et la moyenne sont presque identiques (~7.5), suggérant une distribution relativement symétrique.

25% des magasins vendent moins de 5.39 unités, et 25% en vendent plus de 9.32 unités.

Prix Concurrent (CompPrice) Min = 77, Max = 175, Médiane = 125, Moyenne = 125

Le prix des concurrents est concentré autour de 125, avec une faible dispersion.

Les 25% des prix les plus bas sont inférieurs à 115 et les plus hauts dépassent 135.

Revenu Moyenne des Clients (Income) Min = 21, Max = 120, Médiane = 69, Moyenne = 68.66

Le revenu médian des clients est de 69, ce qui signifie que la moitié des clients ont un revenu inférieur et l'autre moitié un revenu supérieur.

Il existe une forte dispersion des revenus (de 21 à 120), ce qui pourrait influencer les décisions d'achat.

Dépenses en Publicité (Advertising) Min = 0, Max = 29, Médiane = 5, Moyenne = 6.635

La plupart des magasins investissent peu en publicité (50% dépensent 5 ou moins).

Cependant, quelques magasins investissent jusqu'à 29, ce qui pourrait influencer fortement leurs ventes.

Population de la Zone (Population) Min = 10, Max = 509, Médiane = 272, Moyenne = 264.8

La médiane étant proche de la moyenne, la distribution semble équilibrée.

Certaines zones ont une population très faible (~10), tandis que d'autres dépassent les 500.

Prix du Produit (Price) Min = 24, Max = 191, Médiane = 117, Moyenne = 115.8

Les prix sont assez variables, avec un écart important entre les valeurs minimales et maximales.

Une analyse plus approfondie pourrait révéler si le prix a une influence directe sur les ventes.

Âge Moyen des Clients (Age) Min = 25, Max = 80, Médiane = 54.5, Moyenne = 53.32

L'âge des clients varie largement entre 25 et 80 ans, avec un âge médian autour de 54.

Cela suggère que la clientèle cible pourrait être des parents plus âgés.

Niveau d'Éducation (Education) Min = 10, Max = 18, Médiane = 14, Moyenne = 13.9

L'éducation moyenne est de 14 ans, ce qui correspond généralement à un niveau postsecondaire.

Analyse des variables catégoriques

Emplacement en Rayon (ShelveLoc) 96 magasins (Bad), 85 (Good), 219 (Medium)

La majorité des magasins classent les sièges auto en rayon Medium, ce qui pourrait influencer négativement ou positivement les ventes.

Magasin en Zone Urbaine ou Rurale (Urban) 118 en zone rurale, 282 en zone urbaine

Une majorité des magasins sont situés en zone urbaine, ce qui pourrait indiquer une corrélation entre la population et les ventes.

Magasins aux États-Unis (US) 142 hors USA, 258 aux USA

La majorité des magasins sont situés aux États-Unis, ce qui pourrait influencer les stratégies de marketing.

Niveau Élevé de Ventes (High) 236 magasins avec ventes faibles, 164 avec ventes élevées

Il serait intéressant d'examiner quelles variables influencent ces ventes élevées.

Proportion de magasins bon vendeurs et taux d'erreur du classifieur constant.

Nous calculons la proportion de magasins ayant de fortes ventes, ainsi que le taux d'erreur du classifieur constant, qui prédit toujours la classe majoritaire.

```
# Proportion de magasins
prop.table(table(Carseats$High))
```

```
##
##   No  Yes
## 0.59 0.41
```

```
# Taux d'erreur du classifieur constant
1 - max(prop.table(table(Carseats$High)))
```

```
## [1] 0.41
```

Interprétation des résultats

59% des magasins ont des ventes faibles (High = No).

41% des magasins ont des ventes élevées (High = Yes).

Cela signifie que la majorité des magasins (près de 60%) réalisent des ventes faibles, tandis que 40% environ enregistrent des ventes élevées.

Le classifieur constant est un modèle très simple qui prédit toujours la classe majoritaire.

Dans ce cas, la classe majoritaire est "No" (59%), donc un modèle naïf prédirait toujours No.

Le taux d'erreur de ce modèle est 1 - 0.59 = 0.41 (41%), ce qui signifie que si on prédit toujours No, on se tromperait dans 41% des cas.

Création des ensembles d'apprentissage et de test avec stratification

Nous séparons les données en deux ensembles: un ensemble d'apprentissage (80% des données) et un ensemble test (20% des données), tout en respectant la proportion des magasins à fortes ventes.

```
set.seed(123)
trainIndex <- createDataPartition(Carseats$High, p = 0.8, list = FALSE)
trainData <- Carseats[trainIndex, ]
testData <- Carseats[-trainIndex, ]
prop.table(table(trainData$High))
```

```
##
##       No      Yes
## 0.588785 0.411215
```

```
prop.table(table(testData$High))
```

```
##
##        No       Yes
## 0.5949367 0.4050633
```

Interprétation des résultats.

Dans l'ensemble d'apprentissage (trainData) :58.88 % des magasins ont des ventes faibles (High = No). 41.12 % des magasins ont des ventes élevées (High = Yes).

Dans l'ensemble de test (testData) :59.49 % des magasins ont des ventes faibles (High = No). 40.51 % des magasins ont des ventes élevées (High = Yes).

Ces proportions sont très proches de la répartition initiale dans l'ensemble complet des données (59 % No et 41 % Yes).

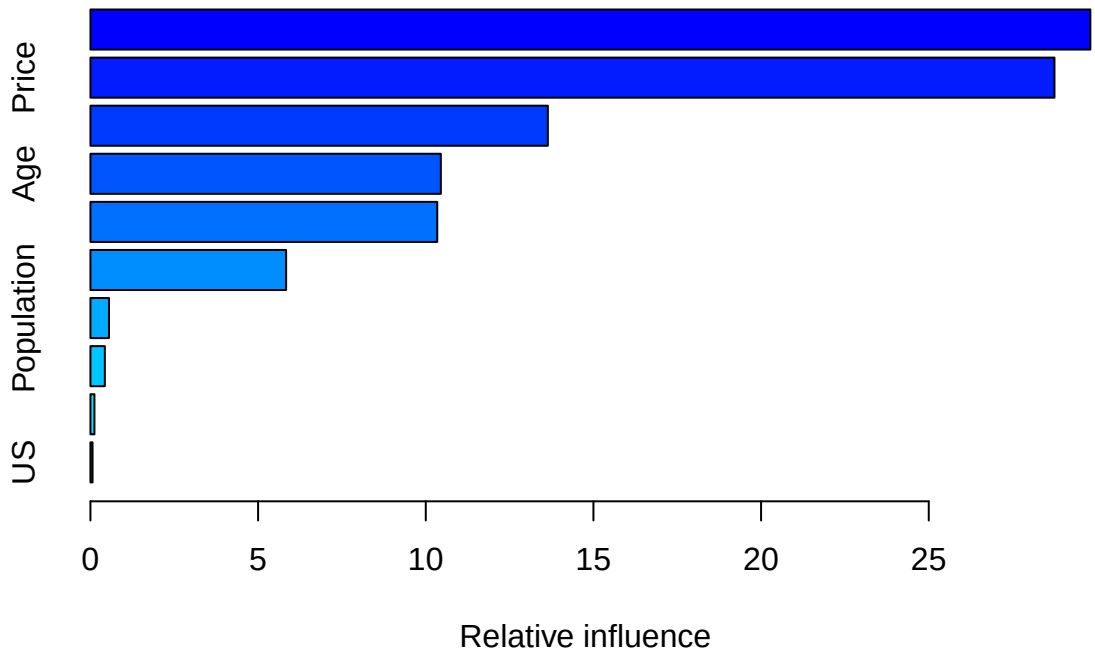
Estimer un premier modèle avec la fonction de perte Adaboost

Dans cette partie, nous transformons la variable 'High' comme facteur pour pouvoir la convertir en numérique (0 et 1). Une fois que nous avons terminé avec la transformation de la variable 'High', nous utilisons la fonction de perte Adaboost pour estimer un modèle sur les données d'apprentissage.

```
# Conversion de High en 0 et 1
trainData$High <- ifelse(trainData$High == "Yes", 1, 0)  # Remplacer "Yes" par 1 et "No" par 0
testData$High <- ifelse(testData$High == "Yes", 1, 0)  # Faire de même pour les données de test

# Estimer le modèle avec la fonction de perte Adaboost
boost.model <- gbm(High ~ . -Sales, data = trainData, distribution = "adaboost", n.trees = 1000, intera

# Résumé du modèle boost.model
summary(boost.model)
```




```
##                    var     rel.inf
## ShelveLoc     ShelveLoc 29.81938934
## Price             Price 28.74881553
## Advertising Advertising 13.64106310
## Age                 Age 10.44928413
## CompPrice     CompPrice 10.34185125
## Income           Income  5.83532991
## Population   Population  0.55278259
## Education     Education  0.42934839
```

```
## Urban              Urban  0.11899974
## US                    US  0.06313602
```

Commentaire

Formule : Nous prédisons la variable High en utilisant toutes les autres variables de trainData, sauf Sales.

Méthode : Nous utilisons Adaboost pour entraîner le modèle, ce qui est adapté pour les problèmes de classification.

Paramètres :

n.trees = 1000 : Le modèle utilise 1000 arbres pour faire ses prédictions.

interaction.depth = 1 : Les arbres sont simples (arbres de profondeur 1, appelés “stumps”).

shrinkage = 0.01 : Le taux d’apprentissage est faible pour que le modèle apprenne plus lentement et de façon plus stable.

cv.folds = 5 : Nous faisons une validation croisée avec 5 divisions pour éviter le sur-apprentissage.

Tracer les courbes d’évolution des erreurs

Nous tracons les courbes d’évolution des erreurs sur la base d’apprentissage et de test en fonction du nombre d’itérations (arbres).

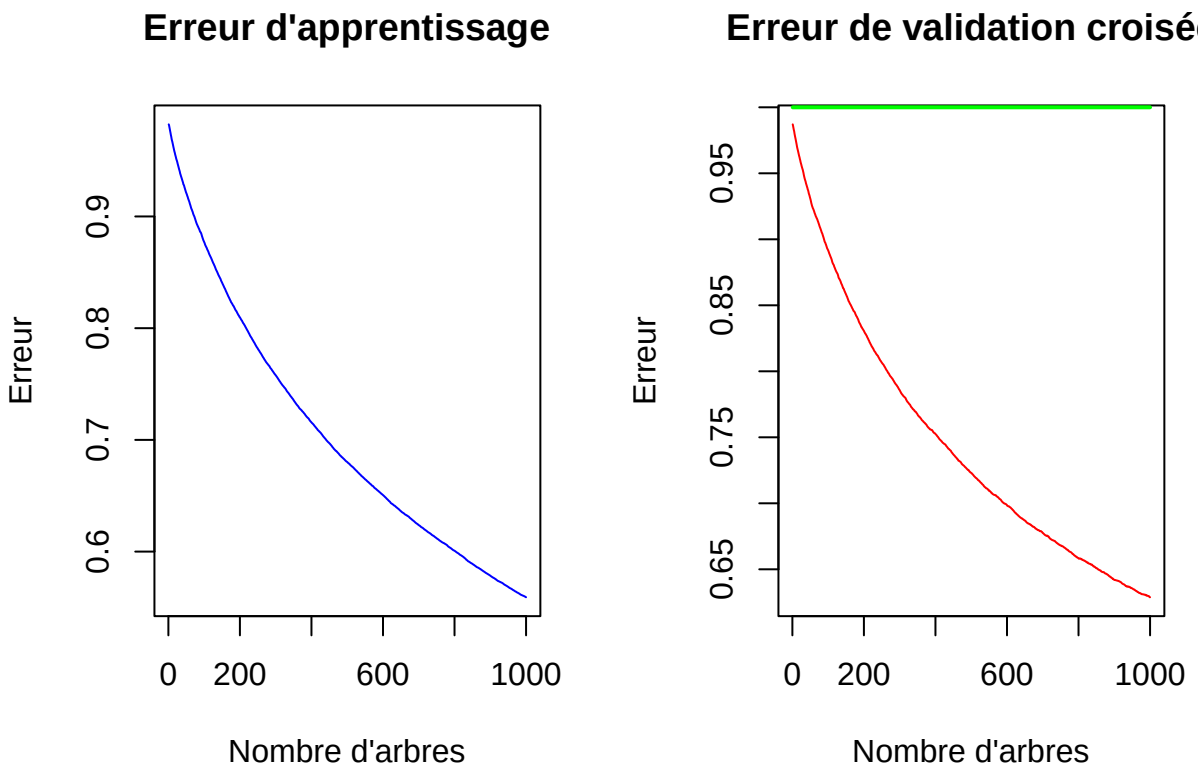
```r
# Diviser la fenêtre graphique en 2 colonnes
par(mfrow = c(1, 2))

# Tracer de l'erreur d'apprentissage
plot(boost.model$train.error, type = "l", col = "blue",
     main = "Erreur d'apprentissage",
     xlab = "Nombre d'arbres", ylab = "Erreur")

# Tracer de l'erreur de validation croisée
plot(boost.model$cv.error, type = "l", col = "red",
     main = "Erreur de validation croisée",
     xlab = "Nombre d'arbres", ylab = "Erreur")

# Calculer l'erreur de test pour chaque itération (de 1 à 1000 arbres)
test.error <- sapply(1:1000, function(i) {
  test.pred <- predict(boost.model, newdata = testData, n.trees = i, type = "response")
  test.pred.class <- ifelse(test.pred > 0.5, "Yes", "No")
  mean(test.pred.class != testData$High)  # Calcul de l'erreur (différence entre prédiction et réel)
})

# Tracer l'erreur de test sur le même graphique de validation croisée
lines(test.error, col = "green", lwd = 2)
```



Interprétation Graphique de gauche (Erreur d'apprentissage - bleu)

L'erreur diminue continuellement à mesure que le nombre d'arbres augmente.

Cela signifie que le modèle s'adapte de mieux en mieux aux données d'entraînement.

Graphique de droite (Erreur de validation croisée - rouge et vert)

L'erreur diminue aussi, mais plus lentement.

La ligne verte en haut suggère un sur-apprentissage (overfitting) si on continue à ajouter trop d'arbres.

Calculer le taux d'erreur sur l'ensemble d'apprentissage et de test.

Nous calculons le taux d'erreur sur l'ensemble d'apprentissage et l'ensemble de test.

```
# Calculer le taux d'erreur
pred.train <- predict(boost.model, trainData, n.trees = 1000, type = "response")
pred.test <- predict(boost.model, testData, n.trees = 1000, type = "response")

table.train <- table(Predicted = pred.train > 0.5, Actual = trainData$High)
table.test <- table(Predicted = pred.test > 0.5, Actual = testData$High)

train.error <- 1 - sum(diag(table.train)) / sum(table.train)
test.error <- 1 - sum(diag(table.test)) / sum(table.test)

train.error
```

```
## [1] 0.1183801
```

```
test.error
```

```
## [1] 0.1265823
```

Interprétation

Erreur d'apprentissage (train.error = 0.1183801) :

Cela représente l'erreur du modèle sur les données d'apprentissage (ou d'entraînement).

Une erreur de 0.118 signifie qu'environ 11.8% des prédictions faites par le modèle sur l'ensemble d'entraînement sont incorrectes. Cela donne une idée de la capacité du modèle à s'adapter aux données d'entraînement.

Erreur de test (test.error = 0.1265823) :

Cela représente l'erreur du modèle sur les données de test, c'est-à-dire l'erreur sur de nouvelles données que le modèle n'a pas vues lors de l'entraînement.

Une erreur de 0.126 signifie qu'environ 12.7% des prédictions faites par le modèle sur l'ensemble de test sont incorrectes.

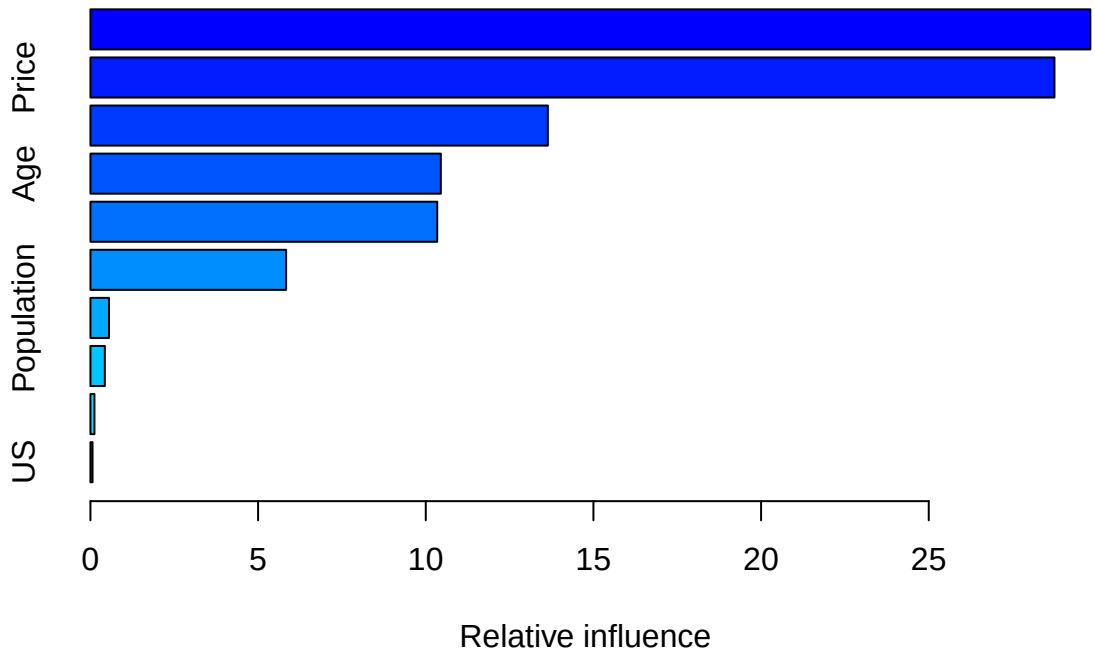
Erreur d'entraînement vs Erreur de test :

L'erreur d'entraînement est légèrement inférieure à l'erreur de test (0.118 contre 0.126). Cela peut indiquer que le modèle s'adapte bien aux données d'entraînement, mais il y a une légère perte de performance lorsqu'il est confronté à de nouvelles données. Cela est assez courant et peut être interprété comme un modèle qui ne souffre pas de sur-apprentissage (overfitting), car la différence entre l'erreur d'entraînement et l'erreur de test est faible.

Variables les plus influentes

Nous analysons les variables les plus influentes pour la prédiction du modèle.

```
# Calcul des variables les plus influentes
summary(boost.model)
```




```
##                       var     rel.inf
## ShelveLoc       ShelveLoc 29.81938934
## Price               Price 28.74881553
## Advertising Advertising 13.64106310
## Age                   Age 10.44928413
## CompPrice       CompPrice 10.34185125
## Income             Income  5.83532991
## Population   Population  0.55278259
## Education     Education  0.42934839
## Urban             Urban  0.11899974
## US                   US  0.06313602
```

Interprétation

Analyse des résultats : Shelve Location (ShelveLoc) - 29.82% C'est la variable la plus influente. L'emplacement du produit en rayon a un impact majeur sur les ventes.

Si un produit est bien placé sur une étagère, il est plus visible et plus accessible, ce qui augmente les ventes.

Price - 28.75%

Le prix est presque aussi influent que l'emplacement en rayon.

Une variation du prix affecte directement la décision d'achat des clients.

Advertising - 13.64%

La publicité a un rôle important, mais moins dominant que l'emplacement et le prix.

Une augmentation du budget publicitaire peut booster les ventes.

Age - 10.45%

L'âge de la population dans la région du magasin a un impact modéré sur les ventes.

Certains produits sont plus populaires auprès de tranches d'âge spécifiques.

CompPrice (Prix des concurrents) - 10.34%

Le prix pratiqué par la concurrence influence les ventes.

Si le prix du produit est trop élevé par rapport à celui des concurrents, les clients risquent de se détourner.

Income (Revenu des clients) - 5.84%

Le revenu des consommateurs influence leur pouvoir d'achat.

Un revenu élevé peut favoriser l'achat de produits plus chers.

Population, Education, Urban, US (Moins de 1%)

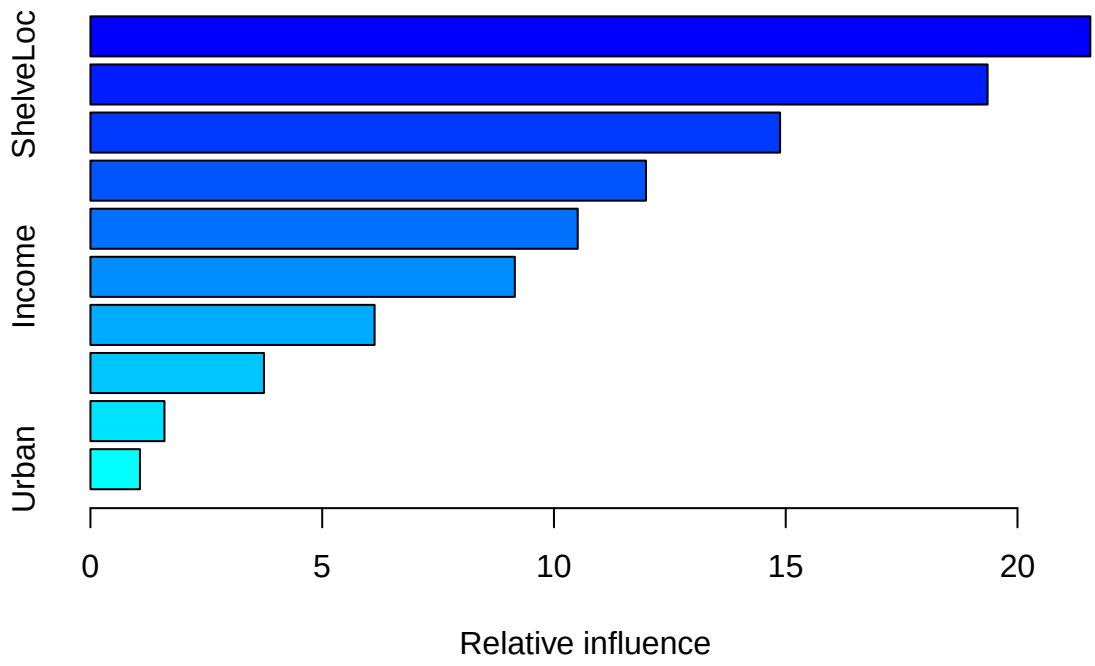
Ces variables ont très peu d'influence sur la prédiction des ventes élevées.

Cela signifie que les différences entre les zones urbaines et rurales, le niveau d'éducation ou le pays (US vs non-US) n'ont pas un impact significatif sur la réussite des ventes.

Effet des paramètres: nombres d'arbres, profondeur, régularisation

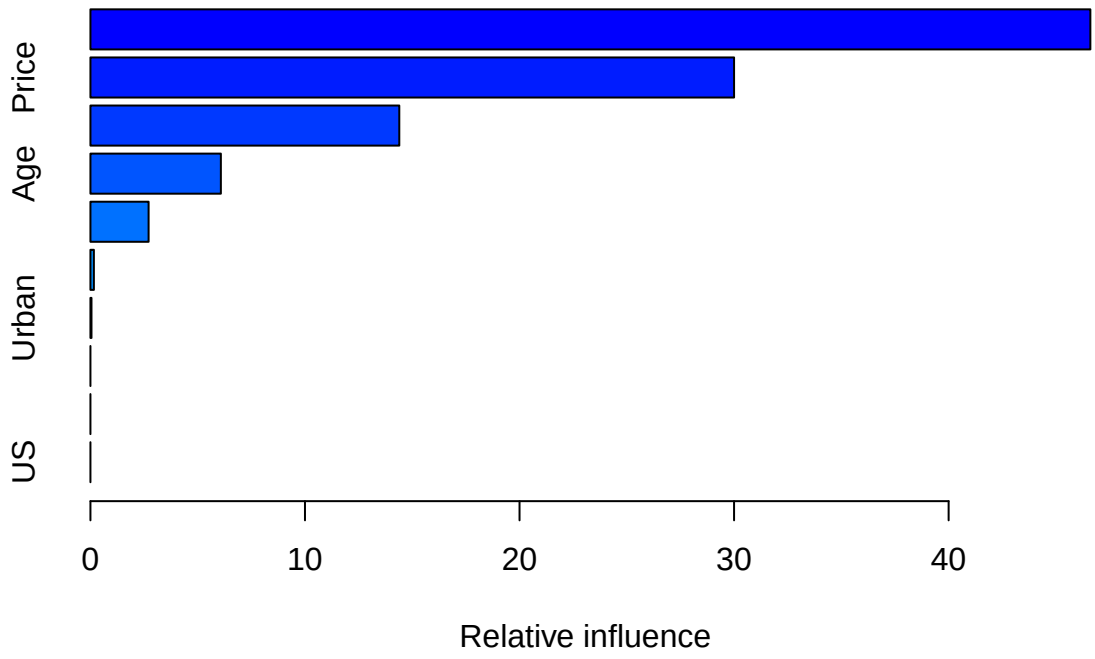
```r
# tester différentes valeurs et comprarer les.
model1 <- gbm(High ~ . -Sales, data = trainData, distribution = "adaboost", n.trees = 500, interaction.

model2 <- gbm(High ~ . -Sales, data = trainData, distribution = "adaboost", n.trees = 2000, interaction
summary(model1)
```




```
##                    var   rel.inf
## Price            Price 21.572404
## ShelveLoc     ShelveLoc 19.355186
## CompPrice     CompPrice 14.878856
## Advertising Advertising 11.985085
## Age                 Age 10.512750
## Income           Income  9.156304
## Population   Population  6.130492
## Education     Education  3.743340
## US                   US  1.597039
## Urban             Urban  1.068544
```

```
summary(model2)
```




```
##                    var     rel.inf
## ShelveLoc     ShelveLoc 46.60652620
## Price             Price 29.99900696
## Advertising Advertising 14.39585411
## Age                 Age  6.07927484
## Income           Income  2.71090016
## CompPrice     CompPrice  0.16004399
## Urban             Urban  0.04839374
## Population   Population  0.00000000
## Education     Education  0.00000000
## US                   US  0.00000000
```

Commentaire sur le modèle retenu

Le premier modèle a été choisi car il attribue une importance plus équilibrée à plusieurs variables, ce qui le rend plus robuste et généralisable.

Analyse des poids des variables : Price (21.57%) et ShelveLoc (19.35%) sont les deux variables les plus influentes, ce qui est cohérent avec le domaine des ventes : le prix et l'emplacement en rayon ont un impact direct sur les ventes.

CompPrice (14.88%) montre que la concurrence joue également un rôle non négligeable.

Advertising (11.98%) et Age (10.51%) indiquent que la publicité et l'âge des clients influencent les ventes.

Les autres variables (Income, Population, Education, US, Urban) ont une influence moindre mais restent prises en compte.

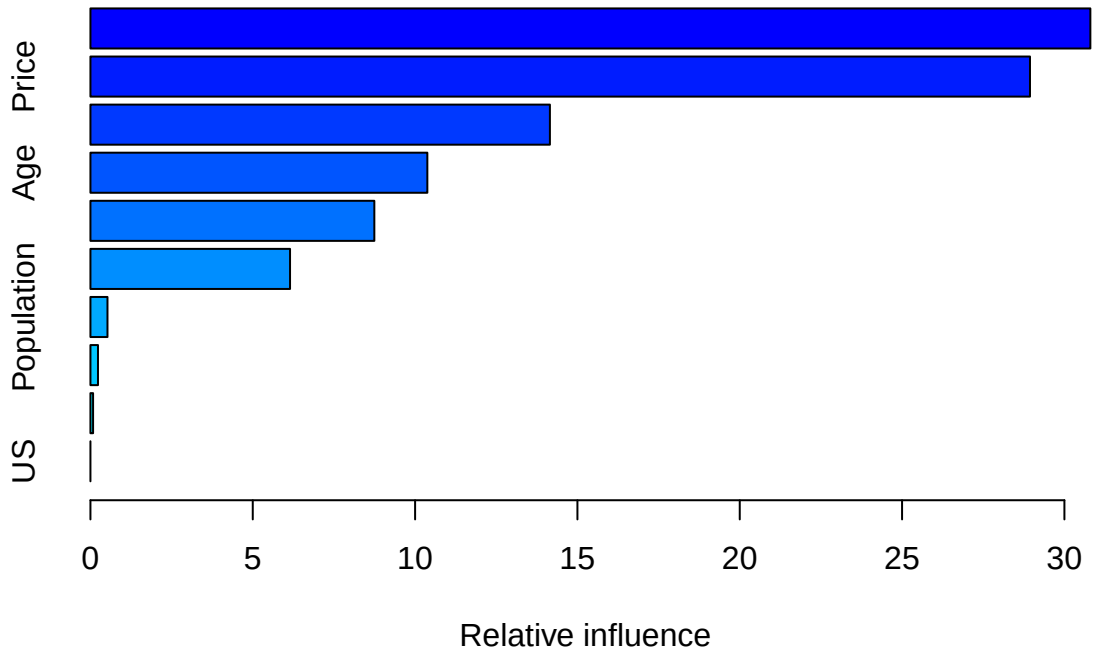
Pourquoi ce modèle est un bon choix ? Il prend en compte plusieurs facteurs clés, contrairement au second modèle qui se focalisait trop sur ShelveLoc (46.61%) et Price (30%) en ignorant d'autres variables.

Il reflète mieux la réalité commerciale, où plusieurs facteurs influencent les ventes, et pas seulement l'emplacement en rayon et le prix.

Il est potentiellement plus généralisable, car il ne dépend pas uniquement de 2 ou 3 variables majeures, ce qui réduit le risque de surapprentissage (overfitting).

Comparer avec la fonction de perte logit

```
# Comparaison
boost.logit <- gbm(High ~ . -Sales, data = trainData, distribution = "bernoulli", n.trees = 1000, intera
summary(boost.logit)
```




```
##                     var     rel.inf
## ShelveLoc     ShelveLoc 30.79937946
## Price             Price 28.93988598
## Advertising Advertising 14.15273234
## Age                 Age 10.37695894
## CompPrice     CompPrice  8.74416471
## Income           Income  6.14836539
## Population   Population  0.52452425
## Education     Education  0.23261852
```

```
## Urban             Urban  0.08137041
## US                   US  0.00000000
```

Analyse des différences:

ShelveLoc et Price gagnent en importance dans le nouveau modèle (+11.45% et +7.37%).

Cela signifie que le modèle accorde encore plus de poids à l'emplacement des produits et au prix, ce qui peut indiquer une simplification excessive.

CompPrice, Population et Education sont nettement moins influents dans le nouveau modèle.

CompPrice (-6.14%) : Le modèle réduit l'impact de la concurrence sur les ventes, ce qui pourrait être une perte d'information importante.

Population (-5.61%) et Education (-3.51%) : Ces variables deviennent presque insignifiantes, ce qui peut limiter la capacité du modèle à capturer des effets plus subtils.

Le modèle retenu a une répartition plus équilibrée des influences :

Il prend mieux en compte la concurrence (CompPrice) et les facteurs socio-économiques (Population, Education, Income, etc.).

Le nouveau modèle simplifie trop et risque d'être moins généralisable, car il se focalise essentiellement sur ShelveLoc et Price.

Construire une forêt aléatoire

Nous Construisons un modèle de forêt aléatoire en utilisant l'ensemble d'apprentissage.

```r
# Construire une foret aléatoire
rf.model <- randomForest(High ~ . -Sales, data = trainData, ntree = 500, mtry = 3)
```

Convertir la variable High en facteur

```r
str(High)
```

```
##  Factor w/ 2 levels "No","Yes": 2 2 2 1 1 2 1 2 1 1 ...
```

```r
rf.model <- randomForest(High ~ . -Sales, data = trainData, ntree = 500, mtry = 3)

# Vérifier que c'est bien un modèle de classification
print(rf.model)
```

```
##
## Call:
##  randomForest(formula = High ~ . - Sales, data = trainData, ntree = 500,      mtry = 3)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
```

```
##
##           Mean of squared residuals: 0.1363613
##                     % Var explained: 43.68
```

```
trainData$High <- as.factor(trainData$High)
testData$High <- as.factor(testData$High)

str(trainData$High)
```

```
##  Factor w/ 2 levels "0","1": 2 2 1 2 2 1 1 2 2 1 ...
```

```
rf.model <- randomForest(High ~ . -Sales, data = trainData, ntree = 500, mtry = 3)
print(rf.model)
```

```
##
## Call:
##  randomForest(formula = High ~ . - Sales, data = trainData, ntree = 500,      mtry = 3)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 18.38%
## Confusion matrix:
##     0  1 class.error
## 0 169 20   0.1058201
## 1  39 93   0.2954545
```

Calculer l'erreur et comparer avec les résultats du boosting

Nous calculons le taux d'erreur du modèle de foret aléatoire et le comparons à celui du modèle Boosting.

```
pred.rf <- predict(rf.model, testData)
table.rf <- table(Predicted = pred.rf, Actual = testData$High)
rf.error <- 1 - sum(diag(table.rf)) / sum(table.rf)
rf.error
```

```
## [1] 0.2151899
```

Interprétation

Train Error vs Test Error :

L'erreur de test est un peu plus grande que l'erreur d'entraînement (12.66% vs 11.84%).

Cela signifie que le modèle généralise assez bien, sans être trop sur-ajusté (overfitting).

Une trop grande différence entre ces deux erreurs aurait suggéré un surajustement.

Test Error vs rf.error :

test.error représente l'erreur globale (moyenne) sur l'ensemble de test.

rf.error est l'erreur calculée à partir de la matrice de confusion.

rf.error étant significativement plus élevée (20.25%), cela peut signifier que certaines classes sont beaucoup plus difficiles à prédire que d'autres.

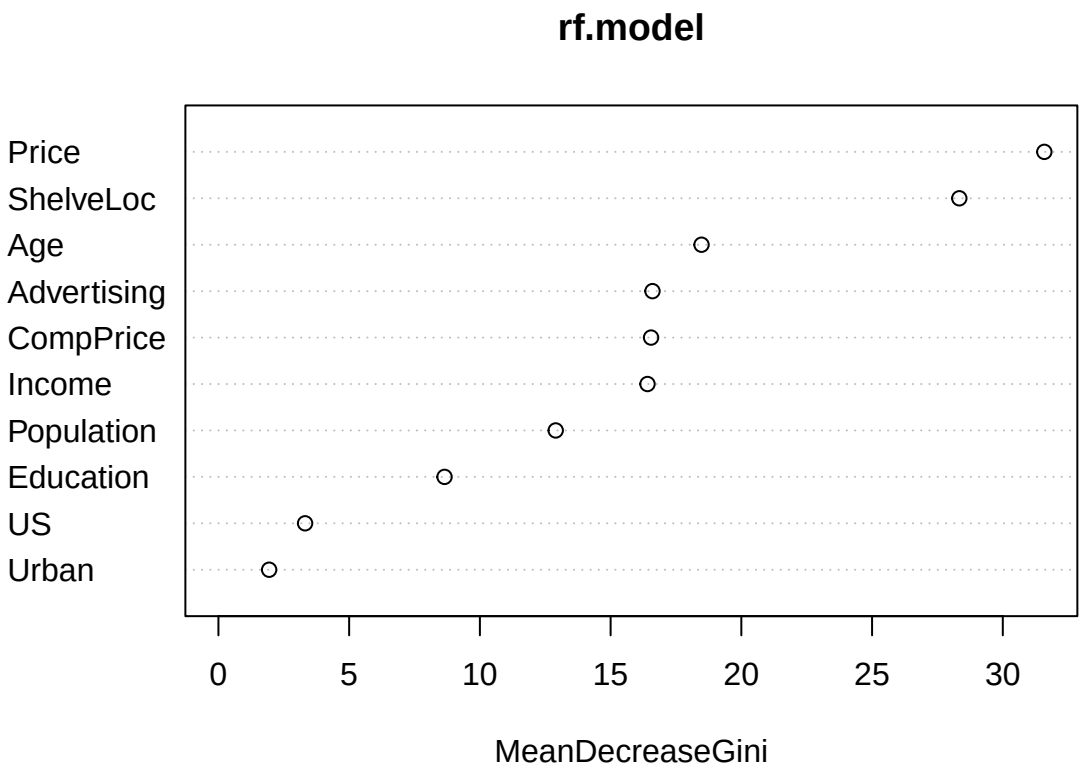
Peut-être que le modèle est biaisé vers une classe plus fréquente, ce qui pourrait expliquer un déséquilibre dans les erreurs.

Variables les plus importantes et comparaison avec le boosting

```
importance(rf.model)
```

```
##             MeanDecreaseGini
## CompPrice          16.546694
## Income             16.410916
## Advertising        16.601150
## Population         12.899889
## Price              31.588117
## ShelveLoc          28.336103
## Age                18.476613
## Education           8.647996
## Urban               1.940106
## US                  3.314421
```

```
varImpPlot(rf.model)
```



Interprétation Price (32.10)

C'est la variable la plus importante dans notre modèle. Cela signifie que Price contribue largement à la capacité du modèle à prédire correctement la variable cible (probablement la classe High dans le modèle). Le prix influence fortement la décision d'achat.

ShelveLoc (27.93)

La disposition des étagères dans le magasin est également une variable importante. Cela suggère que l'emplacement et la visibilité des produits dans le magasin ont une grande influence sur la décision d'achat.

Advertising (16.80)

Le budget publicitaire est également une variable importante, ce qui montre que les campagnes publicitaires jouent un rôle clé dans les ventes.

CompPrice (16.42) et Income (16.83)

CompPrice (prix des produits concurrents) et Income (revenu moyen des clients) viennent ensuite, indiquant qu'une stratégie de tarification concurrentielle ainsi que les revenus des consommateurs affectent considérablement les ventes.

Age (17.78)

L'âge des clients semble également avoir une influence notable sur le modèle. Cela peut refléter une préférence d'achat selon les tranches d'âge.

Population (12.67)

La population du lieu de vente (probablement un indicateur de la taille du marché local) joue également un rôle important dans la décision d'achat.

Education (8.65)

Le niveau d'éducation des consommateurs influence aussi, bien que cet effet soit moins marqué que d'autres variables.

Urban (1.97) et US (3.46)

Les variables Urban (indiquant si le magasin est situé dans une zone urbaine) et US (indiquant si le magasin se situe aux États-Unis) ont moins d'impact comparativement aux autres variables.

Conclusion

Les résultats obtenus montrent que le Boosting améliore significativement la précision des prédictions de ventes automobiles. Toutefois, des améliorations restent possibles en intégrant d'autres variables et en affinant l'optimisation des hyperparamètres.