---
title: "GBM en R"
author: "Víctor Gallego, Roi Naveiro"
date: "21 de abril de 2019"
output: pdf_document

GBM en R

En esta práctica usaremos la librería gbm de R para ajustar un conjunto de árboles de decisión mediante boosting. Lo evaluaremos en el dataset de datos de próstata.

1. Cargar fichero en R y librería gbm

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
data <- read.csv('../../data/prostate.data', header = TRUE, row.names = 1)
```

2. Separar en train/test de acuerdo con los valores de la columna `train`

```
train_col <- which(colnames(data) == "train")
target_col <- which(colnames(data) == "lpsa")
train_set <- data[ data$train, -train_col]
test_set  <- data[!data$train, -train_col]
```

2. (Opcional, solo para ver que gbm es capaz de tratar con variables a diferentes escalas) Escalar las variables para que tengan media 0 y varianza 1 (menos `lpsa`)

```
Xtrain <- scale(train_set[, -target_col])
center <- attr(Xtrain, "scaled:center")
scale  <- attr(Xtrain, "scaled:scale")
Xtest <- scale(test_set[, -target_col], center = center, scale = scale)
```

```
# Con escalado
train <- data.frame(Xtrain, lpsa=train_set[, target_col])
test <- data.frame(Xtest, lpsa=test_set[, target_col])
```

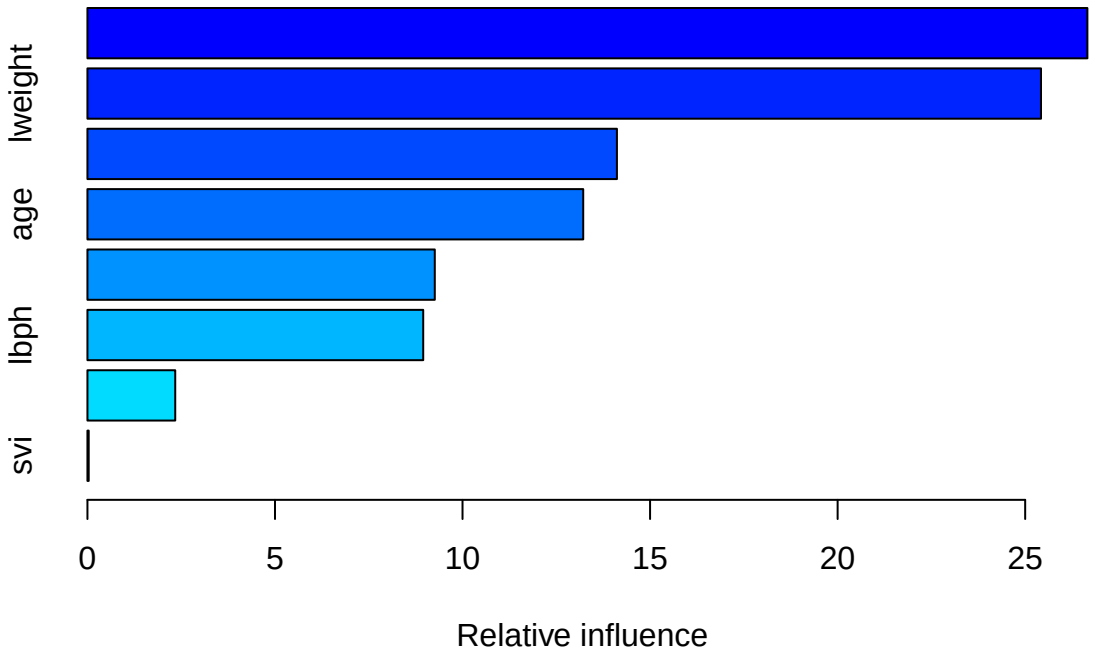
```
# Sin escalado
train <- data.frame(train_set[, -target_col], lpsa=train_set[, target_col])
test <- data.frame(test_set[, -target_col], lpsa=test_set[, target_col])
```

3. Ajustamos un modelo para predecir el valor mediano en función del resto de variables, para 5000 iteraciones y 4 interacciones máximas en cada árbol, y shrinkage de 0.4 Escoger el argumento distribution adecuado

```
model <- gbm(lpsa ~ .,data=train, distribution="gaussian",n.trees=5000, shrinkage = 0.5, interaction.de
```

4. Obtener la importancia de cada variable usando summary

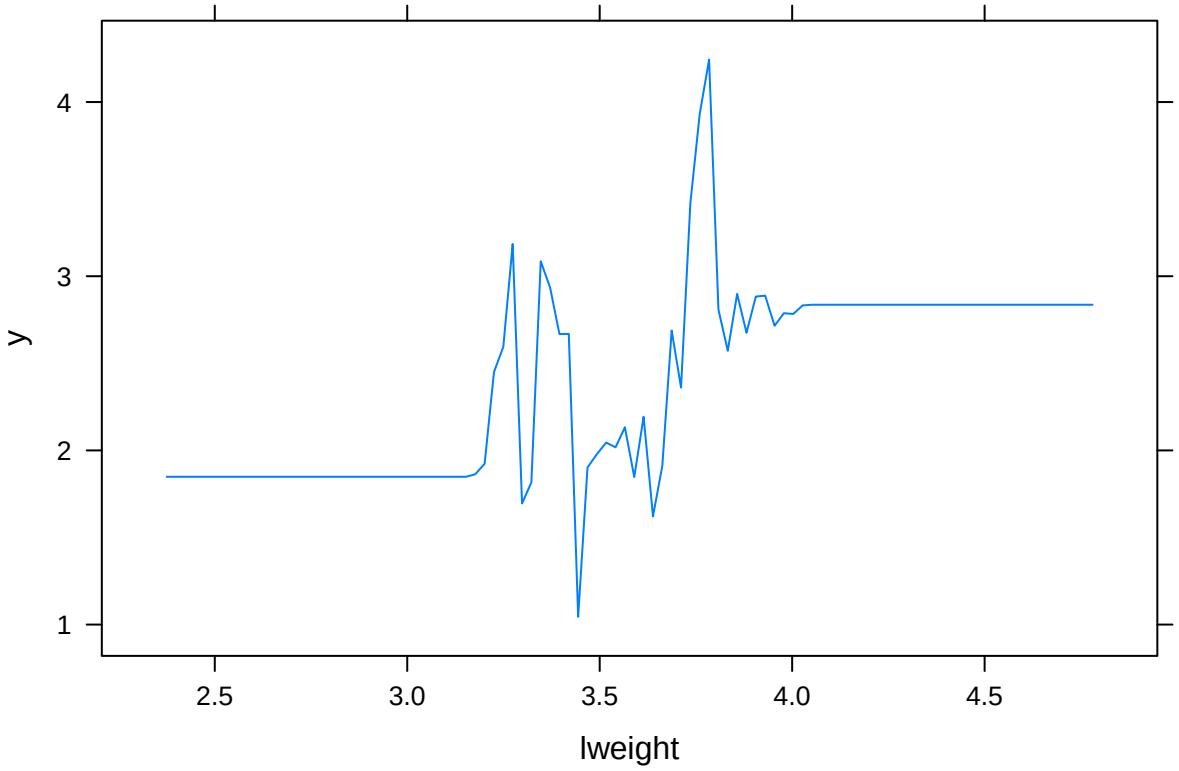
```
summary(model)
```



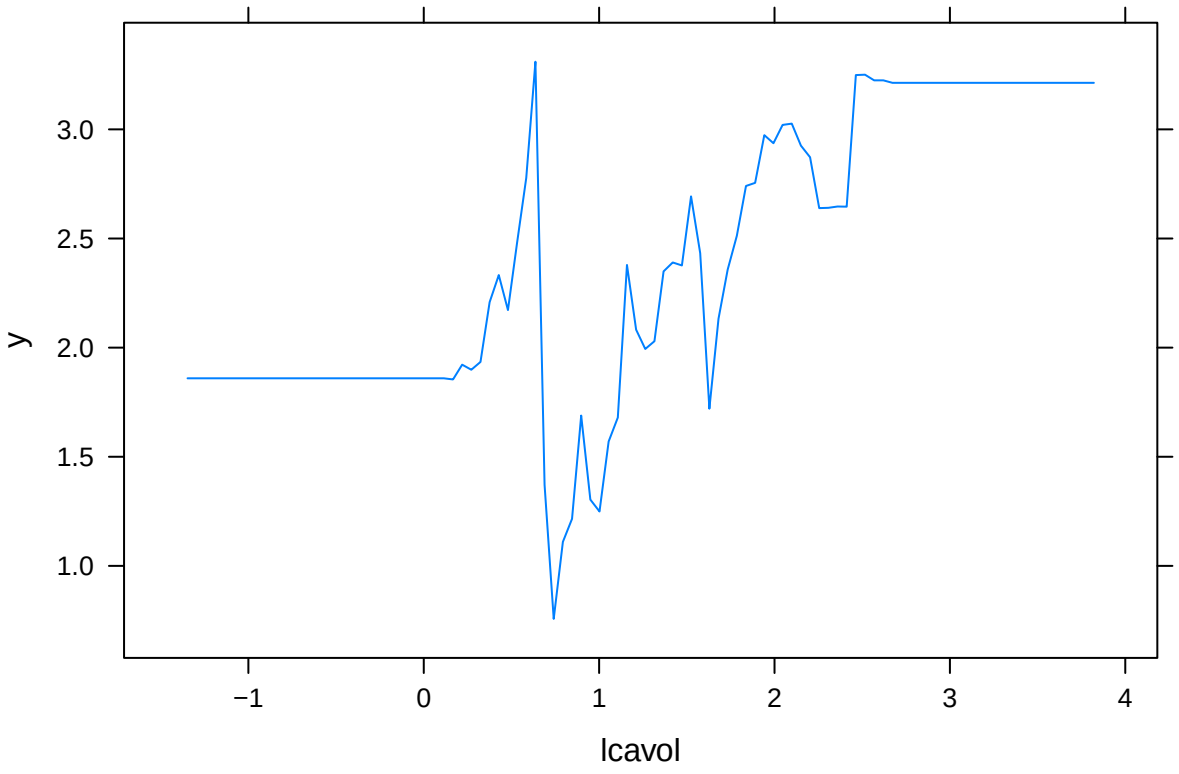

```
##             var     rel.inf
## lcavol   lcavol 26.65892944
## lweight lweight 25.42197296
## pgg45     pgg45 14.11579778
## age         age 13.21716938
## lcp         lcp  9.26036845
## lbph       lbph  8.95152652
## gleason gleason  2.34083929
## svi         svi  0.03339618
```

5. Obtener el gráfico de dependencia parcial para las dos variables más importantes, tanto de forma independiente como de forma conjunta las dos

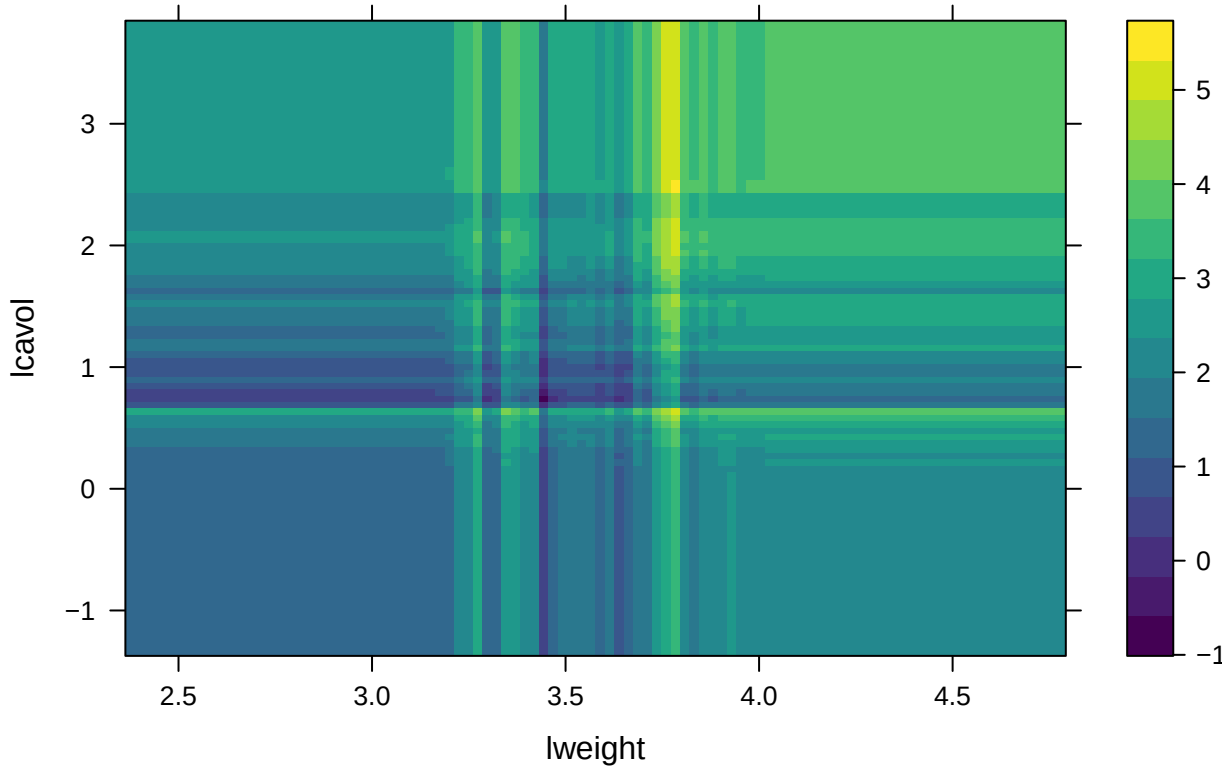
```
plot(model, i="lweight")
```




```
plot(model, i="lcavol")
```

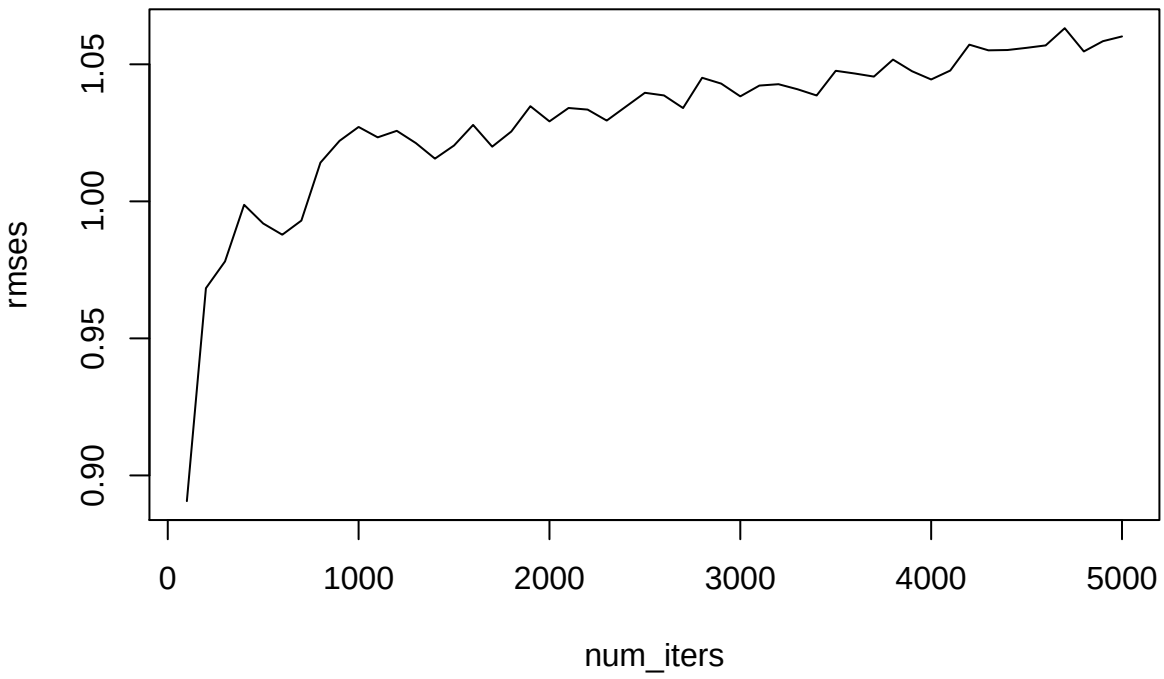



```
plot(model, i.var = c("lweight","lcavol"))
```



6. Representar el RMSE sobre el conjunto de test en función del número de árboles utilizados en la predicción

```
num_iters <- seq(100,5000,100)
yhat.boost <- predict(model,newdata = test, n.trees=num_iters)
rmses <- sqrt(colMeans((yhat.boost - test$lpsa)^2))
plot(num_iters, rmses, typ='l')
```



```r
min(rmses)
```

```
## [1] 0.8906273
```

7. Ahora, usaremos la librería caret para ajustar los siguientes hiperparámetros mediante CV. Recordatorio:

- Usa trainControl para especificar validación cruzada con 10 cortes.
- Usa expand.grid para especificar los siguientes hiperparámetros:
- profundidades: de 1 a 4.
- número de iteraciones: de 50 a 1500 contando de 50 en 50.
- shrinkage: 0.1 o 0.3.
- n.minobsinnode: 5.

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

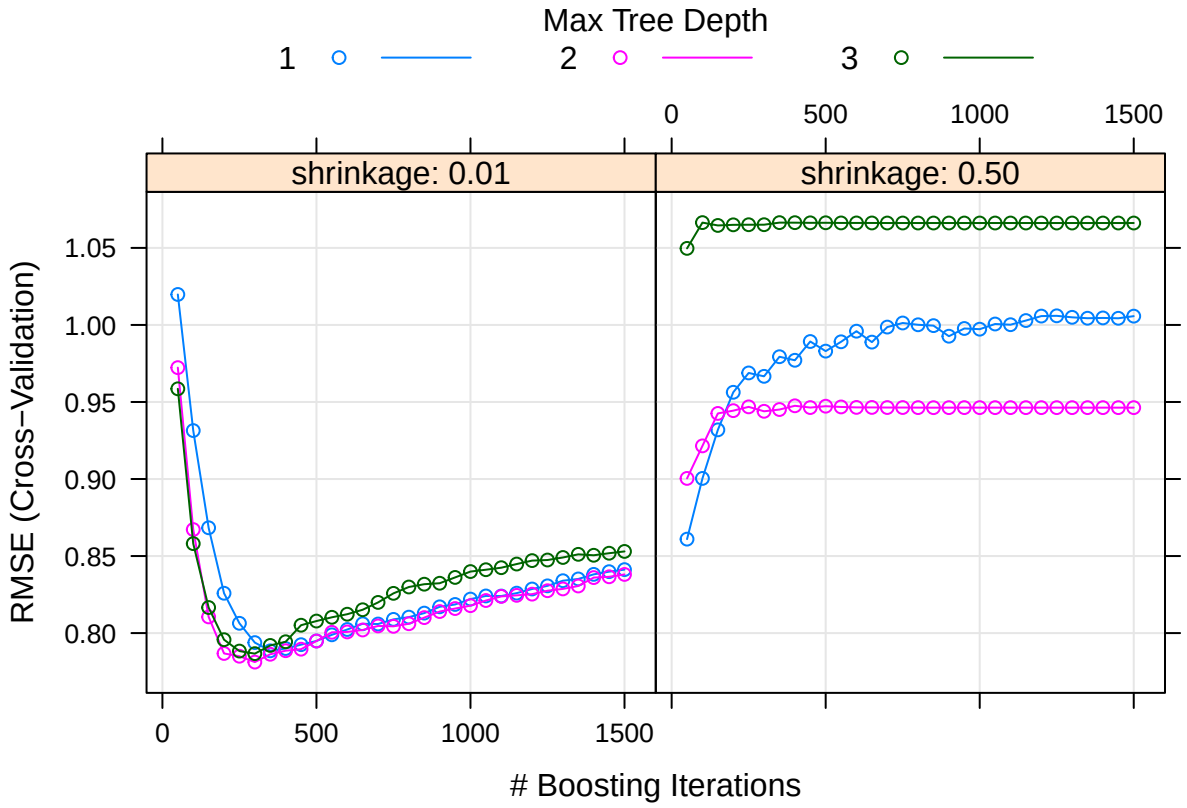
```r
fitControl <- trainControl(## 10-fold CV
                           method = "cv",
                           number = 10)
gbmGrid <-  expand.grid(interaction.depth = c(1, 2, 3),
                        n.trees = (1:30)*50,
                        shrinkage = c(0.5,0.01),
                        n.minobsinnode = 5)
model2 <- train(lpsa ~ ., data = train,
                method = "gbm",
                trControl = fitControl,
                tuneGrid = gbmGrid,
                verbose = FALSE)
```

8. Mostrar los hiperparámetros del mejor modelo, y hacer plot del objeto modelo ajustado

```r
print(model2$bestTune)
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 36     300                 2      0.01              5
```

```r
plot(model2)
```



9. Ajustar gbm al training set entero con los hiperparámetros anteriores. Realizar predicciones sobre el conjunto inicial de test y obtener el nuevo RMSE

```
model_opt <- gbm(lpsa ~ .,data=train, distribution="gaussian",
                 n.trees=model2$bestTune$n.trees,
                 interaction.depth=model2$bestTune$interaction.depth,
                 shrinkage = model2$bestTune$shrinkage)

yhat2 <- predict(model_opt,newdata = test, n.trees = model2$bestTune$n.trees)
rmse <- sqrt(mean((yhat2 - test$lpsa)^2))
print(rmse)
```

```
## [1] 0.7766425
```