Ensambles de Árboles de Decisión

Los árboles de decisión, en su formato más tradicional, poseen la ventaja de su facilidad de interpretación, pero acarrean una dificultad inherente: su propensión a generar modelos que se sobreajustan a los datos observados , generando por ende estimadores de alta varianza. Para subsanar este inconveniente ya hemos visto la posibilidad de aplicar técnicas de regularización llamadas "poda". *(prunning)*

Muestreo "*Bootstrap*"

Método para estimar el desvío de un estadístico muestral.

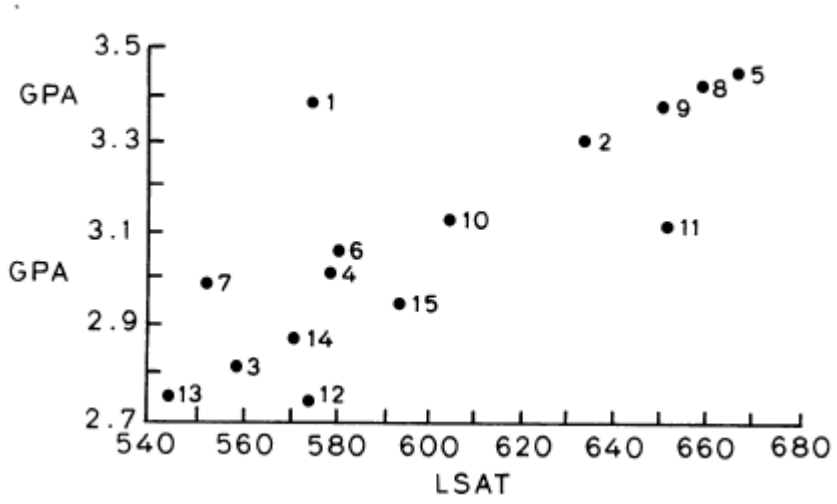
A diferencia del estimador de la media poblacional $\bar{x}$,para el cual (por el Teorema Central del Límite) conocemos su función de distribución y por ende podemos calcular sus parámetros en especial su desvió o error muestral, h ay muchos estdíst5icos (cómo se denomina a los estimadores de parámetros poblaciones) de los que s desconoce su distribución de probabilidad, por lo que no es imposible calcular su desvío nuestra y por ende realizar inferencias estadísticas.

Hay una forma más sencilla de estimar $\sigma(F)$. Tomemos $\hat{F}$ como la **distribución empírica** de x. Para ellos utilizaremos la distribución uniforme.

$$\hat{F} = 1/n \qquad x_1, x_2, \cdots, x_n$$

wue nos permite calcular la varianza de la media muestral como $\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$

Ejemplo: estimación del Coeficiente de correlación



Los puntos representan los promedios del LSAT y del GPA del conjunto de ingresantes a diferentes universidades de EEUU. El coeficiente de correlación observado es $\hat{\theta} = .776$. Queremos estimar ahora el error standard (la precisión) de esta estimación del coef de correlación.

$$\sigma(F) = [\mathrm{Var}_F\{\hat{\theta}(\mathbf{y})\}]^{1/2}$$

Como no conocemos la función de distribución muestral F, la estimaremos mediante un muestreo "bootstrap" $\hat{\sigma} = \sigma(\hat{F})$ donde $\hat{F}$ es la distribución empírica de los valores de $x_i = (\text{LSAT}_i, \text{GPA}_i)$ en este caso es $\hat{F} = 1/15$

En la mayoría de los casos no contamos con una estimación insesgada del desvío standard de este estimador $\hat{\theta}$, como si lo hay para $\bar{x}$. No obstante ello es fácil obtener una estimación numérica aplicando una simulación de Monte Carlo.

$(x_1^*, x_2^*, \cdots, x_n^*)$ indica una muestra "bootstrap" de tamaño **n**, obtenida a partir de extraer *n* observaciones con reposición de la muestra original $\{x_1, x_2, \cdots, x_n\}$

Algoritmo de Simulación de Monte Carlo para "*Bootstrap*"

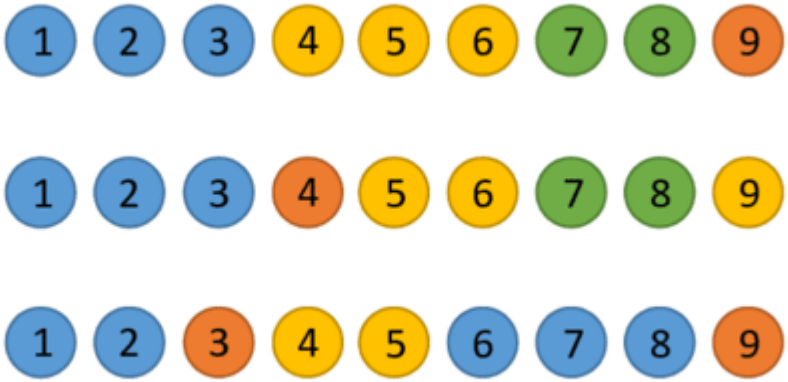
1) Utilizando un generador de nros. aleatorios se extraen una gran cantidad de muestras a partir de la muestra original
2) Para cada muestra bootstrap ***b*** se calcula el estadístico muestral $\hat{\theta}^*(b)$ con $b = 1, 2, \cdots, B$
3) Calcular laos errores standard muestrales mediante:

$$\hat{\sigma}_B = \left(\frac{\sum_{b=1}^{B} \{\hat{\theta}^*(b) - \hat{\theta}^*(\cdot)\}^2}{B-1} \right)^{1/2} \quad \textbf{(3)} \qquad \text{siendo} \qquad \hat{\theta}^*(\cdot) = \frac{\sum_{b=1}^{B} \hat{\theta}^*(b)}{B}$$

Para la mayoría de las situaciones **B** va de 50 a 200. La simulación de Monte Carlo consigue que converge $\hat{\sigma}_B$ a $\hat{\sigma}$ sólo si el tamaño de la muestra bootstrap es igual a *n* (el original)

Ejemplo de muestras "*Bootstrap*"

Muestra ORIGINAL



Ensambles 1: *Bagging*

CART = Classification and Regression Trees

Bagging es el acrónimo de ***"Bootstrap aggregation"***, que es un método para reducir la varianza de un modelo desarrollado por el Prof. Leo Breiman (Univ. de California en **Berkeley**) en 1996.

Los promedios de cualquier variable siempre tiene menor varianza que la variable prop. dicha (σ^2 vs. σ^2/n). En resumidas cuentas: ***promediar los valores de las observaciones de una variable reduce la varianza.***

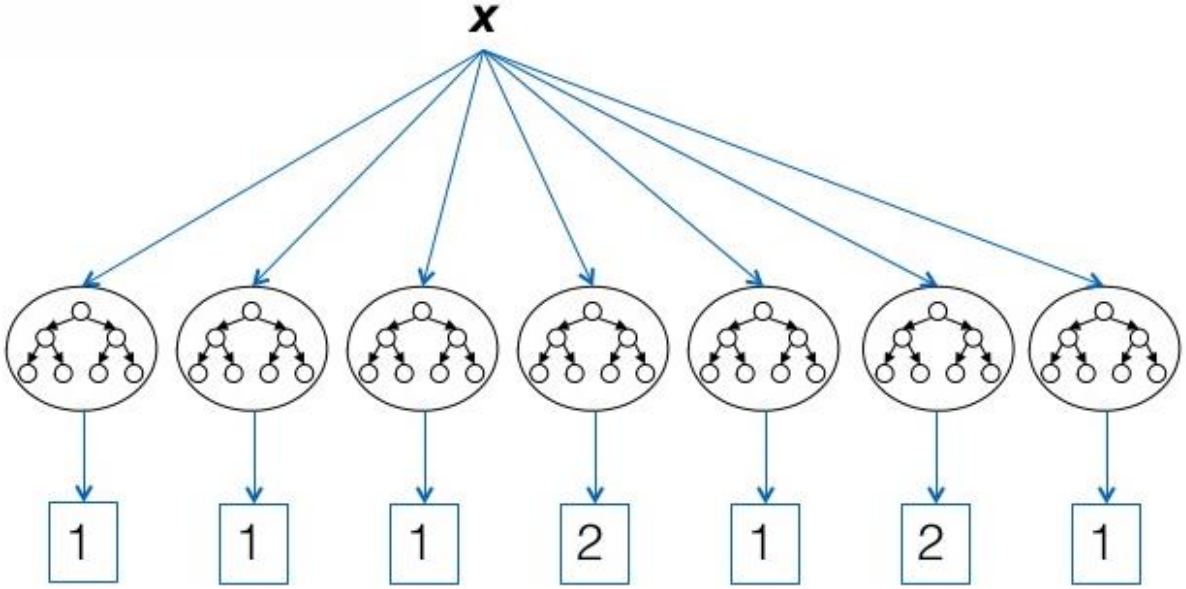
$\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$: B muestras diferentes. Si luego promedio sus estimaciones obtendré un modelo con menor varianza.

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x).$$

Pero como no dispongo de B muestras, obtengo entonces genero muestras *bootstraps,* obteniendo así un estimador ***bagging***

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Para variables categóricas → se elige la clase mayoritaria entre las estimadas por los B modelos



Input:

Dataset L

Ensemble size T

```
1.for t=1 to T:
2.    sample = BootstrapSample(L)
3.    h_t = TrainClassifier(sample)
```

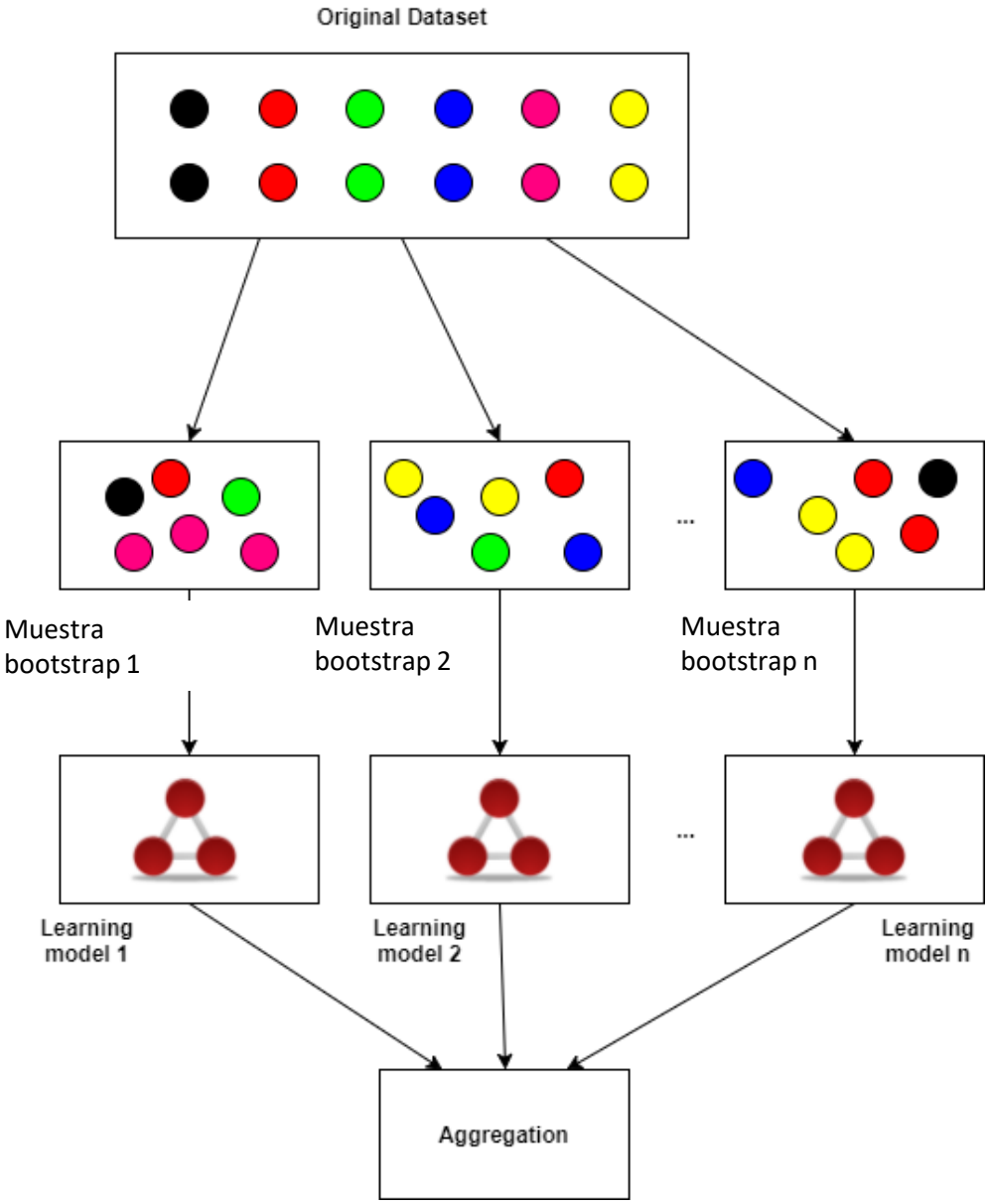
Bootstrap

\+

Output:

$$H(\mathbf{x}) = \underset{j}{\operatorname{argmax}} \left(\sum_{t=1}^{T} I(h_t(\mathbf{x}) = j) \right)$$

Aggregation

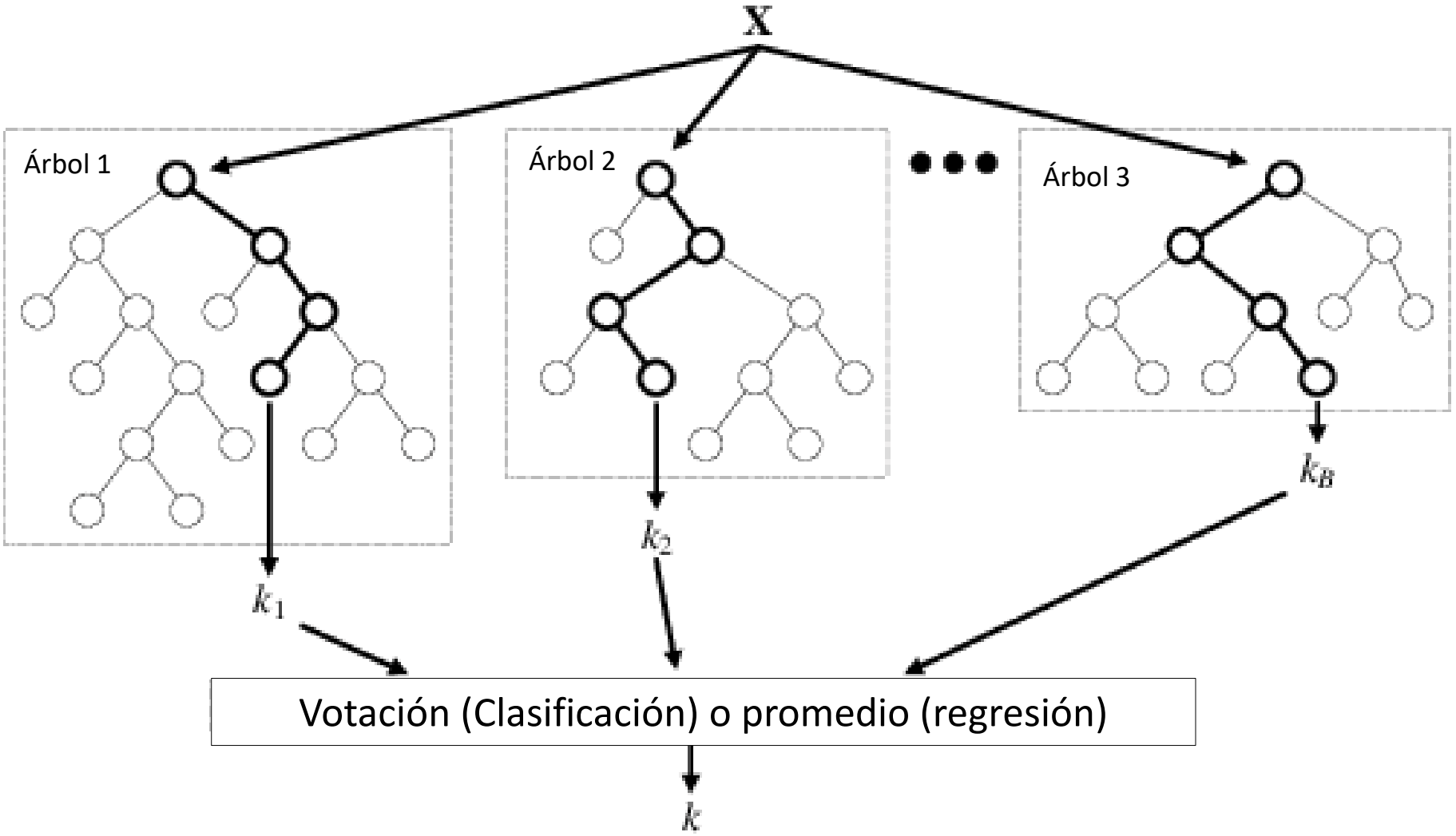

Original Dataset
Muestra bootstrap 1
Muestra bootstrap 2
...
Muestra bootstrap n
Learning model 1
Learning model 2
...
Learning model n
Aggregation

Ensambles 2: *Random Forest*

El algoritmo *de* ***Random Forest*** mejora la performance del *bagging* al ***decorrelacionar*** los árboles entre sí. El procedimiento empieza de la misma manera: tomamos B muestras *bootstrap* de observaciones a partir del dataset original para entrenar en ellas un modelo de árboles de decisión. La diferencia ahora es que en cada ramificación se eligen ***m*** de las ***p*** variables predictoras originales para evaluar sobre cuál de ellas basar la partición binaria. Comúnmente se utiliza un valor $m \approx \sqrt{p}$.
La lógica de este método , y su ventaja inherente, se basa en controlar la preponderancia de las variables con mucha capacidad discriminante en los primeras ramificaciones, pues ellas serán las que primero escogerá cada uno de los árboles de decisión del algoritmo *bagging*, generando así una correlación marcada entre estos.

IMPORTANTE: El promedio de variables correlacionadas siempre tiene varianza mayor que el de variables que no lo están.

Siendo la esencia de los **métodos de ensamble la reducción de la varianza**, es obvio el beneficio que se tiene al aplicar esta mejora del algoritmo de *bagging*


X
Árbol 1
Árbol 2
Árbol 3
k_1
k_2
k_B
Votación (Clasificación) o promedio (regresión)
k

Librerías en R de Bagging y Random Forest

`randomForest` *Classification and Regression with Random Forest*

Description

`randomForest` implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. It can also be used in unsupervised mode for assessing proximities among data points.

Usage

```
## S3 method for class 'formula'
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
## Default S3 method:
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
## S3 method for class 'randomForest'
print(x, ...)
```

Arguments

`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which `randomForest` is called from.
`subset`	an index vector indicating which rows should be used. (NOTE: If given, this argument must be named.)
`na.action`	A function to specify the action to be taken if NAs are found. (NOTE: If given, this argument must be named.)
`x, formula`	a data frame or a matrix of predictors, or a formula describing the model to be fitted (for the `print` method, an `randomForest` object).
`y`	A response vector. If a factor, classification is assumed, otherwise regression is assumed. If omitted, `randomForest` will run in unsupervised mode.
`xtest`	a data frame or matrix (like `x`) containing predictors for the test set.
`ytest`	response for the test set.
`ntree`	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
`mtry`	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in `x`) and regression (p/3)
`replace`	Should sampling of cases be done with or without replacement?
`classwt`	Priors of the classes. Need not add up to one. Ignored for regression.
`cutoff`	(Classification only) A vector of length equal to number of classes. The 'winning' class for an observation is the one with the maximum ratio of proportion of votes to cutoff. Default is 1/k where k is the number of classes (i.e., majority vote wins).
`strata`	A (factor) variable that is used for stratified sampling.
`sampsize`	Size(s) of sample to draw. For classification, if sampsize is a vector of the length the number of strata, then sampling is stratified by strata, and the elements of sampsize indicate the numbers to be drawn from the strata.
`nodesize`	Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).
`maxnodes`	Maximum number of terminal nodes trees in the forest can have. If not given, trees are grown to the maximum possible (subject to limits by `nodesize`). If set larger than maximum possible, a warning is issued.
`importance`	Should importance of predictors be assessed?
`localImp`	Should casewise importance measure be computed? (Setting this to `TRUE` will override `importance`.)
`nPerm`	Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.
`proximity`	Should proximity measure among the rows be calculated?
`oob.prox`	Should proximity be calculated only on "out-of-bag" data?
`norm.votes`	If `TRUE` (default), the final result of votes are expressed as fractions. If `FALSE`, raw vote counts are returned (useful for combining results from different runs). Ignored for regression.
`do.trace`	If set to `TRUE`, give a more verbose output as `randomForest` is run. If set to some integer, then running output is printed for every `do.trace` trees.
`keep.forest`	If set to `FALSE`, the forest will not be retained in the output object. If `xtest` is given, defaults to `FALSE`.
`corr.bias`	perform bias correction for regression? Note: Experimental. Use at your own risk.
`keep.inbag`	Should an `n` by `ntree` matrix be returned that keeps track of which samples are "in-bag" in which trees (but not how many times, if sampling with replacement)
`...`	optional parameters to be passed to the low level function `randomForest.default`.

Ajuste de Hiperparámetros → medida de la complejidad el modelo

Se denomina hiperparámetros a los **parámetros de ajuste** que se desea optimizar para **maximizar la performance de un algoritmo predictivo**. Se diferencian de los parámetros cualitativos que son los que definen al algoritmo, por ejemplo el método y las reglas asociadas a la partición binaria de un árbol de decisión.

Los hiperparámetros fundamentales que tiene mayor preponderancia en relación a su influencia en la performance del modelo predictivo de Bagging y RF son:

mtry: Número de variables predictoras elegidas aleatoriamente para armar el árbol
Al bajar el número de *mtry* reducimos la correlación entre los árboles, y por consiguiente la varianza de las estimaciones (predicciones) mejorando así su capacidad de generalización. A su vez esto afecta el balance sesgo-varianza, por ende estos modelos serán menos precisos pues tendrán más *bias.*
Los valores por default son $\sqrt{p}$ para clasificación y $p/3$ para regresión.

Sampzise: tamaño de la muestra de boostrap para entrenar cada árbol. Tiene el mismo efecto que el mtry: a menor valor menor varianza (y capacidad de generalización) y menor potencia (más bias). Valor por default: si es con **reposición,** nrow del dataset, si no 0,632*nrow.

ntree: cantidad de árboles de decisión a entrenar. Tiene relación con el tamaño de la muestra y la cantidad de variables predictoras (*sampsize y mtry*) pues a menor valor de ellos será necesario más árboles para mejorar la precisión de las predicciones.