

**Curso: Minería de Datos**  
**Docente: Jesús Salinas Flores**

Unidad III: Modelamiento  
usando técnicas  
supervisadas de  
clasificación

**Semana: 12 y 13**  
**Modelos de Ensamble**



## **RESULTADOS DE APRENDIZAJE DE LA UNIDAD III**

- Comprende el modelamiento de los datos y compara diferentes técnicas supervisadas de clasificación usando indicadores de evaluación de modelos

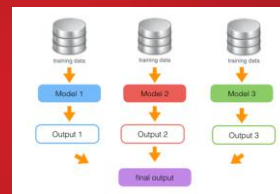
# Modelos de Ensamble

## Contenidos

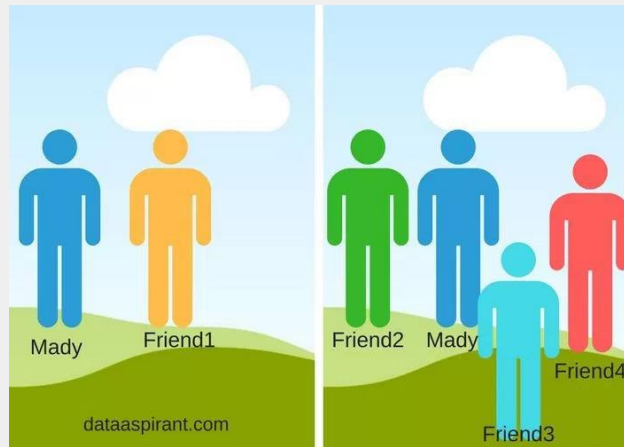
- 12.1 Bagging
- 12.2 Random Forest
- 13.1 Boosting
- 13.2 Selección de variables



## Introducción



## Métodos de Ensamble



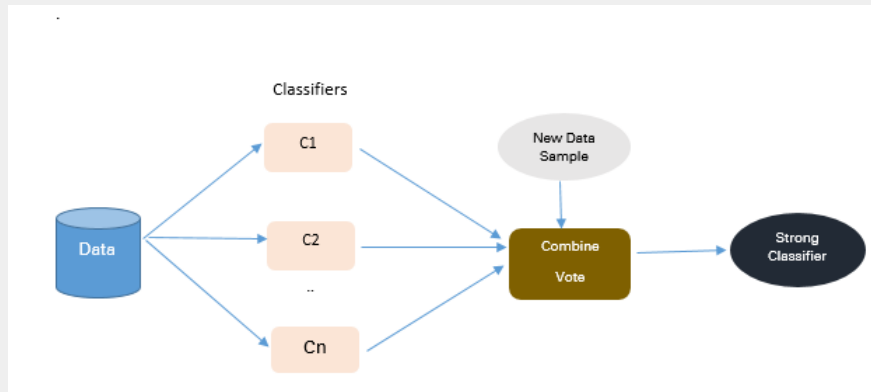
## Métodos de Ensamble

Los métodos de ensamble son técnicas para combinar varios algoritmos de aprendizaje débiles con la finalidad de por construir un algoritmo de aprendizaje más fuerte.



Los métodos de ensamble combinan múltiples hipótesis para formar una mejor hipótesis.

## Métodos de Ensamble



## Características

Evaluar la predicción de un ensamble, requiere mayor computación que un modelo simple.



Algunos de los principales métodos de ensamble son:

Bagging, Boosting y Stacking.



## Bagging

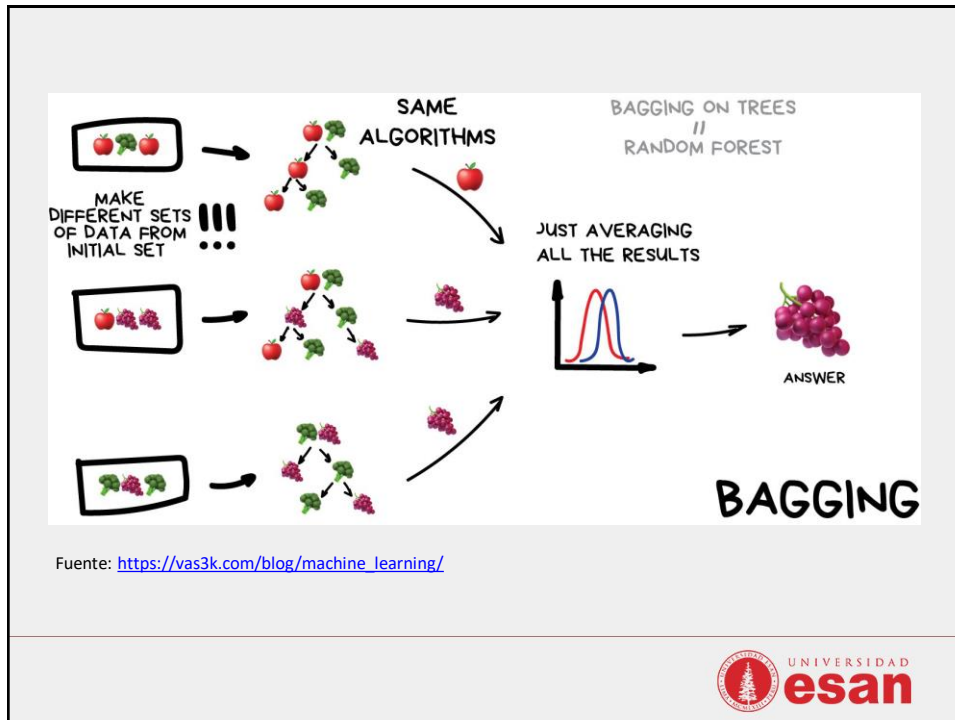


## Introducción

Una forma de mejorar un modelo predictivo es usando la técnica creada por Leo Breiman que denominó Bagging (o Bootstrap Aggregating).



Esta técnica consiste en crear diferentes modelos usando muestras aleatorias con reemplazo y luego combinar o ensamblar los resultados.



## Algoritmo Bagging

Divide los datos de Entrenamiento en distintos subconjuntos de datos, obteniendo como resultado diferentes muestras aleatorias con las siguientes características:

- Muestra uniforme (misma cantidad de individuos en cada set)
- Muestras con reemplazo (los individuos pueden repetirse en el mismo set de datos).
- El tamaño de la muestra es igual al tamaño de los datos de entrenamiento, pero no contiene a todos los individuos ya que algunos se repiten
- Si se usan muestras sin reemplazo, suele elegirse el 50% de los datos como tamaño de muestra

## Algoritmo Bagging

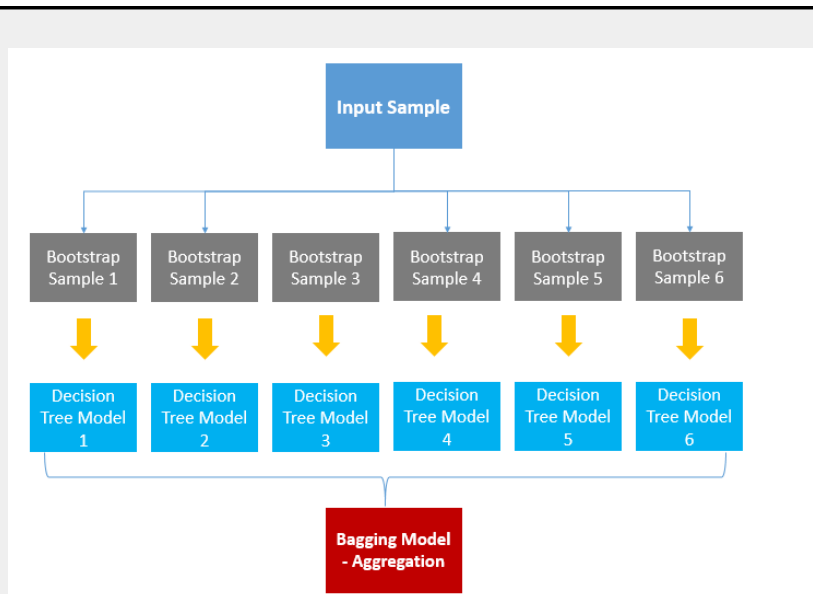
Debido a que se utiliza muestreo con reemplazo, ciertas observaciones ocurrirán más de una vez en una muestra bootstrap, mientras que otras no ocurrirán en absoluto.

$$1 - \left(1 - \frac{1}{n}\right)^n$$

Se puede demostrar que una muestra bootstrap contiene alrededor del 63% de los registros de la data de entrenamiento.

$$1 - \frac{1}{e} \cong 0,63$$

Esto se debe a que cada observación tiene la siguiente probabilidad de ser seleccionada para una muestra bootstrap.





## Random Forest



## Introducción

Random Forest es un algoritmo predictivo que usa la técnica de [Bagging](#) para combinar diferentes arboles, donde cada árbol es construido con observaciones y variables aleatorias.





## Algoritmo Random Forest

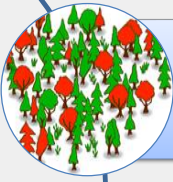


Selecciona individuos al azar (usando muestreo con reemplazo) para crear diferentes set de datos.



Crea un árbol de decisión con cada set de datos, obteniendo diferentes arboles, ya que cada set contiene diferentes individuos y diferentes variables.

## Algoritmo Random Forest



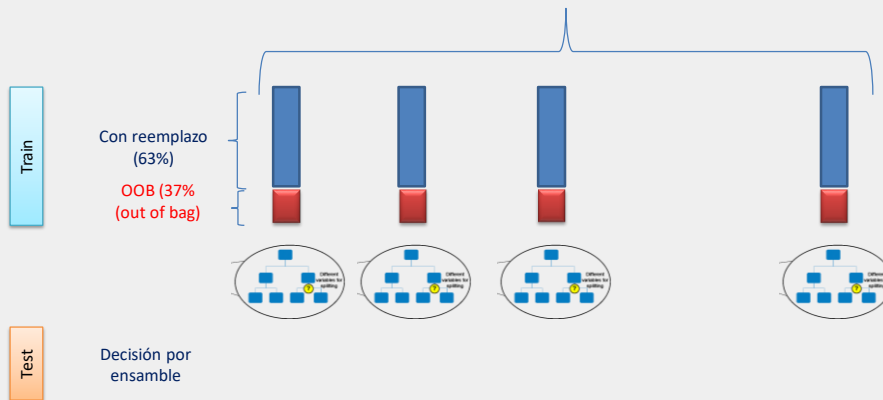
Al crear los arboles se eligen variables al azar en cada nodo del árbol, dejando crecer el árbol en profundidad (sin podar).



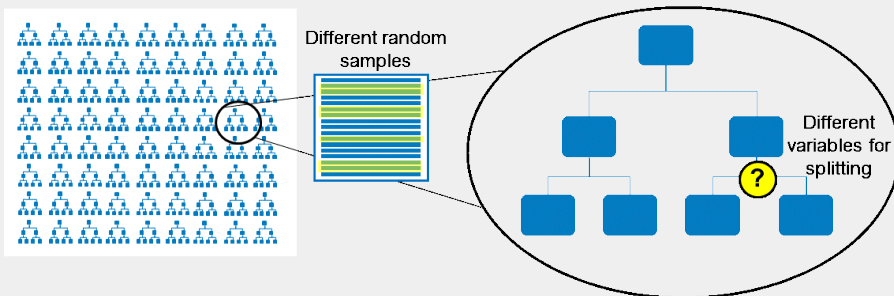
Predice los nuevos datos usando el "voto mayoritario", donde clasificará como "positivo" si la mayoría de los arboles predicen la observación como positiva.

# Algoritmo Random Forest

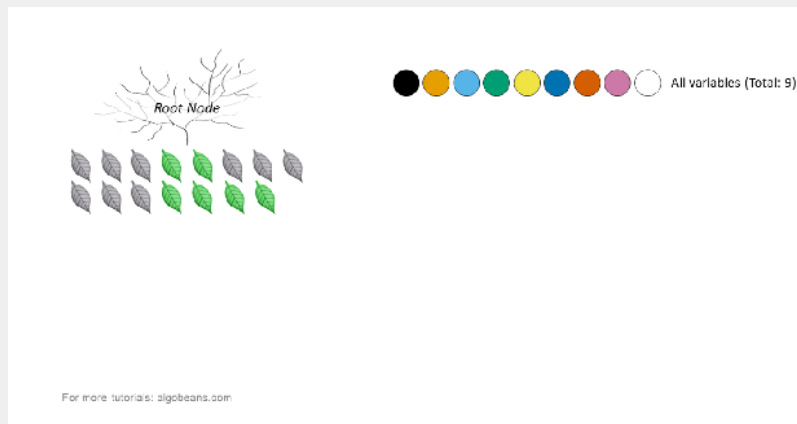
100 árboles con el algoritmo CART sin podar



# Algoritmo Random Forest



## Selección de variables en Random Forest



## Mediciones en Random Forest

OOB Error (out of bag error)

Graficar el error del modelo

Importancia de variables

Reglas de un árbol específico

## OOB error (out of bag error)

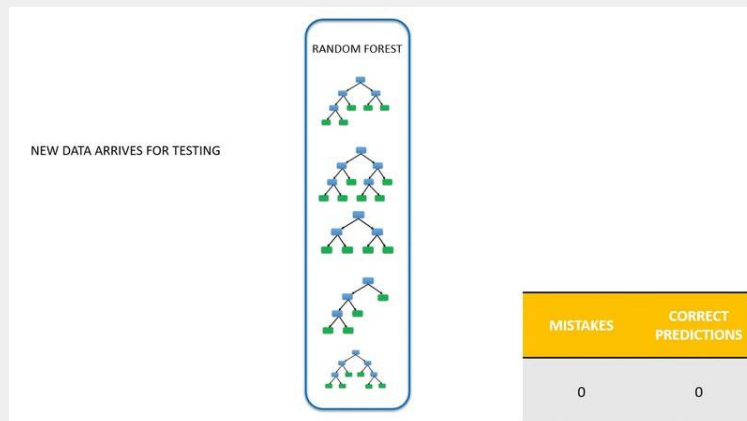
Cuando el algoritmo randomForest selecciona una muestra con reemplazo para crear un árbol en una iteración, algunas observaciones se quedan fuera (out of bag) y no son usadas para crear el árbol.



Para esas observaciones que quedaron fuera del árbol, se hace una predicción y se calcula el error de la predicción. Esto se hace en cada iteración para calcular el error estimado, llamado OOB error.



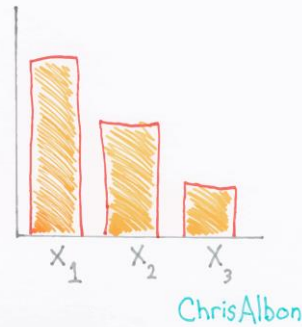
## OOB error (out of bag error)



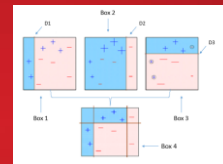
# FEATURE IMPORTANCE

Decision trees make splits that maximize the decrease in impurity.

By calculating the mean decrease in impurity for each feature across all trees we can know that feature's importance.



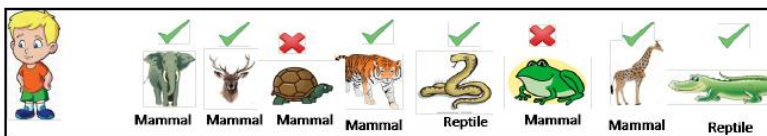
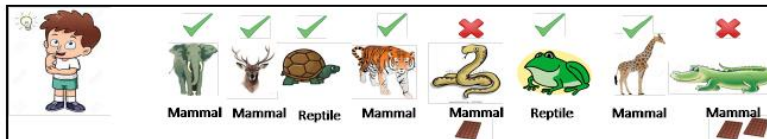
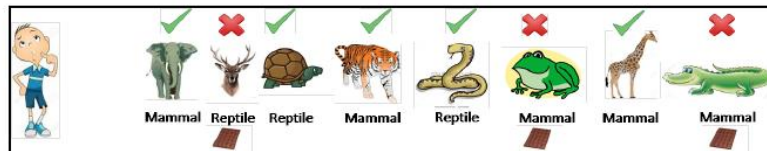
## Boosting

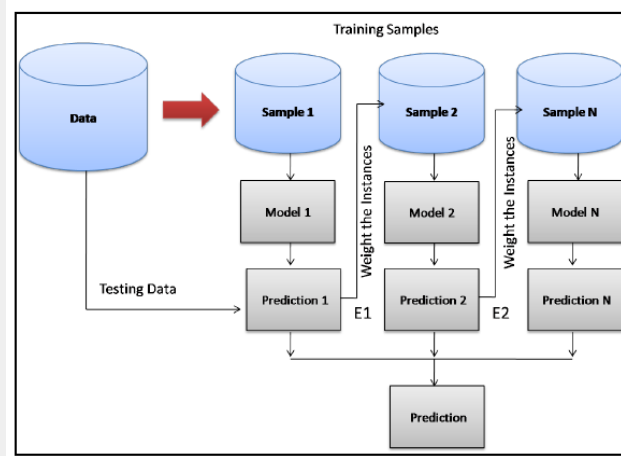


Tomado del libro “Ensemble Machine Learning” de Ankit Dixit (2017)



Animal	Clase
Elefante	Mamífero
Ciervo	Mamífero
Tortuga	Reptil
Tigre	Mamífero
Cobra	Reptil
Sapo	Reptil
Jirafa	Mamífero
Cocodrilo	Reptil

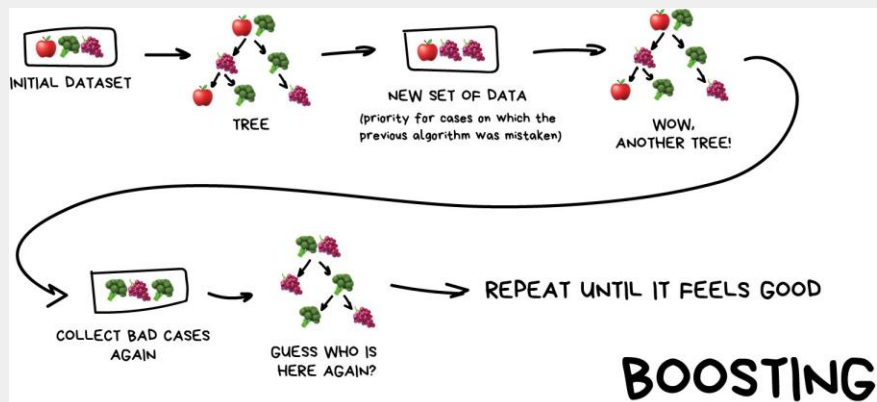




Fuente: Ensemble Machine Learning. Ankit Dixit. Packt. 2017

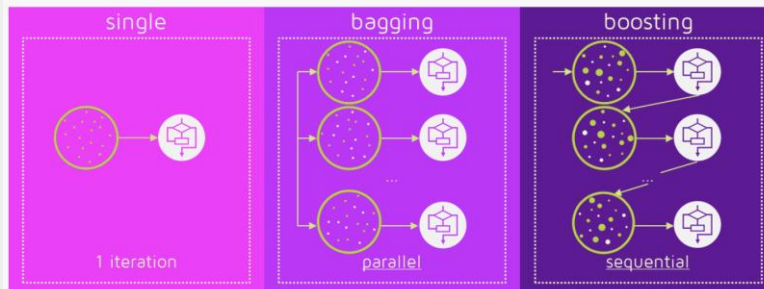
29

## Boosting

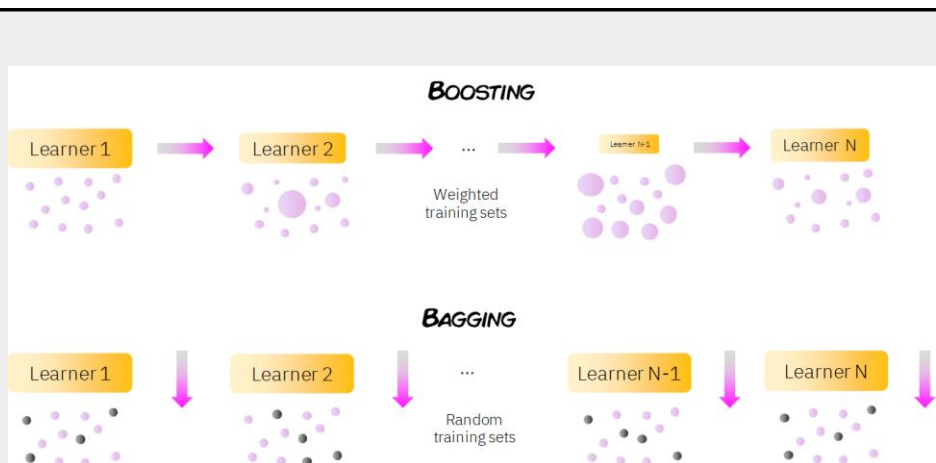


Fuente: [https://vas3k.com/blog/machine\\_learning/](https://vas3k.com/blog/machine_learning/)

## Boosting



Referencia: <https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/>



Fuente: <https://towardsdatascience.com/what-is-boosting-in-machine-learning-2244aa196682>

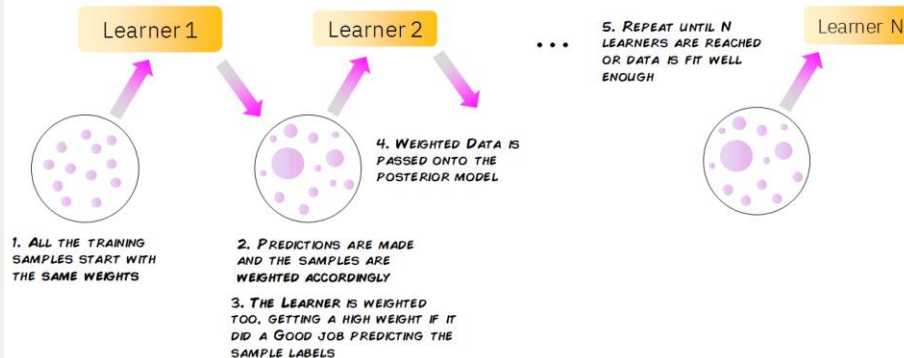


## Algoritmo Boosting

1. Crea muestras aleatorias de los datos de entrenamiento.
2. Entrena un clasificador (Modelo 1) para esta muestra y prueba toda la información de entrenamiento (Sí, todo, no la muestra).
3. Calcula el error para cada predicción de casos. Si el caso está clasificado erróneamente, se aumenta el peso para ese caso y se crea otra muestra (por reemplazo).
4. Repite este procedimiento hasta que se obtenga una alta precisión del sistema.

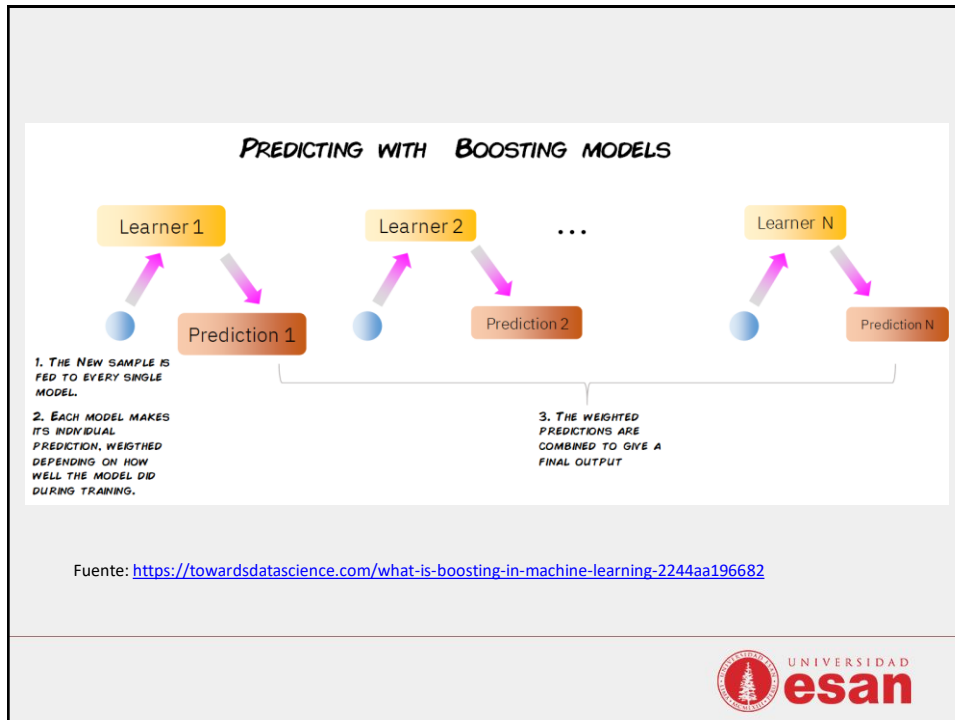


### TRAINING BOOSTING MODELS



Fuente: <https://towardsdatascience.com/what-is-boosting-in-machine-learning-2244aa196682>





## Boosting

- ❖ Los algoritmos originales fueron propuestos por Robert Schapire y Yoav Freund, quienes luego desarrollaron el AdaBoost, un algoritmo adaptativo que ganó el prestigioso Premio Gödel.
- ❖ El término “boosting” hace referencia a una familia de algoritmos que son **capaces de convertir modelos débiles en fuertes**
  - ❖ Un modelo es “débil” cuando tiene una tasa de error importante, pero su rendimiento no es aleatorio (por ejemplo, un 0.5 de tasa de error para un clasificador binario)

## Boosting

- ❖ De manera incremental, la estrategia “boosting” entrena cada modelo con el mismo dataset
  - ❖ Pero con pesos ajustados al error de la última predicción
- ❖ La idea principal es obligar a los modelos a enfocarse en las instancias que dificultan la predicción
- ❖ A diferencia del bagging, el boosting es un método secuencial
  - ❖ Por ello, no se pueden usar las opciones de paralelización



## Boosting

- ❖ Su principal objetivo es reducir el “sesgo”
  - ❖ Por ello, son propensos a sufrir “overfitting”
  - ❖ Así, es fundamental hacer una buena configuración de los parámetros, y así evitar el “overfitting”
- ❖ De esta manera, con la estrategia Boosting:
  - ❖ (1) El primer algoritmo es entrenado con todo el conjunto de datos
  - ❖ (2) El segundo algoritmo, solo procesa los “residuos” (instancias más difíciles) del primero
    - ❖ Le dan más peso a estas instancias que eran más complejas de predecir



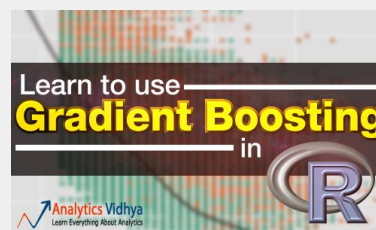
## Boosting

- ❖ Confía en crear una serie de **predictores débiles** que no son buenos para todo el conjunto de datos, pero sí para parte del mismo
- ❖ De esta manera, cada modelo, “*boosts*” (aumenta) el **rendimiento de todo el conjunto de modelos**
- ❖ Algunos ejemplos
  - ❖ XGBoost, GBM, AdaBoost, etc.

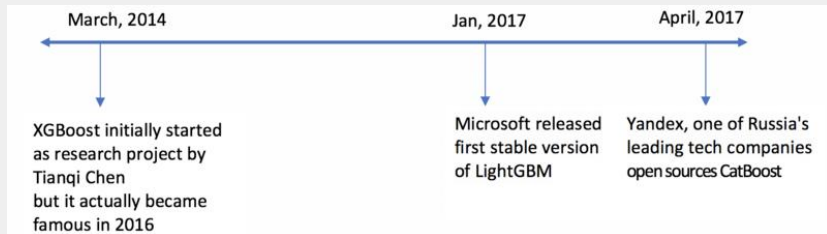


## Algoritmos Boosting

1. AdaBoost (**Adaptive Boosting**)
2. Gradient Boosting Machines
3. Stochastic Gradient Boosting
4. XG Boosting
5. Light Boosting
6. Cat Boosting



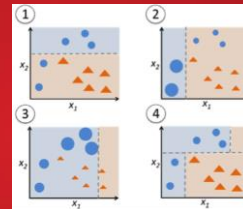
## CatBoost vs. Light GBM vs. XGBoost



Fuente: <https://www.kdnuggets.com/2018/03/catboost-vs-light-gbm-vs-xgboost.html>

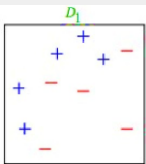


## Algoritmo AdaBoosting



## Algoritmo AdaBoosting (simplificado)

- ❖ Abreviatura de Adaptive Boosting, AdaBoost funciona mediante el proceso de entrenamiento secuencial, prediciendo y actualizando los pesos de las muestras clasificadas erróneamente y de los modelos débiles correspondientes.
- ❖ Se usa principalmente con árboles de decisión **stumps**: árboles de decisión con solo un nodo raíz y dos nodos de salida, donde solo se evalúa una característica de los datos. Como podemos ver, al tener en cuenta solo una característica de nuestros datos para hacer predicciones, cada árbol **stump** es un **weak learner** (clasificador débil). Sin embargo, al combinar muchos de ellos, se puede construir un modelo ensemble muy robusto y preciso.

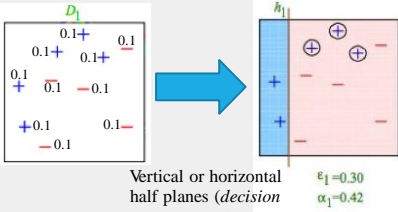


**Fuente:** Material preparado por el Ing. Aldo Meza Rodríguez para el curso de Seminario de Tesis II de la Maestría de Estadística Aplicada de la Universidad Nacional Agraria La Molina.





UNIVERSIDAD  
**esan**



UNIVERSIDAD  
**esan**

$$\begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Vertical or horizontal  
half planes (*decision  
stumps*)

$$\begin{aligned} \epsilon_1 &= 0.30 \\ \alpha_1 &= 0.42 \end{aligned}$$

$\epsilon_n$  = error

$\alpha_n$  = valor de confianza

sign = obtiene el signo de un número

$$\begin{cases} -1 & \text{si número} < 0, \\ 0 & \text{si número} = 0, \\ 1 & \text{si número} > 0. \end{cases}$$

$$D_1$$

0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1

→

$$h_1$$

$$\sum_i w_i' [f(h(x_i) \neq y_i)] \rightarrow \epsilon_1 = 0.30$$

$$\alpha_1 = 0.42$$

$\epsilon_n$  = error

$\alpha_n$  = valor de confianza

sign = obtiene el signo de un número

$$D_1$$

0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1
0.1	+	0.1	+	0.1

→

$$h_1$$

$$\epsilon_1 = 0.30$$

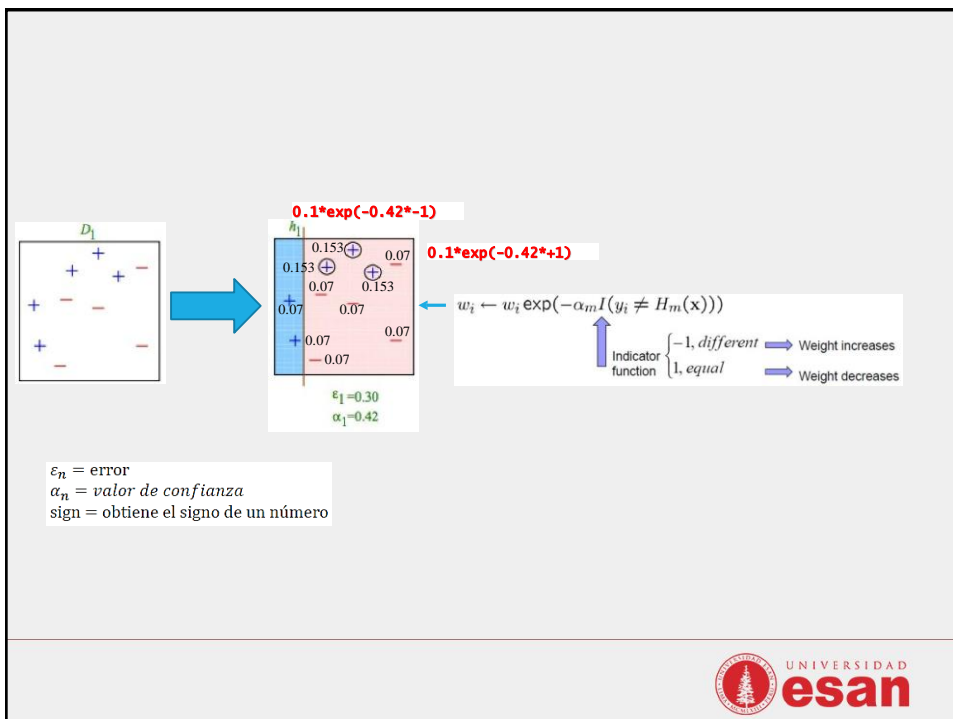
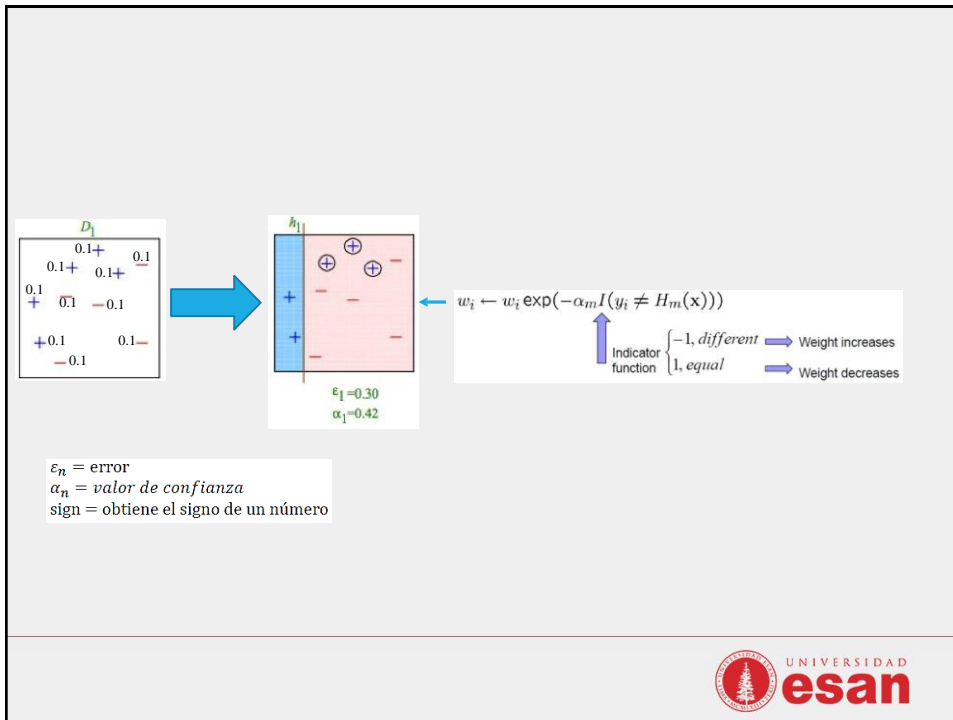
$$\alpha_1 = 0.42$$

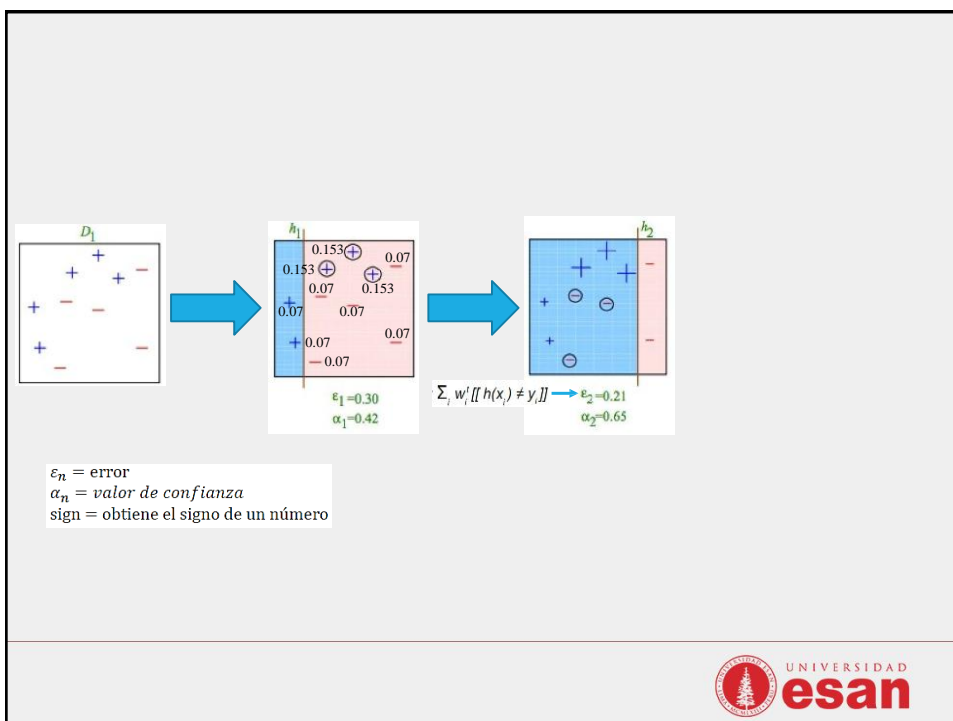
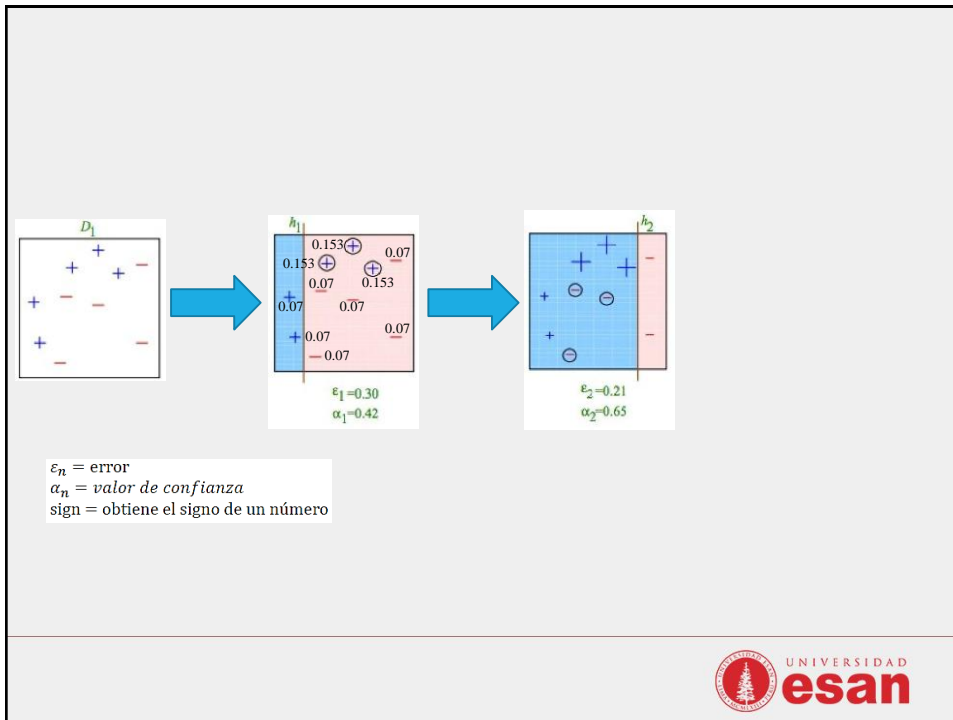
$\epsilon_n$  = error

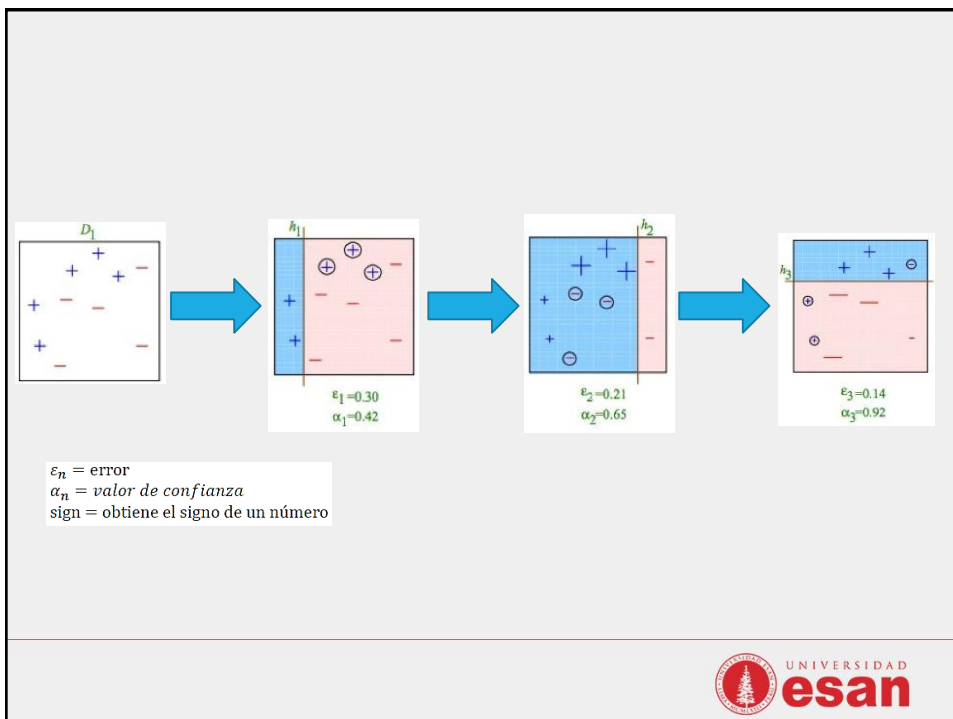
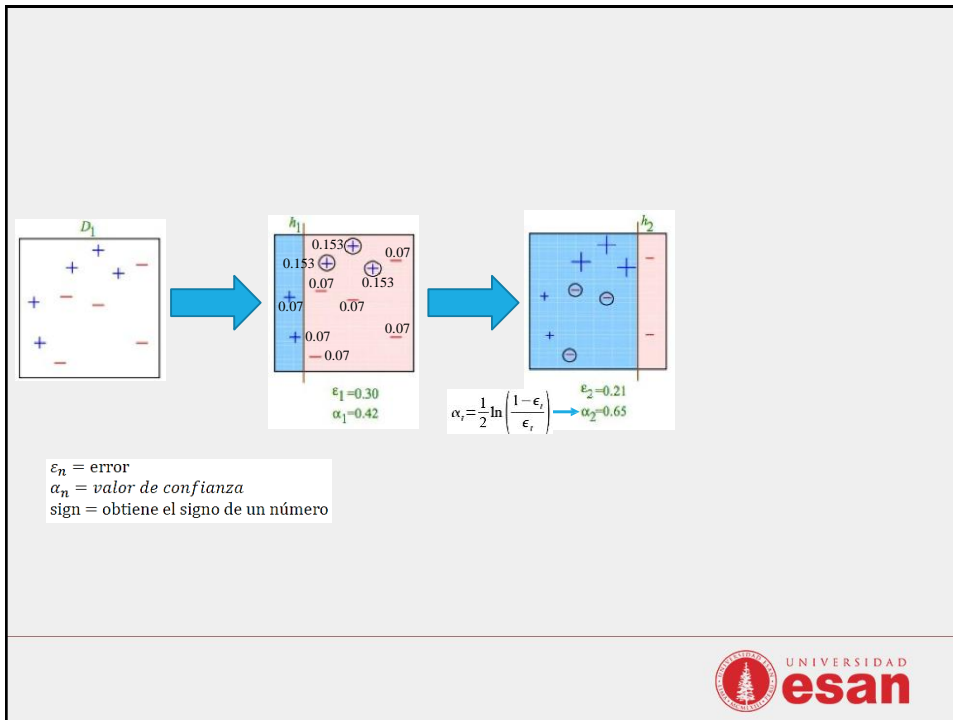
$\alpha_n$  = valor de confianza

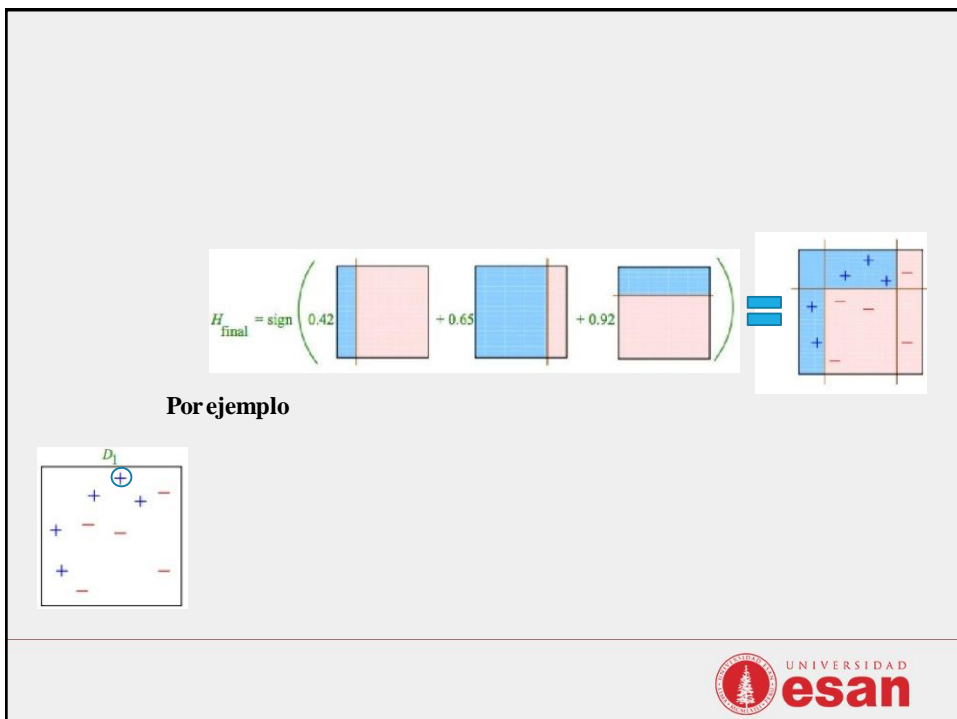
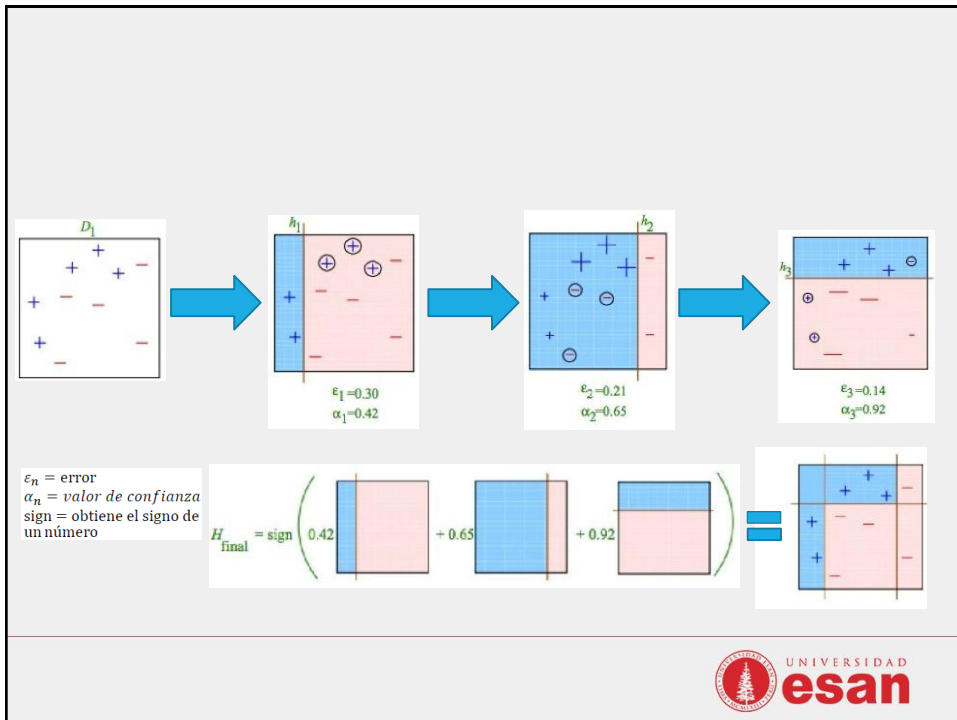
sign = obtiene el signo de un número











$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline & \\ \hline \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline & \\ \hline \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \oplus \rightarrow \text{sign}((0.42 \cdot -1) + (0.65 \cdot 1) + (0.92 \cdot 1)) = \text{sign}(1.15) = 1$$



$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline & 1 \\ \hline \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline & 1 \\ \hline \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \oplus \rightarrow \text{sign}((0.42 \cdot -1) + (0.65 \cdot 1) + (0.92 \cdot 1)) = \text{sign}(1.15) = 1$$




$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline 1 & \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline 1 & \\ \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

**Por ejemplo**

$$D_1 \oplus$$

+	-	+	-
+	-	-	-
+	-	-	-
-	-	-	-

$\rightarrow \text{sign}( (0.42 \cdot -1) + (0.65 \cdot 1) + (0.92 \cdot 1) ) = \text{sign}(1.15) = 1$




$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline 1 & \\ \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline 1 & \\ \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

**Por ejemplo**

$$D_1 \oplus$$

+	-	+	-
+	-	-	-
+	-	-	-
-	-	-	-

$\rightarrow \text{sign}( (0.42 \cdot -1) + (0.65 \cdot 1) + (0.92 \cdot 1) ) = \text{sign}(1.15) = 1 \quad \checkmark$



$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array} \ominus \rightarrow \text{sign}((0.42 * -1) + (0.65 * -1) + (0.92 * 1)) = \text{sign}(-0.15) = -1$$



$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array} \ominus \rightarrow \text{sign}((0.42 * -1) + (0.65 * -1) + (0.92 * 1)) = \text{sign}(-0.15) = -1 \quad \checkmark$$



$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|} \hline 0.42 \\ \hline \end{array} + 0.65 \begin{array}{|c|} \hline 1 \\ \hline \end{array} + 0.92 \begin{array}{|c|} \hline 1 \\ \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \rightarrow \text{sign}((0.42 * -1) + (0.65 * 1) + (0.92 * -1)) = \text{sign}(-0.69) = -1$$

$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|} \hline 0.42 \\ \hline \end{array} + 0.65 \begin{array}{|c|} \hline 1 \\ \hline \end{array} + 0.92 \begin{array}{|c|} \hline 1 \\ \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \rightarrow \text{sign}((0.42 * -1) + (0.65 * 1) + (0.92 * -1)) = \text{sign}(-0.69) = -1 \quad \checkmark$$



$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array} \rightarrow \text{sign}((0.42*1) + (0.65*1) + (0.92*-1)) = \text{sign}(0.15) = 1$$

$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|c|} \hline 0.42 & \\ \hline \hline \end{array} + 0.65 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} + 0.92 \begin{array}{|c|c|} \hline 1 & \\ \hline \hline \end{array} \right) = \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array}$$

Por ejemplo

$$D_1 \begin{array}{|c|c|c|c|} \hline + & + & + & - \\ \hline + & - & - & - \\ \hline + & - & - & - \\ \hline \end{array} \rightarrow \text{sign}((0.42*1) + (0.65*1) + (0.92*-1)) = \text{sign}(0.15) = 1 \quad \checkmark$$

$$H_{\text{final}} = \text{sign} \left( \begin{array}{|c|} \hline 0.42 \\ \hline \end{array} + 0.65 \begin{array}{|c|} \hline 1 \\ \hline \end{array} + 0.92 \begin{array}{|c|} \hline 1 \\ \hline \end{array} \right) = \begin{array}{|c|c|c|} \hline + & + & + \\ \hline + & - & - \\ \hline + & - & - \\ \hline \end{array}$$

Por ejemplo

$$\begin{array}{lcl}
 \begin{array}{|c|} \hline D_1 \\ \hline \end{array} & \rightarrow & \text{sign}(0.42 \cdot 1 + 0.65 \cdot 1 + 0.92 \cdot 1) = \text{sign}(1.15) = 1 \\
 \begin{array}{|c|} \hline + \\ \hline \end{array} & \rightarrow & \text{sign}(0.42 \cdot (-1) + 0.65 \cdot 1 + 0.92 \cdot 1) = \text{sign}(-0.15) = -1 \\
 \begin{array}{|c|} \hline - \\ \hline \end{array} & \rightarrow & \text{sign}(0.42 \cdot (-1) + 0.65 \cdot 1 + 0.92 \cdot (-1)) = \text{sign}(-0.69) = -1 \\
 \begin{array}{|c|} \hline + \\ \hline \end{array} & \rightarrow & \text{sign}(0.42 \cdot 1 + 0.65 \cdot 1 + 0.92 \cdot (-1)) = \text{sign}(0.15) = 1
 \end{array}$$



## Algoritmo AdaBoosting

Given:  $(x_1, y_1), \dots, (x_m, y_m)$  where  $x_i \in \mathcal{X}$ ,  $y_i \in \{-1, +1\}$ .

Initialize:  $D_1(i) = 1/m$  for  $i = 1, \dots, m$ .

For  $t = 1, \dots, T$ :

- Train weak learner using distribution  $D_t$ .
- Get weak hypothesis  $h_t : \mathcal{X} \rightarrow \{-1, +1\}$ .
- Aim: select  $h_t$  to minimize the weighted error:

$$\epsilon_t \doteq \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose  $\alpha_t = \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$ .
- Update, for  $i = 1, \dots, m$ :

$$\begin{aligned}
 D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\
 &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},
 \end{aligned}$$


where  $Z_t$  is a normalization factor (chosen so that  $D_{t+1}$  will be a distribution).

Output the final hypothesis:

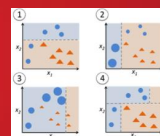

$$H(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(x) \right).$$

Fuente: Boosting. Foundations and Algorithms. Robert E. Schapire Yoav Freund. 2012





## Algoritmo Gradient Boosting Machine (GBM)

## Gradient Boosting Machine (GBM)

1. GBM entrena a cada uno de los clasificadores débiles (**weak learners**) secuencialmente, agregando más y más estimadores, pero en lugar de adaptar los pesos de los datos, intenta predecir los errores residuales cometidos por los estimadores anteriores.
2. Debido a esto, ya no tenemos pesos de muestra, y todos los modelos débiles tienen la misma cantidad de importancia. Nuevamente, la mayoría de las veces, los árboles de decisión se usan como predictores básicos, sin embargo, no son árboles pequeños, sino árboles más grandes y de tamaño fijo.
3. Los GBM usan una tasa de aprendizaje y dan pequeños pasos hacia mejores resultados, de manera conceptual y similar a lo que se hace en **Gradient Descent**.

## Elementos de un Gradient Boosting (1)

1. Una función pérdida a ser optimizada
2. Un clasificador débil para hacer predicciones
3. Un modelo aditivo para añadir clasificadores débiles para minimizar la función pérdida.

(1) Referencia: <https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>



## 1. Función Pérdida

- ❖ Minimizar la función de pérdida significa maximizar la exactitud del clasificador.
- ❖ La función pérdida usada depende del tipo de problema. clasificador débil para hacer predicciones

Rooted Mean Squared Error for regression

$$L = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

LogLoss for binary classification

$$L = -\frac{1}{N} \sum_{i=1}^N (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

mllogloss for multi-classification

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M y_{i,j} \log(p_{i,j})$$

$$p_i = \frac{1}{1 + e^{-y_i}}$$



## 2. Clasificador débil

- ❖ Los árboles de decisión se utilizan como clasificadores débiles
- ❖ Específicamente, se utilizan árboles de regresión que generan valores reales para divisiones y cuya producción se puede sumar, lo que permite agregar salidas de modelos posteriores y "corregir" los residuos en las predicciones.



## 3. Modelo aditivo

- ❖ Los árboles se agregan uno a la vez, y los árboles existentes en el modelo no se cambian.
- ❖ Un procedimiento de descenso de gradiente se utiliza para minimizar la pérdida al agregar árboles.
- ❖ Tradicionalmente, el descenso de gradiente se utiliza para minimizar un conjunto de parámetros, como los coeficientes en una ecuación de regresión o ponderaciones en una red neuronal. Después de calcular el error o la pérdida, los pesos se actualizan para minimizar ese error.



### 3. Modelo aditivo

- ❖ En lugar de parámetros, tenemos submodelos de aprendizaje débiles o, más específicamente, árboles de decisión. Después de calcular la pérdida, para realizar el procedimiento de descenso de gradiente, debemos agregar un árbol al modelo que reduce la pérdida (es decir, seguir el gradiente). Hacemos esto parametrizando el árbol, luego modificamos los parámetros del árbol y nos movemos en la dirección correcta (reduciendo la pérdida residual).
- ❖ En general, este enfoque se denomina descenso funcional de gradiente o descenso de gradiente con funciones.



### Algoritmo para Gradient Boosting

```

1  Initialized all predictions to the sample log-odds:  $f_i^{(0)} = \log \frac{\hat{p}}{1-\hat{p}}$ .
2  for iteration  $j = 1 \dots M$  do
3      Compute the residual (i.e. gradient)  $z_i = y_i - \hat{p}_i$ 
4      Randomly sample the training data
5      Train a tree model on the random subset using the residuals as
        the outcome
6      Compute the terminal node estimates of the Pearson residuals:
        
$$r_i = \frac{1/n \sum_i^n (y_i - \hat{p}_i)}{1/n \sum_i^n \hat{p}_i (1 - \hat{p}_i)}$$

7      Update the current model using  $f_i = f_i + \lambda f_i^{(j)}$ 
8  end

```





## Algoritmo XGBoosting




## XGBoost

- ❖ Al igual que en los GBM, se ajustan los árboles a los residuos de las predicciones de árboles anteriores, sin embargo, en lugar de usar árboles de decisión de tamaño fijo convencionales, XGBoost usa un tipo diferente de árboles: **árboles XGBoost**.
- ❖ Construye estos árboles calculando puntajes de similitud entre las observaciones que terminan en un nodo de salida. Además, XGBoost permite la regularización, reduciendo el posible sobreajuste de nuestros árboles individuales y, por lo tanto, del modelo de conjunto general.
- ❖ XGBoost está optimizado para superar el límite de los recursos computacionales de los algoritmos de árbol potenciados, lo que lo convierte en un algoritmo de alto rendimiento y rápido en términos de tiempo y cálculo.

## XGBoosting

- ❖ Extreme Gradient Boosting (xgboost) es similar al Gradient Boosting pero más eficiente.
- ❖ Tiene algoritmos de resolución de modelo lineal y aprendizaje de árbol.
- ❖ Lo que lo hace rápido es su capacidad para hacer procesamiento en paralelo en una sola PC.
- ❖ Esto hace que xgboost sea al menos 10 veces más rápido que los algoritmos de Gradient Boosting existentes.
- ❖ Es compatible con varias funciones objetivas, incluida la regresión y la clasificación.



## Ventajas del XGBoosting

- ❖ **Regularización:**
- ❖ La implementación estándar de GBM no tiene regularización como XGBoost, por lo tanto, también ayuda a reducir el sobreajuste.
- ❖ De hecho, XGBoost también se conoce como técnica de "refuerzo regularizado".
- ❖ **Procesamiento en paralelo:**
- ❖ XGBoost implementa el procesamiento paralelo y es sorprendentemente más rápido en comparación con GBM.





## Ventajas del XGBoosting

- ❖ **Alta flexibilidad**
- ❖ XGBoost permite a los usuarios definir objetivos de optimización personalizados y criterios de evaluación.
- ❖ **Manejo de valores perdidos**
- ❖ XGBoost tiene una rutina incorporada para manejar los valores perdidos.
- ❖ El usuario debe proporcionar un valor diferente al de otras observaciones y pasarlo como parámetro. XGBoost encuentra un valor perdido en cada nodo y aprende qué camino tomar para los valores perdidos en el futuro.



## Ventajas del XGBoosting

- ❖ **Poda de árboles:**
- ❖ Un GBM dejaría de dividir un nodo cuando encuentre una pérdida negativa en la división.
- ❖ XGBoost por otro lado hace divisiones hasta la max\_depth especificada y luego comienza a podar el árbol hacia atrás y elimina divisiones más allá de las cuales no hay una ganancia positiva.



## Ventajas del XGBoosting

- ❖ **Cross-Validation incorporado**
- ❖ XGBoost permite al usuario ejecutar una validación cruzada en cada iteración del proceso de refuerzo y, por lo tanto, es fácil obtener el número óptimo exacto de iteraciones de refuerzo en una sola ejecución.
- ❖ Esto es diferente a GBM donde se tiene que ejecutar una búsqueda en grid y solo se pueden probar valores limitados.



## Parámetros Booster

- ❖ **nrounds**. El valor por defecto es 100. Controla el número máximo de iteraciones. Para la clasificación, es similar a la cantidad de árboles para crecer. Debe ser ajustado usando CV
- ❖ **eta**: el valor predeterminado se establece en 0.3. Controla la tasa de aprendizaje, es decir, la velocidad a la que nuestro modelo aprende patrones en los datos. Después de cada ronda, reduce los pesos de las características para alcanzar el mejor óptimo. Bajo **eta** conduce a un cálculo más lento. Debe ser apoyado por el aumento de **nrounds**. El rango es de 0 a 1. Por lo general, se encuentra entre 0.01 - 0.3.



## Parámetros Booster

- ❖ **gamma:**
- ❖ El valor predeterminado se establece en 0. El rango es de 0 a  $\infty$ .
- ❖ Controla la regularización (o evita el sobreajuste). El valor óptimo de gamma depende del conjunto de datos y otros valores de parámetros.
- ❖ Cuanto mayor sea el valor, mayor será la regularización. Regularización significa penalizar grandes coeficientes que no mejoran el rendimiento del modelo. default = 0 significa que no hay regularización.
- ❖ Truco de tuning: comience con 0 y compruebe la tasa de error de CV. Si ve que train error >>> test error, ponga gamma en acción. Cuanto más alto sea el gamma, menor será la diferencia en el train y el CV de test. Si no tiene idea de qué valor usar, use gamma = 5 y vea el rendimiento. Recuerde que gamma trae mejoras cuando desea usar árboles poco profundos (max\_depth bajos).



## Parámetros Booster

- ❖ **max\_depth:** el valor predeterminado se establece en 6. Controla la profundidad máxima de un árbol. Cuanto mayor la profundidad, más complejo el modelo y se corre el riesgo de caer en overfitting. El rango es de 1 a  $\infty$ .
- ❖ **min\_child\_weight:** el valor predeterminado se establece en 1. El rango es de 0 a  $\infty$ . En la clasificación, si el nodo de hoja tiene una suma mínima de peso de instancia (calculado por derivada parcial de segundo orden) menor que **min\_child\_weight**, la división de árbol se detiene. En palabras simples, bloquea las posibles interacciones de características para evitar el sobreajuste. Debe ser ajustado usando CV.



## Parámetros Booster

- ❖ **max\_delta\_step**: el valor predeterminado se establece en 0. Paso delta máximo que permitimos que sea la estimación de peso de cada árbol. Si el valor se establece en 0, significa que no hay restricción. Si se establece en un valor positivo, puede ayudar a que la actualización sea más conservadora. Por lo general, este parámetro no es necesario, pero podría ayudar en la regresión logística cuando la clase está extremadamente desequilibrada. Establecerlo en un valor de 1 a 10 podría ayudar a controlar la actualización. El rango es de 0 a  $\infty$ .



## Parámetros Booster

- ❖ **subsample**: el valor predeterminado se establece en 1. Se debe especificar la proporción de submuestras de la instancia de capacitación. Establecerlo en 0.5 significa que XGBoost recolectó aleatoriamente la mitad de las instancias de datos para hacer crecer los árboles y esto evitará el sobreajuste. El rango es de 0 a 1.
- ❖ **colsample\_bytree**: el valor predeterminado se establece en 1. Se debe especificar la proporción de columnas de la submuestra al construir cada árbol. Controla la cantidad de características (variables) suministradas a un árbol. El rango es de 0 a 1. Por lo general los valores van de 0.5 a 0.9.



## Parámetros específicos del Booster Lineal

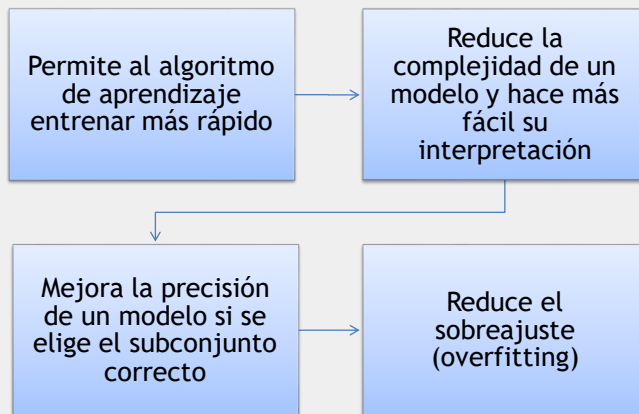
- ❖ **lambda y alpha**: estos son términos de regularización en pesos. El valor predeterminado de Lambda es 1 y alpha es 0.
- ❖ **lambda\_bias**: término de regularización L2 en el sesgo y tiene un valor predeterminado de 0.



## Selección de Variables



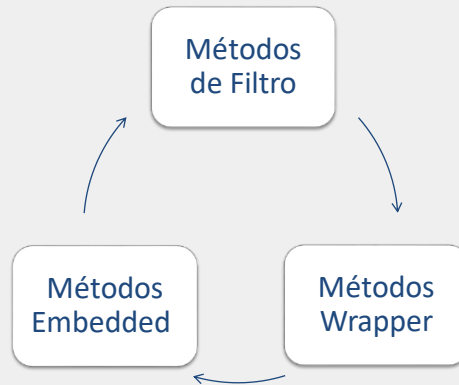
## Importancia de la selección de variables



## Métodos para Seleccionar Variables



## Métodos para seleccionar variables



## Métodos de Filtro

- ❖ Los métodos de filtro se usan generalmente en la etapa de preprocesamiento.
- ❖ La selección de variables es independiente de cualquier algoritmo de aprendizaje automático.
- ❖ En cambio, las variables se seleccionan en función de sus puntajes con pruebas estadísticas para su correlación con la variable respuesta.

## Métodos de Filtro

Variable Predictora	Variable Respuesta	
	Continua	Categórica
Continua	Correlación de Pearson	LDA
Categórica	ANOVA	Chi-Cuadrado



## Métodos Wrapper

- ❖ Se usa un subconjunto de variables y se entrena un modelo para usarlas. Con base a las inferencias que se obtiene del modelo anterior, se decide agregar o eliminar variables al subconjunto.
- ❖ El problema se reduce a un problema de búsqueda.
- ❖ Estos métodos son computacionalmente muy costosos.
- ❖ Algunos ejemplos son: selección de variables hacia adelante, selección de variables hacia atrás, eliminación recursiva de variables y el algoritmo Boruta.

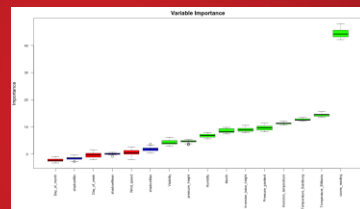


## Métodos Embedded

- ❖ Combina las cualidades de los métodos de filtro y de los métodos Wrapper.
- ❖ Se implementa mediante algoritmos que tienen sus propios métodos de selección de funciones incorporados.
- ❖ Algunos de los ejemplos más populares de estos métodos son la regresión LASSO y RIDGE que tienen funciones de penalización incorporadas para reducir el sobreajuste.



## Algoritmo Boruta



## Ventajas

- ❖ Funciona bien para problemas de clasificación y regresión.
- ❖ Tiene en cuenta las relaciones multivariadas.
- ❖ Es una mejora en la medida de la importancia de variables de Random Forest.



## Ventajas

- ❖ Sigue un método de selección de variables totalmente relevante en el que considera todas las variables que son relevantes para la variable respuesta.
- ❖ En cambio, la mayoría de los otros algoritmos de selección de variables siguen un método óptimo mínimo donde se basan en un pequeño subconjunto de variables que produce un error mínimo en el clasificador elegido.
- ❖ Puede manejar interacciones entre variables

## Algoritmo Boruta

Realiza la mezcla de los valores de las variables predictoras y las une con las variables predictoras originales.

Crea un Random Forest con el conjunto de datos fusionado.

Luego hace una comparación de las variables originales con las variables aleatorias para medir la importancia de la variable.

Solo se consideran importantes las variables que tienen mayor importancia que las variables aleatorias.

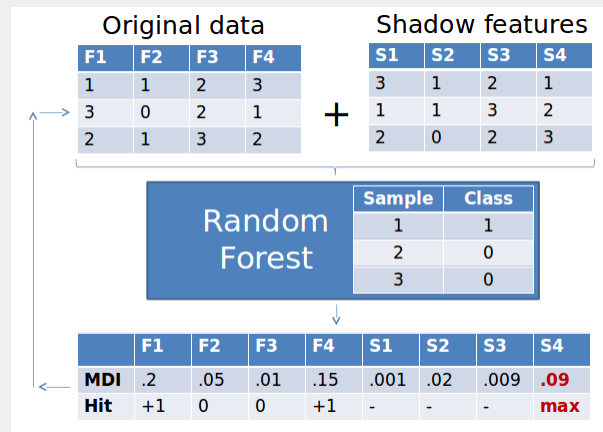


## Algoritmo Boruta

- ❖ Primero, duplica el conjunto de datos y mezcla los valores en cada columna. Estos valores se llaman variables de sombra.
- ❖ Luego, entrena un clasificador (Random Forest) en el conjunto de datos. Al hacer esto, se asegura de que pueda hacerse una idea de la importancia - mediante el **Mean Decrease Accuracy** o el **Mean Decrease Impurity** - para cada una de las variables del conjunto de datos. Cuanto mayor sea el puntaje, mejor o más importante.



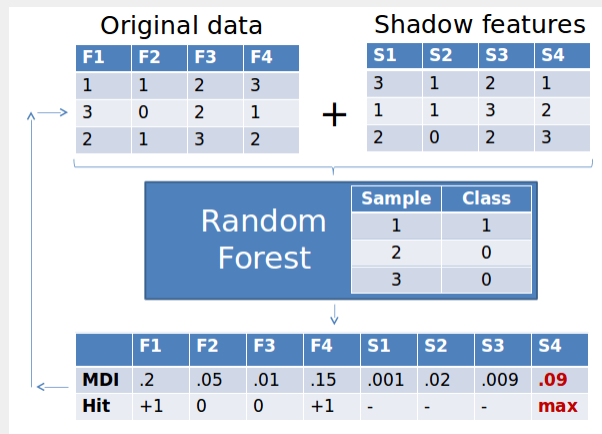
## Algoritmo Boruta



## Algoritmo Boruta

- ❖ Se calcula el puntaje Z. El **Mean of Accuracy Loss** dividido entre la desviación estándar del Accuracy Loss.
- ❖ Luego, el algoritmo verifica cada una de sus variables originales si tienen una mayor importancia.
- ❖ Es decir, si la variable tiene un puntaje Z más alto que el puntaje Z máximo de la mejor de las variables sombras. Si es así, lo registra en un vector. Estos se llaman éxitos. Luego, continuará con otra iteración.
- ❖ Después de un conjunto predefinido de iteraciones, el algoritmo terminará con una tabla de éxitos.

## Algoritmo Boruta



## Algoritmo Boruta

- ❖ En cada iteración, el algoritmo compara los puntajes Z de las copias mezcladas de las variables y de las variables originales para ver si este último se desempeñó mejor que el anterior.
- ❖ Si lo hace, el algoritmo marcará la variable como importante.
- ❖ En esencia, el algoritmo está tratando de validar la importancia de la variable mediante la comparación con copias mezcladas al azar, lo que aumenta la solidez.
- ❖ Esto se hace comparando el número de veces que una variable mejoró con las variables de sombra usando una distribución binomial.

## Ejercicio en aula

Para el caso práctico desarrollado en clase, resuelva e interprete cada uno de los ejercicios propuestos.



## Referencias Unidad III

### Contenido 12 y 13. Modelos de Ensamble

- Cichosz, P. Data Mining Algorithms: Explained Using R. Cap. 15 pp. 401-430

