Clase 7

# Desbalance de Clases

Junio - Julio de 2023

PhD. Débora Chan

# Organización

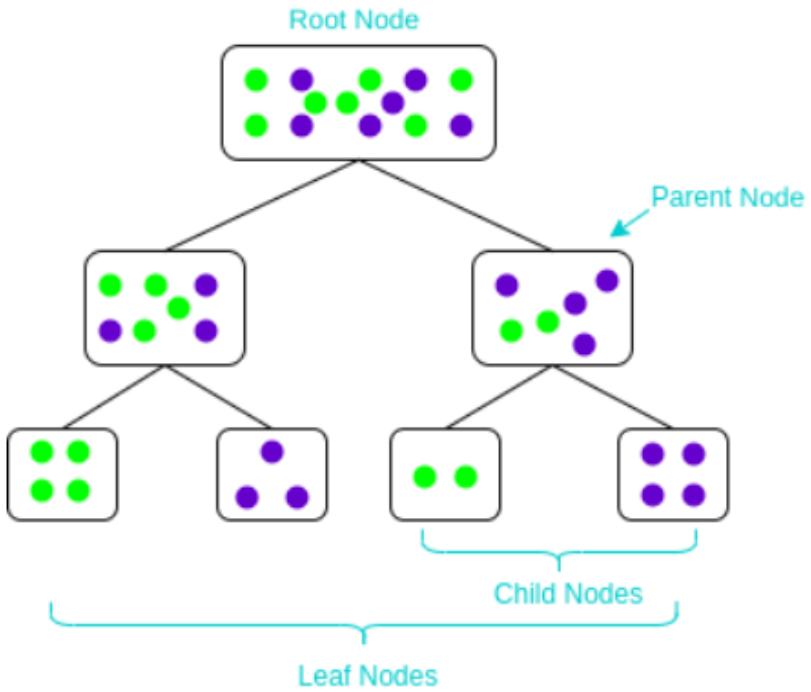
## Qués son?

Los árboles de decisión o clasificación son un modelo de predicción surgido en el ámbito del machine learning (aprendizaje automático), que partiendo de una base de datos crea diagramas de construcciones lógicas que nos ayudan a resolver problemas. También sirven como base para otros algoritmos de aprendizaje automático complicados y ampliamente utilizados, como Random Forest , XGBoost y LightGBM.

El árbol predice la variable dependiente con base en el aprendizaje de reglas de decisión inferidas desde las características que poseen los datos; si la variable dependiente es categórica decimos que es un árbol de clasificación, y si es numérica es un árbol de regresión

# Construcción

Para construir un árbol de clasificación se usa el método recursive binary splitting. Existen varias alternativas, todas ellas con el objetivo de encontrar nodos lo más puros/homogéneos posible.



# División binaria recursiva

## En qué consiste?

Es computacionalmente inviable considerar todas las posibles particiones del predictor, por lo que se opta por una división binaria recursiva (recursive binary splitting), similar al concepto de stepwise selection, consiguiendo también obtener buenos resultados.

El proceso comienza en el punto más alto del árbol (donde todas las observaciones pertenecen a una misma región), y de manera sucesiva se divide el espacio del predictor, donde cada división genera dos nuevas ramas. Se optará además por la mejor división en cada paso, según un criterio, sin tener en cuenta si dicha división mejorará el árbol en futuros pasos o divisiones.

# Nodos

Esta sub-división de los nodos pretende generar nodos relativamente puros (homogéneos). Hay varios criterios para estas subdivisiones cuando la variable objetivo es categórica:

1. **Impureza de Gini**

Es la forma más popular y sencilla de dividir un árbol de decisión. El valor de la impureza de Gini es:

$$Imp.Gini = 1 - Gini = 1 - \sum_{i=1}^{n} p_i^2$$

Cuanto menor sea la impureza de Gini, mayor será la homogeneidad del nodo. La impureza de Gini de un **nodo puro es cero**.

2 **Ganancia de información**: Funciona sobre el concepto de la entropía y viene dado por:

$$Ganancia_{informacion} = 1 - Entropia$$

La entropía se utiliza para calcular la pureza de un nodo. A menor entropía, mayor será la pureza del nodo.
La entropía de un nodo **homogéneo** es **cero**. Dado que restamos la entropía de 1, la ganancia de información es mayor para los nodos más puros con un valor máximo de 1. La expresión de la entropía es:

$$Entropia = \sum_{i=1}^{n} p_i log_2(p_i)$$

## 3 Chi-cuadrado

Funciona en la significación estadística de las diferencias entre el nodo principal y los nodos secundarios. El valor de Chi cuadrado es:

$$Chisq = \sqrt{\frac{obs - esp}{esp}}$$

Esta fórmula nos da el valor de Chi-cuadrado para una clase. Se considera la suma de los valores de Chi-Cuadrado para todas las clases en un nodo para calcular el Chi-Cuadrado para ese nodo. Cuanto mayor sea el valor, mayores serán las diferencias entre los nodos padre e hijo, es decir, mayor será la homogeneidad (el esperado es el del nodo padre y el observado es el del nodo hijo).

# Error de clasificación

Fracción de observaciones de entrenamiento en una región o nodo que no pertenece a la clase más frecuente. Se define como:

$$EC = 1 - max_{k,m}(\widehat{p}_{mk})$$

donde $\widehat{p}_{mk}$ es la proporción de observaciones de entrenamiento en la región m que pertenecen a la clase k.
El error de clasificación no suele ser lo suficientemente sensible en medir la pureza de los nodos para el crear el árbol. Aun así, gini, cross entropy o EC pueden usarse para el la poda del árbol. El error de clasificación es preferible si el objetivo es conseguir la máxima predicción en las predicciones del árbol podado final.

# Podado del árbol (Pruning)

## En qué consiste?

El proceso de división binaria recursiva puede conseguir buenas predicciones con los datos de entrenamiento, ya que reduce el criterio objetivo en este conjunto. Sin embargo esto puede conducir a un sobreajuste, reduciendo la capacidad predictiva para nuevos datos.

Una posible alternativa es construir un árbol hasta el punto en el criterio alcance un cierto nivel generalmente alto. Con esta estrategia obtendremos árboles más pequeños, con menos ramificaciones. Concretamente, la estrategia consistirá en construir un árbol grande y luego podarlo para obtener un sub-árbol que consiga el menor test error, obtenido mediante validación cruzada.

# Ventajas de los Árboles de Clasificación

## Ventajas

- Fácil interpretación.
- Representación gráfica muy intuitiva y posible aun cuando hay más de 3 predictores.
- Fácil manejo de predictores cualitativos sin la necesidad de crear variables dummy.
- No hay necesidad de asumir a priori ninguna relación entre las variables, pudiendo introducir relaciones muy complejas entre las mismas.
- Seleccionan los predictores en forma automática.

# Desventajas de los Árboles de Clasificación

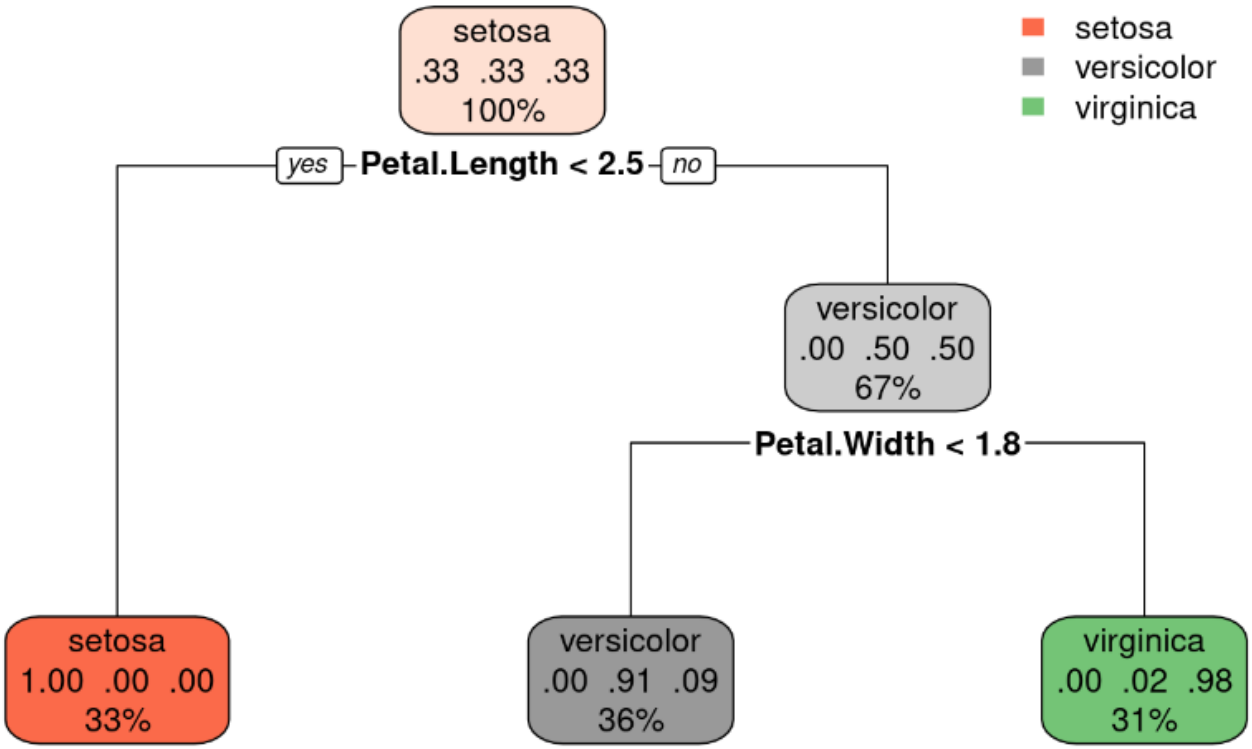
## Alternativa de solución

- Capacidad predictiva superable por otros métodos de regresión/clasificación.
- Este en tipo de clasificación 'débil', pues sus resultados pueden variar mucho dependiendo de la muestra de datos usados para entrenar un modelo.

## Alternativa de solución

Sin embargo, mediante la agregación de varios árboles de decisión, métodos como bagging, random forests y boosting, la capacidad predictiva de los árboles puede mejorarse sustancialmente.

# Interpretación del Árbol de clasificación



# Ver ejemplo 1 RMarkdown

Detección de Spam

# Organización

# Problemas de clasificación

Hay muchos ejemplos de clasificación donde los datos de una de las categorías están muy desbalanceados con la otra (u otras). Algunos ejemplos con los que nos encontramos con mucha frecuencia son:

1. Detección de spam.
2. Detección de fraudes.
3. Predicción de abandono.
4. Detección de positivos (en salud) inmersos en una gran cantidad de negativos.

**En cada uno de estos casos, la clase minoritaria tiene una frecuencia significativamente menor que la e mayoritaria. Sin embargo, la predicción de la clase minoritaria es más importante.**

# ¿Cómo afecta a los algoritmos el desbalance de clases?

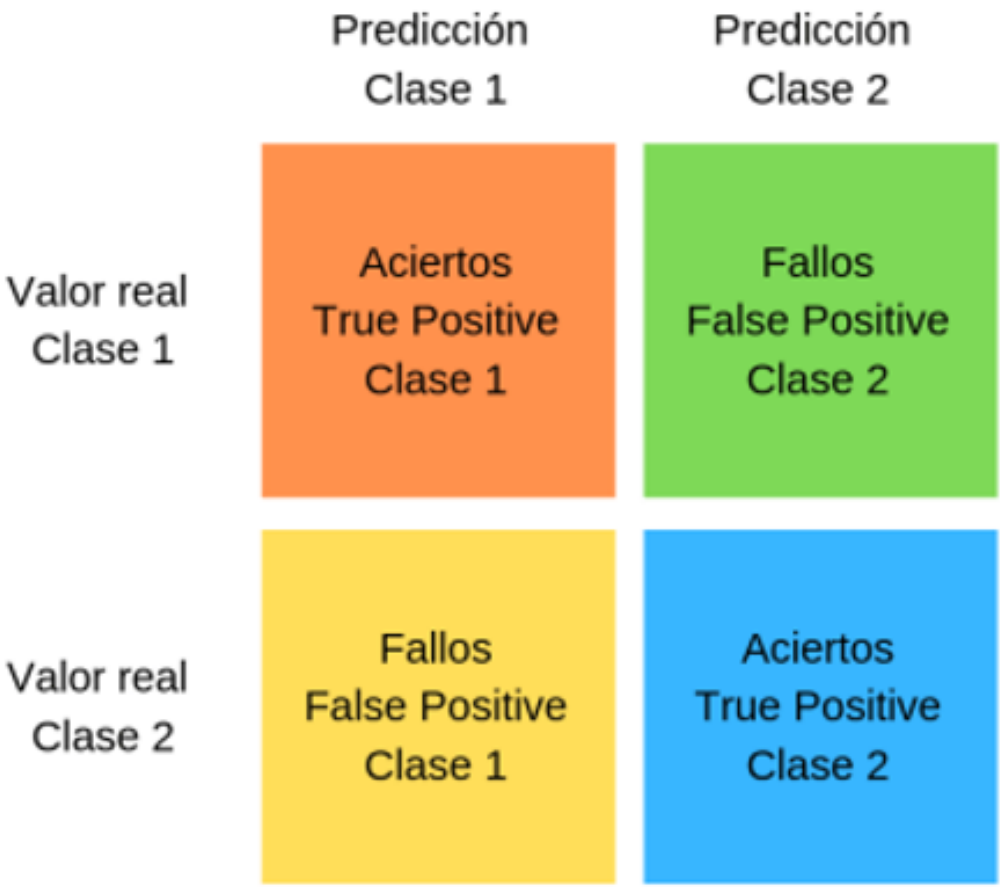
## Qué hace el algoritmo frente al desbalance?

Si le damos a un algoritmo una base de datos con 990 sanos y 10 enfermos, no podemos pretender que el algorimo detecte esta clase. Lo más probable es que siempre clasifique como sano y luego dirá que el acierto es del 99 %, al menos en su etapa de entrenamiento.

Esta supuesta efectividad, puede darnos la falsa idea de que el modelo funciona correctamente.

# ¿Cómo nos permiten darnos cuenta de esto las métricas?

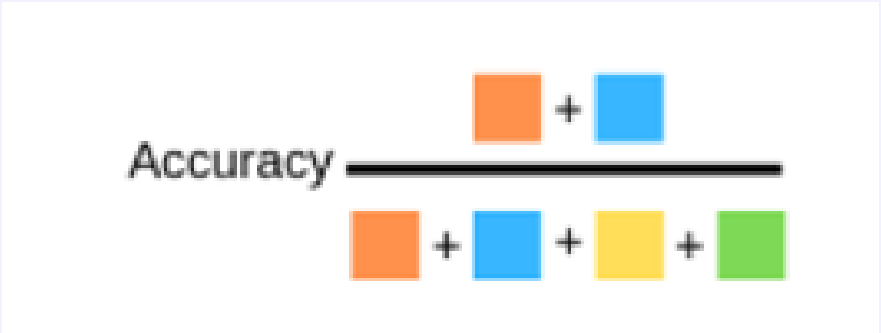
## Matriz de Confusión



# Algunas métricas

## Accuracy

La Accuracy del modelo es básicamente la proporción de predicciones correctas del modelo. En el ejemplo da 99 % aún cuando no hemos logrado identificar a ninguno de la clase minoritaria.
La Precisión de una clase define cuan confiable es un modelo en responder si un punto pertenece a esa clase. Para la clase mayoritaria será del 99 % sin embargo para la minoritaria es del 0 %.



# Algunas métricas

## Recall

El Recall de una clase expresa qué tan bien el modelo detecta a los individuos de esta clase. Para la clase mayoritaria, en nuestro ejemplo, será de 1 y para la minoritaria de 0. En medicina se denomina **sensibilidad** cuando se trata de detectar los casos positivos y **especificidad** cuando se trata de detectar los casos negativos

# Algunas métricas

## Precisión

La Precisión de una nos dice qué tan confiable es un modelo cuando indica que un individuo pertenece a una determinada clase. Para nuestro ejemplo en la clase mayoritaria es del 99 % sin embargo para la clase minoritaria es del 0 %.


En medicina para la clase de positivos se llama VPP (valor predictivo positivo) y para los negativos se denomina VPN (valor predictivo negativo).

# Algunas métricas

## F1-Score

El F1 Score de una clase es la media armónica entre la de precisión y recall. Su expresión simbólica es:

$$F1 - Score = \frac{2\text{precision recall}}{\text{precision} + \text{recall}}$$


Es decir que es una combinación de las métricas de precisión y recall, para la clase minoritaria daría 0.

# Diferentes combinaciones de valores

## Posibles Casos

Las cuatro combinaciones posibles de medida son las siguientes:

- **Alta Precición y Alto Recall**: el modelo maneja perfectamente ambas clases.
- **Alta Precición y Bajo Recall**: el modelo no detecta muy bien la clase, pero si lo logra es confiable.
- **Baja Precición y Alto Recall**: el modelo no detecta muy bien la clase, pero si lo logra es confiable.
- **Baja Precición y Bajo Recall**: el modelo no clasifica correctamente.

.

# ¿Cómo dan las métricas en casos desbalanceados?

## Las métricas

Cuando el dataset está desbalanceado, suele suceder que se obtiene un alto valor de precisión para la clase mayoritaria y un bajo valor de recall para la clase minoritaria.

## Las Estrategias

Además, se dispone de diversas estrategias para tratar de mejorar la clasificación para los casos de datasets desbalanceados.
Las vamos a presentar brevemente a continuación.

# Estrategia I: Ajuste de Parámetros del modelo

Consiste en ajustar parámetros ó métricas del propio algoritmo para intentar equilibrar a la clase minoritaria penalizando a la clase mayoritaria durante el entrenamiento.

Se puede implementar este ajuste de peso en árboles, en regresión logística con el parámetro class_weight= 'balanced' (veremos luego un ejemplo).
Sin embargo, no todos los algoritmos tienen estas posibilidades, en ese caso podemos ajustar la métrica objetivo.

Por ejemplo, redes neuronales por ejemplo podríamos ajustar la métrica de Loss para que penalice a las clases mayoritarias.

# Estrategia II: Modificar el Dataset

## ¿Cómo lo modificamos?

- Podemos eliminar muestras de la clase mayoritaria para reducirlo e intentar equilibrar la situación.
- El 'peligro' es prescindir de muestras importantes, que brindan información y por lo tanto empeorar la calidad del modelo.
- Para seleccionar qué muestras eliminar, deberíamos seguir algún criterio, esto podría sesgar al modelo.
- También podríamos agregar nuevas filas con los mismos valores de las clases minoritarias, pero esto podría conducir al modelo al overfitting.

# Remuestreo

## En Síntesis

Las técnicas de muestreo: agregan patrones o quitan patrones y se pueden clasificar en:

- Sobremuestreo (oversampling): crea muestras vinculadas al patrón de la clase minoritaria.
- Submuestreo (undersampling): descarta registros de la clase mayoritaria con algún criterio o regla definida.
- Híbrido: combina los dos primeros.

# Estrategia III: Muestras Artificiales

- Podemos intentar generar muestras sintéticas (no idénticas) utilizando diversos algoritmos que intentan seguir la tendencia del grupo minoritario.
- Según el método elegido para esta simulación, podemos mejorar los resultados.
- El peligros es crear muestras sintéticas que pueden alterar la distribución 'natural' de esa clase y confundir al modelo en su clasificación.

# Estrategia IV: Métodos de Ensamble Balanceados

## Para qué ensamblar?

Se aprovechan las ventajas del ensamble de metodologías; es decir, entrenar diversos modelos y entre todos ellos obtener el resultado final aplicando una 'votación', por ejemplo.

Hay que asegurarse de que al tomar las muestras de entrenamiento éstas estén equilibradas.

# Otros Métodos disponibles menos comunes

## Recomendación

Ver cuál se ajusta al problema

- Usar algoritmos de Boosting: estos modelos por definición suelen centrarse en mejorar los errores que cometen. Por ejemplo, en un XGBoost, aumentando el número de árboles, podemos ir corrigiendo los errores de los árboles anteriores.
- Darle más peso a las muestras de la clase minoritaria: la regresión logística permite ponderar en mayor medida los elementos según la clase que sean. Dándole mayor peso a los elementos de la clase minoritaria se centrará en ajustarse mejor a esa clase y así, predecir mejor.
- Usar algoritmos de Stacking y algoritmos de aprendizaje por refuerzo: del mismo modo que los Boosting, estos algoritmos permiten ir mejorando los aciertos de la clase minoritaria.

# Ver Ejemplos 2 y 3 de RMarkdown

# Organización

# Objetivos

## Bagging Boosting Random Forest

Nos permiten mejorar sustancialmente el rendimiento predictivo de modelos basados en árboles, aunque con una considerable reducción de la interpretabilidad del modelo final.

Estos métodos también se conocen como métodos de ensemble o métodos combinados, que son los que utilizan múltiples algoritmos de aprendizaje para obtener predicciones que mejoren las que se podrían obtener por medio de cualquiera de los algoritmos individuales.

**Son aplicables a muchos métodos de aprendizaje estadísticos (no solo árboles de decisión) para clasificación.**

# Bagging

## Problema

Los árboles de decisión sufren de alta varianza, lo que significa que, si dividiéramos al azar los datos de entrenamiento en dos grupos y ajustáramos un árbol de decisión a cada mitad, los resultados que obtendríamos podrían ser bastante diferentes.

## Alternativa

Por el contrario, un procedimiento o método con baja varianza dará resultados parecidos aun aplicándose sobre sets de datos distintos. El método de bagging o bootstrap aggregation es un procedimiento utilizado para reducir la varianza de un método de aprendizaje estadístico, usado muy frecuentemente con árboles de decisión.

# Cómo se aplica bagging en clasificación?

**La forma más simples es:**

Dada una observación de test, podemos obtener la clase predicha por cada uno de B árboles, y escoger como predicción final para dicha observación la clase de mayor frecuencia de entre las B predicciones ( de cada árbol).

---

**El número de árboles (B)** a crear no es un parámetro crítico a la hora de aplicar bagging. Ajustar un gran número de árboles no aumentará el riesgo de overfitting, por lo que usaremos un número lo suficientemente alto como para alcanzar la estabilización en la reducción del test error.

# Error Out of Bag(OOB)

## Para estimar el error de clasficación

No es necesario utilizar validación cruzada o simple pues:

- Cada árbol generado por bootstrapping usa alrededor de 2/3 de las observaciones; el tercio restante se conoce como out of bag(OOB).
- Podemos obtener la i-esima predicción para la i-esima observación usando cada uno de los árboles para los cuales esta observación es OOB.
- Con un número suficientemente elevado de árboles (B) el error OOB puede llegar a ser equivalente al error de validación leave-one-out. Además, el método basado en OOB para estimar el test error resulta conveniente cuando se aplica bagging en sets de datos grandes, para los cuales aplicar la validación cruzada sería computacionalmente muy costoso.

# Limitaciones en el uso Out-of-Bag Error

## OOB

- El Out-of-Bag Error no es adecuado cuando las observaciones tienen una relación temporal (series temporales). Como la selección de las observaciones que participan en cada entrenamiento es aleatoria, no respetan el orden temporal y se estaría introduciendo información a futuro.
- El preprocesado de los datos de entrenamiento se hace de forma conjunta, por lo que las observaciones out-of-bag pueden sufrir data leakage. De ser así, las estimaciones del OOB-error son demasiado optimistas.

# Importancia de las variables

- Al combinar múltiples árboles mediante bagging, ya no es posible representar gráficamente el modelo resultante mediante un árbol.
- Se pierde de esta manera la posibilidad de identificar cuáles son las variables más importantes. Por tanto, bagging mejora la predicción del modelo a expensas de la pérdida de interpretabilidad.
- Sin embargo, un modo de poder identificar los predictores tienen mayor importancia es cuantificar la reducción de Gini lograda como resultado de las subdivisiones de ese predictor sobre los B árboles.
- El predictor será más importante cuanto mayor resulte esa reducción promedio.

# Random forests

## Limitaciones del Bagging

Si en un set de datos un predictor es muy important, todos o casi todos los árboles generados por bagging usarán este predictor en la primer subdivisión y esto hará que sean muy similares las particiones y por ende las predicciones. Así no lograremos una mejor clasificación!

## Alternativa

El método de random forests proporciona una mejora a los árboles combinados por bagging en cuanto a que los descorrelaciona, teniendo en cuenta solo un subgrupo de predictores en cada división.
Se construye una cantidad de árboles de decisión a partir de pseudomuestras, pero se escogen entre los p predictores, sólo m para cada árbol ($m \approxeq \sqrt{p}$).

# Cuándo conviene?

- RF logra que en una cantidad de divisiones no se considere a ese predictor importante.
- Usar un número pequeño de m para aplicar random forests puede resultar útil cuando contamos con un gran número de predictores correlacionados.
- Al igual que con bagging, aumentar el número de pseudo-árboles no incrementará el riesgo de overfitting, por lo que en la práctica usamos un valor lo suficientemente alto para conseguir una reducción significativa del error de clasificación.

# Boosting

**Cómo funciona?**

De manera similar al bagging combina un gran número de árboles, pero lo hace en este caso enforma secuancial usando en cada caso información del anterior. Son árboles más pequeños para evitar el sobreajuste.

Otra diferencia es que boosting no utiliza remuestreo por bootstrapping, sino que cada árbol se genera utilizando una versión modificada del set de datos original.

# Los Parámetros del Boosting

## Los Parámetros principales del algoritmo son:

- **Número de árboles (B)**. B se selecciona por validación cruzada; pues a diferencia del bagging y random forests, boosting puede sobreajustarse a los datos si el número de árboles es demasiado alto.
- **Número de divisiones (d)** en cada árbol, que controla el nivel de complejidad. Un valor de d=1 (cada árbol contiene una única división, es decir, un único predictor) suele dar buenos resultados.
- **Parámetro de penalización ($\lambda$)** es la velocidad con el que boosting aprende. Valores comunes para este parámetro suelen ser 0,01 o 0,001, aunque la decisión depende del problema en cuestión. Por ejemplo, un valor muy pequeño de $\lambda$ puede requerir un número elevado de árboles para conseguir buenos resultados.

# Algoritmos de Boosting

Los más utilizados son:

- AdaBoost
- Gradient Boosting
- Stochastic Gradient Boosting

# AdaBoost

## Adaboost (Adaptive Boosting)

Aplicado fundamentalmente a problemas de clasificación, es un algoritmo iterativo que se basa en combinar múltiples **weak learners**, en un único **strong learner** a través de una combinación lineal ponderada. En cada iteración se genera un **weak learner** con un peso asociado. En un escenario de clasificación binaria:

- Se codifican los niveles de la variable respuesta categórica como +1 y -1.
- Se asocia un mismo peso inicial ($w_i$) a todas las observaciones $i$ de entrenemiento.
- Se elige el tipo de modelo para generar los weak learners.
- Se asigna peso a cada uno de los week learners en función de su proporción de aciertos.

# Gradient Boosting

## Gradient Boosting

El proceso iterativo de Gradient Boosting no asigna un peso independiente a cada observación de entrenamiento, sino que hace uso de una función de coste L(yi,f(x)) cuyo gradiente o derivada parcial de la función de coste se pretende minimizar.

- El gradiente se utiliza para encontrar la dirección en la que cambiar los parámetros de los weak learners para reducir el error de predicción en las siguientes iteraciones.
- Se permite el uso de cualquier función de coste diferenciable (que admita derivadas en cualquier dirección).
- Para problemas de clasificación los pasos se repiten K veces en cada iteración m, una por cada clase y luego se obtiene un acoplamiento de los K clasificadores distintos.

# Stochastic Gradient Boosting

## Stochastic Gradient Boosting

Es una variación del Gradient Boosting en el sentido que cada iteración hace uso de un subconjunto aleatorio y sin reemplazo de los datos de entrenamiento disponibles para ajustar los weak learners.

Esto, además de proporcionar un extra de aleatoriedad, permite obtener el error out-of-bag, lo que permite no tener que recurrir a la validación cruzada para la optimización de hiperparámetros cuando los requerimientos computacionales son limitantes.

# Observación Importante

## Clave

La clave para que los métodos de ensemble consigan mejores resultados que cualquiera de sus modelos individuales es que, los modelos que los forman, sean lo más diversos posibles (sus errores no estén correlacionados).

Una analogía para comprender mejor esta observación es: si en un juego de equipos tienen que acertar preguntas sobre temáticas diversas, tendrá mejores chances un equipo integrado por muchos jugadores, cada uno experto en un tema distinto, que un equipo formado por muchos jugadores expertos en un mismo tema o por un único jugador que sepa un poco de todos los temas.

# Predicción de probabilidades

La mayoría de implementaciones de Random Forest, entre ellas la de ranger, permiten predecir probabilidades cuando se trata de problemas de clasificación.

Esta predicción nos da información sobre la seguridad con la que el modelo realiza esta asignación.

Con predict(type="prob"), en lugar de una clasificación, se obtiene la probabilidad con la que el modelo considera que cada observación puede pertenecer a cada una de las clases.

# Ver Ejemplo 4 en RMarkdown