Semana 6

INTRODUCCIÓN A MACHINE LEARNING

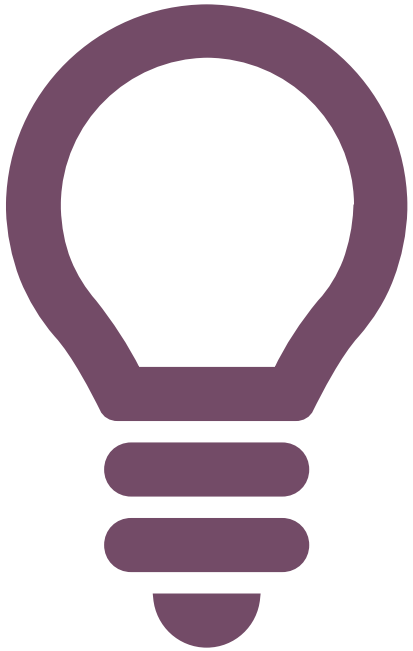
ENSEMBLE METHODS

¿QUÉ ES UN ENSAMBLE?

Concepto básico

- Un **ensamble** en el contexto de Machine Learning es una técnica que combina múltiples modelos individuales (también llamados **modelos base** o **modelos débiles**) para formar un **modelo más robusto y preciso**.
- En lugar de entrenar un solo clasificador o regresor, entrenamos varios y combinamos sus predicciones (por votación, promedio, ponderación, etc.).

¿POR QUÉ USAR UN ENSAMBLE?



Ningún modelo individual es perfecto. Los errores que comete un modelo pueden ser diferentes a los errores que cometen otros. La idea central es que **la combinación de modelos ayuda a compensar los errores de unos con los aciertos de otros.**

TIPOS DE ERRORES QUE UN ENSAMBLE AYUDA A REDUCIR

- **Error por varianza**: modelos muy sensibles a los datos (como árboles de decisión) pueden cambiar drásticamente ante pequeñas variaciones en el conjunto de entrenamiento.
- **Error por sesgo**: modelos muy simples que no capturan la complejidad del problema.
- **Error por ruido**: datos aleatorios o no informativos pueden afectar el modelo.

Un buen ensamble puede **reducir la varianza** (principal ventaja del bagging) y, en algunos casos, también **reducir el sesgo** (como hace el boosting).

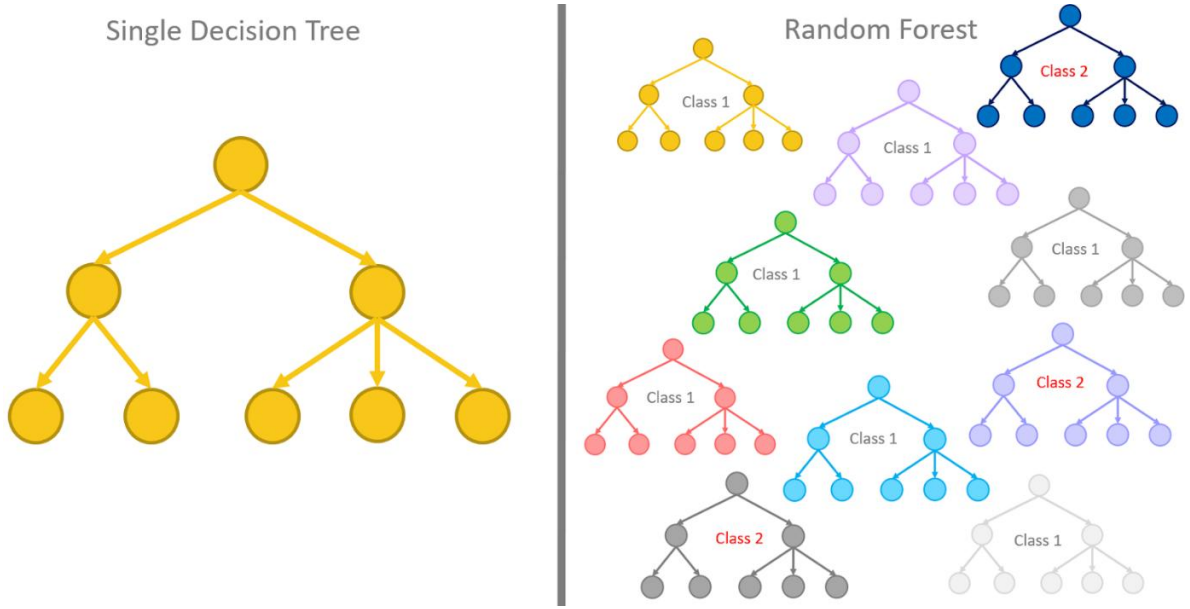
BENEFICIOS DE LOS MÉTODOS DE ENSAMBLE

- Mejor precisión y generalización.
- Mayor robustez ante sobreajuste.
- Capacidad para combinar modelos heterogéneos o complementarios.

BAGGING Y BOOSTING

En el aprendizaje automático, **bagging** y **boosting** son dos estrategias fundamentales para construir modelos de ensamble. Ambos comparten el objetivo de mejorar el rendimiento predictivo al combinar múltiples modelos débiles, pero lo hacen de manera muy diferente.

Un **modelo débil** es aquel que tiene un desempeño apenas mejor que el azar, como un árbol de decisión simple (shallow tree). Tanto bagging como boosting usan varios modelos débiles para construir un modelo fuerte.



¿QUÉ ES UN MODELO DÉBIL?

¿QUÉ ES BAGGING?

Bagging (*Bootstrap Aggregating*) es una técnica que consiste en:

- Entrenar **varios modelos en paralelo**, cada uno sobre una muestra distinta del conjunto de datos original.
- Estas muestras se crean mediante **bootstrap**, es decir, muestreo aleatorio **con reemplazo**.

Luego, se combinan las predicciones de todos los modelos:

- Por mayoría de votos (en clasificación).
- Por promedio (en regresión).

OBJETIVO PRINCIPAL DEL BAGGING

Reducir la **varianza** del modelo.

¿CÓMO AYUDA EL BAGGING?

Bagging combate la varianza entrenando muchos modelos similares (por ejemplo, árboles) en diferentes subconjuntos aleatorios del conjunto de entrenamiento.

Luego:

- **Promedia** (regresión) o **vota** (clasificación) sus predicciones.
- Así, los errores aleatorios de cada modelo tienden a **cancelarse entre sí**.
- El resultado es un modelo **más estable y menos sensible al ruido**.

¿QUÉ ES BOOSTING?

Boosting es una técnica que consiste en:

- Entrenar **varios modelos de forma secuencial**, donde **cada modelo intenta corregir los errores del anterior**.
- Se asigna mayor peso a las observaciones mal clasificadas, forzando al siguiente modelo a enfocarse en ellas.
- Al final, las predicciones se combinan, típicamente mediante una suma ponderada.

OBJETIVO PRINCIPAL DEL BOOSTING

Reducir el **sesgo** del modelo.

BAGGING: BOOTSTRAP AGGREGATION

Bootstrap es un método estadístico de **re-muestreo con reemplazo**, utilizado para estimar la distribución de un estadístico (media, varianza, etc.) a partir de una única muestra.

Definición formal

- Bootstrap consiste en generar múltiples subconjuntos aleatorios del mismo tamaño que el conjunto de datos original, seleccionando observaciones **con reemplazo**.

¿QUÉ SIGNIFICA "CON REEMPLAZO"?

Cada vez que se elige una muestra, **no se elimina del conjunto original**.

Esto permite que un mismo dato pueda aparecer varias veces en una misma muestra bootstrap.

¿QUÉ ES EL TEOREMA DEL LÍMITE CENTRAL?

El **Teorema del Límite Central** (TLC) es un teorema fundamental en estadística que dice lo siguiente:

> Si tomamos muchas muestras aleatorias de un mismo tamaño desde una población, y calculamos la media de cada muestra, entonces **la distribución de esas medias se aproxima a una distribución normal**, **aunque la población original no sea normal**.

Esto ocurre cuando:

- El tamaño de muestra es suficientemente grande ($n \geq 30$ suele ser una regla práctica).
- Las muestras son independientes y aleatorias.

BOOTSTRAP Y TLC

Gracias al TLC, sabemos que esa distribución de estimadores converge hacia una normal (bajo ciertas condiciones), lo que permite:

- Estimar la variabilidad.
- Calcular intervalos de confianza sin conocer la distribución real.
- Validar modelos de forma interna (como con el error **out-of-bag**).

BOOTSTRAP Y TLC

El **Bootstrap** es una técnica práctica de remuestreo.

El **Teorema del Límite Central** le da fundamento teórico: permite justificar por qué podemos usar la distribución de los estadísticos bootstrap como una buena aproximación.

¿QUÉ TAMAÑO TIENEN LOS SUBCONJUNTOS BOOTSTRAP EN UN ENSAMBLE?

Cada subconjunto bootstrap generalmente tiene el **mismo tamaño** que el conjunto de entrenamiento original.

Si el conjunto de entrenamiento tiene *n* ejemplos, entonces cada subconjunto bootstrap también tendrá *n* ejemplos.

¿CÓMO SE ESCOGEN LOS EJEMPLOS?

Con reemplazo:

- Se seleccionan ejemplos **aleatoriamente** del conjunto original.
- Cada vez que se elige un ejemplo, **se vuelve a incluir en el conjunto disponible**, por lo que puede **repetirse** varias veces o **no aparecer** en absoluto.

¿CÓMO SE ESCOGEN LOS EJEMPLOS?

Ejemplo:

Conjunto original (5 ejemplos):

$$D = [x_1, x_2, x_3, x_4, x_5]$$

Subconjunto bootstrap (tamaño 5, con reemplazo):

$$D' = [x_2, x_5, x_2, x_1, x_4]$$

MÉTODOS RELACIONADOS CON BAGGING

- **Pasting**

Pasting es una variante de Bagging donde también se entrenan múltiples modelos en subconjuntos del conjunto original, pero sin reemplazo.

- **Subespacio aleatorio (Random Subspace Method)**

Es una técnica donde, además de muestrear ejemplos (filas), se muestrean subconjuntos aleatorios de características (columnas) para cada modelo.

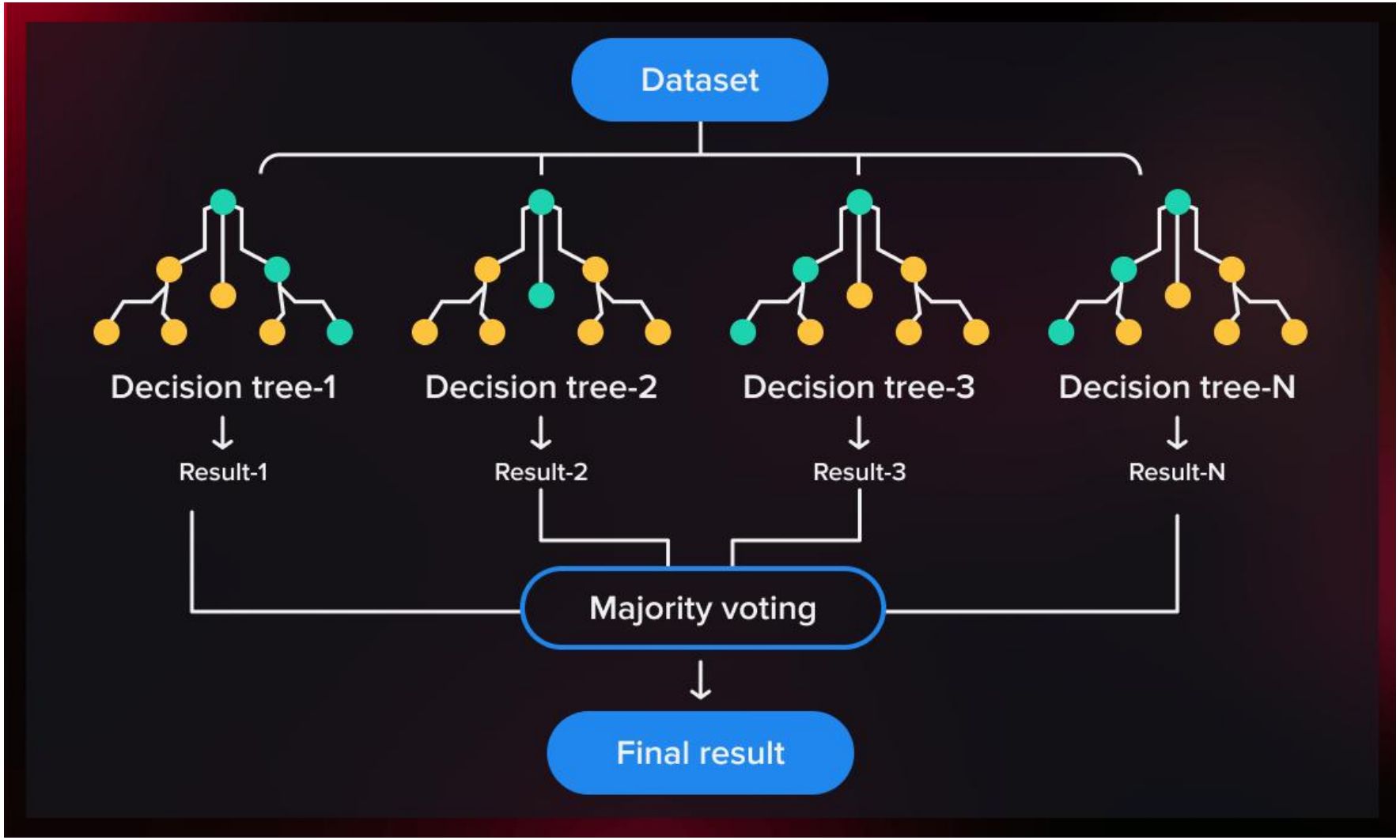
RANDOM FORESTS

¿QUÉ ES UN RANDOM FOREST?

Un Random Forest es un algoritmo de aprendizaje automático basado en ensambles de árboles de decisión.

Se trata de una colección de árboles de decisión entrenados con Bagging, con una mejora adicional: cada árbol se entrena usando un subconjunto aleatorio de características en cada división del árbol.

¿QUÉ ES UN RANDOM FOREST?



¿CÓMO FUNCIONA UN RANDOM FOREST?

1. Bootstrapping (Bagging):

- Se generan múltiples subconjuntos de entrenamiento **con reemplazo**.
- Cada subconjunto se usa para entrenar un árbol distinto.

2. Subespacio aleatorio:

- En cada nodo del árbol, **se elige aleatoriamente un subconjunto de variables**.
- La mejor división se busca **solo dentro de ese subconjunto**, no entre todas las variables.

¿CÓMO FUNCIONA UN RANDOM FOREST?

3. Crecimiento de árboles completos

- Los árboles suelen crecer **sin poda** (muy profundos), lo que aumenta la varianza individual.
- La agregación reduce esta varianza al combinar muchos árboles.

4. Agregación de resultados

- **Clasificación:** se hace una **votación por mayoría** entre los árboles.
- **Regresión:** se toma el **promedio** de las predicciones.

¿CÓMO SE SELECCIONA EL NÚMERO DE ÁRBOLES EN UN RANDOM FOREST?

El número de árboles (n_estimators en scikit-learn, por ejemplo) **es un hiperparámetro**.

No hay una regla fija, pero se suele elegir **entre 100 y 1000 árboles**, dependiendo del problema y del poder de cómputo.

Más árboles suelen mejorar la estabilidad y precisión, pero también aumentan el **costo computacional**.

Se selecciona típicamente mediante **validación cruzada** o búsqueda de hiperparámetros (Grid Search / Random Search).

¿QUÉ SUCEDE EN **CADA NODO DE DECISIÓN**?

En cada nodo, se selecciona **aleatoriamente** un **subconjunto de variables** (no todas).

Se evalúan todas las **divisiones posibles** usando esas variables (por ejemplo, usando entropía, Gini, MSE...).

Se elige la **mejor división** según el criterio del árbol (maximizar ganancia de información, reducir error, etc.).

El nodo se divide y se repite el proceso en los nodos hijos.

ESTE PROCESO CONTINÚA **HASTA**

Se alcanza una **profundidad máxima** (si se especifica),

O el nodo ya no puede dividirse más (por ejemplo, contiene solo un tipo de clase).

ÁRBOL DE DECISIÓN VS ÁRBOL EN RANDOM FOREST

Árbol de Decisión (clásico):

- Usa **todo el conjunto de entrenamiento**.
- En cada nodo, evalúa **todos los features** disponibles.
- Busca la **mejor división global** en base a una métrica como Gini o entropía.
- Tiene una **estructura determinística**: dado el mismo dataset, siempre produce el mismo árbol.
- El árbol es el **modelo final**.

ÁRBOL DE DECISIÓN VS ÁRBOL EN RANDOM FOREST

Árbol dentro de un Random Forest

- Se entrena con un **subconjunto aleatorio de datos**, obtenido con reemplazo (*bootstrap*).
- En cada nodo, se evalúa solo un **subconjunto aleatorio de features** ($\sqrt{n}$ para clasificación).
- Busca la **mejor división dentro de ese subconjunto**, no en todos los features.
- La estructura del árbol es **aleatoria** debido al muestreo y a la selección de features.
- El árbol **es solo uno de muchos**; su salida se combina con otros árboles.

FEATURE IMPORTANCE

Es una medida de **cuánto contribuye una variable** (feature) al desempeño predictivo de un modelo.

Su propósito es indicar **qué tan útil fue cada variable** para tomar decisiones dentro del modelo, según algún criterio de evaluación.

FEATURE IMPORTANCE

1. Basada en la estructura del modelo (impureza)

Usada en árboles de decisión, Random Forests y Gradient Boosting.

- Cada vez que una variable se usa para dividir un nodo, se calcula cuánto **mejora el modelo** (reduce la impureza como Gini o Entropía).

Estas mejoras se **acumulan** para cada variable a lo largo del modelo.

- **Ventaja:** rápida, está integrada al entrenamiento.
- **Desventaja:** sesgada hacia variables con más niveles o variabilidad.

¿CÓMO INTERPRETAR CORRECTAMENTE LA FEATURE IMPORTANCE?

Una variable **importante** mejora la capacidad predictiva del modelo de forma consistente.

Importancia ≠ causalidad: que una variable sea útil para predecir no significa que cause el fenómeno.

Si dos variables están **correlacionadas**, la importancia se puede repartir entre ellas o concentrarse en una sola.

VALIDACIÓN CON ERROR OUT-OF-BAG

¿Qué es el error Out-of-Bag (OOB)?

- Es una forma de estimar el **error de generalización** de un modelo entrenado con **Bagging** (como Random Forest).

- **"Out-of-Bag" = fuera de la bolsa** de muestreo: se refiere a las muestras que **no fueron seleccionadas** en el subconjunto de entrenamiento de un árbol.

VALIDACIÓN CON ERROR OUT-OF-BAG

En promedio, **cada árbol del Random Forest ve solo ~63%** del dataset original (debido al muestreo con reemplazo).

El **otro ~37%** se considera **OOB** para ese árbol.

¿CÓMO SE CALCULA EL OOB **EN RANDOM FORESTS**?

1. Para cada muestra x del conjunto original:
    - Identifica los árboles en los que **no fue usada** para entrenar (es decir, donde fue OOB).
    - Usa **solo esos árboles** para predecir la clase o valor de x.
    - Compara la predicción agregada con la **etiqueta real**.
2. Se calcula el **error promedio** entre las predicciones y las etiquetas reales de todas las muestras.

¿PARA QUÉ SE USA EL OOB?

1. **Estimación del error de generalización**

   - El OOB actúa como una forma **interna de validación**.
   - Permite estimar qué tan bien generaliza el modelo a nuevos datos.
   - Funciona como si tuvieras un conjunto de validación separado, pero sin perder datos para entrenamiento.

2. **Ajuste de hiperparámetros**

   Puedes usar el error OOB para:

   - Elegir el número óptimo de árboles (n_estimators).
   - Ajustar la profundidad de los árboles (max_depth).
   - Modificar max_features o min_samples_leaf.

   Esto evita usar validación cruzada, que es más costosa.

¿PARA QUÉ SE USA EL OOB?

3. Detección de sobreajuste

- Si el **OOB error disminuye** al aumentar los árboles, y luego **se estabiliza**, significa que el modelo está mejorando sin sobreajustar.
- Si empieza a subir, indica **sobreajuste**.

4. Validación rápida en grandes conjuntos de datos

El OOB es útil cuando:

- No puedes permitirte hacer K-Fold Cross Validation por tiempo.
- Quieres una validación confiable sin separar un test set.