

Apuntes tema 3

Aprendizaje supervisado

1. Fundamentos del Aprendizaje Supervisado

Se define por el entrenamiento de algoritmos mediante un **conjunto de datos que ya contiene la respuesta correcta**, denominada **etiqueta (label)** o **target**.

Mecánica Operativa

El sistema opera mediante la interacción de dos elementos clave:

- **Características de entrada (Features):** Variables de **entrada** que describen cada ejemplo.
- **Salida o Etiqueta (Target):** El **valor o categoría** que el modelo debe aprender a predecir.

El objetivo final es que el modelo detecte la **relación** entre las entradas y salidas para poder hacerlo con datos nuevos.

2. Tipos de Problemas y Algoritmos

La naturaleza del problema supervisado se determina según el tipo de salida que se desea obtener.

A. Clasificación

Se utiliza cuando el objetivo es predecir una **clase o categoría discreta** (un conjunto limitado de opciones).

- **Ejemplos:** Detección de spam, diagnóstico médico (positivo/negativo), reconocimiento de imágenes.
- **Algoritmos representativos:** Regresión logística, Árboles de decisión, Random Forest, SVM (Máquinas de Vectores de Soporte) y Redes neuronales.

B. Regresión

Se aplica cuando el objetivo es predecir un **valor numérico continuo** dentro de un rango infinito.

- **Ejemplos:** Predicción de precios de mercado, cálculo de tiempos de entrega, estimaciones meteorológicas.
- **Algoritmos representativos:** Regresión lineal, SVR (Regresión con SVM), Random Forest Regressor y Redes neuronales.

3. El Pipeline de Machine Learning

El flujo de trabajo estándar para desarrollar un modelo supervisado sigue una secuencia lógica de ocho pasos:

1. **Recolección:** Obtención de información relevante.
2. **Etiquetado:** Asignación de la **respuesta correcta** a cada muestra (crucial si el dataset original no las tiene).
3. **Limpieza y Preparación:** Tratamiento de **errores y normalización**.
4. **División del Dataset:** Partición en **conjuntos de entrenamiento, validación y prueba**.
5. **Entrenamiento:** Aplicación del **algoritmo sobre los datos**.
6. **Evaluación y Optimización:** Medición de **precisión** y **ajuste** de parámetros.
7. **Prueba (Test):** Validación con datos nuevos para verificar el **rendimiento real**.
8. **Producción:** Aplicación del modelo en un **entorno real**.

4. Preparación y Limpieza de Datos

La calidad de un modelo es **directamente proporcional** a la calidad de sus datos. Esta fase suele consumir más tiempo que el entrenamiento mismo.

Tratamiento de Registros e Integridad

- **Eliminación de duplicados:** Para evitar que el modelo memorice patrones artificiales en lugar de aprender **patrones reales**.
- **Gestión de valores nulos (missing values):** Contamos con la **eliminación** si son pocos datos los que se borran o **imputación** con una media, mediana o moda.
- **Detección de Outliers (Valores atípicos):** Valores **extremos** que **distorsionan las medias**. Se identifican mediante análisis visual (boxplots) o

métodos estadísticos (Z-score, IQR). Se pueden eliminar o limitar mediante *clipping*.

Transformación de Variables

Los algoritmos requieren que los datos estén en un formato numérico y, frecuentemente, en escalas similares.

Técnica	Descripción	Uso Recomendado
Min-Max Scaling	Transforma valores al rango [0, 1].	Modelos basados en distancias (KNN), Redes Neuronales.
Standard Scaling	Ajusta datos a Media=0 y Desviación=1.	Modelos lineales, SVM, PCA; menos sensible a outliers.
Label Encoding	Asigna un número a cada categoría.	Datos donde existe un orden intrínseco.
One-Hot Encoding	Crea columnas binarias por categoría.	La mayoría de los modelos; evita órdenes artificiales.

5. Estrategias de División y Estratificación

Para garantizar que la evaluación del modelo sea justa y representativa, el dataset se divide generalmente en tres bloques:

- **Entrenamiento (Train):** 60-70% de los datos.
- **Validación (Validation):** 15-20% para ajuste de hiperparámetros.
- **Prueba (Test):** 15-20% para la evaluación final.

La Importancia de la Estratificación

La **estratificación** asegura que la **proporción de cada clase** se mantenga constante en todos los subconjuntos, lo que permite:

- Una **evaluación** más justa y precisa.
- Reducción de la **variabilidad** de las métricas.
- Garantizar que el modelo aprenda a **identificar** la clase minoritaria de manera efectiva.

Aprendizaje supervisado: Regresión lineal

1. Características Principales de la Regresión Lineal

La regresión lineal se define por una **serie de atributos técnicos** que determinan su funcionamiento y aplicación:

- **Aprendizaje Supervisado:** El algoritmo aprende a partir de un conjunto de **datos previamente etiquetados**.
- **Basado en Modelo:** Se centra en la construcción de una **función hipótesis** que represente la relación entre los datos.
- **Modelo Lineal:** Utiliza una función lineal para **realizar predicciones**.
- **Predicción Mediante Suma Ponderada:** El resultado se obtiene **computando** una **suma ponderada** de las características de entrada, a la que se añade una constante de sesgo o *bias*.
- **Salida Continua:** Su propósito fundamental es **predecir valores numéricos** dentro de un rango continuo (por ejemplo, el coste económico de un incidente).

2. Notación y Estructura de Datos

Para el desarrollo del modelo, se utiliza una **notación estandarizada** que permite organizar la información del conjunto de entrenamiento (*dataset*):

Símbolo	Definición
m	Número total de ejemplos en el conjunto de datos.
x	Variable de entrada o característica (<i>feature</i>).
y	Variable de salida o valor objetivo (<i>target</i>).
n	Número de variables de entrada disponibles.
(x, y)	Representación de un único ejemplo de entrenamiento.

3. La Función Hipótesis

La función hipótesis es la **expresión matemática** que aproxima la relación entre las variables de entrada y la salida. El **proceso general** sigue el flujo: **Conjunto de datos** → **Función hipótesis** → **Modelo** → **Predicciones**.

3.1. Tipos de Regresión Lineal

Dependiendo del número de variables de entrada, la función varía:

1. **Regresión Lineal Univariable (Unidimensional):** Se emplea cuando solo existe una característica de entrada. Su fórmula es: $h_{\theta}(x) = \theta_0 + \theta_1 x$
2. **Regresión Lineal Multivariable (Multivaluada):** Se utiliza cuando existen múltiples características de entrada (x_1, x_2, \dots, x_n): $\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \dots + \theta_N x_N$

3.2. Parámetros del Modelo

El comportamiento y la precisión de la recta de regresión dependen de los parámetros θ :

- θ_0 es el punto con el que corta en el eje y.
- θ_1 indica la pendiente (inclinación de la recta).

4. Evaluación mediante la Función de Coste

Una vez inicializados los parámetros, es necesario medir qué tan precisas son las predicciones del modelo. Para ello se utiliza la **función de coste** o función de error.

4.1. Error Cuadrático Medio (MSE)

Es la métrica más utilizada en regresión lineal. Cuantifica la diferencia entre la predicción de la función hipótesis y el valor real del dato etiquetado. El cálculo sigue estos pasos:

1. Calcular la diferencia entre la predicción y el valor real para cada ejemplo.
2. Elevar dicha diferencia al cuadrado (lo que asegura valores positivos y penaliza errores grandes).
3. Sumar todos los errores individuales.
4. Calcular el promedio dividiendo entre el número de ejemplos (m) y, por convención matemática, entre 2.

Un valor de coste alto indica que el modelo no se ajusta a la tendencia de los datos, mientras que un valor bajo indica un ajuste óptimo.

5. Optimización: Descenso por Gradiente (Gradient Descent)

El objetivo del entrenamiento es encontrar los valores de θ_0 y θ_1 que minimicen la función de coste. La técnica predominante para lograr esto es el **Descenso**

por Gradiente.

5.1. El Concepto de Mínimo Global

En la **regresión lineal**, la función de coste suele tener una **forma convexa**. Esto garantiza la existencia de un único **mínimo global**, que es el punto exacto donde el **error es el menor posible**.

5.2. Funcionamiento del Algoritmo

El **Descenso por Gradiente** ajusta los parámetros de forma iterativa siguiendo estos principios:

- **Cálculo del Gradiente:** El gradiente es equivalente a la **derivada o pendiente** de la función en un punto dado.
- **Dirección del Ajuste:** El gradiente indica la **dirección en la que el error aumenta**; por lo tanto, el algoritmo mueve los parámetros en la dirección opuesta para reducir el error.
- **Proceso Iterativo:** El algoritmo **modifica** gradualmente θ hasta que la **pendiente es cero** (línea horizontal), lo cual indica que se ha alcanzado el mínimo global y los parámetros son óptimos.

Este proceso de minimización del error mediante el ajuste de parámetros es lo que constituye técnicamente el **entrenamiento** de un algoritmo de Machine Learning.

Aprendizaje Supervisado: Regresión logística

1. Definición y Propósito del Modelo

La regresión logística es una técnica de **clasificación** que computa una **suma ponderada** de las características de entrada más un sesgo (*bias*), aplicando posteriormente una **función de activación** para determinar la pertenencia a una clase.

- **Clasificación Binaria:** El modelo intenta predecir **valores discretos**:
 - **0:** Clase negativa (ej. "no spam").
 - **1:** Clase positiva (ej. "spam").

- **Naturaleza:** Se considera un **modelo lineal generalizado**.

2. Diferencias entre Regresión Lineal y Logística

Aunque ambos modelos comparten la base de una combinación lineal de variables, sus **objetivos** y **comportamientos** son diferentes:

Característica	Regresión Lineal	Regresión Logística
Tipo de predicción	Valores continuos en rango amplio.	Valores discretos (normalmente 0 o 1).
Transformación	Ninguna (suma ponderada directa).	Función logística (sigmoide) aplicada al resultado.
Sensibilidad a datos anómalos	Alta. Valores extremos desplazan la recta y alteran el límite de decisión.	Baja. El resultado está siempre acotado entre 0 y 1.

El fallo de la regresión lineal en clasificación

Al intentar aplicar **regresión lineal** a problemas de **clasificación**, la aparición de ejemplos adicionales o valores atípicos puede **aplanar la recta de regresión**. Esto provoca que el límite de decisión se **desplace incorrectamente**, generando predicciones erróneas.

3. Construcción de la Función Hipótesis

La función hipótesis ($h_{\theta}(x)$) en regresión logística garantiza que las **predicciones estén siempre en el intervalo [0, 1]**.

La Función Sigmoide

Para lograr este acotamiento, se utiliza la función sigmoide o logística ($g(z)$):

$$g(z) = 1 / (1 + e^{-z})$$

Propiedades de la sigmoide:

- Tiene una característica **forma en "S"**.
- Toma el valor **0.5** cuando la entrada (z) es 0.
- Tiende a **1** para valores **positivos grandes**.
- Tiende a **0** para valores **negativos grandes**.

Definición Matemática

Para un problema con **una sola característica de entrada**, la hipótesis se expresa como:

$$h_{\theta}(x) = 1 / (1 + e^{-(\theta_0 + \theta_1 x)})$$

4. Interpretación y Probabilidad

La salida de la función hipótesis es la **probabilidad** de que un ejemplo **pertenezca a la clase positiva** ($y=1$).

- Si el modelo devuelve 0.7, indica un **70% de probabilidad** de que el correo sea spam.
- Esta salida continua **evita las distorsiones por datos anómalos** que afectan a la regresión lineal simple.

5. La Función de Coste

La **función de coste tradicional** de la regresión lineal (mínimos cuadrados) **no** es apta para la **regresión logística**. Al combinarla con la función sigmoide, se genera una función de error **no convexa**.

- **Problema de la no convexidad:** Presenta **múltiples mínimos locales**, lo que impide que los algoritmos de optimización aseguren encontrar el menor valor de error posible.
- **Solución:** Se define una **función de coste convexa** que garantiza un único mínimo global.

Función de Coste Unificada

La función utilizada **penaliza las predicciones incorrectas** de forma logística:

$$J(h_{\theta}(x), y) = -y \log(h_{\theta}(x)) - (1 - y) \log(1 - h_{\theta}(x))$$

Comportamiento de la penalización:

- Si la etiqueta real es $y=1$ y el modelo predice 0.1 (cercano a 0), el **coste es muy alto**.
- Si la etiqueta real es $y=0$ y el modelo predice 0.9 (cercano a 1), el **coste es muy alto**.
- A medida que la predicción se acerca al valor real, el **coste tiende a cero**.

6. Límite de Decisión (*Threshold*)

El *threshold* es el umbral que **transforma** la probabilidad continua en una **decisión discreta**.

- **Regla estándar:**
 - Si $h\theta(x) \geq 0.5 \rightarrow$ Clase 1.
 - Si $h\theta(x) < 0.5 \rightarrow$ Clase 0.

Ajuste del Umbral según el Contexto

El valor de 0.5 es habitual pero no obligatorio. **Puede modificarse** dependiendo del **tipo de error que se desee minimizar**:

1. **Threshold bajo (ej. 0.3):** Clasifica más ejemplos como clase 1. Útil para reducir **falsos negativos** (ej. diagnóstico médico donde no se quiere omitir un caso positivo).
2. **Threshold alto (ej. 0.7):** Clasifica menos ejemplos como clase 1. Útil para reducir **falsos positivos** (ej. evitar que correos legítimos importantes terminen en la carpeta de spam).

El límite de decisión actúa como una **frontera que divide las clases** en el espacio de las características de entrada.

Aprendizaje supervisado: KNN

1. Fundamentos del K-NN como Aprendizaje Supervisado

K-NN se clasifica como un algoritmo de **aprendizaje supervisado** porque requiere de un conjunto de entrenamiento previamente etiquetados. Cada instancia en este conjunto está compuesta por:

- **Características de entrada (Features):** Variables de **entrada** que describen cada ejemplo.
- **Salida o Etiqueta (Target):** El **valor o categoría** que el modelo debe aprender a predecir.

Este algoritmo es valorado por ser uno de los **métodos más accesibles**.

2. La Características del "Lazy Learning"

K-NN no construye un modelo explícito durante el entrenamiento. Esta naturaleza se resume en la siguiente comparativa:

- **Algoritmos convencionales:** "Aprenden **antes** y predicen **rápido**".
- **k-NN (Lazy Learning):** "Aprende **poco** y predice **lento**".

El entrenamiento consiste exclusivamente en **almacenar los datos de entrada**. En consecuencia, todo el procesamiento ocurre en el momento exacto en que se solicita una predicción.

3. Mecánica de Funcionamiento

El proceso operativo es **secuencial** y se activa ante cada **nuevo ejemplo sin etiqueta**:

1. **Recepción:** Se introduce un **dato nuevo**.
2. **Cálculo de Distancia:** Se mide la distancia entre el **nuevo ejemplo** y **todos los puntos** almacenados.
3. **Selección:** Se identifican los **k ejemplos más cercanos** (los "vecinos").
4. **Decisión:**
 - En **clasificación**, se asigna la **clase más frecuente** entre los vecinos.
 - En **regresión**, se calcula la **media de los valores de los vecinos**.

4. El Rol Crítico del Parámetro k

La elección del valor de k determina la **estabilidad y precisión** del modelo:

Valor de k	Impacto en el Modelo	Riesgos Asociados
Pequeño (ej. $k=1$)	Toma decisiones basadas en un único vecino. Muy sensible a valores atípicos.	Sobreajuste (Overfitting) y alta sensibilidad al ruido.
Grande	El modelo es más estable y reduce la influencia del ruido.	Subajuste (Underfitting) y pérdida de detalles locales importantes.

En la práctica, el valor óptimo se halla mediante **procesos de validación y pruebas de rendimiento**.

5. Métrica de Distancia y el Escalado de Características

Medidas de Similitud

Para cuantificar la cercanía entre puntos, k-NN emplea **métricas matemáticas**. La más común es la **Distancia Euclídea**, que representa la **línea recta entre dos puntos**. Otras alternativas incluyen:

- **Manhattan**: Suma de **diferencias absolutas**.
- **Minkowski**: Una generalización de **diversas métricas**.
- **Chebyshev**: Basada en la **distancia máxima en cualquier dimensión**.

Necesidad Obligatoria de Escalado

k-NN es extremadamente **sensible** a la **magnitud de los datos**. Si una variable tiene un rango mayor frente a otra menor, la **variable mayor dominará** el cálculo de distancia, invalidando la similitud real. Por ello, es importante aplicar:

- **StandardScaler**: Ajuste a **media 0** y **desviación 1**.
- **MinMaxScaler**: Transformación de **valores al rango entre 0 y 1**.

6. Aplicabilidad: Clasificación vs. Regresión

La versatilidad del k-NN le permite abordar dos tipos de problemas fundamentales:

- **Clasificación**: La predicción se basa en un **sistema de votación**. Por ejemplo, si con $k=5$, tres vecinos pertenecen a la "Clase A" y dos a la "Clase B", el resultado final será "Clase A".
- **Regresión**: La predicción es un **valor numérico derivado de la media**. Si los vecinos más cercanos tienen valores de 200, 220 y 210, la predicción resultante será 210.

7. Evaluación de Ventajas y Limitaciones

El uso de k-NN debe ser estratégico, considerando sus fortalezas y debilidades inherentes:

Ventajas

- **Implementación** y **comprensión** sumamente sencillas.

- **No** requiere una fase de **entrenamiento compleja**.
- **Versatilidad** para tareas de **clasificación y regresión**.
- Excelente desempeño en **conjuntos de datos pequeños**.

Limitaciones

- **Eficiencia:** Muy **lento** durante la fase de predicción.
- **Memoria:** Exige almacenar la **totalidad del dataset**.
- **Sensibilidad:** Vulnerable al **ruido** y dependiente **crítico** del escalado.
- **Escalabilidad:** Poco apto para **grandes volúmenes de datos** (*datasets* masivos).

En conclusión, k-NN es una herramienta **poderosa y fundamental** para entender el aprendizaje supervisado, ideal para contextos con **datos limitados** y características bien definidas, aunque suele ceder ante modelos como SVM o árboles de decisión en entornos de alta complejidad.

Aprendizaje supervisado: Máquinas de Vectores de Soporte (SVM)

1. Fundamentos del Aprendizaje en SVM

El aprendizaje automático consiste en la **detección de patrones** para realizar predicciones precisas. En este contexto, SVM debe equilibrar dos problemas críticos que afectan a la mayoría de los modelos:

Desafíos del Ajuste de Modelos

Situación	Comportamiento del Modelo	Resultado
Subajuste (Underfitting)	El modelo es demasiado simple ; no detecta patrones importantes.	Error alto tanto en entrenamiento como en pruebas.
Buen ajuste	Aprende patrones generales y tendencias reales .	Predicciones correctas con datos nuevos.
Sobreajuste (Overfitting)	El modelo memoriza el ruido y los datos específicos .	Error bajo en entrenamiento , pero alto en validación/test .

Las SVM destacan por mitigar estos problemas mediante la **construcción de modelos robustos**, siendo especialmente eficaces cuando las **fronteras de decisión** son difíciles de definir.

2. Funcionamiento y Construcción del Límite de Decisión

A diferencia de otros modelos lineales que buscan cualquier línea que separe las clases, la SVM busca el **mejor límite de decisión posible**.

El Concepto de Margen

El margen es la distancia entre el **límite de decisión** y los **ejemplos de entrenamiento más cercanos** de cada clase. El objetivo primordial de SVM es maximizar este margen.

- **Vectores Soporte (Support Vectors):** Son los puntos o ejemplos que se encuentran exactamente sobre las **líneas del margen**. Estos ejemplares son los únicos que determinan la posición del modelo; si se añaden datos fuera del margen, el modelo no varía.
- **Ventaja de Generalización:** Al situar la frontera lo más **lejos** posible de los **vectores soporte**, se reduce la probabilidad de **clasificar erróneamente** nuevos datos que caigan cerca de la frontera de decisión.

3. Tipos de Modelos SVM según la Naturaleza de los Datos

El algoritmo puede **adaptarse** dependiendo de si los datos son **linealmente separables** o no.

3.1. Clasificación de Margen Duro (Hard Margin Classification)

Este enfoque es estricto y solo es aplicable en situaciones ideales donde los **datos** están perfectamente **separados** y no presentan ruido.

- **Restricción:** No permite ningún error de clasificación en el **conjunto de entrenamiento**.
- **Sensibilidad:** Es extremadamente vulnerable a **valores anómalos (outliers)**. Un solo dato mal ubicado puede obligar al modelo a **recalcular** un límite de decisión subóptimo o, en casos de solapamiento, impedir la creación de un modelo.

3.2. Clasificación de Margen Blando (Soft Margin Classification)

Es la variante más utilizada en aplicaciones reales debido a su **flexibilidad**.

- **Objetivo:** Permitir ciertos **errores controlados** para obtener un modelo que represente mejor la **tendencia global**.
- **Robustez:** Minimiza la influencia de los *outliers* y evita que **ejemplos extremos distorsionen** la frontera de decisión.

4. El Hiperparámetro C y la Regulación del Modelo

La **flexibilidad** en una SVM se controla mediante el **hiperparámetro C**, que define el grado de penalización de los errores de clasificación durante el entrenamiento.

- **C Alto (Comportamiento de Margen Duro):**
 - Penaliza **fuertemente** los errores.
 - El modelo intenta **clasificar correctamente cada ejemplo**.
 - Riesgo elevado de **sobreajuste** por ajustarse excesivamente a los datos.
- **C Bajo (Comportamiento de Margen Blando):**
 - Penaliza **menos** los errores, permitiendo clasificaciones erróneas controladas.
 - El modelo es **más flexible y robusto** frente a datos ruidosos.
 - Incorpora más **vectores soporte** para definir la frontera basándose en la concentración de datos y no en excepciones.

El **valor óptimo de C** depende de la **naturaleza del dataset** y del **nivel de ruido**, por lo que su selección requiere técnicas sistemáticas de validación.

5. Kernels y Separación No Lineal

En muchos escenarios reales, las clases **no pueden separarse mediante una línea recta** (datos no linealmente separables). SVM resuelve esto transformando el espacio de los datos.

El Truco del Kernel (Kernel Trick)

SVM utiliza funciones matemáticas denominadas **kernels**.

- **Función:** Calcula la **similitud entre ejemplos** como si estuvieran en un espacio de mayor dimensión, sin necesidad de realizar la transformación explícita de los datos.
- **Eficiencia:** Permite trabajar con modelos **no lineales** manteniendo una complejidad computacional razonable.

Tipos de Kernels Comunes

1. **Lineal:** Para **separaciones básicas**.
2. **Polinómico:** Genera **límites de decisión curvos** sin la explosión de características de la regresión polinómica tradicional.
3. **RBF (Gaussiano):** Utilizado para **mapeos complejos**.
4. **Sigmoide:** Otra variante para **estructuras específicas**.

Análisis de Modelos de Aprendizaje Supervisado: Árboles de Decisión y Random Forest

1. Fundamentos de los Árboles de Decisión

Los árboles de decisión son **clasificadores no lineales** ampliamente utilizados tanto en el ámbito académico como profesional. Su funcionamiento se basa en la **construcción de un modelo** que define **límites de decisión** a través de una **estructura jerárquica** de reglas tipo "if-then-else".

Características Principales

- **Versatilidad:** Capacidad para **predecir valores continuos** y **valores discretos**.
- **No Linealidad:** Construyen **límites** mediante particiones sucesivas.
- **Interpretabilidad:** El proceso de decisión es **sencillo** y **fácil** de seguir.

Mecanismo de Funcionamiento

El algoritmo busca **valores de corte** en las características de entrada para separar las clases de la mejor manera posible.

1. **Nodo raíz:** Aplicación de una **primera regla**.

2. **Partición:** Los ejemplos que cumplen la regla se asignan a un **nodo hoja** (nodo puro) o a una rama que requiere más divisiones.
3. **Jerarquía:** Se aplican **reglas sucesivas** hasta alcanzar una predicción final.

2. Limitaciones de los Árboles de Decisión Individuales

A pesar de sus ventajas, los árboles de decisión presentan cuatro limitaciones críticas:

Limitación	Descripción
Sobreajuste (Overfitting)	Tendencia a crear estructuras excesivamente complejas que memorizan el ruido y los detalles específicos de los datos de entrenamiento, perdiendo capacidad de generalización.
Óptimos Locales	La construcción se basa en el mejor corte inmediato por nodo (optimización local), lo que no garantiza una estructura óptima a nivel global .
Inestabilidad	Alta sensibilidad a pequeñas perturbaciones ; una mínima variación en el conjunto de entrenamiento puede resultar en un árbol con una estructura completamente diferente.
Complejidad en Datos con Ruido	El problema del sobreajuste se agrava en datasets con muchas características o valores atípicos .

3. Ensemble Learning y el Método Bagging

El **Ensemble Learning** se basa en el principio de que **combinar las predicciones de varios modelos** genera **resultados superiores** a los de un **modelo individual**. Esta técnica se aplica habitualmente en las fases finales de un proyecto de Machine Learning para **mejorar la robustez**.

El Proceso de Bagging (Bootstrap Aggregating)

El Bagging es la base del algoritmo Random Forest y consta de los siguientes pasos:

1. **Datasets Bootstrap:** Se generan **B subconjuntos del dataset original** mediante muestreo aleatorio con reemplazo.
2. **Entrenamiento:** Se entrena un **modelo independiente** sobre cada subconjunto.

3. **Ejemplos Out-of-Bag (OOB):** Los **datos que no se incluyen** en un subconjunto específico se denominan OOB y sirven para **evaluar el rendimiento** del modelo sin necesidad de un conjunto de validación externo (OOB score).
4. **Agregación:**
 - **Clasificación:** Votación mayoritaria entre **todos los modelos**.
 - **Regresión:** Cálculo de la **media** o **mediana** de las predicciones.

4. Random Forest: Sinergia de Aleatoriedad y Agregación

Un **Random Forest** es un **conjunto de árboles de decisión** que colaboran para realizar una predicción conjunta. Su robustez proviene de dos fuentes de aleatoriedad:

Fuentes de Aleatoriedad

- **Aleatoriedad en los Datos (Bagging):** Cada árbol se entrena con una **versión distinta** de los datos originales.
- **Aleatoriedad en las Características:** En cada división de un nodo, el árbol busca en un **subconjunto aleatorio** de estas. Esto descorrelaciona los árboles entre sí, mejorando la calidad **global del bosque**.

Beneficios del Modelo

- **Reducción del Sobreajuste:** Al promediar múltiples árboles, el **error** de generalización **disminuye**.
 - **Estabilidad Mejorada:** Las **variaciones** en los datos tienen un **impacto reducido** en el modelo final.
 - **Precisión Elevada:** El consenso de **múltiples modelos** suele capturar mejor el patrón general de los datos.
-

5. Hiperparámetros Críticos en Random Forest

El rendimiento de un Random Forest depende del **ajuste de sus hiperparámetros**, los cuales regulan la complejidad y el coste computacional:

Hiperparámetro	Función Impacto
<code>n_estimators</code>	Número de árboles en el bosque. Valores altos mejoran el rendimiento sin causar sobreajuste, pero aumentan el coste computacional.
<code>max_depth</code>	Profundidad máxima de los árboles. Limitarla ayuda a evitar el sobreajuste y favorece la generalización.
<code>max_features</code>	Número de características a considerar en cada nodo. Esencial para la aleatoriedad del modelo
<code>min_samples_split</code>	Mínimo de ejemplos para dividir un nodo; evita divisiones basadas en pocos datos.
<code>min_samples_leaf</code>	Mínimo de ejemplos en una hoja; ayuda a suavizar el modelo en presencia de ruido.
<code>bootstrap</code>	Define si se usa muestreo con reemplazo. Si es <code>True</code> , habilita el uso de métricas OOB.