

# Material de Catedra

# Machine Learning (Aprendizaje Automatico)

**Diego Parrás** (diegoparras@economicas.uba.ar)
Universidad de Buenos Aires. Facultad de Ciencias Económicas.
Secretaría Académica.

**2025**



**Contacto**
Facultad de Ciencias Económicas – UBA
Av. Córdoba 2122 – CABA – Argentina
E-mail: biblioteca.digital@fce.uba.ar

# Machine Learning

## (Aprendizaje Automático)

**Diego Parrás**

# Contenido

**Antes de comenzar**

En un mundo impulsado por la transformación digital y la automatización, el aprendizaje automático se ha posicionado como una herramienta esencial en diversos campos, desde las finanzas y el marketing hasta la medicina y la ingeniería. Como subcampo clave de la inteligencia artificial, el aprendizaje automático permite a los sistemas aprender de los datos, adaptarse a nuevas situaciones y mejorar su rendimiento sin intervención humana directa. Este material, "Introducción al Aprendizaje Automático", ofrece una exploración clara y concisa de los fundamentos que sustentan esta tecnología.

Comenzaremos con una revisión de los conceptos básicos y la historia del aprendizaje automático, proporcionando el contexto necesario para entender cómo y por qué estas técnicas se han vuelto indispensables en la era de los datos. A continuación, exploraremos distintos enfoques y modelos, desde el aprendizaje supervisado hasta el no supervisado y por refuerzo, detallando sus características y aplicaciones.

En secciones posteriores, se abordan las etapas clave en el desarrollo de un proyecto de aprendizaje automático: desde la recolección y preparación de datos, la selección y entrenamiento de modelos, hasta la validación y ajuste final. Este recorrido busca proporcionar una visión comprensiva, permitiendo a los lectores familiarizarse con los desafíos y las oportunidades que presenta el aprendizaje automático en la práctica.

Finalizaremos con una mirada a los desafíos más relevantes que enfrentan estos modelos, como la calidad de los datos y la selección de algoritmos adecuados. Aunque estos temas pueden resultar complejos, el objetivo es presentarlos de forma accesible y didáctica, facilitando así la comprensión incluso para quienes se adentran por primera vez en este campo.

Desde el Centro de Estudios en Modelos de Inteligencia Artificial aplicados a las Ciencias Económicas de la FCE-UBA (CeMIACE) dependiente de la Secretaría de Extensión Universitaria, Bienestar Estudiantil y desarrollo Sustentable, hemos desarrollado este material con la expectativa de que sirva como un recurso introductorio valioso y manejable para cualquier interesado en el aprendizaje automático. Esperamos que este trabajo no solo brinde los conocimientos necesarios para iniciar en esta disciplina, sino también inspire a explorar más a fondo las vastas posibilidades que ofrece.

Lic. Diego Parrás (Director del CeMIACE)

# Conceptos de Machine Learning (Aprendizaje Automático)

### 1.1. Definición.

El aprendizaje automático (AA), aprendizaje automatizado, aprendizaje de máquinas o aprendizaje computacional *(Machine Learning*, ML*)* es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Según Tom Mitchell (1997) se dice que un programa de computadora aprende de la experiencia (E) en una determinada tarea (T), evaluando su desempeño con una medida (P), Si (P) medido en (T) aumenta si aumenta (E).

### 1.2. Orígenes del término.

Arthur Samuel introdujo formalmente el término en su trabajo "*Some Studies in Machine Learning Using the Game of Checkers*" del año 1959, en donde se destaca que: "una computadora puede ser programada de forma tal que aprenderá a jugar mejor a las damas que la persona que la programó. No solo eso, sino que puede hacerlo en un período relativamente corto de tiempo cuando se le proveen sólo las reglas del juego, un sentido de dirección y una lista redundante e incompleta de parámetros que se supone tienen alguna relación con el juego, pero cuyas señales y pesos relativos son desconocidos y no especificados".

### 1.3. Un Caso de presentación: Filtro de SPAM.

En nuestros programas de gestión de correos electrónicos, normalmente disponemos de un filtro para los correos electrónicos no deseados, identificados por los usuarios. Este filtro de spam es un programa basado en aprendizaje automático que, a partir de ejemplos de correos no deseados (señalados por los usuarios) y correos normales (no spam, también conocidos como "*ham*"), aprende a identificar el spam. Los ejemplos que el sistema utiliza para su aprendizaje constituyen el **conjunto de entrenamiento**, y cada ejemplo dentro de este se conoce como **instancia de entrenamiento**. La parte del sistema de aprendizaje automático que se encarga de aprender y hacer predicciones se denomina **modelo**. Entre los modelos más comunes se encuentran las redes neuronales y los bosques aleatorios.

Siguiendo la definición de Mitchell, la tarea (T) consiste en identificar correos no deseados en los nuevos mensajes recibidos, la experiencia (E) viene dada por los datos de entrenamiento y se debe establecer una medida de rendimiento (P). Un ejemplo de esta podría ser la proporción de correos electrónicos clasificados correctamente. Esta medida específica se conoce como precisión y es frecuentemente utilizada en tareas de clasificación.

### 1.3.1. Enfoque tradicional

Consideremos cómo deberíamos escribir un filtro de spam utilizando técnicas de programación tradicionales (figura 1.1).

Primero deberíamos examinar cómo suele estar estructurado el *spam*. Podríamos notar que algunas palabras o frases (como "oferta", "tarjeta de crédito", "gratis" y "asombroso") tienden a aparecer recurrentemente en la línea de asunto. Deberíamos intentar encontrar algunos otros patrones en el nombre del remitente, el cuerpo del correo electrónico y otras partes de éste.

Luego deberíamos escribir un algoritmo de detección para cada uno de los patrones que hayamos encontrado y, si todo funciona correctamente, nuestro programa marcaría los correos electrónicos como *spam* si se detectaran estos patrones.

Probaríamos nuestro programa y, eventualmente, repetiríamos los pasos anteriores hasta que fuera lo suficientemente bueno para lanzarse.

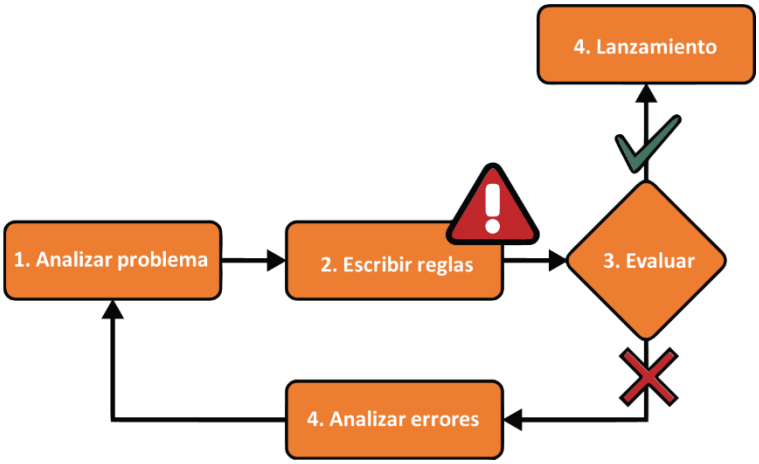


Figura 1.1 Enfoque tradicional

Dado que el problema es difícil, nuestro programa probablemente se convertirá en una larga lista de reglas complejas, bastante difíciles de mantener.

### 1.3.2. Enfoque basado En Aprendizaje Automático.

Un filtro de *spam* basado en técnicas de aprendizaje automático aprende automáticamente qué palabras y frases son buenos predictores de spam al detectar patrones de palabras inusualmente frecuentes en los ejemplos de spam en comparación con los

ejemplos de spam (figura 1.2). El programa es mucho más corto, más fácil de mantener y probablemente más preciso.

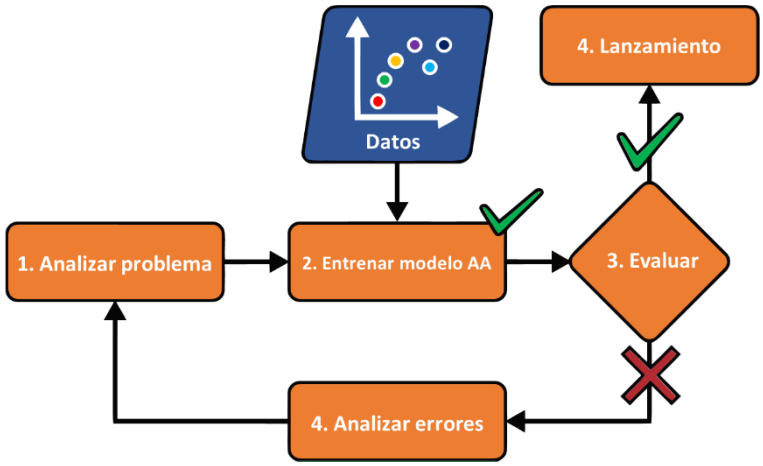


Figura 1.2 Enfoque de Aprendizaje automático

### 1.3.3. Retroalimentando el modelo con datos de salida.

¿Qué sucede si los emisores de spam descubren que sus correos con la palabra "gratis" son sistemáticamente bloqueados? Podrían optar por usar la palabra “gratuito” en su lugar. En este caso, sería esencial actualizar un filtro de *spam* basado en programación tradicional para que también identifique correos con "gratuito". Si los emisores de spam continúan adaptando sus estrategias para evadir el filtro, resultará necesario crear constantemente nuevas reglas.

En contraste, un filtro de *spam* que emplea técnicas de aprendizaje automático puede adaptarse automáticamente. Este tipo de filtro aprenderá por sí solo que la palabra "gratuito" ha comenzado a aparecer con frecuencia inusual en los correos marcados como *spam* por los usuarios, y comenzará a filtrarlos adecuadamente (figura 2.3).

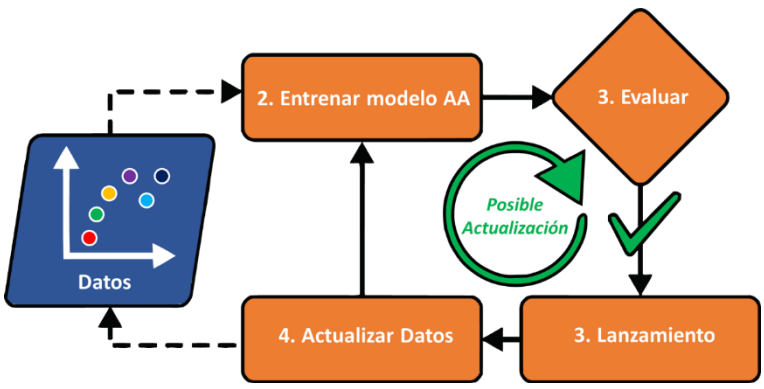


Figura 2.3 Actualización automática a los cambios.

### 1.4. Aprendizaje automático para la comprensión de problemas.

El aprendizaje automático se destaca especialmente en abordar problemas demasiado complejos para los métodos tradicionales o aquellos que carecen de un algoritmo predefinido. Al ser alimentados con grandes cantidades de datos, los modelos de aprendizaje automático tienen el potencial de revolucionar nuestra comprensión y enfoque hacia problemas complejos (figura 1.4).

Los modelos de aprendizaje automático son susceptibles de análisis para comprender qué han aprendido, aunque este proceso puede ser complejo en modelos más sofisticados. Por ejemplo, después de entrenar un filtro de spam para identificar eficientemente correos no deseados, es posible examinarlo para descubrir cuáles palabras y frases específicas son consideradas indicadores clave de *spam*. Esta inspección puede revelar cómo el modelo prioriza ciertas características sobre otras, ofreciendo así una visión detallada de su proceso de toma de decisiones. Este nivel de transparencia, además de ayudar a mejorar el modelo, proporciona información muy valiosa para entender mejor la naturaleza cambiante de los correos no deseados y adaptar estrategias para combatirlos eficazmente.

En ocasiones, este análisis nos puede revelar correlaciones inesperadas o tendencias emergentes, lo que resulta en un entendimiento más profundo del problema en cuestión. El proceso de explorar extensas colecciones de datos para identificar patrones subyacentes se conoce como minería de datos *(data mining)*, y en esta tarea, el aprendizaje automático demuestra una eficacia notable.

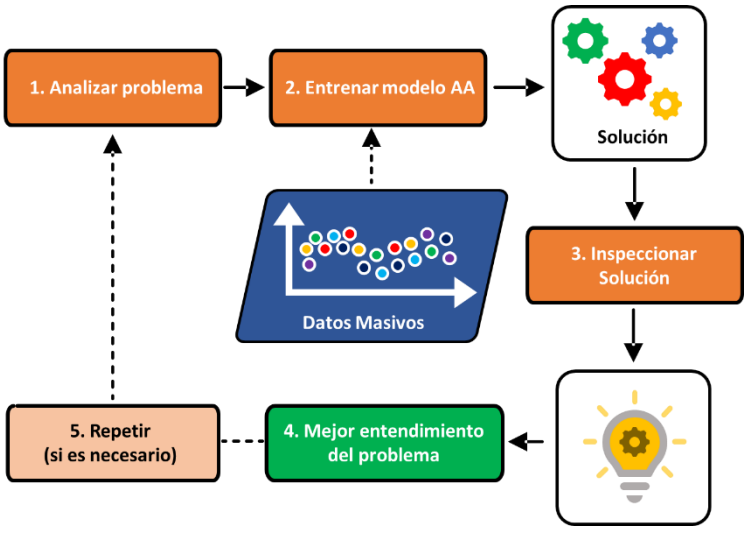


Figura 1.4 Mejorar comprensión del problema con Aprendizaje Automático.

### 1.5. Tipos de sistemas de aprendizaje automático

Dada la variedad de sistemas de aprendizaje automático existentes, resulta práctico clasificarlos en categorías basándonos en varios criterios clave:

**Tipo de Supervisión durante el Entrenamiento**: Incluye sistemas supervisados, no supervisados, semi-supervisados, y auto-supervisados, entre otros.

**Capacidad de Aprendizaje Incremental**: Se refiere a si el sistema es capaz de aprender continuamente (aprendizaje en línea) o si requiere de un conjunto de datos fijo para el entrenamiento (aprendizaje por lotes).

**Método de Operación**: Algunos sistemas trabajan mediante la comparación de nuevos datos con datos conocidos, mientras que otros detectan patrones en los datos de entrenamiento para construir un modelo predictivo, al estilo de los métodos científicos (aprendizaje basado en instancias versus aprendizaje basado en modelos).

Estos criterios no son mutuamente excluyentes. Por ejemplo, un filtro de spam avanzado podría ser un sistema de aprendizaje automático supervisado, basado en modelos y capaz de aprender en línea.

#### 1.5.1. Supervisión durante el entrenamiento

Los sistemas de AA se pueden clasificar según la cantidad y el tipo de supervisión que reciben durante el entrenamiento. Las principales categorías son: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje auto-supervisado, aprendizaje semi-supervisado y aprendizaje por refuerzo.

##### 1.5.1.1. Aprendizaje supervisado

En el aprendizaje supervisado, el conjunto de entrenamiento que alimenta al algoritmo incluye las soluciones deseadas, llamadas etiquetas (figura 1.5).

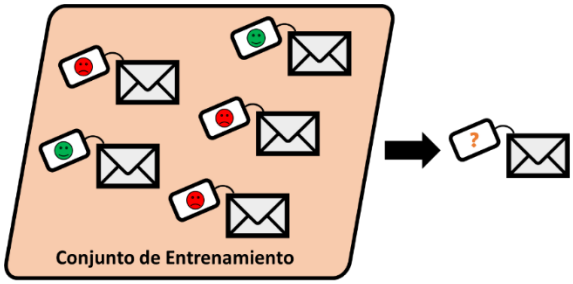


Figura 1.5. Conjunto de entrenamiento etiquetado para clasificación de spam.

**1.5.1.1.1. Algoritmos de Clasificación.**

Una tarea típica de aprendizaje supervisado es la clasificación, la cual utiliza un determinado algoritmo para asignar datos en categorías específicas. Los algoritmos de clasificación comunes son Bayes Ingenuo *(Naive Bayes)*, Clasificadores Lineales *(Linear Classifiers)*, Máquinas de Vectores de Soporte *(Support Vector Machines, SVM)*, Árboles de Decisión (*Decision Trees)*, K Vecinos Más Cercanos *(K-Nearest Neighbors, KNN)* y bosques aleatorios (*Random Forests*), que se describen con más detalle a continuación. Cada uno de estos algoritmos tiene sus propias fortalezas y debilidades, y la elección entre ellos depende en gran medida de la naturaleza del problema de clasificación específico que estemos abordando.

**Bayes Ingenuo *(Naive Bayes).***

*Naive Bayes* es una técnica de clasificación en AA que se basa en el Teorema de Bayes. Este teorema se utiliza para calcular la probabilidad posterior $P(y \mid x)$ de una clase $y$ dado un vector de características $x$. La fórmula general para Naive Bayes es:

$$P(y \mid x) = \frac{P(x \mid y) \cdot P(y)}{P(x)}$$

Es ampliamente utilizado debido a su simplicidad y efectividad, especialmente en tareas como la clasificación de texto (por ejemplo, distinguir entre correos spam y no spam, o analizar el sentimiento de un texto).

La suposición fundamental de "*naive*" (ingenua) en *Naive Baye*s es que cada característica en el conjunto de datos es independiente de las demás, dado que ya se conoce la clase. Esto significa que la presencia (o ausencia) de una característica no afecta la presencia (o ausencia) de otra. Bajo este supuesto, la fórmula resultante es la siguiente.

$$P(y \mid x) = \prod_i P(x_i \mid y) \cdot P(y)$$

Aunque esta es una suposición bastante fuerte y a menudo no es verdadera en datos reales, sorprendentemente, Naive Bayes puede funcionar muy bien en la práctica.

*Naive Bayes* utiliza probabilidades para hacer predicciones. Por ejemplo, en la clasificación de correos electrónicos, calcula la probabilidad de que un correo sea spam o no, basándose en las palabras que contiene. Durante la fase de entrenamiento, el algoritmo aprende la frecuencia de las palabras en correos clasificados como spam y no spam. Esto le

permite estimar la probabilidad de que un correo nuevo pertenezca a una categoría u otra. Para un nuevo correo, Naive Bayes calcula la probabilidad de que éste sea spam utilizando las probabilidades aprendidas. Si la probabilidad de ser spam es mayor que la de no serlo, el correo se clasifica como spam, y viceversa.

A pesar de su suposición de independencia, el algoritmo tiende a ser muy eficaz en la práctica. Su simplicidad lo hace rápido y fácil de implementar, y a menudo ofrece un rendimiento comparable a clasificadores más complejos.

Hay varios tipos de clasificadores *Naive Bayes*, y cada uno se utiliza según el tipo de datos que se maneja:

**Gaussiano.** En el modelo Gaussiano, se asume que los valores de las características son continuos y se distribuyen siguiendo una distribución gaussiana (normal). Es especialmente útil cuando las características tienen valores continuos, como podría ser la medición de temperaturas, alturas, etc. En este caso, se calcula la media y la desviación estándar de las características para cada clase.

**Multinomial** El modelo Multinomial *Naive Bayes* es apropiado para características que representan frecuencias o recuentos. Es muy usado en la clasificación de textos, donde las características son, por ejemplo, la frecuencia de palabras o frecuencias de términos. En este enfoque, se utiliza la distribución multinomial para modelar la probabilidad de diferentes características.

**Bernoulli.** El modelo de Bernoulli *Naive Bayes* es útil cuando las características son variables binarias (variables dicotómicas que toman valores 0 o 1). Este modelo es similar al multinomial, pero está diseñado para datos binarios. Se utiliza mucho en situaciones donde las características son principalmente de naturaleza binaria, como en la clasificación de textos con presencia o ausencia de palabras.

**Categórico.** Similar a los modelos multinomial y Bernoulli, el modelo categórico es utilizado para variables discretas. La diferencia principal es que mientras que el multinomial trabaja con recuentos, el modelo categórico está diseñado para características que representan categorías.

**Complementario.** Este es una variación del modelo multinomial y se utiliza principalmente en el ámbito de la clasificación de textos. Fue desarrollado para abordar el problema del sesgo en conjuntos de datos con una distribución de clases desigual.

**Clasificadores Lineales *(Linear Classifiers)***.

Los clasificadores lineales, son algoritmos que utilizan una función lineal para separar diferentes clases (figura 1.6). Estos modelos asumen que las clases pueden ser separadas por una línea (en 2D), un plano (en 3D) o un hiperplano (en dimensiones superiores). Son simples, rápidos y eficientes, ideales para problemas que presenten una clara división lineal.

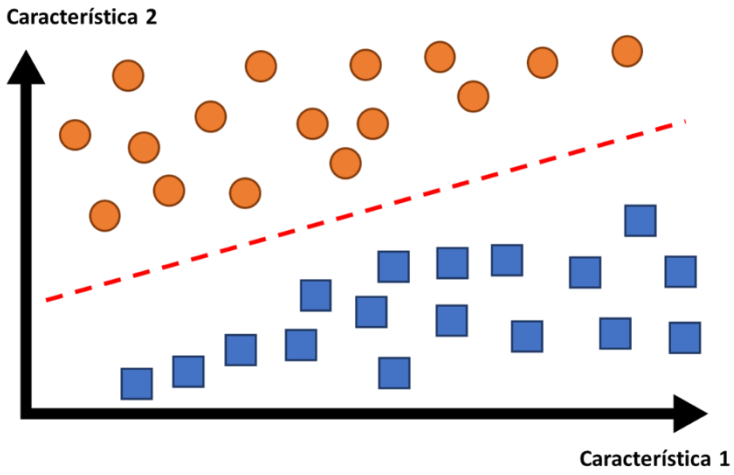


Figura (1.6). Solución de un problema de clasificación linealmente separable.

Entre los clasificadores lineales, algunos son particularmente destacados y ampliamente utilizados en diversas aplicaciones debido a su eficacia, flexibilidad y robustez. Los más importantes y comúnmente usados son los siguientes.

**Regresión Logística**. Aunque se llama regresión logística, este algoritmo se usa principalmente para la clasificación binaria. Modela la probabilidad de que una instancia pertenezca a una de las dos clases utilizando la función logística. Es muy útil en situaciones donde se necesita predecir la presencia o ausencia de un característica (como sí/no, verdadero/falso) (figura 1.7).

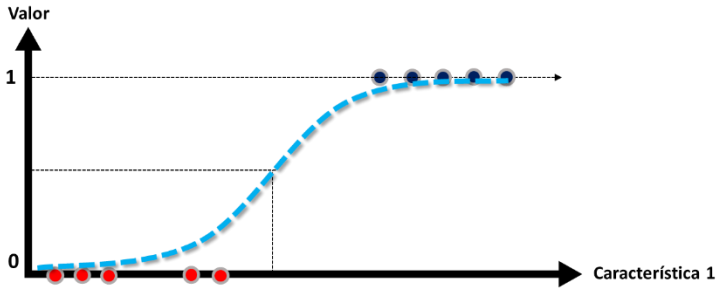


Figura 1.7. Regresión logística binaria.

**Clasificador de Vectores de Soporte Lineal *(Linear Support Vector Machines, Linear SVC).*** Es un algoritmo de clasificación que busca el hiperplano que maximiza el margen entre las clases. Este enfoque es particularmente efectivo en espacios de alta dimensión y cuando las clases son linealmente separables. La ventaja principal del Linear SVC radica en su eficacia en manejar datos con muchas dimensiones y en su capacidad para proporcionar una frontera de decisión clara y definida. Sin embargo, enfrenta limitaciones en conjuntos de datos muy grandes, ya que no escala bien con el número de muestras, lo que puede llevar a un aumento en el tiempo de computación y los requisitos de memoria.

**Análisis Discriminante Lineal *(Linear Discriminant Analysis, LDA).*** Es un método estadístico que busca encontrar la combinación lineal óptima de características que mejor separa dos o más clases distintas dentro de un conjunto de datos. La ventaja principal del modelo radica en su enfoque en maximizar la distancia entre las medias de diferentes clases mientras se minimiza la variación dentro de cada clase. Sin embargo, el *LDA* tiene la limitación de suponer que las características siguen distribuciones gaussianas y que todas las clases comparten matrices de covarianza iguales, lo que puede no ser realista en todos los conjuntos de datos.

**Máquina de Soporte de Vectores *(Support Vector Machine, SVM)*.**

Hemos visto que el algoritmo SVC lineal es un tipo de algoritmo utilizado para clasificación lineal. Se puede pensar a las Máquinas de Vectores de soporte como una extensión de los *Linear SCV* que utilizan una estrategia, para enfrentarse a escenarios en los que la separación de los grupos es de tipo no lineal, de aumento en las dimensiones del espacio original. Aunque los grupos no sean linealmente separables en el espacio original no implica que no lo sean en uno con más dimensiones. La imagen presenta dos grupos cuya separación en dos dimensiones no es lineal, pero sí lo es al añadir una tercera dimensión (figura 1.8).

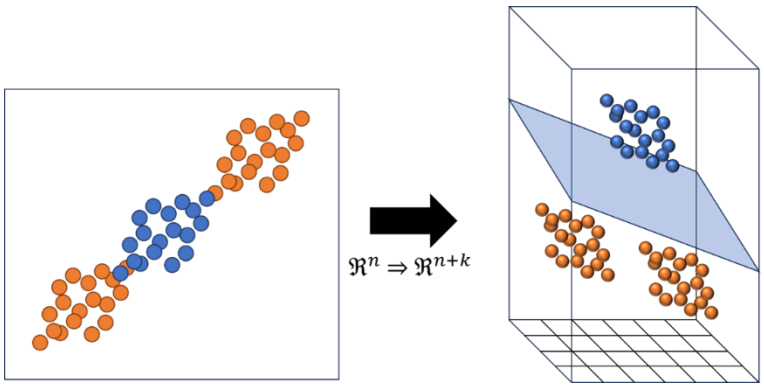


Figura (1.8). Solución de un problema de clasificación no linealmente separable en dos dimensiones mediante el uso de SVM para aumentar su dimensionalidad.

En el algoritmo SVM, para aumentar la dimensionalidad del espacio original se utilizan funciones especiales conocidas como kernels. Estos kernels transforman los datos de entrada a un espacio de características más adecuado para la separación lineal. Los tipos más comunes de kernels en SVM son:

**Kernel Lineal.** Es una generalización del SVM Lineal. Consiste simplemente en configurar el kernel con una función lineal, ideal para datos linealmente separables, donde una línea o hiperplano puede separar claramente las clases. Su ventaja radica en no incrementar la complejidad del modelo innecesariamente. Se debería utilizar solamente si se desea flexibilidad a la hora de implementar distintos kernels para analizar el problema.

**Kernel Polinómico.** Este kernel convierte los datos a un espacio polinomial de grado definido. Resulta útil cuando las relaciones entre los atributos son más complejas que simples relaciones lineales. Un grado polinómico demasiado alto implica un espacio de características por demás sofisticado lo que puede provocar en algunos casos problemas con el modelo

**Kernel de Base Radial (RBF) o Gaussiano.** Es uno de los kernels más populares y versátiles, efectivo en espacios de alta dimensión y en situaciones donde las relaciones entre los datos no son claramente lineales. El kernel RBF es capaz de manejar relaciones entre atributos que no son ni lineales ni polinómicas.

**Árboles de Decisión *(Decision Trees).***

En el contexto del aprendizaje automático, los árboles de decisión son modelos predictivos que representan una serie de decisiones y sus posibles consecuencias. Tal como su nombre lo indica, la estructura de este tipo de modelos se puede asemejar a la de un árbol, ya

que se compone de nodos, ramas y hojas (Figura 1.9). A continuación, describiremos sus principales componentes.

**Nodo Raíz.** Este es el primer nodo del árbol de decisión. No tiene ningún nodo "padre" (es decir, no proviene de otro nodo). Contiene la condición o pregunta que divide el conjunto de datos inicial en dos o más subconjuntos. Podemos pensar en el nodo raíz como el punto de partida de cualquier decisión en el árbol.

**Nodo Hijo.** Los nodos hijos son los nodos que resultan de dividir otro nodo. Cada nodo en el árbol (excepto el nodo raíz) es un nodo hijo de algún otro nodo. Un nodo hijo representa una de las respuestas a la pregunta o condición planteada en el nodo padre. Los nodos hijos pueden ser otros nodos de decisión (lo que lleva a más divisiones) o pueden ser nodos hoja.

**Hoja (o Nodo Hoja).** Un nodo hoja es un nodo sin hijos. Es el final de una rama en el árbol. Representa una decisión final o un resultado en el proceso de toma de decisiones. En el contexto del aprendizaje automático, un nodo hoja generalmente representa la clasificación final o el valor de predicción para las instancias que llegan a este punto en el árbol.

**Rama.** Cada rama emerge de un nodo y conduce a otro nodo, ya sea un nodo hijo o un nodo hoja. La idea detrás de las ramas es ilustrar cómo se llega a diferentes conclusiones o resultados en función de una serie de decisiones o criterios.

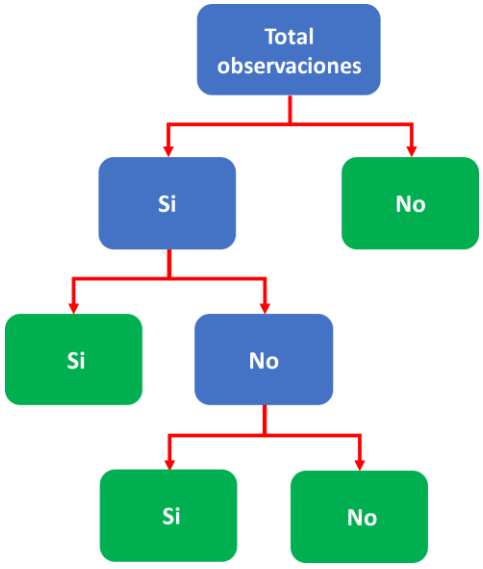


Figura (1.9). Ejemplo de árbol de decisión. En azul se presentan los nodos de decisión, en verde los nodos hojas y en rojo las ramas.

**K Vecinos Más Cercanos *(K-Nearest Neighbors, K-NN)*.**

El algoritmo K-NN clasifica nuevos datos basándose en ejemplos existentes. Para clasificar un nuevo dato, busca en los datos disponible y encuentra cuáles son los 'K' más cercanos a este nuevo dato. 'K' es un número elegido a priori, como 3, 5, 10, etc., y representa cuántos vecinos queremos considerar (figura 1.10). Una vez hecho esto, clasifica el nuevo dato según la categoría más común entre sus 'K' vecinos más cercanos. Es un enfoque directo y efectivo, especialmente en situaciones donde los datos se superponen mucho entre categorías. Sin embargo, si tenemos una cantidad enorme de datos, K-NN puede volverse lento porque necesita examinar y comparar con muchos vecinos cada vez que clasifica un nuevo dato.

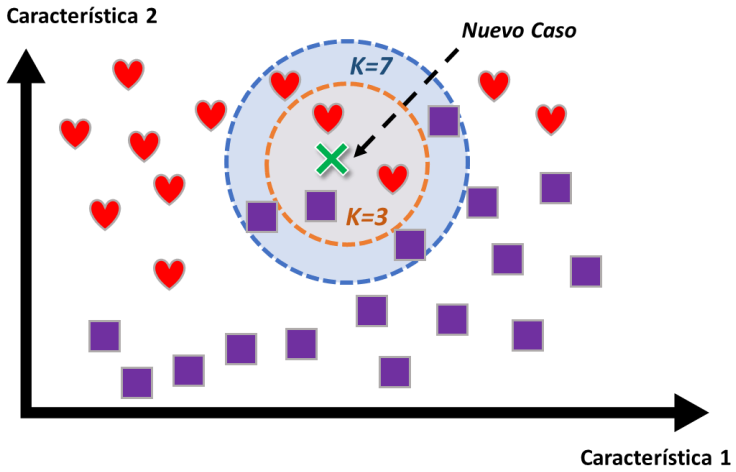


Figura (1.10). Ejemplo de algoritmo K-NN. El valor de "K" indica la cantidad de datos conocidos más cercanos al nuevo caso que se tomarán para realizar la clasificación.

**Bosques Aleatorios *(Random Forests)*.**

El algoritmo *Random Forest* funciona creando un "bosque" de árboles de decisión. Cada árbol de decisión en el bosque es un modelo que hace predicciones o clasificaciones basándose en una serie de preguntas y respuestas sobre los datos.

Cuando se presenta un nuevo dato para clasificar o predecir algo con *Random Forest*, el algoritmo no se basa en un único árbol de decisión, sino que consulta a una gran cantidad de ellos. Cada árbol de decisión es entrenado con un subconjunto de los datos, lo que significa que cada árbol tiene una perspectiva ligeramente diferente y se especializa en diferentes aspectos de los datos.

Para llegar a una decisión final, *Random Forest* aplica un proceso llamado "votación por mayoría". Esto significa que cada árbol en el bosque da su predicción, y la respuesta más común entre todos los árboles es la que elige *Random Forest* como resultado final (figura 1.11). Este método de combinar múltiples modelos para mejorar la precisión y la robustez se conoce como "*ensemble learning*". *Ramdon Forest* utiliza una técnica denominada *"Bootstrap Aggregating" (Bagging)* que consiste en crear múltiples conjuntos de datos separados a partir de nuestro conjunto de datos original. Esto se hace mediante un proceso llamado *"Bootstrap"*, que es básicamente tomar muestras aleatorias de los datos con reemplazo. Esto significa que el mismo dato puede aparecer más de una vez en un conjunto de muestras.

*Random Forest* es particularmente eficaz porque reduce el riesgo de sobreajuste (cuando un modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos) y es muy adaptable a una amplia variedad de tareas de clasificación y regresión. Además, al utilizar múltiples árboles, es menos probable que el modelo se vea afectado por el ruido en los datos de entrenamiento.

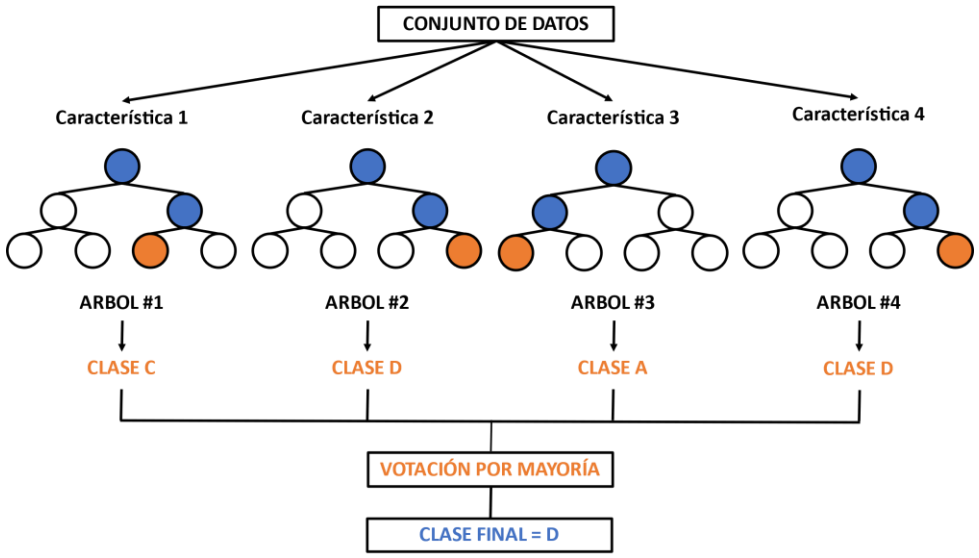


Figura (1.11). *Random Forest* utiliza el método de *bagging* para el entrenamiento (y combinación) de modelos y la votación por mayoría para llegar al consenso.

**1.5.1.1.2. Algoritmos de Regresión.**

Los algoritmos de regresión en el aprendizaje automático son esenciales para predecir valores numéricos continuos basados en datos previos (figura 1.12).

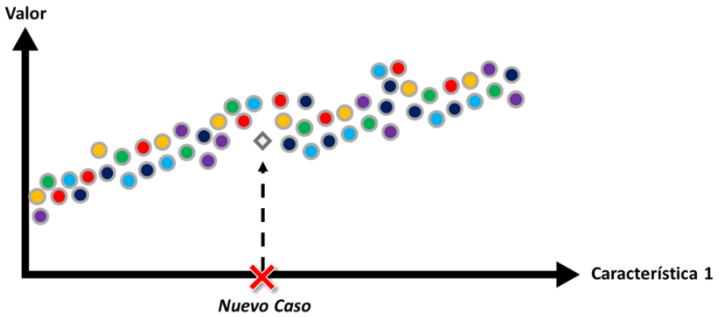


Figura 1.12. En los problemas de regresión se desea predecir un valor, dada una característica de entrada.

A continuación, presentaremos los principales algoritmos de regresión que se utilizan en aprendizaje automático.

**Regresión Lineal**. La regresión lineal es uno de los algoritmos más fundamentales y simples en el aprendizaje automático, usado para predecir un valor numérico basándose en la relación lineal entre variables. Funciona bien cuando la relación entre las variables independientes (predictores) y la variable dependiente (objetivo) es aproximadamente lineal. Es efectiva para entender cómo varía el valor de una variable al cambiar otra, y su sencillez la hace fácil de implementar y de interpretar (figura 1.13).

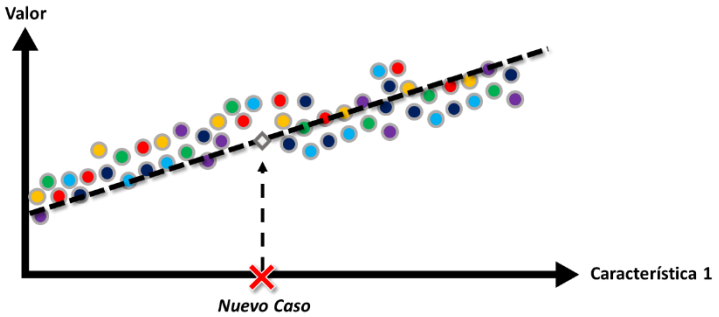


Figura 1.13. Regresión lineal.

**Regresión Polinómica**. La regresión polinómica extiende la regresión lineal permitiendo relaciones más complejas entre las variables. En lugar de una línea recta, ajusta una curva polinómica a los datos, lo que la hace más adecuada para modelos donde la relación entre las variables independientes y la dependiente no es estrictamente lineal (figura 1.14).

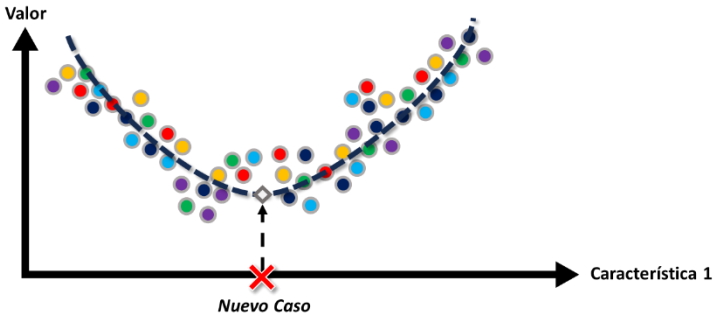


Figura 1.14. Regresión Polinómica.

Existen además algoritmos de clasificación que se pueden utilizar para problemas de regresión, a continuación, mencionaremos los más importantes:

**Árboles de Decisión para Regresión**. Los árboles de decisión se pueden aplicar a problemas de regresión. Dividen los datos en segmentos más pequeños, o "ramas", basándose en diferentes criterios, y cada rama del árbol da como resultado un valor numérico. Son útiles por su facilidad de interpretación y por su capacidad de manejar datos no lineales.

***Random Forest* para Regresión**. En *Random Forest* para regresión, se combinan múltiples árboles de decisión, cada uno construido sobre una muestra de los datos. Las predicciones de todos estos árboles se promedian para obtener la predicción final. Este método mejora la precisión y reduce el riesgo de sobreajuste, haciendo que el modelo sea robusto frente a datos variados.

**Máquinas de Vectores de Soporte para Regresión (SVR)**: *SVR* es una adaptación de las Máquinas de Vectores de Soporte, utilizada para problemas de regresión. Busca un hiperplano en un espacio multidimensional que mejor ajuste los datos, tratando de minimizar el error. Es eficaz en conjuntos de datos de alta dimensión y es conocido por su capacidad de manejar relaciones no lineales.

### 1.5.1.2. Aprendizaje no supervisado.

El aprendizaje no supervisado resulta especialmente valioso en situaciones donde se desconoce la estructura subyacente de los datos o cuando se busca una exploración inicial para identificar patrones interesantes o inesperados. En contraste con el aprendizaje supervisado, donde los modelos se entrenan con datos etiquetados, el aprendizaje no supervisado no utiliza respuestas o etiquetas predefinidas. Esto implica un enfoque más

exploratorio y analítico, donde los algoritmos deben discernir la estructura de los datos sin ningún tipo de instrucción explícita.

Uno de los usos más comunes del aprendizaje no supervisado es en el agrupamiento (o clustering), donde se agrupan instancias similares. Esto es útil en campos como la biología para la clasificación de genes, en marketing para la segmentación de clientes, o en redes sociales para identificar grupos de intereses similares. Algunos algoritmos populares en esta categoría incluyen K-means, DBSCAN y algoritmos jerárquicos.

La reducción de dimensionalidad, otra aplicación clave del aprendizaje no supervisado, es esencial cuando se trabaja con datos de alta dimensión, como en la genómica o el procesamiento de imágenes. Técnicas como el análisis de componentes principales (PCA) y *el t-distributed Stochastic Neighbor Embedding* (t-SNE) son fundamentales para simplificar los datos manteniendo su estructura esencial, facilitando así su visualización y análisis.

En la detección de anomalías, el aprendizaje no supervisado puede identificar patrones de datos que no se ajustan a lo esperado. Esto es esencial en, por ejemplo, la detección de fraudes, el monitoreo de redes y la seguridad informática. Por otro lado, la detección de novedades se centra en identificar nuevas o inusuales instancias de datos, lo que es importante en el ámbito de la vigilancia o en la monitorización de condiciones en maquinarias para mantenimiento predictivo.

Finalmente, la generación de reglas de asociación es un área importante del aprendizaje no supervisado que se utiliza en análisis de mercado, como por ejemplo en la técnica de "market basket analysis", para descubrir relaciones entre variables en grandes bases de datos. Herramientas como el algoritmo *A priori* son ejemplos de cómo se pueden identificar estas asociaciones de manera eficiente.

#### 1.5.1.2.1. Agrupamiento (Clustering).

El agrupamiento o *clustering*, en el contexto del aprendizaje no supervisado, es un método de análisis de datos que se utiliza para identificar estructuras o patrones intrínsecos en un conjunto de datos sin etiquetar. El objetivo principal del *clustering* es agrupar o segmentar un conjunto de datos en varias categorías o "*clusters*" de tal manera que los datos dentro de cada grupo sean más similares entre sí en comparación con los de otros grupos (figura 1.15).

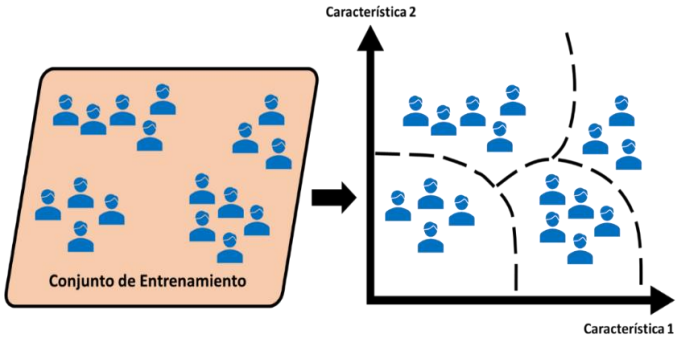


Figura 1.15. Un conjunto de entrenamiento sin etiquetar y un algoritmo de agrupamiento para aprendizaje automático no supervisado

Existen diversas técnicas de *clustering,* entre las cuales se encuentran el *clustering* particional, el *clustering* jerárquico, el *DBSCAN* y el *clustering* basado en modelos.

***Clustering* Particional.**

El *clustering* particional divide los datos en un número específico de *clusters* sin establecer ninguna estructura jerárquica, asignando cada punto de datos a un solo cluster, buscando optimizar cierto criterio, como la minimización de la varianza dentro del cluster o la maximización de la distancia entre los *clusters*. Este tipo de *clustering* es especialmente útil cuando se tiene una idea previa del número de clusters en los datos.

***K-Means***. Es el algoritmo de *clustering* particional más reconocido, ampliamente utilizado por su eficiencia y simplicidad. Este algoritmo busca dividir un conjunto de datos en un número predefinido de 'K' *clusters*, con el objetivo de minimizar la varianza dentro de cada uno de estos grupos. A través de un proceso iterativo, *K-Means* reasigna puntos a los *clusters* en función de su proximidad al centroide (punto central) de cada *cluster*, lo que lo hace particularmente útil en aplicaciones como la segmentación de clientes, la categorización de documentos y la compresión de imágenes. Una de las principales ventajas de *K-Means* es su eficiencia computacional y la facilidad de implementación. Sin embargo, presenta desventajas, como la necesidad de especificar el número de clusters de antemano y su sensibilidad a los valores iniciales de los centroides y a la presencia de valores atípicos en los datos.

***K-Medoids***. También conocido como *PAM (Partitioning Around Medoids)*, es un algoritmo de *clustering* que comparte similitudes con *K-Means*, pero con una diferencia clave: en lugar de calcular centroides, *K-Medoids* elige puntos de datos reales como representantes, o *"medoids"*, de cada cluster. Esta característica distintiva hace que *K-Medoids* sea más

robusto frente a valores atípicos en comparación con *K-Means*, lo que lo convierte en una opción preferible en aplicaciones donde los centroides calculados no son representativos o no aplicables, como en el caso de los datos categóricos. Una de las ventajas principales de *K-Medoids* es su menor sensibilidad a los valores atípicos, aunque, como desventaja, es más costoso desde el punto de vista computacional que *K-Means*, particularmente en conjuntos de datos de gran tamaño. Este algoritmo es especialmente útil en situaciones donde se requiere una representación más precisa y resistente de los clusters.

***Fuzzy C-Means (FCM).*** Este método introduce un enfoque más flexible en la asignación de *clusters*. A diferencia de los métodos tradicionales, donde cada punto de datos pertenece a un único cluster, el algoritmo asigna a cada punto un grado de pertenencia a cada *cluster*, lo que permite un solapamiento suave entre ellos. Esta característica lo hace particularmente efectivo en situaciones donde los límites entre los *clusters* no están claramente definidos, como en casos donde la categorización no es absoluta. Una ventaja significativa de este enfoque es que proporciona información valiosa sobre la incertidumbre y la pertenencia parcial de los puntos a los *clusters*. Sin embargo, este algoritmo es más complejo y computacionalmente más costoso en comparación con métodos de *clustering* particionales tradicionales, lo que puede ser una limitación en ciertos contextos de análisis de datos.

**Clustering Jerárquico *(Hierarchical Clustering).***

Esta técnica de organiza los datos en una estructura de árbol o dendrograma, ya sea mediante un enfoque aglomerativo (fusionando *clusters*) o divisivo (dividiendo *clusters*). Es particularmente útil en el análisis de datos biológicos y otros campos donde la representación jerárquica es informativa. Aunque no necesita un número predefinido de *clusters* y proporciona una interpretación intuitiva a través del dendrograma, es computacionalmente más costoso y menos efectivo para conjuntos de datos grandes.

**Algoritmo de Propagación de Afinidad *(Affinity Propagation).*** Este algoritmo se basa en el concepto de "pasar mensajes" entre puntos de datos, considerándolos todos como posibles ejemplos y determinando iterativamente los más representativos. Útil en reconocimiento de imágenes y situaciones donde no está claro cuántos *clusters* podrían ser apropiados, este método no requiere definir el número de *clusters* de antemano y puede ser más informativo y preciso que *K-Means*, aunque es más costoso computacionalmente.

**Enlace Ward *(Ward's Linkage).*** El método de enlace Ward busca minimizar la suma de cuadrados dentro de todos los *clusters*, fusionando en cada paso aquellos que resultan en el menor incremento en la suma total de cuadrados. Útil en muchas aplicaciones científicas, tiende a crear *clusters* más compactos y esféricos, pero puede ser sensible a valores atípicos.

**Enlace Promedio *(Average Linkage)*.** Este método fusiona *clusters* basándose en la distancia promedio entre todos los pares de puntos en los *clusters* considerados para la fusión. Es efectivo en situaciones donde los *clusters* se fusionan naturalmente y no necesariamente tienen formas esféricas. Aunque es menos susceptible a valores fueras de escala en comparación con el enlace completo, puede ser difícil de interpretar en conjuntos de datos con estructuras de *clusters* complejas.

**Enlace Completo *(Average Linkage).*** En el enlace completo, los *clusters* se fusionan basándose en la distancia máxima entre los puntos en los *clusters* considerados para la fusión. Adecuado para conjuntos de datos donde es importante que los puntos en un *cluster* estén cercanos entre sí, tiende a encontrar *clusters* bien separados y es menos susceptible a los valores fuera de escala que el enlace promedio, aunque puede crear *clusters* de tamaños desiguales y es sensible a las variaciones en la estructura de los datos.

***Clustering* Espectral.** El *clustering* espectral crea un mapa de las similitudes entre los puntos, comparando sus distancias. Luego, utiliza un enfoque matemático, basado en autovalores, para transformar dimensionalmente estos puntos con el objetivo de encontrar los grupos ocultos. Esta transformación permite que el algoritmo identifique *clusters*, incluso si tienen formas irregulares o si no están claramente separados. Es especialmente útil en escenarios como la segmentación de imágenes o en el análisis de redes sociales. Su desventaja radica en su exigencia en términos de recursos computacionales y en su requerimiento de una cuidadosa selección de parámetros para funcionar correctamente.

**Clustering basado en Densidad.**

El *clustering* basado en densidad es un enfoque en el análisis de datos que agrupa puntos en función de la densidad, es decir, la cercanía o concentración de puntos en una región específica. Este tipo de *clustering* es diferente a los métodos anteriores, ya que no asume que los *clusters* tengan una forma o tamaño específico, y es capaz de identificar *clusters* basados en la densidad de puntos en un espacio.

**DBSCAN *(Density-Based Spatial Clustering of Applications with Noise).*** Este algoritmo agrupa puntos cercanos en regiones de alta densidad, marcando como valores fuera de escala aquellos en áreas de baja densidad. Este algoritmo es efectivo para identificar *clusters* de formas arbitrarias en datos espaciales. No necesita un número predeterminado de *clusters* y puede manejar *outliers*, pero la selección de parámetros puede ser complicada, y su rendimiento varía en datos de alta dimensión.

**OPTICS (Ordering Points To Identify the Clustering Structure).** similar a *DBSCAN*, aborda algunas de sus limitaciones al identificar clusters en datos de densidad variable. Útil en la detección de anomalías y conjuntos de datos espaciales, no requiere especificar el número de clusters y puede identificar clusters en datos con variaciones de densidad. Sin embargo, elegir los parámetros adecuados puede ser complicado.

***Clustering* basado en modelos.**

El *clustering* basado en modelos es un enfoque de agrupación en el aprendizaje no supervisado que utiliza modelos estadísticos para determinar la estructura de los *clusters* dentro de un conjunto de datos. A diferencia de los métodos anteriores, los algoritmos de *clustering* basado en modelos asumen que los datos provienen de una mezcla de varias distribuciones estadísticas. Cada distribución representa un *cluster*, y los datos se agrupan en función de cuál distribución es más probable que los haya generado.

**Clustering Basado en Mezclas Gaussianas *(Gaussian Mixture Models, GMM).*** Este algoritmo utiliza un enfoque probabilístico para el agrupamiento de datos, asumiendo que los datos provienen de una mezcla de varias distribuciones gaussianas (o normales), cada una representando un *cluster* diferente. En este modelo, cada punto de datos tiene una probabilidad de pertenecer a cada una de las distribuciones gaussianas. La optimización del *GMM* implica ajustar los parámetros de estas distribuciones (como la media y la varianza) para maximizar la probabilidad de los datos observados. Este método es especialmente adecuado para datos complejos y multidimensionales, y es capaz de identificar *clusters* que tienen formas elípticas y tamaños variables. Una ventaja clave del *GMM* es su flexibilidad en la forma de los *clusters* y su capacidad para asignar probabilidades de pertenencia, ofreciendo una perspectiva más matizada sobre cómo se agrupan los datos.

**Clustering Basado en Modelos Bayesianos *(Bayesian Model-Based Clustering, BMBC)*.** El *Clustering* Basado en Modelos Bayesianos se enfoca en utilizar principios de inferencia

bayesiana para determinar la pertenencia a los *clusters*. Estos modelos incorporan una medida de incertidumbre y utilizan probabilidades previas sobre los parámetros del modelo, lo que permite incorporar conocimientos o suposiciones previas en el proceso de agrupamiento. En ***BMBC*** se destacan los *modelos de mezclas bayesianas* y los *procesos de Dirichlet*. Los modelos de mezclas bayesianas utilizan la inferencia bayesiana para estimar la probabilidad de cada *cluster*, permitiendo un agrupamiento más preciso y flexible. Los procesos de Dirichlet, por otro lado, ajustan automáticamente el número de clusters basándose en los datos, adecuados para situaciones donde la cantidad de clusters no es conocida de antemano. Ambos métodos ofrecen un enfoque adaptable y detallado para el clustering en conjuntos de datos complejos.

***Clustering Basado en Modelos de Markov Ocultos* (*Hidden Markov Model Clustering, HMM*).** Este enfoque se aplica especialmente en el análisis de datos secuenciales o temporales. En un *HMM*, se asume que los datos son generados por un proceso de Markov con estados ocultos, donde cada estado representa un cluster. El modelo busca identificar estos estados ocultos basándose en las observaciones de los datos. Los *HMM* son particularmente útiles para analizar secuencias en las que la dependencia temporal entre las observaciones es importante, como en las series temporales financieras, el análisis del lenguaje o las secuencias genéticas. Estos modelos pueden capturar la dinámica temporal y las transiciones entre diferentes estados o clusters, proporcionando una comprensión detallada de la estructura subyacente en los datos secuenciales.

**1.5.1.2.2. Visualización de Datos**

Los algoritmos de visualización son capaces de tomar grandes volúmenes de datos complejos y producir una representación visual en dos o tres dimensiones (figura 1.16). Estos algoritmos intentan mantener intacta la estructura original de los datos para facilitar su interpretación y el descubrimiento de patrones ocultos. Por ejemplo, podríamos visualizar los patrones de compra de clientes en un centro comercial, lo que nos ayudaría a entender mejor el comportamiento del consumidor. ***T-Distributed Stochastic Neighbor Embedding (t-SNE)*** es un algoritmo muy utilizado para visualización de alta dimensión. Reduce la dimensionalidad de los datos de manera efectiva para facilitar su visualización en 2D o 3D.

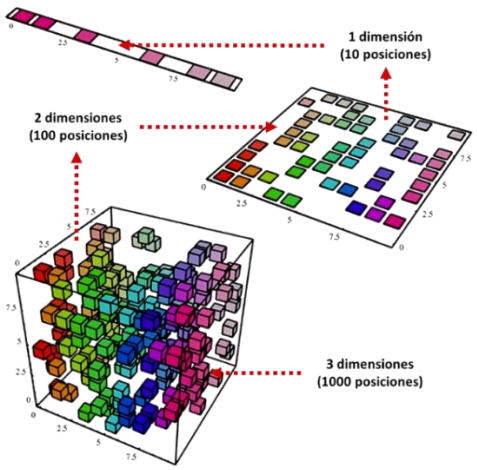


Figura 1.16. Un ejemplo de cómo el algoritmo t-SNE facilita la visualización de datos.

### 1.5.1.2.3. Reducción de Dimensionalidad

En la reducción de dimensionalidad, el objetivo es simplificar los datos sin perder información importante (figura 1.17). Esto se logra combinando características correlacionadas en una sola. Por ejemplo, en un conjunto de datos sobre salud, la altura y el peso podrían combinarse en un único indicador de índice de masa corporal. Esto, además de simplificar el análisis, mejora la eficiencia de los algoritmos posteriores ya que permite una mayor velocidad de ejecución, una mejor ocupación de espacio en memoria, y en alguno casos, un mejor funcionamiento general. ***El Análisis de Componentes Principales (Principal Component Analysis, PCA)*** es un método popular para reducir la dimensionalidad de los conjuntos de datos, identificando y manteniendo solo las direcciones (componentes principales) en las cuales hay más variabilidad.

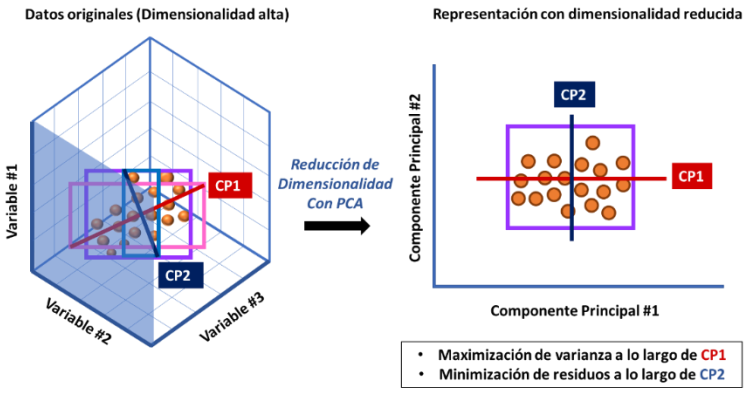


Figura 1.17. Reducción de dimensionalidad con algoritmo PCA.

#### 1.5.1.2.4. Detección de Anomalías.

La detección de anomalías es utilizada para identificar comportamientos inusuales o datos atípicos. Por ejemplo, en un sistema de monitoreo de tráfico, podría utilizarse para detectar vehículos que se comportan de manera inusual, lo que podría indicar condiciones de tráfico anormales o emergencias. Durante el entrenamiento, el algoritmo se familiariza con patrones de tráfico normales y luego es capaz de identificar desviaciones significativas de estos patrones. (figura 1.18). ***One-Class SVM*** es adecuado para la detección de anomalías en datos no etiquetados.

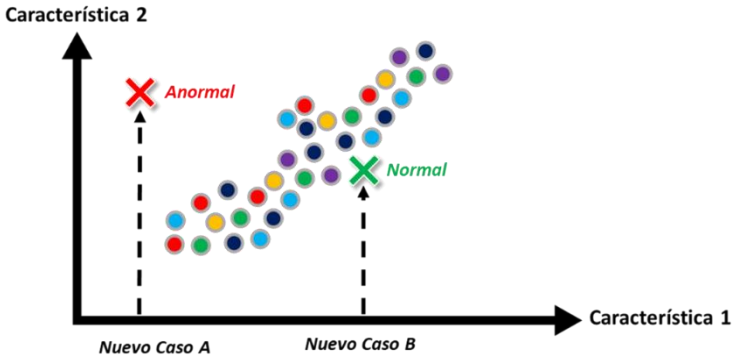


Figura 1.18. Detección de anomalías

#### 1.5.1.2.5. Detección de Novedades

La detección de novedades se enfoca en identificar instancias nuevas y distintas de todas las del conjunto de entrenamiento. Por ejemplo, en un sistema de vigilancia de fauna, un algoritmo de detección de novedades podría identificar la aparición de una especie animal no registrada previamente en el área. ***One-Class SVM*** puede también ser utilizado para la detección de novedades.

#### 1.5.1.2.6. Reglas de Asociación

El aprendizaje de reglas de asociación busca encontrar relaciones entre atributos en grandes conjuntos de datos. Por ejemplo, en el análisis de datos de una aplicación móvil, podríamos descubrir que los usuarios que juegan ciertos tipos de juegos también tienden a descargar determinadas aplicaciones de productividad, lo que puede resultar de mucha utilidad para estrategias de marketing cruzado y recomendaciones de aplicaciones. ***Algoritmo A priori*** es comúnmente usado para extraer reglas de asociación. Busca combinaciones

frecuentes de ítems en bases de datos transaccionales y es muy usado en análisis de cestas de mercado.

#### 1.5.1.3. Aprendizaje semi-supervisado

Dado que el etiquetado de datos suele llevar mucho tiempo y ser costoso, suelen ocurrir muy frecuentemente la existencia de set de datos con muchas instancias sin etiquetar y pocas instancias etiquetadas. Algunos algoritmos pueden manejar datos parcialmente etiquetados. Esto se llama aprendizaje semi-supervisado (figura 1.19). En la figura apreciamos algunas observaciones etiquetadas en dos clases (corazones y cuadrados) y una gran cantidad de observaciones sin etiquetar (círculos) que ayudan a clasificar una nueva observación (la cruz) en la clase de corazón en lugar de en la clase de cuadrado (aunque esté más cerca de las observaciones etiquetadas como cuadrados).

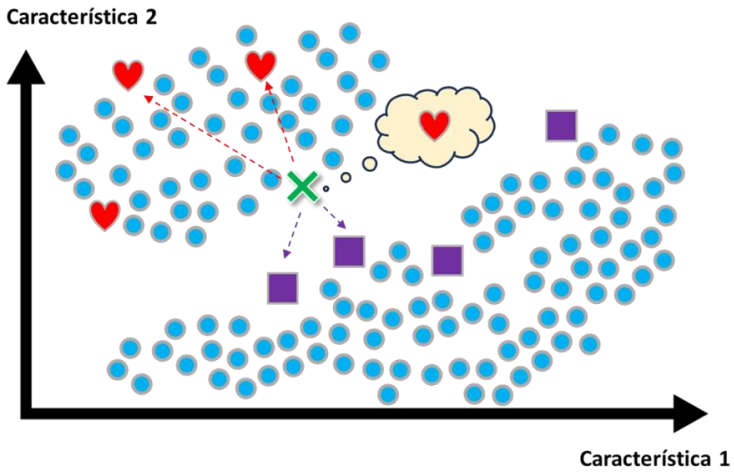


Figura 1.19. Aprendizaje semi-supervisado.

Algunos servicios de alojamiento de fotos, como *Google Photos*, son buenos ejemplos de esto. Una vez que subimos nuestras fotos al servicio, este reconoce automáticamente, mediante un algoritmo no supervisado de agrupación, que la misma persona A aparece en determinadas imágenes, mientras que otra persona B aparece en otras (pudiendo ambas personas aparecer juntas en la misma imagen). El sistema requiere únicamente que identifiquemos a cada persona mediante una etiqueta individual. Una vez hecho esto, podrá reconocer y nombrar a todos los individuos en cada fotografía, lo que resulta sumamente práctico para facilitar la búsqueda de fotos.

La mayoría de los algoritmos de aprendizaje semi-supervisado combinan elementos tanto del aprendizaje supervisado como del no supervisado. Se utilizan en situaciones donde se dispone de una gran cantidad de datos no etiquetados junto con un conjunto más pequeño de datos etiquetados. Algunos de los algoritmos más utilizados se detallan a continuación.

**1.5.1.3.1. Autoentrenamiento *(Self-training)*.**

Es uno de los métodos más simples. Primero, se entrena un modelo con el conjunto de datos etiquetado y luego se usa ese modelo para predecir etiquetas en el conjunto no etiquetado. Las predicciones más confiables se agregan al conjunto de entrenamiento y el proceso se repite. Es probablemente el método más simple y uno de los más utilizados debido a su facilidad de implementación. Su popularidad se debe a que no requiere un marco de trabajo complicado y puede aplicarse en una variedad de contextos con diferentes modelos de aprendizaje supervisado.

**1.5.1.3.2. Coentrenamiento *(Co-training)*.**

En este enfoque, se entrenan dos modelos en paralelo en diferentes vistas (subconjuntos de características) del mismo conjunto de datos. Cada modelo aprende y luego proporciona etiquetas para el otro modelo, mejorando iterativamente ambos modelos. Se utiliza especialmente en contextos donde los datos se pueden dividir naturalmente en dos conjuntos de características distintas y complementarias.

**1.5.1.3.3. Propagación de Etiquetas *(Label Propagation)*.**

Este método utiliza la similitud entre ejemplos para propagar las etiquetas desde los datos etiquetados a los no etiquetados. Construye un grafo donde los nodos son ejemplos y los bordes representan similitudes, y luego utiliza este grafo para extender las etiquetas. Este método es ampliamente utilizado debido a su eficiencia en tareas donde la estructura de los datos puede ser representada efectivamente en un grafo, como en redes sociales o en análisis de texto.

**1.5.1.4. Aprendizaje auto-supervisado.**

En el ámbito del aprendizaje automático, nos encontramos con el aprendizaje auto-supervisado, una metodología que consiste en la generación autónoma de un conjunto de datos etiquetado a partir de uno que originalmente carece de etiquetas. Esta técnica posibilita la posterior implementación de algoritmos de aprendizaje supervisado.

Para ilustrar este concepto, tomemos como ejemplo un conjunto de fotografías de perros sin información específica sobre sus razas. Iniciamos el proceso ocultando selectivamente características distintivas en estas imágenes, como partes de la cara o del pelaje, y posteriormente entrenamos un modelo para predecir y restaurar estas áreas ocultas. Durante la fase de entrenamiento, las imágenes parcialmente ocultadas funcionan como entradas y las imágenes originales completas como etiquetas (figura 1.20).

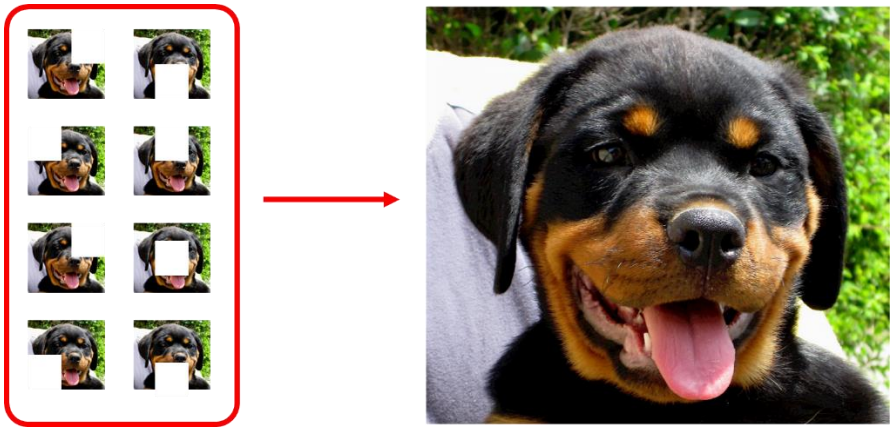

Figura 2.20. Aprendizaje auto-supervisado

El modelo desarrollado tiene aplicaciones intrínsecas, como en la restauración de imágenes deterioradas. Sin embargo, su utilidad se extiende a adaptaciones para tareas más específicas. Por ejemplo, si el objetivo es la clasificación de perros por raza, el modelo inicial de reconstrucción de imágenes puede ser ajustado para identificar características raciales distintivas. Habiendo adquirido la capacidad de diferenciar entre atributos caninos variados, el modelo se perfecciona finalmente con un conjunto de datos que incluye etiquetas específicas de razas, facilitando así una clasificación precisa.

Los técnicas de aprendizaje auto supervisado más utilizados son *Contrastive Learning, Autoencoders* con Perturbaciones, *Predictive Coding*, *Clustering*, Redes Siamesas, Redes Tripletas y *Self-training with Noisy Student*.

**1.5.1.4.1. Contrastive Learning.**

Esta técnica aprende representaciones útiles de los datos al maximizar la similitud entre pares de datos "positivos" (generalmente, diferentes vistas del mismo dato) y minimizar la similitud entre pares "negativos" (datos diferentes).

**1.5.1.4.2. *Autoencoders* con Perturbaciones.**

Los *autoencoders* son redes neuronales entrenadas para reconstruir su entrada en la salida. En el aprendizaje auto-supervisado, se introduce una perturbación (como el enmascaramiento de parte de la entrada) y el modelo aprende a reconstruir la entrada original a partir de esta versión perturbada.

**1.5.1.4.3. Predictive Coding.**

Esta técnica se basa en predecir una parte de los datos dados los demás. Por ejemplo, predecir la próxima palabra en una secuencia o el próximo fotograma en un video.

**1.5.1.4.4. Clustering.**

Los algoritmos de *clustering* basan su estrategia en la capacidad que tienen para organizar los datos 'clusters' agrupados en función a sus características inherentes, sin necesidad de etiquetas previas. Posteriormente, estas asignaciones de clusters se utilizan como sustitutos de etiquetas en un contexto de aprendizaje supervisado.

**1.5.1.4.5. Redes Siamesas y Redes Tripletas.**

Las redes siamesas aprenden a diferenciar entre pares de entradas, mientras que las redes de tripletas extienden este concepto a triples de datos (un dato "ancla", uno "positivo" similar y uno "negativo" diferente). Estas redes se utilizan para aprender representaciones en las que datos similares están cerca en el espacio de representación, mientras que los datos diferentes están lejos.

**1.5.1.4.6. Self-training with Noisy Student.**

Esta técnica implica primero entrenar un modelo (el "profesor") en un conjunto de datos etiquetados, luego usar este modelo para etiquetar datos no etiquetados, y finalmente entrenar un nuevo modelo (el "estudiante") en este conjunto de datos ampliado, incluyendo tanto etiquetas reales como generadas.

Cabe resaltar que, a pesar de utilizar conjuntos de datos no etiquetados inicialmente, el aprendizaje auto-supervisado se diferencia del aprendizaje no supervisado, que se centra en tareas como la agrupación o la detección de anomalías. Al emplear etiquetas generadas durante el proceso de entrenamiento, el aprendizaje auto-supervisado se alinea más estrechamente con el aprendizaje supervisado, enfocándose en tareas como la clasificación y la

regresión. Por lo tanto, es pertinente considerar el aprendizaje auto-supervisado como una categoría distinta dentro del espectro del aprendizaje automático.

**1.5.1.5. Aprendizaje por refuerzo.**

El aprendizaje por refuerzo representa un paradigma distinto dentro del ámbito de la inteligencia artificial. En este enfoque, el agente (una entidad de aprendizaje en este contexto) interactúa activamente con su entorno, toma decisiones, ejecuta acciones y, basado en las consecuencias de esas acciones, recibe recompensas o penalizaciones. Esta metodología se orienta hacia el autoaprendizaje del agente para desarrollar una estrategia óptima, conocida como política, que maximice la acumulación de recompensas a lo largo del tiempo.

La política en el aprendizaje por refuerzo se refiere a un conjunto de reglas o directrices que el agente sigue para decidir sus acciones en función del estado actual del entorno. El programa *AlphaGo* de *DeepMind* ilustra este concepto de manera sobresaliente. En 2017, el sistema alcanzó un hito al derrotar a *Ke Jie*, el jugador número uno del mundo en Go, un juego notorio por su complejidad estratégica. Este logro fue posible gracias a que *AlphaGo* desarrolló su política ganadora a través del análisis de millones de partidas y numerosas sesiones de juego contra sí mismo. Durante sus partidas contra los expertos humanos, *AlphaGo* aplicó la política aprendida sin realizar ajustes en tiempo real, un enfoque conocido como “aprendizaje fuera de línea”.

Para comprender mejor cómo un agente aprende y desarrolla su política en un entorno más sencillo, tomemos el juego de TIC TAC TOE como ejemplo. Un agente que aprende a jugar TIC TAC TOE comienza sin conocimiento previo del juego. Inicialmente, sus movimientos pueden ser aleatorios, pero con cada partida, el agente observa los resultados de sus acciones (ganar, perder, empatar) y ajusta gradualmente su estrategia para maximizar las victorias. A través de este proceso iterativo de prueba y error, el agente aprende qué movimientos son más efectivos en diferentes situaciones, formando así su política (mejor estrategia) de juego (figura 2.21).

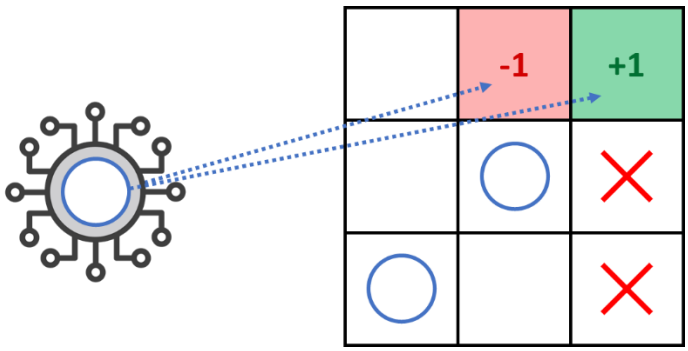


Figura 2.21. Aprendizaje por refuerzo.

Los algoritmos de aprendizaje automático por refuerzo permiten a los agentes aprender a tomar decisiones eficaces basándose en las recompensas obtenidas por sus acciones. A continuación, presentaremos los más utilizados.

**1.5.1.5.1. Q-Learning.**

*Q-Learning* se transformó en uno de los algoritmos más fundamentales y populares en el aprendizaje por refuerzo. Se basa en aprender una función de valor, denominada *Q-function*, que estima la utilidad de una acción en un estado determinado. El agente utiliza esta función para elegir las acciones que maximizan la recompensa esperada.

**1.5.1.5.2. Deep Q-Networks (DQN).**

*DQN* extiende el *Q-learning* utilizando redes neuronales profundas. Esto permite manejar estados y acciones en espacios de alta dimensión, haciéndolo adecuado para problemas más complejos, como juegos de video. *DQN* se hizo famoso por su éxito en aprender a jugar juegos de Atari directamente desde los píxeles de la pantalla.

**1.5.1.5.3. Policy Gradient Methods.**

Los métodos *Policy Gradient* se centran directamente en aprender la política óptima, en lugar de una función de valor. Ajustan la política para maximizar las recompensas mediante el cálculo del gradiente de la expectativa de recompensa.

**1.5.1.5.4. Actor-Critic Methods.**

Combinan ideas de *Q-learning* y *Policy Gradients*. Utilizan dos redes neuronales: el "actor" que propone una acción dada un estado, y el "crítico" que evalúa la acción. Al trabajar juntos, mejoran tanto la política de decisión como la estimación de valor.

Cada uno de estos algoritmos tiene sus fortalezas y aplicaciones particulares, y la elección de uno sobre otro dependerá de las especificidades del problema en cuestión. Aprendizaje por lotes versus aprendizaje en línea.

### 1.5.2. Posibilidad de aprendizaje incremental.

Otro criterio utilizado para clasificar los sistemas de aprendizaje automático es si el sistema puede aprender incrementalmente a partir de un flujo de datos entrantes.

#### 1.5.2.1. Aprendizaje por lotes.

En el aprendizaje por lotes, el sistema es incapaz de aprender de forma incremental, debiendo entrenarse mediante la utilización de todos los datos disponibles (aprendizaje por stock). Por lo general, esto requerirá mucho tiempo y recursos informáticos, por lo que normalmente se realiza sin conexión. Se entrena el sistema primero, y luego se implementa en un entorno de producción, donde opera aplicando lo aprendido sin actualizar sus conocimientos. Esta modalidad se conoce como aprendizaje fuera de línea *(offline learning)*.

Una limitación notable de los modelos entrenados mediante este método es que su rendimiento tiende a decaer con el tiempo, esto ocurre porque el modelo, una vez entrenado, no evoluciona, mientras que el entorno y los datos sí lo hacen. Este fenómeno a menudo se denomina deterioro del modelo *(model rot)* o deriva de datos *(data drift).* La solución consiste en volver a entrenar periódicamente el modelo con datos actualizados. La frecuencia de esta actualización varía según la aplicación: un modelo para clasificar tipos de flores requerirá actualizaciones menos frecuentes que uno destinado a predecir tendencias en el mercado financiero.

#### 1.5.2.2. Aprendizaje en línea.

En el aprendizaje en línea *(online learning)*, el sistema se entrena de forma incremental, procesando secuencialmente instancias de datos ya sea de manera individual o en pequeños conjuntos conocidos como mini-lotes. Cada etapa de aprendizaje resulta ser rápida y económica, permitiendo al sistema adaptarse y aprender de nuevos datos en tiempo real a medida que estos son recibidos. Esta metodología facilita la actualización continua del modelo y su adaptación a tendencias emergentes o cambios en los patrones de datos (figura 1.22).

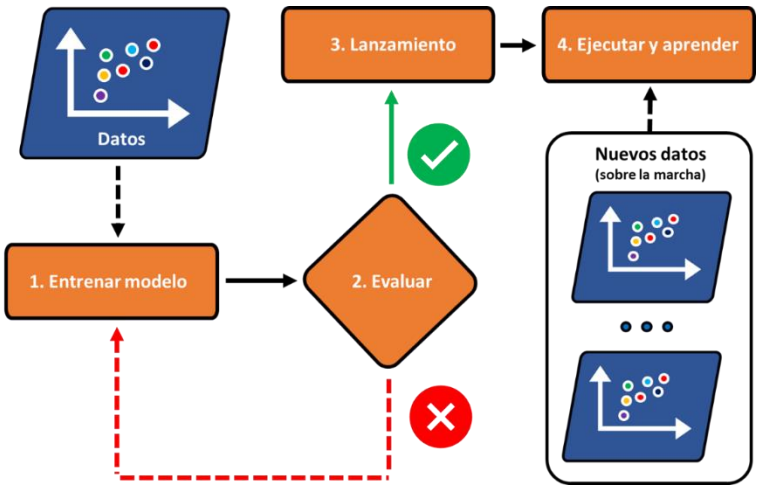


Figura 1.22. Aprendizaje en línea.

El aprendizaje en línea se destaca como una técnica esencial para sistemas que requieren una adaptación rápida a contextos dinámicos, como en la identificación de tendencias emergentes en el mercado de valores. Esta metodología resulta particularmente beneficiosa en escenarios con recursos computacionales restringidos, siendo ideal para el entrenamiento de modelos directamente en dispositivos móviles.

La eficiencia del aprendizaje en línea queda manejada al manejar conjuntos de datos de gran tamaño, imposibles de procesar en su totalidad dentro de la memoria estándar de una computadora. Este escenario se conoce como aprendizaje fuera del núcleo *(out-of-core learning)*. En esta modalidad, el algoritmo procesa los datos de manera fraccionada: carga un segmento de datos, lleva a cabo una etapa de entrenamiento con ese fragmento y luego continúa con el siguiente segmento, hasta completar el análisis del conjunto de datos completo.

A pesar de su denominación, el aprendizaje en línea no siempre se realiza en paralelo o simultáneamente con las operaciones del sistema donde se implementa. A menudo, este proceso se ejecuta de manera aislada o fuera de línea. Por ello, podría ser más preciso considerarlo como un método de aprendizaje incremental o secuencial (figura 1.23).

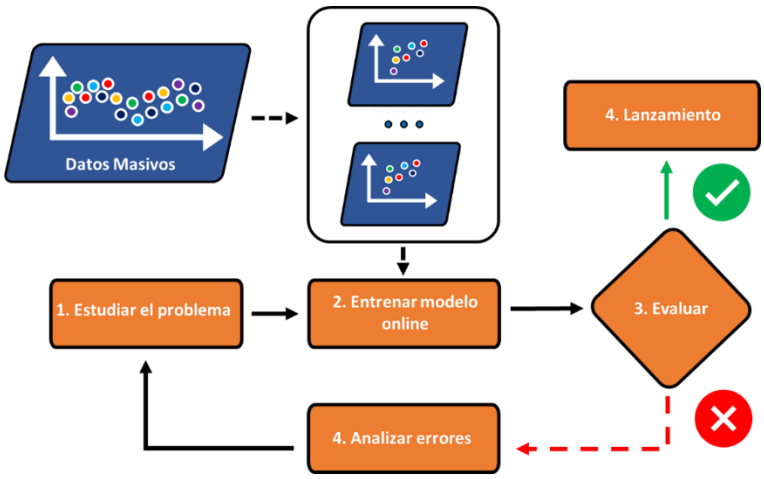


Figura 1.23. Manejo de errores utilizando Aprendizaje en línea.

Un parámetro importante en los sistemas de aprendizaje en línea es la velocidad de adaptación a los datos variables, conocida como la tasa de aprendizaje. Al configurar una tasa de aprendizaje elevada, el sistema se ajustará rápidamente a la información reciente, pero esto puede llevar a una pérdida acelerada de información previa. Por ejemplo, en un filtro de *spam*, una tasa alta podría resultar en que solo se identifique el spam más reciente, ignorando variantes anteriores.

En contraposición, una tasa de aprendizaje reducida hace que el sistema sea más estable y tenga más inercia. Esto significa que aprenderá de manera más gradual y será menos susceptible a las variaciones causadas por el ruido en los nuevos datos o a secuencias de datos atípicos *(outliers)*. Si bien esta configuración puede hacer que el sistema sea más resistente a cambios repentinos y erráticos en los datos, también puede retrasar la incorporación de información relevante y actualizada. La elección de la tasa de aprendizaje adecuada, por lo tanto, implica encontrar un equilibrio entre la adaptabilidad y la estabilidad del sistema en respuesta a los datos en evolución.

### 1.5.3. Metodología de generalización del aprendizaje.

Otra manera de clasificar los sistemas de aprendizaje automático es según su capacidad de generalización. En esencia, la mayoría de las tareas en este campo implican realizar predicciones. Esto significa que, partiendo de un conjunto de datos de entrenamiento, el sistema debería ser capaz de realizar predicciones acertadas para casos que no ha encontrado previamente. Si bien es importante que el sistema muestre un rendimiento sólido

en los datos de entrenamiento, esto no es suficiente por sí solo. El objetivo principal es lograr un alto rendimiento en instancias nuevas y desconocidas.

Para lograr esta generalización, existen dos enfoques predominantes: el aprendizaje basado en instancias y el aprendizaje basado en modelos. Cada uno de estos enfoques tiene una metodología distinta para abordar y aprender de los datos, adaptándose de manera diferente a las instancias nuevas con las que se encuentran.

**1.5.3.1. Aprendizaje basado en instancias.**

El aprendizaje basado en la memorización es una técnica elemental para entrenar sistemas, como un filtro de correo no deseado (*spam*), aunque puede resultar limitada en su efectividad. Por ejemplo, si un filtro de spam se basa exclusivamente en la memorización, solo identificará como spam aquellos correos que sean idénticos a los ya marcados previamente, lo cual restringe notablemente su capacidad de detección.

Una estrategia más sofisticada implica programar el filtro para que reconozca correos no solo idénticos, sino también aquellos que sean sustancialmente similares a los ya catalogados como spam. Esto se logra implementando una métrica de similitud entre los correos. Por ejemplo, un método simple para medir esta similitud podría ser contar el número de palabras compartidas entre los correos. De esta manera, si un correo electrónico tiene muchas palabras en común con uno identificado previamente como spam, entonces el sistema lo clasificará como tal. Este método se conoce como aprendizaje basado en instancias.

En el enfoque del aprendizaje basado en instancias, el sistema guarda ejemplos específicos y los utiliza para evaluar nuevas instancias, comparándolas con los ejemplos almacenados (o una selección de ellos) usando una métrica de similitud. Como se ilustra en la figura 1.24, una nueva instancia (correo electrónico) se catalogaría como "*spam*" (indicado por un corazón en el gráfico) si la mayoría de las instancias más similares pertenecen a esa categoría.

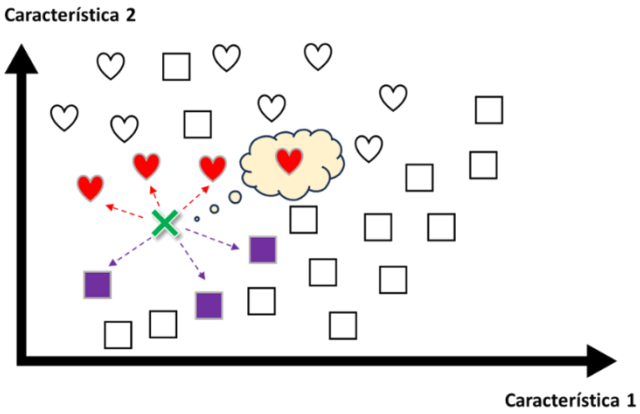


Figura 1.24. Aprendizaje basado en instancias.

**1.5.3.2. Aprendizaje basado en modelos.**

El aprendizaje basado en modelos representa un pilar esencial en el campo de la inteligencia artificial y el aprendizaje automático. A diferencia del aprendizaje basado en instancias, que responde a ejemplos individuales, este enfoque se concentra en desarrollar un modelo general que refleje las tendencias y patrones inherentes a un conjunto de datos. Este modelo se utiliza después para realizar predicciones o tomar decisiones con respecto a nuevos datos, como se muestra en la figura 1.25.

Tomando el ejemplo de un filtro de *spam*, en lugar de limitarse a memorizar casos específicos de correos no deseados, un enfoque basado en modelos se enfocaría en identificar características generales comunes en los correos spam. Esto puede incluir desde palabras clave específicas hasta patrones en el uso del lenguaje, o incluso detalles más sutiles como la frecuencia de ciertas palabras. Una vez construido y entrenado con ejemplos previos, el modelo está capacitado para examinar nuevos correos electrónicos y determinar con mayor precisión si son spam, basándose en las características aprendidas.

Este método presenta varias ventajas. Primero, es más flexible y adaptable, capaz de manejar variaciones y cambios en los datos. Segundo, proporciona una comprensión más profunda y explicaciones más claras de por qué ciertos elementos son clasificados de cierta manera, ya que el modelo ofrece una representación más abstracta y entendible de los datos.

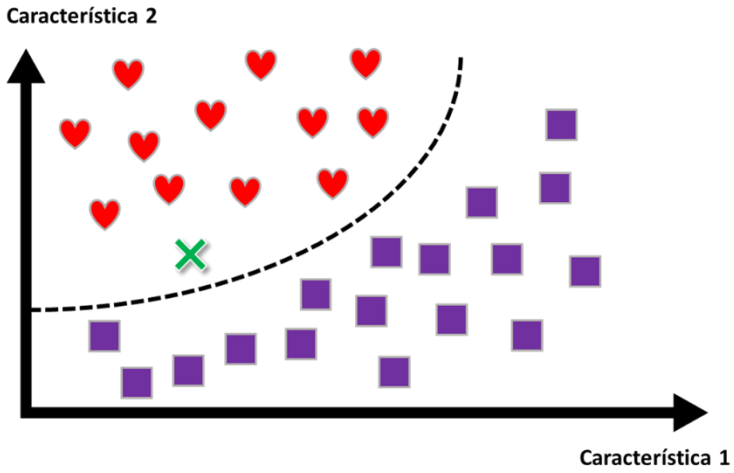


Figura 1.25. Aprendizaje basado en modelos.

## 1.6. Etapas en los proyectos de aprendizaje automático.

El desarrollo de proyectos en el campo del aprendizaje automático puede entenderse de manera simplificada a través de la identificación de cuatro etapas fundamentales. Este esquema proporciona un marco claro y estructurado para abordar los desafíos inherentes a la creación de sistemas eficientes de aprendizaje automático. Al desglosar el proceso en estas etapas clave, se facilita una mejor planificación, ejecución y evaluación de los proyectos, asegurando así que cada aspecto crítico del aprendizaje automático sea adecuadamente considerado y tratado. Cada etapa tiene su importancia y contribuye significativamente al éxito del proyecto en su conjunto.

### 1.6.1. Estudio de los Datos.

La etapa de exploración y análisis del conjunto de datos es una fase fundamental en cualquier proyecto de aprendizaje automático. Durante este proceso, se realiza un examen detallado de los datos para comprender profundamente sus características intrínsecas. Esto incluye analizar la distribución estadística de las variables, identificar posibles correlaciones o relaciones entre diferentes variables, y detectar la presencia de anomalías o valores atípicos que podrían influir en el rendimiento del modelo. Además, resulta de extrema importancia identificar y abordar adecuadamente los valores faltantes, ya que su manejo inadecuado puede llevar a conclusiones erróneas o a un modelo sesgado.

El análisis detallado en esta etapa no solo ayuda a formar una base sólida para el modelo a desarrollar, sino que también es fundamental para la selección de técnicas de preprocesamiento y transformación de datos adecuadas. Por ejemplo, la comprensión de la

distribución de los datos puede influir en la elección de métodos de normalización o estandarización. De igual manera, el reconocimiento de patrones o tendencias subyacentes en los datos puede guiar la ingeniería de características, que es un paso esencial para mejorar la capacidad predictiva del modelo.

### 1.6.2. Selección del Modelo.

La selección del modelo adecuado en un proyecto de aprendizaje automático es un paso crítico que se fundamenta en un análisis exhaustivo de los datos y una comprensión clara del problema específico que se necesita resolver. Esta decisión se basa en una serie de factores clave que incluyen, pero no se limitan a, la naturaleza del problema a abordar - ya sea clasificación, regresión, agrupamiento, entre otros -, la cantidad y el tipo de datos disponibles, y los requerimientos específicos de rendimiento y eficiencia del modelo.

Cada tipo de modelo de aprendizaje automático tiene sus fortalezas y limitaciones, y la elección debe alinearse con las características inherentes de los datos y los objetivos del proyecto. Por ejemplo, algunos modelos pueden ser más adecuados para conjuntos de datos de gran tamaño, mientras que otros pueden ser más eficientes en el manejo de datos con muchas características. Del mismo modo, ciertos modelos pueden ser preferibles debido a su capacidad para manejar complejidades específicas del problema, como datos no lineales o la necesidad de interpretabilidad del modelo.

La selección de un modelo no es una decisión que se deba tomar a la ligera, sino que requiere una consideración cuidadosa de varios aspectos técnicos y prácticos para garantizar que el modelo elegido sea el más adecuado para el conjunto de datos y el problema en cuestión. Una elección adecuada en esta etapa es fundamental para el éxito del proyecto de aprendizaje automático.

### 1.6.3. Entrenamiento del Modelo.

La fase de entrenamiento del modelo es una etapa fundamental donde se afinan y optimizan los parámetros del modelo elegido para alcanzar el mínimo en una función de costo predefinida. Esta optimización es necesaria para permitir que el modelo extraiga patrones y aprenda efectivamente de los datos proporcionados durante el entrenamiento. El proceso involucra suministrar al modelo un conjunto de datos y luego ajustar sus parámetros de manera iterativa, a través de varias rondas de aprendizaje, con el fin de incrementar su precisión y su habilidad para generalizar bien sobre datos no vistos.

En esta etapa, el modelo se somete a un proceso riguroso de aprendizaje donde se busca equilibrar la precisión en los datos de entrenamiento con la capacidad de generalizar a nuevas instancias. Este balance es fundamental para evitar problemas como el sobreajuste, donde el modelo aprende los detalles y el ruido en los datos de entrenamiento a expensas de perder la capacidad de realizar predicciones acertadas en datos no entrenados. El objetivo final de esta fase es desarrollar un modelo que no solo se desempeñe bien en los datos con los que fue entrenado, sino que también mantenga un alto nivel de rendimiento al enfrentarse a nuevos conjuntos de datos, reflejando así su robustez y versatilidad.

### 1.6.4. Implementación y Predicción.

Después de completar la etapa de entrenamiento, el modelo se pone en acción para llevar a cabo predicciones o inferencias sobre nuevos datos. Esta fase es la culminación del proceso de aprendizaje automático, donde el modelo se enfrenta al desafío de aplicar sus conocimientos adquiridos a situaciones y conjuntos de datos previamente desconocidos. La capacidad de realizar predicciones acertadas o tomar decisiones informadas en escenarios no vistos durante el entrenamiento es el indicador definitivo del éxito de un modelo de aprendizaje automático.

En la práctica, esta etapa implica la evaluación de la eficacia del modelo en el mundo real o en condiciones similares a las que enfrentará en su uso cotidiano. Aquí es donde se puede observar la verdadera capacidad del modelo para generalizar, una habilidad esencial que determina su utilidad práctica. La precisión, la fiabilidad y la relevancia de las predicciones o decisiones tomadas por el modelo son aspectos críticos que se examinan en esta fase.

El éxito de un proyecto de aprendizaje automático depende en gran medida de la eficacia con la que se aborden sus etapas iniciales. Las dos primeras fases, el estudio de los datos y la selección del modelo, son tal vez las más importantes ya que establecen una sólida fundación para el desarrollo posterior del proyecto. No debemos olvidar atender correctamente a las etapas de entrenamiento del modelo y su implementación para la inferencia o predicción ya que el entrenamiento debe ajustar con precisión los parámetros del modelo para optimizar su rendimiento, y la fase de inferencia debe demostrar la capacidad del modelo para aplicar lo aprendido a nuevos datos de manera efectiva y fiable.

## 1.7. Principales desafíos del aprendizaje automático.

Los principales desafíos en el campo del aprendizaje automático se centran en dos áreas fundamentales: la selección adecuada del modelo y el manejo eficiente de los datos para el entrenamiento. Los problemas potenciales se dividen en dos categorías clave: la primera se relaciona con la utilización de datos de baja calidad o inadecuados, mientras que la segunda se puede asociar a la elección de un modelo inapropiado.

### 1.7.1. Problemas relacionados con la calidad y la relevancia de los datos.

La calidad y relevancia de los datos utilizados para el entrenamiento son fundamentales. Los datos que son incorrectos, incompletos, sesgados o que no representan adecuadamente el problema a resolver pueden conducir a un modelo mal entrenado. Esto incluye problemas como datos ruidosos, valores atípicos no gestionados, o un conjunto de datos que no refleja la variabilidad real del entorno en el que se implementará el modelo.

#### 1.7.1.1. Cantidad insuficiente de datos de entrenamiento.

Un desafío significativo en el aprendizaje automático es la cantidad insuficiente de datos de entrenamiento. Para que los algoritmos funcionen de manera óptima y desarrollen patrones precisos y generalizables, a menudo requieren una gran cantidad de ejemplos. Dependiendo de la complejidad del problema, esto puede variar desde miles hasta millones de instancias.

La escasez de datos puede limitar la capacidad del modelo para aprender eficazmente, llevando a un aprendizaje incompleto o a una generalización pobre. Esto es particularmente crítico en modelos más complejos que necesitan una gran cantidad de datos para capturar y aprender las sutilezas y variaciones presentes en los datos reales. La falta de datos suficientes puede hacer que el modelo sea incapaz de ajustarse adecuadamente a la diversidad de casos que podría encontrar en un entorno real, lo que resulta en un rendimiento poco eficaz cuando se enfrenta a nuevos datos o situaciones.

#### 1.7.1.2. Datos de entrenamiento no representativos.

Para que los modelos de aprendizaje automático funcionen adecuadamente en aplicaciones reales, es fundamental que los conjuntos de datos de entrenamiento reflejen de manera adecuada la variedad de situaciones que el modelo podría encontrar.

Incluso con grandes volúmenes de datos, si estos presentan sesgos debido a una selección inadecuada o a una cobertura insuficiente de todas las variables relevantes, los modelos podrán ser poco eficaces para situaciones no vistas durante el entrenamiento lo que resulta en predicciones o decisiones inexactas.

**1.7.1.3. Datos de baja calidad.**

La calidad de los datos de entrenamiento es un factor determinante para el éxito del modelo. Los datos que contienen errores, valores atípicos o ruido, como aquellos provenientes de mediciones deficientes, pueden dificultar significativamente la capacidad del sistema para identificar patrones subyacentes.

El proceso de limpieza y preparación de datos es una etapa sumamente importante en el aprendizaje automático, ya que establece la base para el entrenamiento de un modelo preciso y eficiente. Una inversión de tiempo adecuada en esta fase puede significar la diferencia entre un modelo que funciona bien y uno que no logra cumplir con los objetivos deseados.

La limpieza de datos puede incluir una variedad de técnicas, dependiendo de los problemas específicos presentes en el conjunto de datos. Las más destacables son el tratamiento de valores atípicos y el manejo de valores faltantes.

**Tratamiento de Valores Atípicos.** En situaciones donde los datos presentan valores claramente atípicos o anómalos, puede ser adecuado eliminar estos puntos o corregir los errores de forma manual para mejorar la precisión del modelo.

**Manejo de Valores Faltantes.** Cuando faltan ciertas características en los datos (por ejemplo, si un porcentaje de clientes no proporcionó su edad), se debe tomar una decisión sobre cómo abordar estos vacíos. Las opciones incluyen descartar la característica faltante, excluir los casos con datos faltantes, imputar los valores (por ejemplo, utilizando la mediana de la edad) o entrenar modelos con y sin la característica para comparar su rendimiento.

**1.7.1.4. Características irrelevantes.**

En un proyecto de aprendizaje automático, el desarrollo de un conjunto de características robusto y relevante es fundamental para su correcto desempeño. Este proceso, denominado "ingeniería de características", implica una serie de pasos estratégicos diseñados para optimizar la calidad y relevancia de los datos que alimentarán al modelo. Estos pasos incluyen:

**Selección de Características.** Este paso implica identificar y elegir aquellas características que aportan más valor y relevancia al modelo. El objetivo es filtrar aquellas características que son redundantes o que no contribuyen significativamente al rendimiento predictivo del modelo. La selección efectiva de características puede mejorar la eficiencia del modelo y, a menudo, su precisión.

**Extracción de Características.** En este paso, se combinan o transforman características existentes para formar nuevas que sean más significativas para el modelo. Los métodos de reducción de dimensionalidad, como el Análisis de Componentes Principales, son herramientas valiosas en este proceso. Estas técnicas pueden revelar combinaciones de datos que son más informativas y menos ruidosas que las características originales.

**Creación de Nuevas Características.** Este aspecto se centra en ampliar y enriquecer el conjunto de datos con nueva información. A veces, las características más relevantes para un problema no están presentes en los datos originales y deben ser desarrolladas o recopiladas. La creación de nuevas características puede involucrar el análisis de tendencias, la combinación de diferentes fuentes de datos, o incluso el uso de algoritmos para generar características sintéticas.

Cada uno de estos pasos en la ingeniería de características es fundamental para construir un modelo de aprendizaje automático eficiente y preciso. Al asegurarse de que el modelo se alimente con datos de alta calidad y relevantes, se aumenta significativamente la probabilidad de alcanzar una eficiente proyecto.

### 1.7.2. Problemas relacionados con modelos inapropiados.

La elección de un modelo inapropiado puede derivar de una falta de alineación con las características específicas del conjunto de datos o los objetivos del proyecto. Esto puede llevar a un rendimiento subóptimo, donde el modelo no puede capturar adecuadamente las complejidades o patrones de los datos, resultando en predicciones inexactas o generalizaciones deficientes.

#### 1.7.2.1. Sobreajuste de los datos de entrenamiento.

El problema del sobreajuste del modelo se da, cuando éste ajusta muy bien a los datos de entrenamiento, pero no es capaz de generalizar correctamente.

Por ejemplo, si entrenamos un modelo para predecir el rendimiento escolar basándonos en una variedad de factores, incluyendo detalles tan específicos como el color del lápiz favorito de un estudiante, el modelo podría encontrar "patrones" que realmente no son más que coincidencias en el conjunto de entrenamiento. Esto podría llevar a conclusiones erróneas, como que usar un lápiz de color azul está relacionado con mejores calificaciones, lo cual es improbable que se mantenga en un grupo más amplio de estudiantes. (figura 1.26).

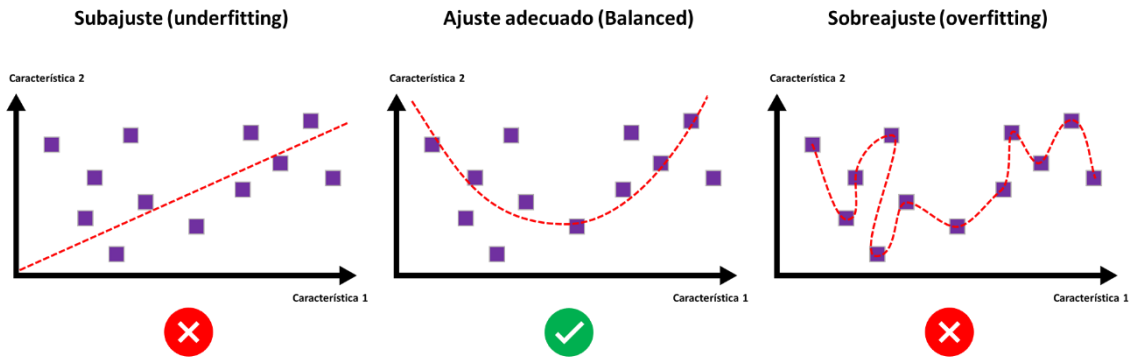


Figura 1.26. Comparativa ajustes de modelos.

**1.7.2.2. Subajuste en los Datos de Entrenamiento.**

El subajuste se produce cuando un modelo es demasiado simple para comprender la estructura subyacente de los datos, lo que resulta en predicciones inexactas. Un ejemplo claro sería un modelo lineal utilizado para predecir el rendimiento académico basándose únicamente en la asistencia a clases, ignorando factores como el tiempo de estudio o el ambiente de aprendizaje. Este modelo probablemente fallará al predecir con precisión debido a su simplicidad.

Existen estrategias que permiten abordar el problema del subajuste en el modelo, a continuación, detallaremos las más importantes.

**Utilizar un Modelo Más Complejo.** Elegir un modelo con más parámetros puede ayudar a capturar la complejidad y las variaciones de los datos de manera más efectiva. Por ejemplo, un modelo que considere múltiples factores como la asistencia, el tiempo de estudio y la participación en clase podría predecir mejor el rendimiento académico.

**Mejorar las Características para el Aprendizaje.** La ingeniería de características puede ser crucial. Esto podría incluir agregar nuevas características relevantes, como el promedio de calificaciones pasadas, o transformar características existentes para reflejar mejor las relaciones en los datos.

**Reducir las Restricciones en el Modelo.** Disminuir la regularización permite que el modelo sea más flexible. Por ejemplo, ajustar el hiperparámetro de regularización para permitir que el modelo se ajuste más libremente a los datos puede ayudar a capturar su estructura de manera más eficaz.

Estas estrategias buscan equilibrar la complejidad del modelo con su capacidad para representar fielmente los datos, mejorando así la precisión y la efectividad de las predicciones del modelo.

**1.7.2.3. Regularización utilizando hiperparámetros.**

Para hacer un modelo más simple y reducir el riesgo de sobreajuste, aplicamos un proceso llamado regularización. Por ejemplo, en un modelo lineal tenga dos parámetros, $\theta_1$ y $\theta_0$. (esto le da al algoritmo de aprendizaje dos grados de libertad para adaptar el modelo a los datos de entrenamiento) podemos modificar tanto la altura ($\theta_0$) como la pendiente ($\theta_1$) de la línea. Si forzamos $\theta_1 = 0$, el algoritmo tendría solo un grado de libertad y le resultaría mucho más difícil ajustar los datos correctamente: todo lo que podría hacer éste es mover la línea hacia arriba o hacia abajo para acercarse lo más posible a las instancias de entrenamiento, terminando alrededor de la media.

Si permitimos que el algoritmo modifique $\theta_1$ pero lo obligamos a mantenerlo pequeño, entonces el algoritmo de aprendizaje tendrá efectivamente entre uno y dos grados de libertad. Producirá un modelo que es más simple que uno con dos grados de libertad, pero más complejo que uno con un solo grado. El objetivo es encontrar el equilibrio adecuado entre ajustar perfectamente los datos de entrenamiento y mantener el modelo lo suficientemente simple para garantizar que se generalice bien.

La cantidad de regularización que se aplicará durante el aprendizaje se puede controlar mediante un hiperparámetro. Un hiperparámetro es un parámetro de un algoritmo de aprendizaje (no del modelo). Como tal, no se ve afectado por el propio algoritmo de aprendizaje; debe establecerse antes del entrenamiento y permanece constante durante el entrenamiento. En general, ajustar hiperparámetros requiere un equilibrio cuidadoso y una comprensión de cómo afectan el comportamiento del modelo aplicado a nuestro conjunto de datos.

#### 1.7.2.4. Ajuste de los hiperparámetros.

El ajuste de los hiperparámetros es una parte crucial del proceso de modelado en aprendizaje automático, ya que estos parámetros pueden tener un impacto significativo en el rendimiento del modelo. La relación entre los valores de los hiperparámetros y cómo estos afectan al sobreajuste o subajuste varía dependiendo del tipo de modelo y del hiperparámetro específico.

##### 1.7.2.4.1. Regularización en SVM.

En las Máquinas de Vectores de Soporte, el hiperparámetro $C$ juega un papel fundamental en la regularización del modelo. Un valor alto de $C$ resulta en menos regularización. Esto significa que el modelo se enfoca en clasificar correctamente la mayor cantidad posible de ejemplos de entrenamiento, aunque esto pueda llevar a un límite de decisión más complicado y menos generalizable, aumentando el riesgo de sobreajuste, ya que el modelo puede empezar a aprender el ruido y las peculiaridades específicas del conjunto de entrenamiento, en lugar de capturar patrones generales que serían aplicables a nuevos datos.

##### 1.7.2.4.2. Regularización en Modelos Lineales.

En el caso de modelos lineales como la regresión lineal o logística, los hiperparámetros $\lambda_1$y $\lambda_2$ tienen un comportamiento inverso al de $C$ en *SVM*. Aquí, un valor más alto del hiperparámetro implica una mayor regularización, lo que puede resultar en un subajuste si el modelo se vuelve demasiado restringido y no puede capturar adecuadamente las relaciones subyacentes en los datos. Las regularizaciones más utilizadas para los modelos lineales son.

**Regularización L1** ***(Lasso).*** En el contexto de la modelización predictiva, como en la regresión para estimar el precio de una vivienda, a menudo nos encontramos con un conjunto amplio de variables predictoras o características. Sin embargo, no todas estas características son igualmente informativas o relevantes para la predicción. La Regularización $L1$, conocida como *Lasso (Least Absolute Shrinkage and Selection Operator)* "Operador de Selección y Encogimiento Absoluto Mínimo"*,* agrega una penalización equivalente al valor absoluto de la magnitud de los coeficientes al modelo de regresión.

$$L1_{MSE} = MSE + \lambda_1 \sum_{j=1}^{n} |\beta_j|$$

Donde:

- $L1_{MSE}$ es el Error Cuadrático medio regularizado por $L1$,
- $MSE$ es el Error Cuadrático medio sin regularizar,
- $\lambda_1$ es la hiperparámetro de regularización de $L1$,
- $\beta_j$ son los coeficientes del modelo.

Este enfoque tiene una consecuencia notable impulsa a ciertos coeficientes a ser exactamente cero (esto se conoce como esparcidad). En términos prácticos, esto resulta en un modelo que ignora algunas de las características menos significativas por lo que, además de simplificar el modelo, ayuda a mejorar su interpretabilidad y rendimiento al centrarse en las variables más relevantes.

**Regularización L2 (Ridge).** Por otro lado, la Regularización $L2$, también conocida como *Ridge*, adopta un enfoque ligeramente diferente. En lugar de penalizar la suma de los valores absolutos de los coeficientes, penaliza la suma de sus cuadrados.

$$L2_{MSE} = MSE + \lambda_2 \sum_{j=1}^{n} \beta_j^2$$

Donde:

- $L2_{MSE}$ es el Error Cuadrático medio regularizado por L2,
- $MSE$ es el Error Cuadrático medio sin regularizar,
- $\lambda_2$ es la hiperparámetro de regularización de L21,
- $\beta_j$ son los coeficientes del modelo.

Esta sutil diferencia conduce a un efecto distinto: en lugar de impulsar coeficientes a cero, reduce su magnitud general. Esto es particularmente valioso en situaciones donde hay una alta correlación entre las características (multicolinealidad), ya que previene que el modelo se vuelva excesivamente sensible a una sola variable. La Regularización $L2$ mejora la estabilidad numérica del modelo y reduce el riesgo de sobreajuste, asegurando que el modelo no solo funcione bien en los datos de entrenamiento, sino que también mantenga un buen rendimiento en datos nuevos.

***ElasticNet.***

*ElasticNet* es una combinación de las regularizaciones $L1$ y $L2$ que busca lograr un equilibrio entre la selección de variables de *Lasso* y la reducción de varianza de *Ridge*. Es especialmente útil cuando hay múltiples características correlacionadas entre sí, ya que Lasso podría seleccionar de manera arbitraria una de ellas, mientras que *Ridge* las incluiría todas.

$$EN_{MSE} = MSE + \lambda_1 \sum_{j=1}^{n} |\beta_j| + \lambda_2 \sum_{j=1}^{n} \beta_j^2$$

Donde:

- $L2_{MSE}$ es el Error Cuadrático medio regularizado por L2,
- $MSE$ es el Error Cuadrático medio sin regularizar,
- $\lambda_1, \lambda_2$ es la hiperparámetro de regularización de L1 y L2 respectivamente,
- $\beta_j$ son los coeficientes del modelo.

*ElasticNet* puede capturar lo mejor de ambos mundos, manteniendo un grupo de características correlacionadas juntas y reduciendo la magnitud de sus coeficientes de manera uniforme.

**1.7.2.4.3. Regularización en arboles de decisión y bosques aleatorios.**

La regularización en estos modelos basados en árboles se realiza mediante la configuración de varios hiperparámetros que controlan la complejidad del modelo. Algunos de los hiperparámetros más importantes se mencionan a continuación.

**Profundidad Máxima del Árbol *(max_depth)*.** Controla la profundidad máxima que puede alcanzar el árbol. Un árbol más profundo puede capturar más detalles, pero también es más propenso al sobreajuste.

***Número Mínimo de Muestras para Dividir un Nodo (min_samples_split).*** Especifica el número mínimo de muestras que debe tener un nodo para que se considere para una división. Un valor más alto reduce el riesgo de sobreajuste.

**Número Mínimo de Muestras en un Nodo Hoja *(min_samples_leaf).*** Establece el número mínimo de muestras que debe tener un nodo hoja. Aumentar este número puede suavizar el modelo y evitar sobreajustes.

**Número Máximo de Características Consideradas para Dividir *(max_features).*** Determina el número máximo de características consideradas para las divisiones. Limitar este número puede prevenir el sobreajuste, especialmente en bosques aleatorios.

***Número Máximo de Hojas del Árbol (max_leaf_nodes).*** Limita el número de nodos hoja en el árbol. Un control efectivo de este hiperparámetro ayuda a encontrar un buen equilibrio entre aprender suficientemente de los datos de entrenamiento y mantener la capacidad del modelo para generalizar a datos no vistos.

Estos hiperparámetros actúan como controles para regular la complejidad de los árboles de decisión y los bosques aleatorios. Ajustar estos hiperparámetros adecuadamente resulta fundamental para equilibrar la capacidad del modelo de capturar patrones complejos en los datos y su habilidad para generalizar bien a nuevos datos no vistos. El proceso de encontrar el conjunto óptimo de valores de hiperparámetros suele realizarse mediante técnicas como la validación cruzada.

#### 1.7.2.4.4. Generalización a otros modelos.

No todos los hiperparámetros tienen un impacto directo en el equilibrio entre sobreajuste y subajuste. Algunos influyen en otros aspectos del modelo, como la convergencia (por ejemplo, la tasa de aprendizaje en métodos de gradiente descendente) o la estructura del modelo (como el número de capas en una red neuronal). Estos hiperparámetros pueden afectar la eficiencia del entrenamiento, la estabilidad del modelo y su capacidad para aprender patrones complejos.

Dependiendo del modelo y del contexto del problema, diferentes hiperparámetros tendrán diferentes efectos en el rendimiento del modelo. Una comprensión clara de cómo cada hiperparámetro influye en el modelo es fundamental para lograr un equilibrio óptimo entre sobreajuste y subajuste, y para construir modelos que sean tanto precisos como generalizables.

### 1.8. Pruebas y Validaciones

La efectividad de un modelo de aprendizaje automático en situaciones reales solo puede determinarse probándolo con datos nuevos. Aunque es posible implementar el modelo directamente en producción y observar su rendimiento, esto puede ser arriesgado, especialmente si el modelo no funciona adecuadamente, lo cual podría generar insatisfacción en los usuarios.

Un enfoque más seguro y efectivo es dividir los datos en dos conjuntos distintos: un conjunto de entrenamiento y un conjunto de prueba. El modelo se entrena utilizando el conjunto de entrenamiento y luego se evalúa con el conjunto de prueba. El error cometido por el modelo en este último conjunto se conoce como error de generalización (o error fuera de muestra), proporcionando una medida de cómo el modelo se desempeñará en casos que no ha visto durante el entrenamiento.

Si se observa que el modelo tiene un bajo error de entrenamiento, pero un alto error de generalización, esto indica que el modelo está sobre ajustando los datos de entrenamiento. Esto significa que, aunque el modelo funciona bien con los datos de entrenamiento, no logra generalizar efectivamente a nuevos datos, lo cual es un indicador clave para realizar ajustes y mejoras en el modelo. La evaluación precisa mediante el conjunto de prueba es, por tanto, un paso crítico para garantizar que el modelo sea robusto y fiable en su aplicación práctica.

### 1.8.1. Segmentación habitual de Datos para Entrenamiento y Pruebas

En la práctica común del aprendizaje automático, es una norma general dividir los datos disponibles en dos segmentos: un conjunto de entrenamiento y un conjunto de pruebas *(Train-Test Split)*. Frecuentemente, se utiliza aproximadamente el 80% de los datos para el entrenamiento y el 20% restante para las pruebas. Esta proporción, sin embargo, puede ajustarse según el tamaño y las características específicas del conjunto de datos.

Por ejemplo, en casos donde el conjunto de datos es excepcionalmente grande, como aquellos que contienen millones de instancias, incluso una fracción más pequeña reservada para pruebas puede ser adecuada. Si se tiene un conjunto de 10 millones de instancias, asignar solo el 1% para pruebas resultaría en un conjunto de prueba de 100,000 instancias. Esta cantidad suele ser más que suficiente para evaluar con precisión el error de generalización del modelo.

Este enfoque permite un uso eficiente de los datos, asegurando que el modelo se entrene con un conjunto de datos amplio, al mismo tiempo que se reserva un conjunto de prueba de tamaño adecuado para una evaluación fiable del rendimiento del modelo.

### 1.8.2. Método de validación de reserva *(Holdout Validation).*

En este método, además de los conjuntos de entrenamiento y prueba, se separa un tercer conjunto de datos para la validación (figura 1.27). Este conjunto se utiliza para

realizar ajustes en los parámetros del modelo, sintonización de hiperparámetros, o para evitar el sobreajuste. Mientras que el conjunto de prueba se reserva para la evaluación final y objetiva del modelo, el conjunto de validación se utiliza para la optimización y mejora continua del modelo durante el proceso de entrenamiento.

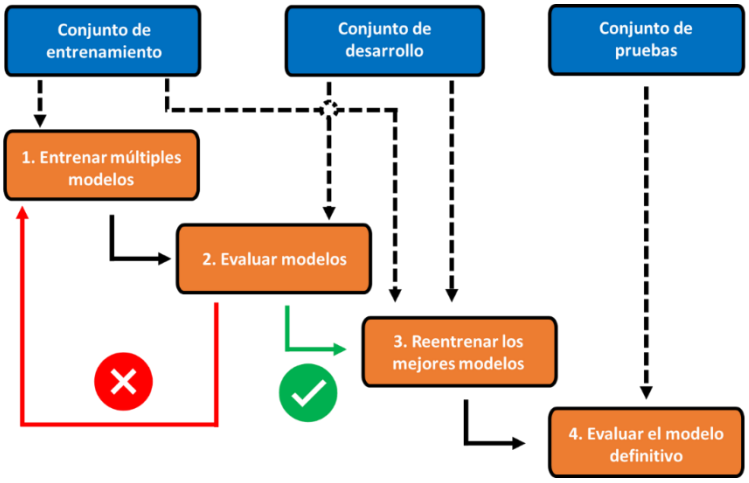


Figura 1.27. Selección de modelos utilizando validación de retención.

Los métodos anteriores son simples y fáciles de implementar y requieren menor tiempo computacional que el método de validación cruzada que explicaremos a continuación. En contraposición, estos métodos pueden llevar a una mayor variabilidad del rendimiento del modelo, sobre todo en casos donde el conjunto de datos es pequeño ya que una parte de los datos no se utiliza para el entrenamiento.

### 1.8.3. Validación Cruzada *(Cross-Validation).*

El método de Validación Cruzada *(Cross-Validation)* consiste en dividir el conjunto de datos completo en 'k' subconjuntos (o *'folds'*). Luego, el modelo se entrena con 'k-1' de estos subconjuntos y se prueba con el subconjunto restante. Este proceso se repite 'k' veces, con cada subconjunto sirviendo una vez como conjunto de prueba. Finalmente, el rendimiento del modelo se evalúa tomando el promedio de los resultados obtenidos en cada iteración.

#### 1.8.3.1. Método K-Folds.

En este método se divide el conjunto de datos en 'k' subconjuntos de manera aleatoria. En cada iteración de la validación cruzada, uno de estos subconjunto se utiliza como conjunto de prueba y el resto como conjunto de entrenamiento. Este proceso se repite 'k' veces, asegurando que cada subconjunto se utilice una vez como conjunto de prueba.

### 1.8.4. Stratified K-Folds.

*K-Folds* no tiene en cuenta la distribución de clases dentro de los subconjuntos, lo que puede ser problemático para conjuntos de datos desequilibrados. En *Stratified K-Folds*, los datos se dividen de tal manera que cada subconjunto contenga aproximadamente la misma proporción de cada clase objetivo como en el conjunto de datos completo.

Esto significa que si, por ejemplo, tenemos un conjunto de datos con dos clases y una de ellas representa el 20% de los datos y la otra el 80%, cada subconjunto en *Stratified K-Folds* tendrá aproximadamente la misma proporción de 20/80 de estas dos clases.

Mientras que el *Train-Test Split* y el *Holdout Validation* son más rápidos y sencillos, pueden no ser suficientes para conjuntos de datos complicados o críticos. La *Cross Validation*, aunque más exigente en términos de recursos, ofrece una evaluación más detallada y confiable, crucial para el desarrollo de modelos de aprendizaje automático robustos y eficaces. La elección entre estos métodos depende de las características específicas del conjunto de datos y los requisitos del problema en cuestión.

### 1.8.5. Desajuste de Datos

Frecuentemente nos encontramos con la situación de tener acceso a una gran cantidad de datos para entrenar nuestros modelos, pero que estos datos no sean completamente representativos de los que usen en un entorno de producción. Consideremos, por ejemplo, el desarrollo de una aplicación móvil para identificar tipos de aves. Podemos descargar fácilmente millones de imágenes de aves de Internet, pero estas no reflejarían con precisión las fotos que los usuarios tomarán con la aplicación en sus dispositivos móviles. Imaginemos que solo contamos con 1.000 fotos realmente tomadas con la aplicación.

Como vimos anteriormente es muy importante que nuestros conjuntos de validación y prueba sean lo más representativos posible de los datos esperados en producción. Por lo tanto, deberíamos componerlos exclusivamente con imágenes tomadas con la aplicación. Podríamos dividir estas fotos de manera equitativa entre el conjunto de validación y el de prueba, cuidando de no tener duplicados en ambos. El problema es que, si después de entrenar nuestro modelo con las imágenes de Internet observamos un rendimiento deficiente en el conjunto de validación, no podremos determinar si esto se debe a un sobreajuste del modelo a los datos de entrenamiento o a un desajuste entre las imágenes de Internet y las de la aplicación móvil.

Para evitar este dilema, podemos reservar algunas de las imágenes de Internet en otro conjunto, denominado conjunto de entrenamiento-desarrollo (*training-dev set*, figura 1.28). Después de entrenar el modelo con el conjunto de entrenamiento, lo evaluamos en el *training-dev.* Si su rendimiento es bajo, probablemente estemos sobreajustando al conjunto de entrenamiento. En tal caso, debemos intentar simplificar o regularizar el modelo, conseguir más datos de entrenamiento o limpiar los existentes. Pero si el modelo funciona bien en el *training-dev* y mal en el conjunto de desarrollo (*dev set*), el problema probablemente se debe al desajuste de datos. Para solucionarlo, podríamos procesar las imágenes de Internet para que se asemejen más a las que se tomarán con la aplicación y luego reentrenar el modelo. Finalmente, una vez que el modelo rinde bien en ambos conjuntos (*training-dev* y *dev*), lo evaluamos en el conjunto de prueba para estimar su rendimiento probable en la producción.


Figura 1.28. Entrenamiento con datos escasos.

En otras palabras, cuando los datos reales son escasos (derecha), se pueden utilizar datos abundantes similares (izquierda) para el entrenamiento y conservar algunos de ellos en un conjunto de desarrollo-entrenamiento para evaluar el sobreajuste. Posteriormente, los datos reales se utilizan para evaluar la discrepancia de datos (conjunto de desarrollo) y para evaluar el rendimiento del modelo final (conjunto de prueba).

## 1.9. Medidas de Rendimiento para algoritmos de aprendizaje automático supervisado.

En el campo del aprendizaje automático supervisado, la evaluación precisa del rendimiento de los algoritmos es fundamental para entender su efectividad y aplicabilidad en diversas situaciones prácticas. Las "Medidas de Rendimiento para Algoritmos de Aprendizaje Automático Supervisado" abarcan una serie de métricas y técnicas estadísticas diseñadas para cuantificar la precisión, fiabilidad y utilidad de los modelos predictivos, tanto en clasificación como en regresión.

### 1.9.1. Medidas para algoritmos de clasificación.

En este punto estudiaremos las principales medidas de rendimiento para evaluar y comparar algoritmos de clasificación en aprendizaje automático. Estas métricas ayudan a entender cuán efectivo es un modelo al realizar predicciones. Las más comunes incluyen:

#### 1.9.1.1. Exactitud *(Accuracy)*.

La exactitud es una métrica que indica cuál es la proporción de predicciones correctas frente al total de predicciones. La Exactitud proporciona una medida del rendimiento general del modelo sin centrarse en una clase en particular.

$$Exactitud = \frac{Verdaderos\ Positivos\ +\ Verdaderos\ Negativos}{Total\ de\ Predicciones}$$

Esta medida es útil cuando las clases están equilibradas, es decir, cuando el número de instancias en cada clase es más o menos el mismo. Ofrece una visión rápida y sencilla de cómo se desempeña el modelo en general. Sin embargo, la Exactitud puede ser engañosa en situaciones donde hay un desequilibrio significativo entre las clases (por ejemplo, en un conjunto de datos donde el 95% de las instancias son de una clase y solo el 5% de otra). En tales casos, un modelo podría predecir simplemente la clase mayoritaria para la mayoría de las instancias y aun así lograr una alta Exactitud, a pesar de no ser eficaz en identificar correctamente las instancias de la clase minoritaria.

#### 1.9.1.2. Precisión *(Precision).*

La Precisión mide la proporción de predicciones positivas correctas respecto al total de predicciones positivas. Está enfocada en minimizar los Falsos Positivos.

$$Precisión\ = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos\ +\ Falsos\ Positivos}$$

Una alta precisión del modelo es fundamental en situaciones donde los Falsos Positivos son costosos o problemáticos (por ejemplo, en filtros de spam de correo electrónico, donde clasificar un correo legítimo como spam es indeseable).

##### 1.9.1.2.1. Precisión del Conjunto de Entrenamiento.

Esta métrica se calcula usando el mismo conjunto de datos que se utilizó para entrenar el modelo. En otras palabras, mide qué tan bien el modelo puede predecir los resultados en los datos que ya ha visto durante el entrenamiento.

Una alta precisión en el conjunto de entrenamiento indica que el modelo ha aprendido bien los patrones de estos datos específicos. Sin embargo, una precisión extremadamente alta en el conjunto de entrenamiento podría ser una señal de sobreajuste, especialmente si esta precisión es mucho más alta que la precisión obtenida en el Conjunto de Pruebas.

**1.9.1.2.2. Precisión del Conjunto de Prueba.**

Por otro lado, la precisión del conjunto de datos, a menudo referida como precisión en el conjunto de prueba o conjunto de validación, se calcula utilizando datos que el modelo no ha visto durante el entrenamiento. Esto puede ser un conjunto de prueba separado o un conjunto de validación.

Esta métrica es definitoria porque proporciona una evaluación más realista de cómo el modelo se desempeñará en el mundo real, en datos no vistos anteriormente. Una alta precisión en el conjunto de prueba o validación indica que el modelo no solo ha aprendido los patrones de los datos de entrenamiento, sino que también puede generalizar bien a nuevos datos. Esto es esencial para un modelo robusto y práctico en aplicaciones del mundo real.

Ambas métricas son importantes para evaluar un modelo de aprendizaje automático. Mientras que la precisión del conjunto de entrenamiento nos dice cuánto ha aprendido el modelo de los datos de entrenamiento, la precisión del conjunto de prueba nos informa sobre la capacidad del modelo para generalizar a nuevos datos. Idealmente, debemos buscar un equilibrio: una alta precisión en ambos conjuntos indica un modelo que ha aprendido bien y puede generalizar bien.

**1.9.1.3. Sensibilidad (Tasa de Verdaderos Positivos).**

La Sensibilidad *(Recall)* Indica qué proporción de casos positivos reales fueron identificados correctamente. Está enfocado en minimizar los Falsos Negativos.

$$Sensibilidad = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$

Una alta sensibilidad de modelo resulta particularmente importante en situaciones donde es crítico capturar todos los casos positivos (por ejemplo, en diagnósticos médicos, donde es vital no pasar por alto un caso positivo de una enfermedad).

**1.9.1.3.1. Sensibilidad del Conjunto de Entrenamiento.**

La sensibilidad conjunto de entrenamiento se calcula utilizando los datos que se usaron para entrenar el modelo. Mide la proporción de casos positivos reales que fueron correctamente identificados por el modelo como positivos, dentro del conjunto de entrenamiento. Una alta sensibilidad en el conjunto de entrenamiento indica que el modelo es eficaz en la identificación de casos positivos de los datos con los que fue entrenado.

Sin embargo, una sensibilidad muy alta en el conjunto de entrenamiento podría ser indicativa de sobreajuste, especialmente si esta sensibilidad es significativamente más alta que la obtenida en el conjunto de pruebas. El sobreajuste implica que el modelo ha aprendido los datos de entrenamiento a tal grado que puede haber capturado ruido o patrones específicos de ese conjunto de datos, lo que podría afectar negativamente su capacidad de generalizar.

**1.9.1.3.2. Sensibilidad del Conjunto de Pruebas.**

La sensibilidad en el conjunto de pruebas se calcula utilizando datos que no fueron vistos por el modelo durante el entrenamiento. Esta métrica evalúa la capacidad del modelo para identificar correctamente los casos positivos en un conjunto de datos nuevo y no utilizado durante el entrenamiento. Una alta sensibilidad en el conjunto de pruebas es un indicador de que el modelo no solo aprendió los patrones de los datos de entrenamiento, sino que también puede generalizar bien y detectar efectivamente los casos positivos en nuevos datos.

La sensibilidad en el conjunto de pruebas es crucial para evaluar cómo se desempeñará el modelo en situaciones reales, donde los datos pueden diferir de los datos de entrenamiento.

Mientras que la sensibilidad del conjunto de entrenamiento nos indica cómo el modelo reconoce los casos positivos de los datos con los que fue entrenado, la sensibilidad del conjunto de pruebas nos informa sobre la capacidad del modelo para identificar casos positivos en datos nuevos y no vistos. Para un modelo de clasificación robusto y confiable, es importante tener un alto rendimiento tanto en el conjunto de entrenamiento como en el de pruebas.

**1.9.1.4. Especificidad (Tasa de Verdaderos Negativos).**

La Especificidad indica qué proporción de casos negativos reales fueron identificados correctamente. Está enfocada en minimizar los Falsos Positivos.

$$Especificidad = \frac{Verdaderos\ Negativos}{Verdaderos\ Negativos + Falsos\ Positivos}$$

Esta métrica es especialmente importante en situaciones donde los falsos positivos tienen consecuencias graves. Por ejemplo, en pruebas médicas, una baja especificidad (es decir, una tasa alta de falsos positivos) podría llevar a pacientes sanos a recibir tratamientos innecesarios o a experimentar ansiedad adicional.

Junto con la sensibilidad (o tasa de verdaderos positivos), la especificidad ayuda a proporcionar una imagen completa de la capacidad del modelo para distinguir ambas clases.

**1.9.1.5. Puntuación F1 *(F1 Score).***

La Puntuación F1 Combina tanto la Precisión como la Sensibilidad en un solo número, proporcionando un indicador holístico del rendimiento del modelo. Se calcula como la media armónica de ambas medidas, proporcionando un balance entre estas dos métricas. Está enfocada en ofrecer un equilibrio entre la exactitud de las predicciones positivas y detección de casos positivos reales.

$$F1 = 2 \times \frac{Precision \times Sensibilidad}{Precision + Sensibilidad}$$

Esta medida es especialmente útil en contextos donde tanto los falsos positivos como los falsos negativos son perjudiciales. Por ejemplo, en un modelo de clasificación médica, no es deseable diagnosticar erróneamente una enfermedad (falsos positivos) ni pasar por alto un diagnóstico correcto (falsos negativos). Un modelo con alta precisión, pero baja sensibilidad, o viceversa, tendrá una puntuación F1 más bajo que un modelo que equilibra ambas.

La puntuación F1 es muy importante en el contexto de existencia de clases desbalanceadas. Por ejemplo, si en un conjunto de datos la mayoría de los casos son negativos, un modelo podría tener una alta precisión simplemente prediciendo siempre la clase negativa, pero esto no sería útil. La puntuación F1 considera tanto la precisión como la sensibilidad, asegurando que el modelo sea efectivo en identificar la clase positiva.

Si bien es una medida robusta, no es siempre la mejor para todas las situaciones. Por ejemplo, en ciertos contextos, es posible que se quiera priorizar la precisión sobre la sensibilidad, o viceversa, dependiendo de las consecuencias de los falsos positivos y falsos negativos.

#### 1.9.1.6. Curva ROC (Receiver Operating Characteristic).

La curva ROC es una herramienta gráfica utilizada para evaluar la capacidad de discriminación de un modelo de clasificación binaria en diferentes umbrales de decisión, mostrando en un gráfico bidimensional la relación entre la Sensibilidad (Tasa de Verdaderos Positivos) en el eje vertical y el complemento de la Especificidad (Tasa de Falsos Positivos) en el eje horizontal.

Una curva que se eleva rápidamente hacia la esquina superior izquierda indica un buen rendimiento. Cuanto más cerca esté la curva del borde superior izquierdo del gráfico, mejor será la capacidad del modelo para distinguir entre las clases positiva y negativa.

#### 1.9.1.7. Área Bajo la Curva *(AUC).*

El AUC *(Area Under the Curve)* representa de manera integral el área que se encuentra bajo la Curva ROC, ofreciendo así una medida consolidada del rendimiento del modelo a través de diversos umbrales de clasificación. Este indicador oscila entre 0 y 1, donde un valor de 1 denota un modelo perfecto, capaz de clasificar con precisión todas las instancias positivas y negativas. Por otro lado, un AUC de 0.5 indica un rendimiento que no supera la aleatoriedad.

Su valor radica en su capacidad para proporcionar un resumen unificado de la eficacia del modelo en distinguir entre las clases a lo largo de todos los posibles niveles de umbral, lo que aporta los siguientes beneficios:

**Evaluación Independiente del Umbral**. Proporcionando una medida que no depende de un umbral específico, lo cual es ventajoso ya que diferentes aplicaciones pueden requerir diferentes umbrales de decisión.

**Comparación de Modelos**. Facilitando la comparación directa de modelos diferentes, permitiendo elegir el modelo con mejor capacidad de discriminación general entre clases.

**Resiliencia a Desbalance de Clases**. Siendo especialmente útil en contextos donde las clases están desbalanceadas, ya que considera tanto los verdaderos positivos como los falsos positivos a lo largo de todos los umbrales.

En conjunto, la Curva ROC y el AUC ofrecen una visión completa del rendimiento de un modelo de clasificación, más allá de lo que las métricas individuales como la Precisión, la

Sensibilidad o la Exactitud pueden proporcionar. Permiten evaluar cómo el rendimiento del modelo varía con diferentes umbrales de clasificación y proporcionan una comprensión clara de la capacidad del modelo para diferenciar entre clases, lo que es de suma importancia en muchas aplicaciones prácticas (figura 1.29).

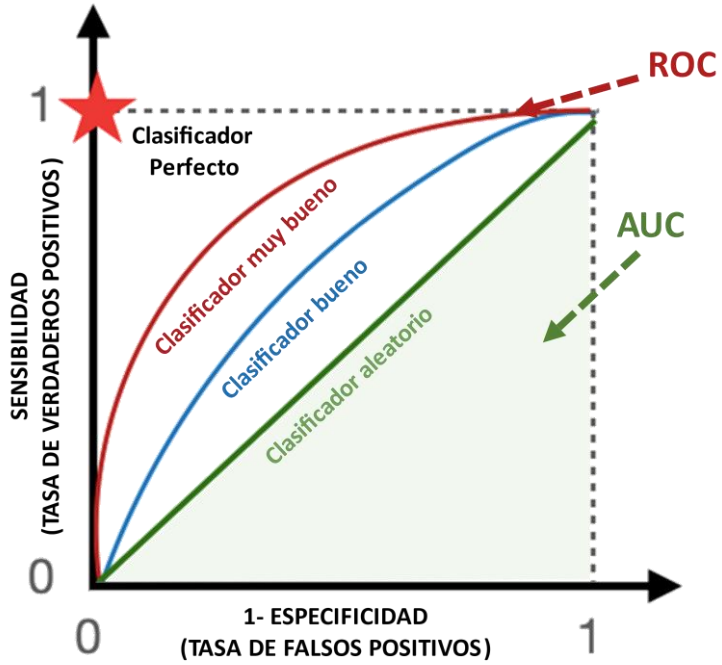


Figura 1.29. Gráfico de ROC y AUC.

**1.9.1.8. Matriz de Confusión *(Confusion Matrix).***

La Matriz de Confusión el rendimiento de los algoritmos de clasificación. Es una tabla que permite la visualización del rendimiento de un algoritmo en términos de la precisión de sus predicciones. Se utiliza para comparar las clasificaciones reales con las predicciones realizadas por el modelo. Generalmente se presenta en un formato de 2x2 en problemas de clasificación binaria, pero puede tener dimensiones mayores para problemas de clasificación con múltiples clases. En una matriz de confusión típica para un problema binario, los términos comunes son:

**Verdaderos Positivos (TP).** Casos en los que el modelo predijo correctamente la clase positiva.

**Falsos Positivos (FP).** Casos en los que el modelo predijo incorrectamente la clase negativa como positiva.

**Verdaderos Negativos (TN).** Casos en los que el modelo predijo correctamente la clase negativa.

**Falsos Negativos (FN).** Casos en los que el modelo predijo incorrectamente la clase positiva como negativa.

La matriz de confusión generalmente se organiza como se muestra en la figura 1.30.

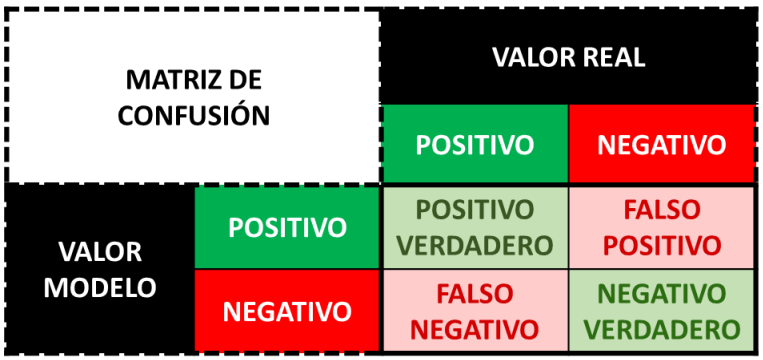


Figura 1.30. Matriz de confusión.

### 1.9.1.9. Matriz de Confusión normalizada (*Normalized Confusion Matrix*).

Una matriz de confusión normalizada muestra el rendimiento de un modelo de clasificación en términos de proporciones o porcentajes en lugar de números absolutos. Cada elemento de la matriz representa la proporción de predicciones para una clase real frente a una clase predicha, facilitando la comparación del rendimiento del modelo entre clases, especialmente en conjuntos de datos desbalanceados. Esta normalización ayuda a entender mejor cómo se distribuyen los errores de clasificación y las predicciones correctas relativas al número total de casos en cada clase (figura 1.31).

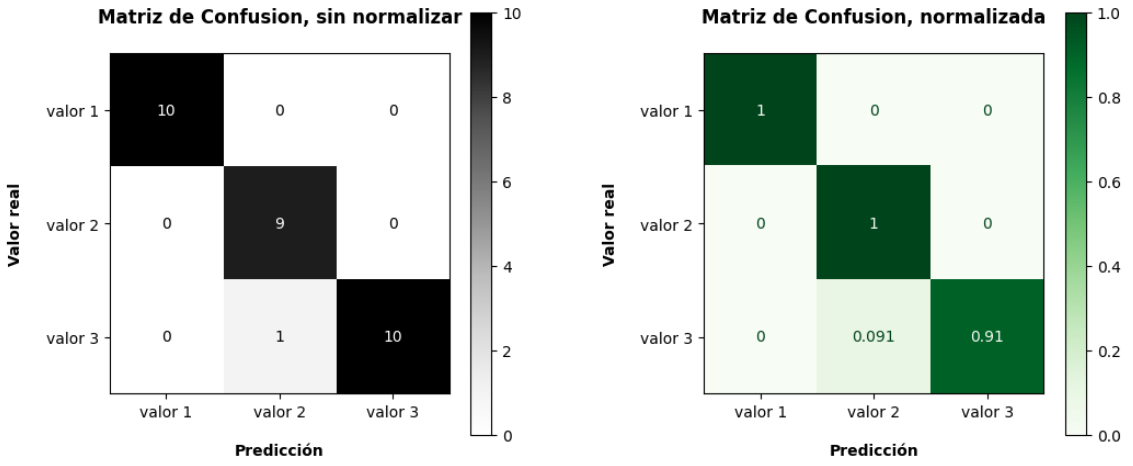

Figura 1.31. Matrices de confusión sin normalizar (izquierda) y normalizada (derecha).

Cada una de estas métricas tiene sus propias fortalezas y debilidades y debe ser seleccionada en función del contexto específico del problema y lo que se considera más importante: ¿es más crítico evitar falsos positivos, capturar todos los casos positivos, o encontrar un equilibrio entre ambos? La elección de la métrica adecuada es crucial para obtener una evaluación precisa del rendimiento de un modelo de clasificación.

### 1.9.2. Medidas para algoritmos de regresión

En los problemas de regresión, se utilizan distintas medidas de rendimiento para evaluar la precisión de los algoritmos de regresión. Estas métricas ayudan a determinar que tan bien el modelo predice los valores continuos. A continuación, presentaremos las más comunes.

#### 1.9.2.1. Error Cuadrático Medio (MSE - Mean Squared Error).

El MSE calcula el promedio de los cuadrados de los errores, es decir, la diferencia cuadrada entre los valores observados y los predichos. El MSE mide la calidad del modelo, con valores más bajos indicando un mejor ajuste. Es útil cuando se quiere penalizar más fuertemente los errores grandes y se está menos preocupado por la presencia de valores atípicos

$$MSE = \frac{1}{n}\sum_{1=1}^{n}(y_i - \hat{y}_1)^2$$

Donde:

- $MSE$ es el Error Cuadrático Medio,
- $y_i$ es el valor original,
- $\hat{y}_1$ es la predicción del modelo.

#### 1.9.2.2. Raíz del Error Cuadrático Medio (RMSE - Root Mean Squared Error).

El RMSE representa la raíz cuadrada del MSE y proporciona una medida de error en las mismas unidades que la variable dependiente, facilitando la interpretación del valor de esta métrica. Su principal desventaja es que es más costoso computacionalmente.

$$RMSE = \sqrt{MSE}$$

#### 1.9.2.3. Error Absoluto Medio (MAE - Mean Absolute Error).

El MAE calcula el promedio de las diferencias absolutas entre los valores observados y los predichos, proporcionando una idea de la magnitud del error sin considerar su dirección. Es preferible cuando los valores atípicos no son deseables en la predicción o cuando se busca una interpretación más directa y uniforme de los errores.

$$MAE = \frac{1}{n}\sum_{1=1}^{n}|y_i - \hat{y}_1|$$

#### 1.9.2.4. Coeficiente de Determinación ($R^2$ - R-squared).

El $R^2$ representa la proporción de la varianza de la variable dependiente que ha sido explicada por las variables independientes en el modelo. El coeficiente de determinación varía entre 0 y 1, donde un valor más alto indica un mejor ajuste del modelo a los datos.

$$R^2 = 1 - \frac{\text{Suma de Cuadrados de los Residuos}}{\text{Suma Total de Cuadrados}}$$

Respecto al coeficiente de determinación, es importante recordar que su valor coincide con el cuadrado del Coeficiente de Determinación de Fischer $r$.

#### 1.9.2.5. R cuadrado ajustado.

A diferencia de $R^2$, el R cuadrado ajustado ajusta la medida para el número de predictores en el modelo, proporcionando una evaluación más precisa en presencia de múltiples variables. Es una versión modificada de $R^2$ que penaliza el exceso de predictores que no contribuyen significativamente al modelo. En la fórmula $n$ representa el tamaño de la muestra, mientras que $p$ representa el número de variables aleatorias del modelo.

$$R^2_{ajustado} = 1 - (1 - R^2) \times \frac{n-1}{n-p-1}$$

### 1.10. Medidas y técnicas de evaluación de algoritmos de clustering.

El *clustering* es un componente esencial del aprendizaje no supervisado, y su evaluación eficaz es crucial para garantizar la calidad y relevancia de los resultados obtenidos. Las técnicas de evaluación de *clustering* proporcionan medios para juzgar la efectividad de un algoritmo de *clustering* en términos de cómo agrupa los datos. Estas técnicas varían desde métodos heurísticos hasta métricas cuantitativas.

### 1.10.1. Métricas cuantitativas.

En el análisis de *clustering*, las métricas cuantitativas son esenciales para evaluar la calidad de los clusters formados por diferentes algoritmos. Estas métricas ofrecen una manera de cuantificar qué tan bien un algoritmo ha agrupado los datos, basándose en varios criterios como la compactación, separación, y la correspondencia con alguna verdad de base o etiquetas conocidas. A continuación, presentaremos algunas de las métricas cuantitativas más importantes.

#### 1.10.1.1. Índice de Jaccard (Jaccard Index).

El Índice de Jaccard mide qué tan similares son dos conjuntos al comparar la cantidad de elementos comunes (intersección) con la cantidad de elementos totales (unión) de estos conjuntos. El valor del índice varía de 0 a 1. Un valor de 0 indica que no hay elementos compartidos, mientras que un valor de 1 indica que los conjuntos son idénticos ya que comparten todos los elementos. Los valores entre 0 y 1 indica algún grado de superposición de elementos. La fórmula es la siguiente:

$$J(A,B) = \left|\frac{A \cap B}{A \cup B}\right|$$

Donde:

$A$ y $B$ son dos clusters o dos conjuntos de etiquetas (una del algoritmo de *clustering* y otra del conjunto de etiquetas reales o de otro algoritmo de *clustering*).

$|A \cap B|$ es el número de elementos que ambos *clusters* (o conjuntos de etiquetas) tienen en común.

$|A \cup B|$ es el número total de elementos únicos en ambos *clusters*.

#### 1.10.1.2. Puntuación de Silueta (Silhouette Score).

Mide qué tan bien está agrupado cada punto en su cluster, basándose en la distancia al cluster más cercano en comparación con la distancia dentro de su propio cluster. Un valor cercano a +1 indica una asignación adecuada, mientras que un valor cercano a -1 sugiere una asignación incorrecta. La fórmula es la siguiente:

$$S(i) = \frac{b(i) - a(i)}{\max\{a\,(i), b(i)\}}$$

Donde:

- $S(i)$ es el *Silhouette Score* para el i-ésimo punto,
- $a(i)$ es la distancia media del i-ésimo punto a los otros puntos en el mismo cluster,
- $b(i)$ es la distancia media del i-ésimo punto al cluster más cercano al que no pertenece.

#### 1.10.1.3. Índice Davies-Bouldin (Davies-Bouldin index).

Evalúa la eficacia del *clustering* enfocándose en dos aspectos fundamentales: la compactación y la separación de los *clusters*.

La compactación se refiere a la proximidad entre los puntos dentro de un mismo *cluster*. En un *cluster* idealmente compacto, los puntos están muy cercanos entre sí, lo que se traduce en una baja variabilidad interna del *cluster*.

Por otro lado, la separación entre los *clusters* alude a la distancia relativa entre los diferentes *clusters*. Para un *clustering* efectivo, es deseable que los *clusters* estén suficientemente separados entre sí, lo que demuestra una clara diferenciación y distinción entre los grupos. Valores más bajos indican un mejor rendimiento de *clustering*. La fórmula es la siguiente:

$$DB = n1 \sum_{i=1}^{n} max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

Donde:

- $\sigma_i$ es la dispersión media de los puntos en el *cluster* $i$,
- $d(c_i, c_j)$es la distancia entre los centros de los *clusters* $i$ y $j$, y *n cluster*,
- $n$ es el número total de *clusters*.

#### 1.10.1.4. Coeficiente de Dunn (Dunn's Index).

Evalúa específicamente la relación entre la mínima distancia inter-cluster (separación) y el máximo diámetro intra-cluster (compactación). Un valor alto del coeficiente de Dunn indica una buena separación entre los clusters con respecto a su tamaño. Es muy sensible a los valores fuera de escala ya que un solo punto puede afectar significativamente el diámetro máximo del cluster. La utilización del Coeficiente de Dunn es preferible cuando la separación clara entre clusters es más importante, pero requiere atención en la presencia de valores fuera de escala. La fórmula es la siguiente:

$$D = min_{1 \leq i \leq n} \left( min_{j \neq 1} \frac{\delta(ci, cj)}{max_{1 \leq k \leq n} \Delta(c_k)} \right)$$

Donde:

- $\delta(ci, cj)$ es la distancia entre los clusters $i$ y $j$,

- $\Delta(c_k)$ es el diámetro del cluster más grande.
- $n$ es el número total de clusters.

Si bien el Índice de Davies-Bouldin y el Coeficiente de Dunn trabajan sobre la compactación y la separación de los clusters, el primero ofrece una evaluación más general del clustering, mientras que el segundo proporciona una evaluación más sensible a la disposición y separación de los clusters, siendo más afectado por la presencia de outliers y la distribución de los puntos dentro de los clusters. La fórmula es la siguiente:

#### 1.10.1.5. Índice Calinski-Harabasz (Varianza Ratio Criterion).

Esta métrica se basa en la idea de que un buen clustering no solo debe agrupar datos similares de manera compacta (dispersión dentro de los clusters), sino que también debe asegurarse de que los diferentes clusters estén claramente separados entre sí (dispersión entre clusters). La fórmula es la siguiente:

$$CH = \frac{B(k) \,/\, (k - 1)}{W(k) \,/\, (n - k)}$$

Donde:

- $B(k)$ es la suma de las distancias cuadradas entre los centros de los clusters y el centro global,
- $W(k)$ es la suma de las distancias cuadradas dentro de cada cluster,
- $k$ es el número de clusters,
- $n$ es el número total de puntos.

#### 1.10.1.6. Información Mutua Ajustada (Adjusted Mutual Information).

Mide la información compartida entre dos *clusters*, es decir, evalúa cuánto se puede aprender sobre una agrupación al observar la otra.

Es útil cuando se conocen las verdaderas etiquetas. Esto ocurre, por ejemplo, cuando se tiene un conjunto de datos etiquetados y se desea evaluar qué tan bien un algoritmo de clustering ha podido reproducir estas etiquetas verdaderas mediante la agrupación.

Al comparar la agrupación generada por el algoritmo con las etiquetas verdaderas, la AMI ofrece una medida cuantitativa de la similitud entre estas dos asignaciones, teniendo en cuenta la posibilidad de que cualquier coincidencia pueda deberse simplemente al azar. Un

valor de AMI cercano a 1 indica que la agrupación generada por el algoritmo es muy similar a las etiquetas verdaderas. Un valor cercano a 0 sugiere que la similitud entre las agrupaciones no es mayor que la que se esperaría por azar. En otras palabras, cualquier coincidencia entre los clusters y las etiquetas verdaderas no es estadísticamente significativa. La fórmula es la siguiente:

$$AMI(U,V) = \frac{MI(U,V) - E[MI(U,V)]}{max\big(H(U),H(V)\big) - E[MI(U,V)]}$$

Donde:

- $AMI(U,V)$: Es la Información Mutua Ajustada entre dos clusters $U$ y $V$.
- $E[MI(U,V)]$: Es el valor esperado de la $MI$ entre las agrupaciones $U$ y $V$.
- $H(U)$ y $H(V)$: Son las entropías de las agrupaciones $U$ y $V$, respectivamente. La entropía mide la cantidad de incertidumbre o variabilidad en la agrupación.

**1.10.1.7. Índice de Rand Ajustado (Ajusted Rand index).**

Esta métrica mide la similitud entre dos asignaciones de clustering contando el número de pares de elementos que son asignados de manera consistente (es decir, asignados al mismo cluster o a clusters diferentes) en ambas agrupaciones teniendo en cuenta la posibilidad de que coincidencias ocurran por azar. Esto se hace normalizando el índice de tal manera que el valor esperado del ARI para dos agrupaciones aleatorias sea 0.

Un valor de 1 indica una similitud perfecta entre las dos agrupaciones, es decir, las asignaciones de clustering son exactamente las mismas. Un valor cercano a 0 sugiere poca o ninguna similitud por encima de lo que se esperaría por azar. Valores negativos son posibles. La fórmula es la siguiente:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}$$

Donde:

- $n_{ij}$: Es el número de puntos que están tanto en el grupo $i$ de la agrupación $U$ como en el grupo $j$ de la agrupación $V$.
- $a_i$: Es el número de puntos en el grupo $i$ de la agrupación $U$.

- $b_j$: Es el número de puntos en el grupo $j$ de la agrupación $V$.
- $n$: Es el número total de puntos.
- $\binom{k}{2}$Es el número de combinaciones posibles de $k$ puntos tomados de dos en dos, lo que representa el número total de pares de puntos.

Aunque la AMI y el ARI tienen propósitos similares y ambos están ajustados para el azar, difieren en su enfoque matemático y en su sensibilidad a diferentes aspectos de la estructura del clustering. Mientras que AMI mide cuánta información se gana sobre una asignación de clustering al conocer la otra, ARI mide la frecuencia con la que las asignaciones coinciden o difieren.

**1.10.1.8. Homogeneidad, Completitud y Medida V.**

Estas métricas proporcionan una forma de medir cuán bien las asignaciones de clustering coinciden con las etiquetas verdaderas. Son particularmente útiles cuando se tienen las etiquetas verdaderas de los datos y se desea evaluar qué tan bien un algoritmo de clustering ha logrado reproducir estas etiquetas, siendo importantes para asegurarnos de que el clustering no solo agrupe los datos de manera efectiva, sino que también respete la estructura subyacente natural de las clases en los datos.

**Homogeneidad**. La homogeneidad de un clustering se refiere a qué tan bien los clusters contienen solo miembros de una sola clase. Un cluster es completamente homogéneo si todos sus elementos pertenecen a una sola clase verdadera. Un valor de 1 significa homogeneidad perfecta, indicando que cada cluster contiene elementos de una única clase. Un valor menor que 1 implica una mezcla de clases dentro de los clusters.

**Completitud**. La completitud mide si todos los miembros de una clase verdadera están asignados al mismo cluster. Un clustering es completamente completo si todos los elementos de una clase están en un solo cluster, sin importar la distribución de otras clases en ese cluster. Un valor de completitud de 1 indica que todos los miembros de una clase están juntos en un solo cluster. Un valor menor indica que los miembros de una misma clase han sido repartidos en varios clusters.

**Medida V (V-Measure)**. La Medida V es una métrica que combina homogeneidad y completitud en una sola medida armónica, proporcionando una evaluación equilibrada de la calidad del clustering. Un valor de la Medida V cercano a 1 indica un equilibrio perfecto entre homogeneidad y completitud, sugiriendo que el clustering ha sido capaz de agrupar

perfectamente las clases verdaderas en clusters separados y completos. Un valor más bajo indica una disminución en una o ambas métricas.

### 1.10.2. Métodos Heurísticos de Evaluación.

Los métodos heurísticos de comparación en el análisis de clustering son enfoques menos formales y más intuitivos que se utilizan para evaluar la calidad de los clusters. A diferencia de las métricas cuantitativas, que proporcionan una evaluación numérica precisa, los métodos heurísticos se basan en reglas generales o en la experiencia para hacer juicios sobre la efectividad del clustering. Estos métodos son particularmente útiles cuando no disponemos de etiquetas verdaderas o cuando queremos obtener una visión más cualitativa del rendimiento del algoritmo de clustering. El más utilizado es el método del codo.

#### 1.10.2.1. Método del Codo (Elbow Method).

El método del codo es una técnica utilizada para determinar el número óptimo de clusters en un conjunto de datos. Se realiza ajustando un algoritmo de clustering, como K-Means, con diferentes cantidades de clusters y calculando la distorsión promedio en cada caso. La distorsión mide cuán cerca están los puntos de datos de sus respectivos centroides en un cluster.

A medida que aumentamos el número de clusters, la distorsión tiende a disminuir. El punto en el que la disminución se vuelve más gradual y forma una curva en forma de codo en un gráfico se considera el número óptimo de clusters, lo que ayuda en la toma de decisiones sobre la segmentación de datos. En la figura 1.32 se visualiza que el número óptimo de clusters es 3.

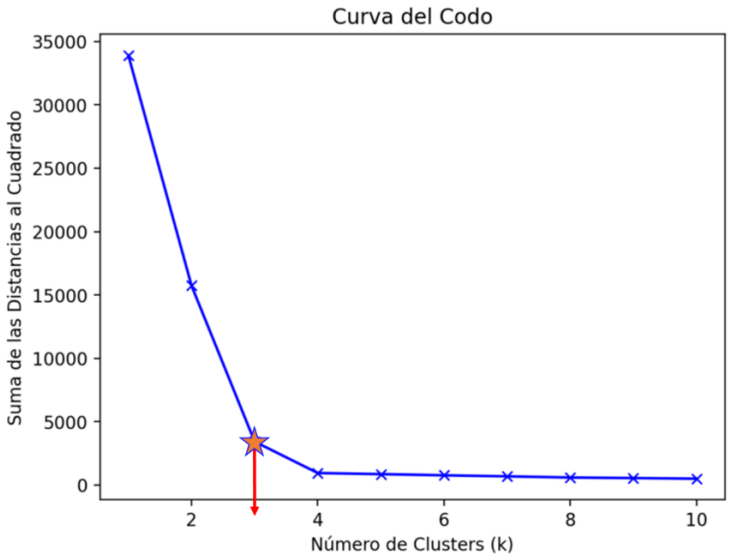


Figura (1.32). Método del codo para obtener el número de clusters óptimo.

## 1.11. Normalización de datos.

La normalización es una técnica de pre-procesamiento de datos que se utiliza para ajustar las características o atributos de los datos dentro de un rango específico y mejorar su interpretación. En el contexto de Aprendizaje Automático, se utiliza para mejorar la eficiencia y precisión de los algoritmos.

Esta técnica se utiliza para estandarizar los datos y reducir el impacto de las diferencias en la escala y la magnitud de los atributos de los datos. Por ejemplo, si una característica tiene valores en el rango de 1 a 1000 y otra característica tiene valores en el rango de 1 a 5, la característica con valores más grandes tendrá una influencia dominante en el modelo de aprendizaje automático. A continuación, detallaremos las técnicas de normalización más utilizadas en Aprendizaje Automático.

### 1.11.1. Normalización Min-Max.

Esta técnica re escala linealmente todas las características a un intervalo específico, generalmente. Es útil cuando los datos tienen valores extremos o rangos muy diferentes, y se desea que todas las características tengan el mismo rango. Su gran desventaja radica en que es muy sensible a los valores atípicos.

$$N_{x_i} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

Donde:

- $N_{x_i}$ es el valor normalizado,
- $x_i$ es el valor original,
- $x_{min}$ es el valor mínimo en el conjunto de datos,
- $x_{max}$ es el valor máximo en el conjunto de datos.

### 1.11.2. Normalización Z-Score o Estandarización.

Esta técnica transforma los datos para que tengan una media de 0 y una desviación estándar de 1. Es útil cuando los datos tienen una distribución normal y se desea que las características tengan una media de 0 y una desviación estándar de 1. Es menos sensible a los valores atípicos que la normalización Min-Max. La fórmula para la normalización Z-Score es:

$$Z_{x_i} = \frac{x_i - \mu_X}{\sigma_X}$$

Donde:

- $Z_{x_i}$ es el valor estandarizado,
- $x_i$ es el valor original,
- $\mu_X$ es la media de los datos,
- $\sigma_X$ es el desvío estándar de los datos.

### 1.11.3. Estandarización modificada.

En la estandarización modificada se toma los valores $Z_{x_i}$ y se los divide por la raíz cuadrada del tamaño de la muestra $\breve{Z}_{x_i} = \frac{Z_{x_i}}{\sqrt{n}}$. Esto asegura que la suma de los cuadrados de los valores estandarizados de esta manera sea igual a 1, es decir:

$$\sum_{i=1}^{n} \breve{Z}_{x_i}^2 = 1$$

Lo que se busca con esto es que la varianza de los valores estandarizados sea igual a la varianza original. A este concepto se lo denomina preservación de la varianza, algo que resulta muy importante para ciertos algoritmos.

$$\sum_{i=1}^{n} \breve{Z}_{x_i}^2 = 1 \Rightarrow \sum_{i=1}^{n} \frac{(x_i - \mu)^2}{n} = \sigma^2$$

### 1.11.4. Escalado por Máximo Absoluto.

Esta técnica escala cada característica dividiéndola por el valor máximo absoluto en cada característica. Esto asegura que cada característica esté en el rango de -1 a 1. Asegurándose mantener el 0 en el conjunto de datos de respuesta, lo que es útil en algunos algoritmos. La fórmula es:

$$S_{x_i} = \frac{x_i}{|x_{max}|}$$

Donde:

- $S_{x_i}$ es el valor normalizado,
- $x_i$ es el valor original,
- $x_{max}$ es el valor máximo en el conjunto de datos.

### 1.11.5. Normalización Robusta (Robust Scaler, RS).

Este método se utiliza para escalar características utilizando estadísticas que son robustas a valores atípicos. En lugar de restar la media (como en la estandarización), se resta la mediana de los datos. y escala los datos según el rango intercuartílico *(IQR)* (el rango entre el 25% y el 75% de los datos), ayudando a centrar los datos sin dar demasiado peso a los valores atípicos. La fórmula es:

$$RS_{x_i} = \frac{x_i - MEDIANA_X}{IRQ_X}$$

Donde:

- $RS_{x_i}$ es el valor normalizado,
- $x_i$ es el valor original,
- $MEDIANA_X$ es la mediana de los datos.
- $IRQ_X$ es el rango intercuartílico de los datos.

## 1.12. Búsqueda del Mejor Algoritmo: El Teorema del No Almuerzo Gratis.

El Teorema del No Almuerzo Gratis (No Free Lunch Theorem, NFL), formulado por David Wolpert y William Macready en 1997, es un principio esencial en la teoría de la optimización y el aprendizaje automático. Este teorema sostiene que, al considerar todas las posibles funciones promediadas, ningún algoritmo de búsqueda puede superar a otro de

manera universal. En términos de aprendizaje automático, esto implica que no existe un único modelo o algoritmo que sea óptimo para todas las tareas o conjuntos de datos posibles.

La relevancia de este teorema radica en destacar la importancia de seleccionar y afinar cuidadosamente el modelo de aprendizaje automático de acuerdo con las necesidades específicas de cada problema. No hay una solución única que funcione para todos los escenarios, y la efectividad de un algoritmo puede variar significativamente dependiendo de las características particulares del conjunto de datos y de la naturaleza del problema que se intenta resolver. Por lo tanto, la adaptación y personalización del enfoque de aprendizaje automático son cruciales para alcanzar resultados óptimos.

### 1.13. Revisión de los Fundamentos del Aprendizaje Automático.

A lo largo de este bloque, hemos realizado un análisis exhaustivo del aprendizaje automático. Dada la amplitud de conceptos cubiertos, es natural sentirse abrumado. En este apartado, haremos un resumen de los aspectos clave.

**Definición.** el aprendizaje automático se centra en permitir que las máquinas mejoren en tareas específicas a través del aprendizaje a partir de datos, en vez de depender exclusivamente de la programación basada en reglas predefinidas.

**Formas de clasificación.** Existen distintas categorías dentro de los sistemas de aprendizaje automático, incluyendo los métodos supervisados, no supervisados, auto supervisados y por refuerzo, los que operan por lotes o en tiempo real, y los basados en instancias frente a los basados en modelos.

**Aspectos claves con datos.** El éxito de un sistema de aprendizaje automático depende en gran medida de la calidad de los datos que forman el conjunto de entrenamiento. Un conjunto deficiente —pequeño, no representativo, con datos erróneos o irrelevantes (siguiendo la máxima de "basura entra, basura sale")— puede perjudicar significativamente el rendimiento.

**Aspectos claves con los modelos.** El equilibrio en la complejidad del modelo es fundamental: un modelo demasiado simple puede no captar los patrones subyacentes en los datos (subajuste), mientras que uno excesivamente complejo podría ajustarse en demasía a los datos de entrenamiento, perdiendo efectividad en casos nuevos (sobreajuste).

**Validación y ajustes.** Un aspecto fundamental en el aprendizaje automático es la evaluación y ajuste continuos de los modelos. Tras entrenar un modelo, no basta con asumir su eficacia en situaciones nuevas. Es imprescindible evaluar su desempeño y hacer ajustes según sea necesario.

### 1.14. Preguntas y respuestas del módulo.

Antes de abordar los casos de aplicación práctica, presentaremos una serie de preguntas y sus posibles respuestas correspondientes a los conceptos más importantes que se expusieron:

1. **Definir el concepto de aprendizaje automático**

   El aprendizaje automático implica la construcción de sistemas capaces de aprender a partir de datos. Aprender, en este contexto, significa mejorar en la realización de una tarea específica, basándose en una medida de rendimiento establecida.

2. **Nombrar cuatro tipos de aplicaciones en las que destaca el aprendizaje automático**

   El aprendizaje automático sobresale en: resolver problemas complejos sin soluciones algorítmicas conocidas, reemplazar extensas listas de reglas ajustadas manualmente, construir sistemas que se adaptan a entornos cambiantes, y ayudar a los humanos en procesos de descubrimiento de conocimiento, como la minería de datos.

3. **¿Qué es un conjunto de entrenamiento etiquetado?**

   Un conjunto de entrenamiento etiquetado es aquel que incluye, para cada instancia de datos, la solución deseada o "etiqueta", proporcionando así ejemplos claros de las respuestas esperadas.

4. **¿Cuáles son las dos tareas supervisadas más comunes?**

   Las tareas de aprendizaje supervisado más habituales en el campo de la inteligencia artificial son la regresión y la clasificación, ambas enfocadas en aprender a partir de datos etiquetados para luego poder realizar predicciones o clasificaciones informadas sobre nuevos datos no vistos anteriormente.

   La regresión se centra en predecir valores continuos y se utiliza en situaciones donde el objetivo es determinar la magnitud de una variable, como el precio de una casa basándose en sus características. Ejemplos típicos de regresión incluyen la regresión lineal y los árboles de regresión.

   Por otro lado, la clasificación se ocupa de asignar categorías discretas a los datos, como identificar si un correo electrónico es spam. Esta tarea se maneja con técnicas como los

K-vecinos más cercanos o las redes neuronales, y es fundamental en problemas donde las respuestas son categóricas, como en la clasificación binaria o multiclase.

5. **¿Cuándo se debería realizar un proceso de reducción de dimensionalidad?**

   La reducción de dimensionalidad es útil cuando se tienen conjuntos de datos con muchas variables, lo cual incrementa los costos computacionales y complica la interpretación y el análisis. Este proceso ayuda a mejorar el rendimiento de los modelos de aprendizaje automático, facilita la visualización de los datos y previene el sobreajuste.

6. **Nombrar cuatro tareas comunes no supervisadas**

   Entre las tareas no supervisadas más comunes se encuentran la agrupación, visualización, reducción de dimensionalidad y aprendizaje de reglas de asociación.

   **Agrupamiento (Clustering).** Consiste en agrupar objetos similares en conjuntos, utilizándose para descubrir estructuras ocultas en los datos, como segmentar clientes por preferencias.

   **Visualización.** Transforma grandes conjuntos de datos en gráficos y diagramas para revelar patrones y tendencias, facilitando la comprensión y el análisis de los datos.

   **Reducción de Dimensionalidad.** Reduce la cantidad de variables en un conjunto de datos, conservando las más importantes para simplificar el análisis sin perder información significativa.

   **Aprendizaje de Reglas de Asociación.** Identifica relaciones significativas entre variables en grandes bases de datos, como descubrir productos que se compran juntos frecuentemente.

7. **¿Qué tipo de algoritmo es el adecuado para segmentar clientes en varios grupos?**

   Para la segmentación de clientes sin definiciones grupales previas, se utilizaría un algoritmo de agrupación (aprendizaje no supervisado). En cambio, si se conocen los grupos objetivo, se emplearía un algoritmo de clasificación (aprendizaje supervisado), alimentándolo con múltiples ejemplos de cada grupo para clasificar a los clientes en estas categorías.

8. **¿El problema de la detección de spam de enmarca como un problema de aprendizaje supervisado o no supervisado?**

   La detección de spam se enmarca como un problema de aprendizaje supervisado, ya que implica alimentar al algoritmo con correos electrónicos y sus respectivas etiquetas (spam o no spam).

9. **¿Qué es un sistema de aprendizaje en línea?**

   Un sistema de aprendizaje en línea es aquel que aprende de manera incremental, adaptándose rápidamente a datos cambiantes y a sistemas autónomos, y tiene la capacidad de manejar grandes volúmenes de información.

10. **¿Qué es el aprendizaje fuera del núcleo?**

    El aprendizaje fuera del núcleo se refiere a algoritmos capaces de procesar enormes cantidades de datos que superan la capacidad de la memoria principal de una computadora. Estos algoritmos dividen los datos en mini-lotes y aplican técnicas de aprendizaje en línea para aprender de estos fragmentos.

11. **¿Qué tipo de algoritmo se basa en una medida de similitud para hacer predicciones?**

    Los sistemas de aprendizaje basados en instancias utilizan medidas de similitud para hacer predicciones. Estos sistemas memorizan los datos de entrenamiento y, ante una nueva instancia, buscan las instancias aprendidas más similares para basar sus predicciones.

12. **¿Cuál es la diferencia entre un parámetro de modelo y un hiperparámetro de modelo?**

    Un parámetro del modelo es un componente del modelo que el algoritmo de aprendizaje ajusta para hacer predicciones (como la pendiente en un modelo lineal). En cambio, un hiperparámetro es un parámetro del algoritmo de aprendizaje en sí (como la cantidad de regularización), y no del modelo.

13. **¿Qué buscan los algoritmos basados en modelos? ¿Cuál es la estrategia más común que utilizan para tener éxito? ¿Cómo hacen predicciones?**

    Los algoritmos basados en modelos buscan valores óptimos para los parámetros del modelo para que este generalice bien a nuevas instancias. Comúnmente, se entrenan

minimizando una función de coste, que evalúa el rendimiento del sistema en los datos de entrenamiento y penaliza la complejidad del modelo si está regularizado. Las predicciones se realizan alimentando las características de nuevas instancias en la función de predicción del modelo, utilizando los valores de parámetros hallados por el algoritmo de aprendizaje.

**14. Nombrar cinco de los principales desafíos del aprendizaje automático.**

En el campo del aprendizaje automático, varios desafíos significativos pueden afectar la efectividad de los modelos y algoritmos.

**Falta de datos.** Ocurre cuando la cantidad insuficiente de datos de entrenamiento puede impedir que el modelo aprenda eficazmente. Esto está estrechamente relacionado con la mala calidad de los datos, que incluye errores, inconsistencias y ruido en los datos, lo que puede llevar a resultados poco fiables.

**No representatividad de los datos.** Sucede cuando los conjuntos de datos no reflejan adecuadamente la realidad o el contexto en el que se aplicará el modelo, resultando en un rendimiento deficiente en situaciones reales.

**Características no informativas.** Son aquellas que no aportan información útil para la tarea de aprendizaje, también pueden complicar el entrenamiento y la interpretación del modelo.

**Modelos demasiado simples (subajuste).** Ocurre cuando los modelos no logran capturar adecuadamente la complejidad y los patrones subyacentes presentes en los datos. Esto resulta en un rendimiento deficiente, ya que el modelo no puede hacer predicciones precisas ni generalizar bien a nuevos datos.

**Modelos excesivamente complejos (sobreajuste).** Ocurre cuando los modelos se adaptan demasiado a los datos de entrenamiento, perdiendo la capacidad de generalizar a nuevos datos.

Estos desafíos requieren un enfoque cuidadoso en la selección, preprocesamiento de los datos y en la elección del modelo adecuado para asegurar un aprendizaje eficaz y resultados confiables.

15. **Si el modelo funciona muy bien con los datos de entrenamiento, pero no se generaliza bien a nuevas instancias, ¿qué está sucediendo? Nombre tres posibles soluciones.**

    Cuando un modelo de aprendizaje automático muestra un alto rendimiento con los datos de entrenamiento, pero no logra generalizar efectivamente a nuevos datos, es probable que esté experimentando sobreajuste. Para abordar este problema, se pueden adoptar las siguientes estrategias:

    **Obtener más datos.** Ampliar el conjunto de entrenamiento puede ayudar al modelo a aprender patrones más generales y no solo las particularidades de un conjunto de datos limitado.

    **Simplificar el modelo.** Optar por un modelo más sencillo o reducir la complejidad del actual puede ser efectivo. Esto puede incluir elegir un algoritmo con menos tendencia a sobreajustar, disminuir el número de parámetros o características del modelo, o implementar técnicas de regularización que penalizan la complejidad excesiva.

    **Reducir el ruido en los datos de entrenamiento.** Mejorar la calidad de los datos de entrenamiento, eliminando o corrigiendo errores y anomalías, puede permitir que el modelo se enfoque en patrones relevantes en lugar de sobreajustarse a irregularidades o ruido en los datos.

    Estas soluciones buscan lograr un equilibrio donde el modelo es lo suficientemente general para aplicarse a datos nuevos, manteniendo un buen rendimiento en los datos de entrenamiento.

16. **¿Qué es un conjunto de prueba y por qué se debería usar?**

    Un conjunto de prueba se utiliza para estimar el error de generalización que un modelo cometerá en nuevas instancias, ofreciendo una evaluación de su rendimiento antes de lanzarlo en producción.

17. **¿Cuál es el propósito de un conjunto de validación?**

    El propósito principal de un conjunto de validación es ofrecer una forma de comparar diferentes modelos o configuraciones de hiperparámetros. Se utiliza para validar la

efectividad del modelo y realizar ajustes en los hiperparámetros antes de la fase de prueba final, garantizando así una selección adecuada y optimizada del modelo.

18. **¿Por qué es recomendable normalizar el conjunto de datos en aprendizaje automático?**

    Normalizar los conjuntos de datos en aprendizaje automático es crucial ya que iguala la escala de las características, lo que es especialmente importante en algoritmos sensibles a la magnitud de los datos. Esto facilita la convergencia más rápida en algoritmos de optimización como el descenso del gradiente y reduce el sesgo hacia características con valores más grandes. Además, ayuda a interpretar mejor los resultados en modelos como la regresión lineal y minimiza los problemas numéricos, asegurando cálculos más precisos y estables.

19. **¿Qué es el conjunto de entrenamiento-desarrollo, cuándo se necesita y cómo se debe usar?**

    El conjunto de entrenamiento-desarrollo, o conjunto train-dev, es necesario cuando existe la posibilidad de un desajuste entre los datos de entrenamiento y los datos de los conjuntos de validación y prueba.

    Es una subdivisión del conjunto de entrenamiento, en la cual el modelo no se entrena. Se utiliza para evaluar el rendimiento del modelo después de entrenarlo en el resto de los datos de entrenamiento. Si el modelo presenta buen rendimiento en el conjunto de entrenamiento, pero no en el train-dev, indica sobreajuste.

    Si rinde bien en ambos, pero falla en la validación, sugiere un desajuste significativo entre los conjuntos de entrenamiento y de validación/prueba. En este caso, se debe mejorar la calidad de los datos de entrenamiento para que sean más representativos de los conjuntos de validación y prueba.

20. **¿Qué problemas pueden surgir si se ajustan los hiperparámetros utilizando el conjunto de prueba?**

    Ajustar los hiperparámetros usando el conjunto de prueba puede llevar a un sobreajuste de este conjunto específico. Esto significa que el modelo se optimizará excesivamente para funcionar bien en los datos de prueba, perdiendo generalización y potencialmente teniendo un rendimiento peor en datos nuevos y no vistos. Como

resultado, la evaluación del error de generalización será inexacta y demasiado optimista, lo que puede llevar a una falsa confianza en el rendimiento del modelo.

**21. ¿Existe universalmente un mejor algoritmo en aprendizaje automático?**

No existe universalmente un "mejor" algoritmo de aprendizaje automático. La efectividad de un algoritmo depende del contexto específico, incluyendo la naturaleza de los datos, el problema a resolver, y los recursos computacionales disponibles. Este concepto está respaldado por el "Teorema de *No Free Lunch*", que establece que ningún algoritmo es superior para todos los problemas de optimización. Por lo tanto, la selección del algoritmo adecuado depende siempre del caso de uso específico.

### 1.15. Aplicaciones Prácticas.

A partir de este punto, comenzaremos a aplicar algunos conceptos vistos en los puntos anteriores utilizando la librería ***Scikit Learn*** disponible en Python. Para simplificar todo lo posible la operatoria de la codificación, y orientar nuestros esfuerzos en la comprensión de los conceptos adquiridos, recomendamos utilizar alguno de los servicios de IA Generativa basados en LLMs, estilo ChatGPT, existentes para la generación de código y su posterior ejecución en algún sistema online de ejecución de código Python al estilo de Google Colab.

#### 1.15.1. Aprendizaje supervisado. Clasificación con dataset de flores de Iris.

El conjunto de datos de Iris (figura 1.33) es un clásico en el campo del aprendizaje automático y la estadística, y es frecuentemente utilizado para tareas de clasificación y para explicar conceptos de aprendizaje automático. Fue introducido por el estadístico y biólogo Ronald Fisher en 1936. Este conjunto consiste en 150 registros de tres especies diferentes de flores de iris (***Iris setosa = 0***, ***Iris versicolor = 1*** e ***Iris virginica = 2***), con 50 muestras de cada una. Para cada muestra, se miden y registran cuatro características (*features*) numéricas:

- Longitud del Sépalo (en centímetros).
- Ancho del Sépalo (en centímetros).
- Longitud del Pétalo (en centímetros).
- Ancho del Pétalo (en centímetros).

Además de estas cuatro características numéricas, el conjunto de datos incluye una quinta etiqueta de clase, que identifica a cuál de las tres especies de iris pertenece cada muestra. Esto hace que el conjunto de datos de Iris sea ideal para problemas de clasificación supervisada, donde el objetivo es aprender a distinguir las flores en base a las medidas de sus sépalos y pétalos. El trabajo con el conjunto de datos de flores de Iris es a menudo considerado el “Hola Mundo” debido a su simplicidad y accesibilidad, ofreciendo una manera efectiva de demostrar la clasificación, el preprocesamiento y la validación de modelos en un formato claro y manejable.


Figura 1.33. Flores Iris en sus tres variedades.

#### 1.15.1.1. Clasificación I.

**Supongamos que tenemos una flor de iris con las siguientes medidas.**

| | |
|---|---|
| Longitud del Sépalo (cms.) | 5,0 |
| Ancho del Sépalo (cms.) | 3,6 |
| Longitud del Pétalo (cms.) | 1,4 |
| Ancho del Pétalo (cms.) | 0,2 |

*A.* Generar una predicción respecto a la especie a la que corresponde utilizando el algoritmo *Naive Bayes* Gaussiano.

*B.* Generar una predicción respecto a la especie a la que corresponde utilizando un Árbol de decisión.

*C.* Generar una predicción respecto a la especie a la que corresponde utilizando Linear *SVC*.

*D.* Generar una predicción respecto a la especie a la que corresponde usando Regresión Logística.

#### 1.15.1.2. Clasificación II.

**Utilizando Máquinas de Vectores de Soporte con kernel lineal y el conjunto de datos de IRIS. Se pide.**

*A.* Entrenar un modelo con un *train-test Split* de 80% de datos para entrenamiento y un 20% de datos para pruebas. La semilla debe estar ubicada en 0,42. El hiperparámetro de regularización *C* debe ser de 0,025.

*B.* Obtener por pantalla la precisión del conjunto de entrenamiento y la del conjunto de prueba.

*C.* Realizar una validación cruzada con el método k-fold de 5 subconjuntos.

D. *Realizar las predicciones utilizando el conjunto de pruebas y comparar con los valores reales.*
E. *Presentar la matriz de confusión sin normalizar y la matriz de confusión normalizada en formato texto y en formato gráfico.*
F. *Realizar un reporte de clasificación del modelo.*

### 1.15.2. Aprendizaje supervisado. Regresión con Conjunto de datos de Diabetes.

El conjunto de datos de diabetes *(_diabetes_dataset*) es también uno de los clásicos cuando iniciamos nuestro recorrido en las técnicas de Aprendizaje Automático. Está conformado por 442 observaciones en donde se analizan una serie de 10 variables numéricas que se suelen utilizar para explicar los valores de progresión de la enfermedad, un año después del inicio del estudio. Este dataset se popularizó en el estudio de Efron, Hastie, Johnstone y Tibshirani titulado "Last Angle Regression" que fue publicado en el 2004.

**Características del Conjunto de Datos.**

- Número de Instancias: 442
- Número de Atributos: Las primeras 10 columnas son valores predictivos numéricos
- Objetivo: La columna 11 es una medida cuantitativa de la progresión de la enfermedad un año después del inicio del estudio
- Información de los Atributos:

- ***Edad.*** Edad en años (0)
- ***sexo.*** (1)
- ***bmi.*** Indice de masa corporal (2)
- ***bp.*** Presión arterial media (3)
- ***s1 tc.*** Colesterol total en suero (4)
- ***s2 ldl.*** Lipoproteínas de baja densidad (5)
- ***s3 hdl.*** Lipoproteínas de alta densidad (6)
- ***s4 tch.*** Colesterol total / HDL (7)
- ***s5 ltg.*** Posiblemente logaritmo del nivel de triglicéridos en suero (8)
- ***s6 glu.*** Nivel de azúcar en sangre (9)

La versión incorporada en Scikit Learn trabaja con datos con estandarización modificada. La versión original, con datos sin estandarizar, se encuentra disponible en:

https://www4.stat.ncsu.edu/~boos/var.select/diabetes.tab.txt.

### 1.15.2.1. Regresión Lineal.

**Ejercicio de Regresión Lineal con Datos de Diabetes. Se tiene a disposición un conjunto de datos normalizados con la técnica de la estandarización modificada (conjunto "toy" – incorporado en el paquete Scikit Learn) y otro conjunto de datos con las unidades originales (conjunto "web" – disponible en la web citada anteriormente). Se pide.**

*A. Selección de Características:*

- *i.* Implementar una función que permita al usuario introducir un número del 0 al 9, correspondiente a una de las características del conjunto de datos. (Validar la entrada del usuario y de que el número esté dentro del rango permitido).
- *ii.* Mostrar el nombre de la característica seleccionada.

*B. Creación y Entrenamiento de Modelos de Regresión Lineal:*

- *i.* Dividir, para ambos datasets, los datos en conjuntos de entrenamiento y prueba.
- *ii.* Entrena un modelo de regresión lineal para ambos conjunto de datos.

*C. Evaluación de Modelos:*

- *i.* Calcular, para ambos modelos, el coeficiente de determinación ($R^2$).
- *ii.* Calcular, para ambos modelos, el error cuadrático medio (MSE).
- *iii.* Calcular, para ambos modelos, la raíz del error cuadrático medio (RMSE)
- *iv.* Calcular, para ambos modelos, el error absoluto medio (MAE).

*D. Visualización de Resultados:*

- *i.* Generar gráficos de dispersión con las líneas de regresión para cada modelo.
- *ii.* Automatizar el nombre de los títulos y las etiquetas para que reflejen correctamente la característica utilizada en el análisis.

*E. Interpretación y Conclusión:*

- i. Interpretar las métricas de rendimiento y los gráficos generados.
- ii. ¿Cómo influye la característica seleccionada en la progresión de la diabetes?

### 1.15.3. Aprendizaje no supervisado. Conjunto de datos IRIS

#### 1.15.3.1. Clusterización I.

A partir del conjunto de datos de IRIS del paquete Scikit Learn de Python. Utilizando las características “Ancho del Sépalo” y “Longitud del Sépalo” Se pide.

*A. Aplicación del Algoritmo K-Means:*

- *i.* Implementar el algoritmo K-Means para agrupar los datos en tres grupos.
- *ii.* Asignar las etiquetas de agrupamiento a cada instancia en el conjunto de datos.

*B. Visualización de los Resultados:*

- *i.* Visualizar los resultados del agrupamiento utilizando un diagrama de dispersión.
- *ii.* Usa diferentes colores para representar los distintos grupos.
- *iii.* Marcar los centroides de cada grupo.

*C. Análisis y Conclusión:*

i. Analizar los resultados obtenidos. ¿Qué ventajas y desventajas tiene el algoritmo k-means?

#### 1.15.3.2. Clusterización II.

A partir del conjunto de datos de IRIS del paquete Scikit Learn de Python. Se pide.

*A. Aplicación y Optimización de K-Means:*

- *i.* Implementar el algoritmo K-Means con un número inicial de 3 clusters.
- *ii.* Utilizar el método del codo para explorar la distorsión para diferentes números de clusters y determinar visualmente el número óptimo de clusters.

*B. Evaluación con el Puntaje de Silueta:*

i. Calcular el puntaje de silueta para tu modelo K-Means.

*C. Reducción de Dimensionalidad con PCA:*

i. Aplicar PCA para reducir la dimensionalidad de los datos a dos componentes para una visualización efectiva.

*D. Visualización de Clusters:*

i. Visualizar los clusters en un espacio de dos dimensiones. Observar cómo se agrupan los datos y cómo se relacionan los clusters con los componentes principales.

*E. Análisis y Conclusión:*

i. Analizar los resultados obtenidos, centrándote en la efectividad del número de clusters elegido y en las características de los clusters visualizados.
ii. Reflexionar sobre la utilidad del puntaje de silueta y la reducción de dimensionalidad en el contexto del aprendizaje no supervisado.

### 1.16. Bibliografía

Géron, Aurélien- *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow_ Concepts, Tools, and Techniques to Build Intelligent Systems-O'Reilly Media (2019).*

Parrás, Diego- Inteligencia Artificial. Fundamentos y Aplicaciones. Fondo Editorial Edicon. (2023)

Raschka, Sebastian- *Machine Learning with PyTorch and Scikit-Learn, Packt Publishing (2022).*