Curso de Inteligencia Artificial

Métodos de Clasificación

Prof. Carlos A. Toro N.
carlos.toro.ing@gmail.com
2022

Objetivos

Al finalizar esta presentación, usted podrá:

- ❑ Dominar la definición de algunos de los algoritmos de clasificación más usados.
- ❑ Conocer las métricas de evaluación de desempeño de algoritmos de clasificación.
- ❑ Implementar algoritmos de clasificación usando *Sci-kit Learn*.

Contenidos

- Algoritmos de clasificación típicos
- Métricas de evaluación de desempeño algoritmos de clasificación
- *Ejercicios prácticos*

Motivación

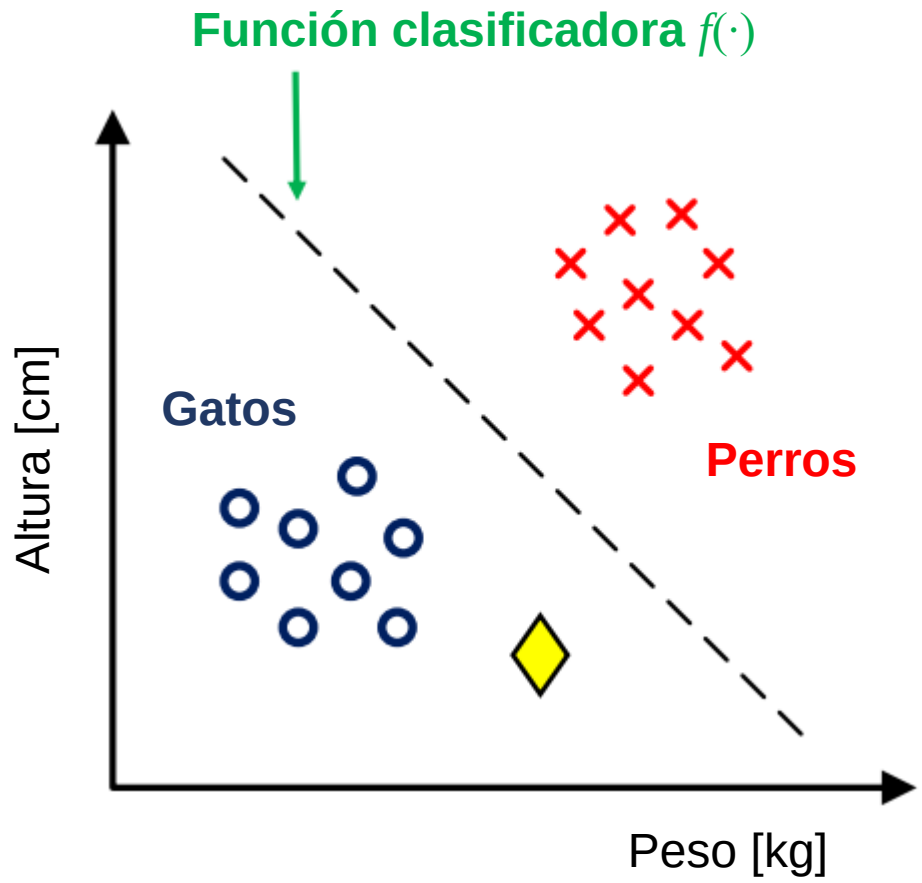
Motivación

- En clases anteriores vimos modelos de **regresión lineal** donde asumimos que la variable de respuesta es cuantitativa (medible). Sin embargo, en muchas situaciones, la variable de respuesta es **cualitativa (categórica)**.
- Las variables cualitativas toman valores en un conjunto (sin orden) C = {“cat_1”, … , “cat_n”}, por ejemplo:
  - **Color de ojos** ∈ {“café”, “azul”, “verde”}
  - **Email** ∈ {“spam”, “no spam”}
  - **Fruta** ∈ {“madura”, “no madura”}
- Así, el problema de **clasificación** consiste en **estimar resultados categóricos** usando un conjunto de variables o características (*features*) dado.
- Estimar la categoría para una observación representada con variables predictoras se denomina **clasificar esa observación**.
- A veces también nos interesará estimar la **probabilidad** de que una observación pertenezca a alguna categoría, luego, la **categoría más probable** se elige como la clase de dicha observación.

Motivación

❑ Ejemplo: clasificación de gatos vs perros

- Supongamos que medimos el **peso** y **altura** de algunos **perros y gatos**.
- Nuestro objetivo es aprender una función $f(\cdot)$ que nos diga si un vector de entrada $\boldsymbol{x} = [x_1, x_2]$ pertenece a un perro o a un gato.
  - x_1: peso
  - x_2: altura
- Pregunta: cómo clasificará el modelo el punto ◆ ?



Motivación

- En esta sección revisaremos de forma general los siguientes algoritmos para resolver la tarea de clasificación:

1. Regresión Logística
2. K-Vecinos más Cercanos (KNN, *K-Nearest Neighbors*)
3. Árboles de Decisión
4. Máquinas de Soporte Vectorial (MSV, o *Support Vector Machines*, SVM, en inglés)
5. Bayes Ingenuo (*Naive Bayes*)
6. Métodos de Aprendizaje Combinado (*Ensemble Learning*)

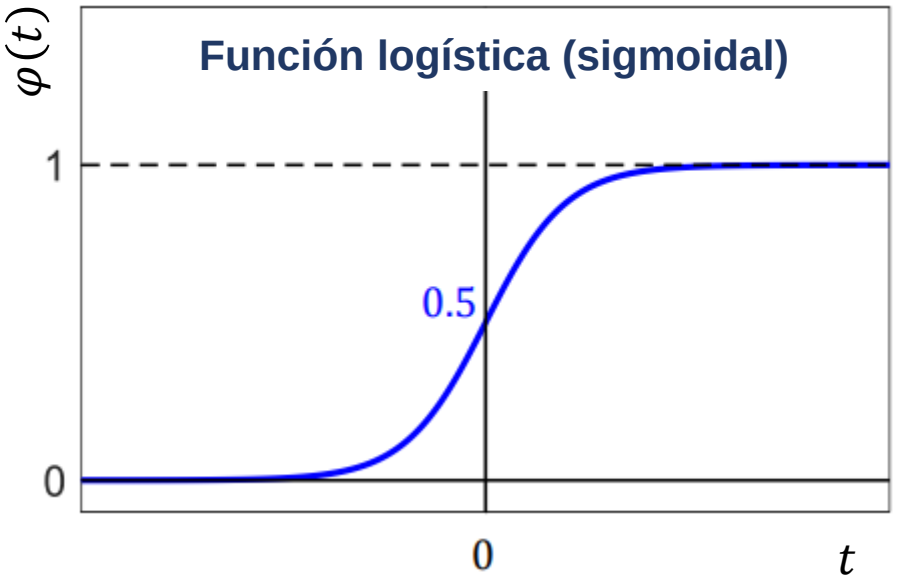
Algoritmos de Clasificación

1. Regresión Logística

Función logística

- La **función logística o sigmoidal** $\varphi(t)$ puede ser usada para clasificar observaciones binarias:
- Cuando t es grande, $\varphi(t) \to 1$
- Cuando t es pequeño, $\varphi(t) \to 0$

$$\varphi(t) = \frac{1}{1 + e^{-t}}$$



https://en.wikipedia.org/wiki/Sigmoid_function

1. Regresión Logística

1. Regresión logística:

- El modelo de regresión logística es entonces una modificación de la regresión lineal para trabajar con **categorías o resultados binarios** (ej. 0 o 1).
- El modelo se define de la siguiente manera:

$$P(Y = 1|x) = \varphi(t) = \varphi(wx + b) = \frac{1}{1 + e^{-(wx+b)}}$$

donde w indica el peso, b indica el sesgo, y $wx+b$ se considera como la función lineal de x. La clase con un valor de probabilidad más alto será la clase de x.

  - Si $(wx + b) \gg 0 \rightarrow \varphi(wx + b) \gg 0.5 \rightarrow P(Y = 1|x) \approx 1$, por lo tanto x es clasificada como 1
  - Si $(wx + b) \ll 0 \rightarrow \varphi(wx + b) \ll 0.5 \rightarrow P(Y = 1|x) \approx 0$, por lo tanto x es clasificada como 0

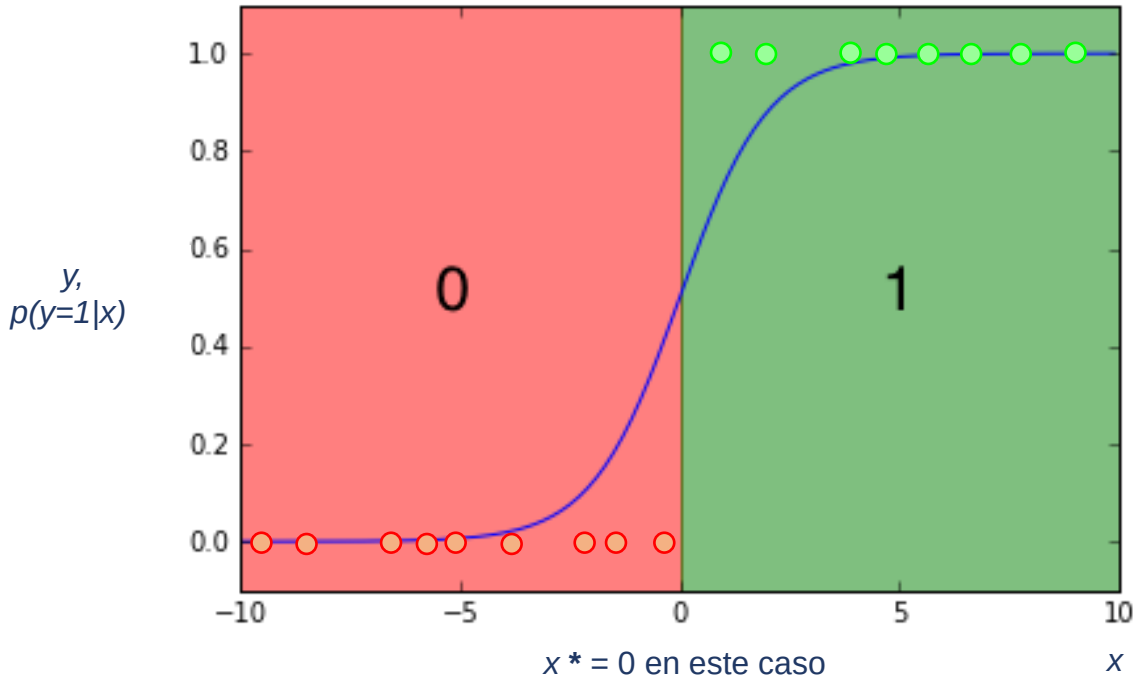
- Luego, el objetivo será encontrar los parámetros w y b para lograr que el modelo clasifique correctamente los datos, para esto, se define una función de costo que se usa durante el entrenamiento del modelo, esta se conoce como **función de entropía cruzada** (lo retomaremos posteriormente en el curso).

1. Regresión Logística

1. Regresión logística: Frontera de decisión

- Dependiendo del número de características que describan las observaciones, será más fácil o complejo visualizar la frontera de decisión que define el modelo.
- La imagen muestra la frontera de decisión óptima cuando tenemos un modelo de regresión logística (típicamente con umbral P = 0.5) para un problema con datos descritos por una variable descriptora.

Frontera de decisión dada por un recta vertical, en x* = - b/w (p = 0.5), la sigmoide nos ayuda a asignar una probabilidad a la decisión por cada punto.

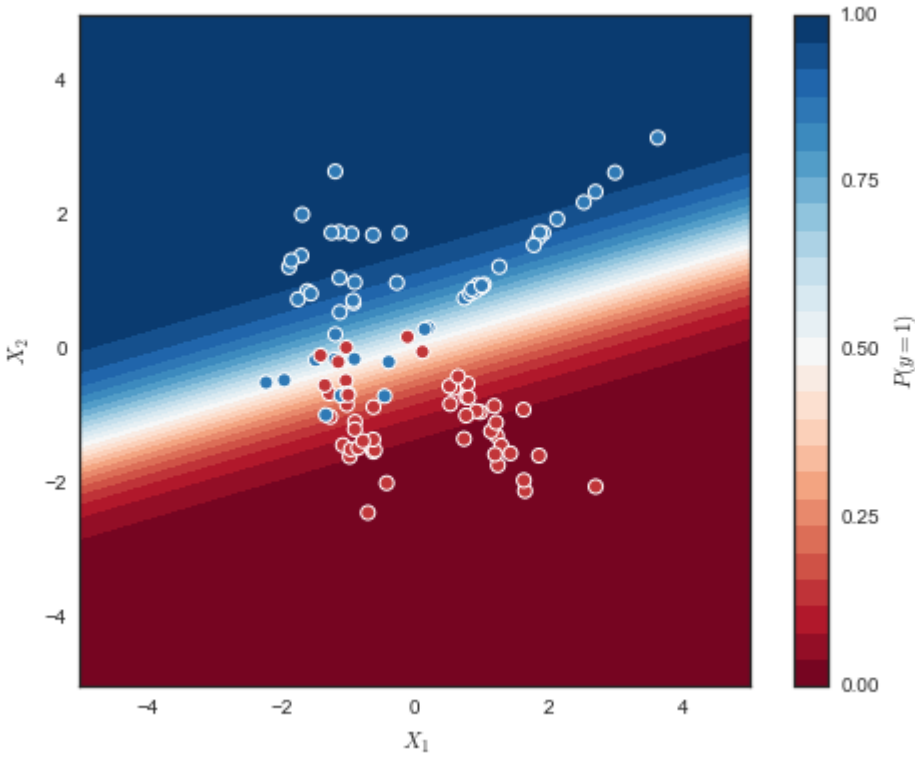


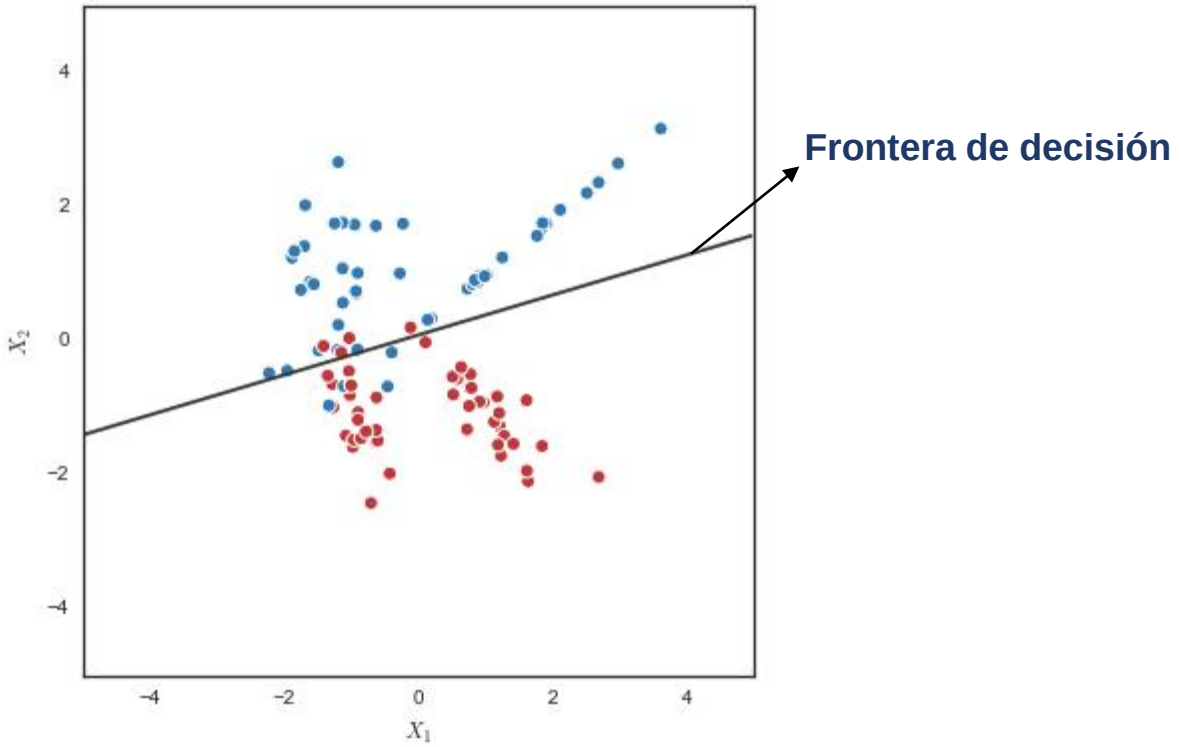
Ojo: p(y=0|x) = 1- p(y=1|x)

1. Regresión Logística

1. Regresión logística: Frontera de decisión

- Frontera de decisión óptima en umbral p = 0.5 y mapa de probabilidades cuando tenemos un modelo de regresión logística y dos variables descriptoras.



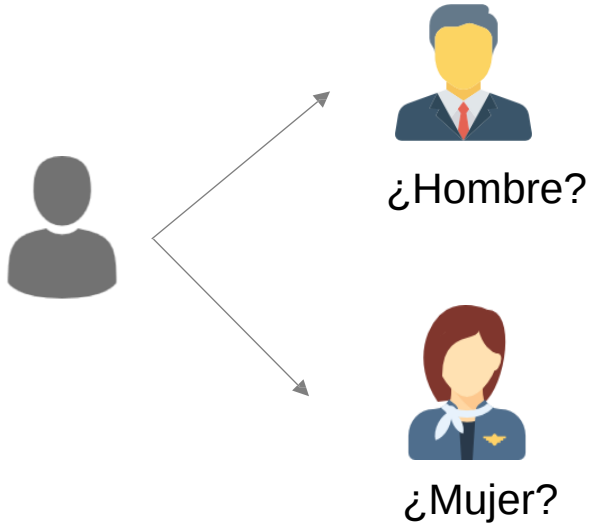


Los colores indican la probabilidad de pertenencia a una de las clases

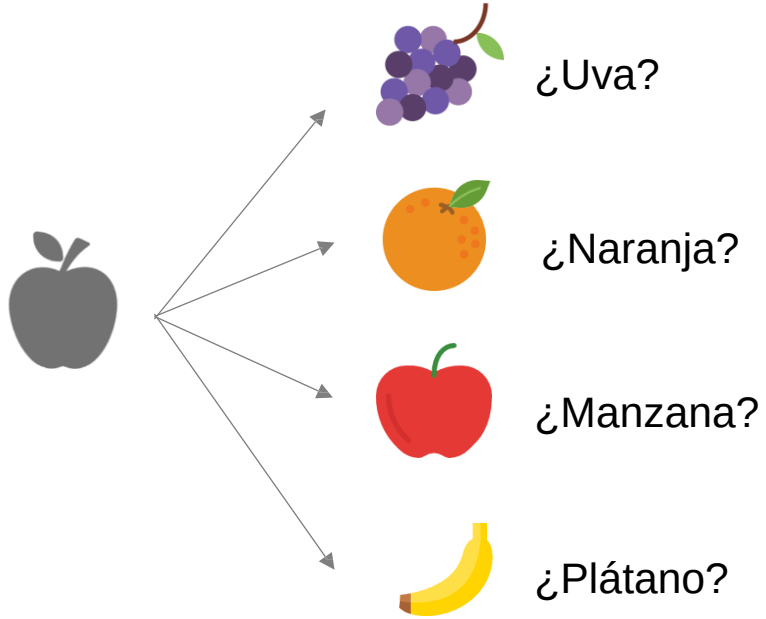
1. Extensión de Regresión Logística: Función Softmax (cont.)

- La regresión logística sólo se aplica a problemas de clasificación binaria (y a problemas multiclase estimado varios modelos por separado). Para problemas de clasificación multiclase, se puede utilizar la **función Softmax** directamente.

Problema de clasificación binaria



Problema de clasificación de varias clases



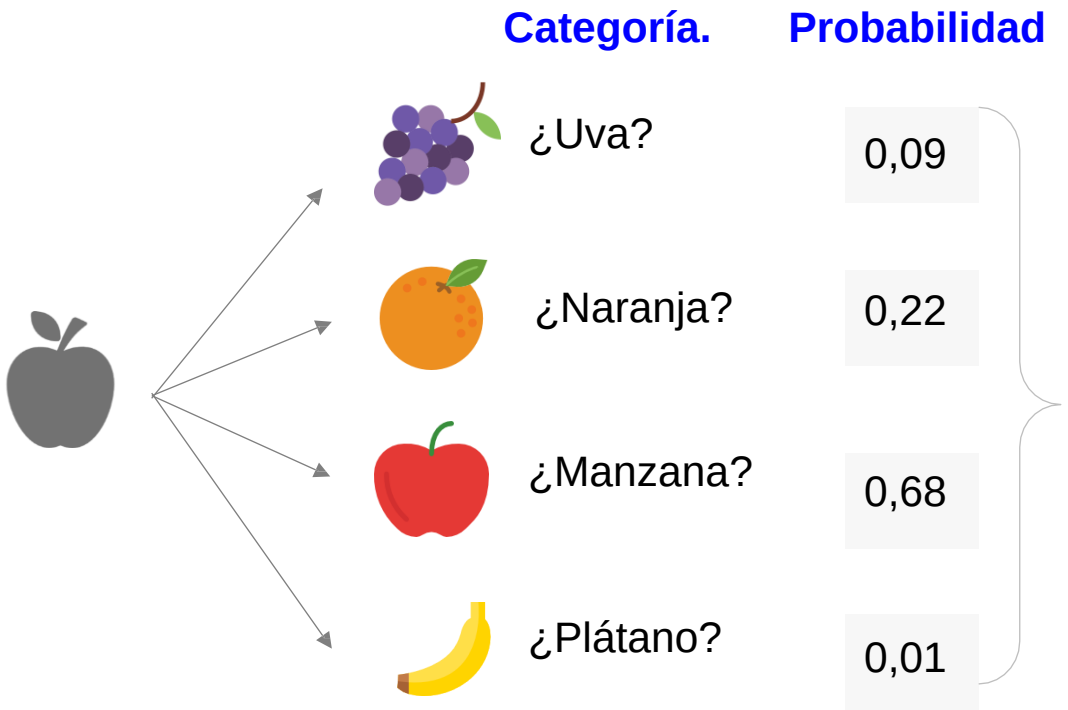
1. Extensión de Regresión Logística: Función Softmax (cont.)

- **La regresión Softmax es una generalización de la regresión logística que podemos usar para la clasificación de K clases.**
- La función Softmax se utiliza para asignar un vector K-dimensional de valores reales arbitrarios a otro vector K-dimensional de valores reales, donde cada elemento de este nuevo vector está en el intervalo (0, 1).
- La función de probabilidad de regresión de Softmax es la siguiente:

$$p(y=k \mid x;w) = \frac{e^{w_k^T x}}{\sum_{l=1}^{K} e^{w_l^T x}}, k = 1, 2 \cdots, K$$

1. Extensión de Regresión Logística: Función Softmax (cont.)

- Softmax asigna una probabilidad a cada clase en un problema de clasificación multiclase. Estas probabilidades deben sumar 1. Ejemplo:



- **Suma de todas las probabilidades:**

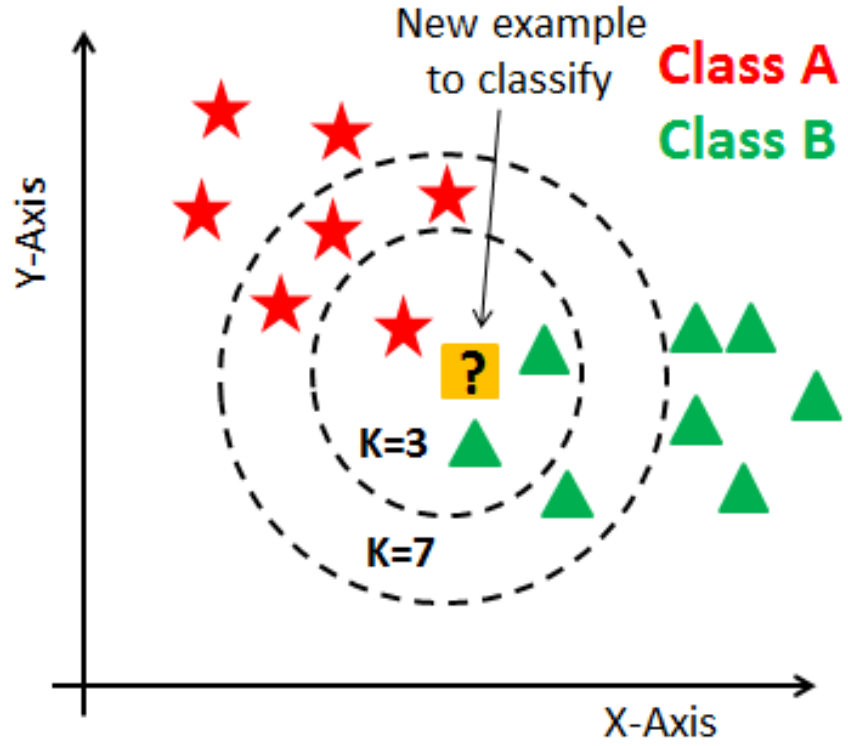
  0.09 + 0.22 + 0.68 + 0.01 = 1

- Lo más probable es que esta foto sea una manzana.

2. K-Vecinos más Cercanos (KNN, *K-Nearest Neighbors*)

2. K-Vecinos más Cercanos:

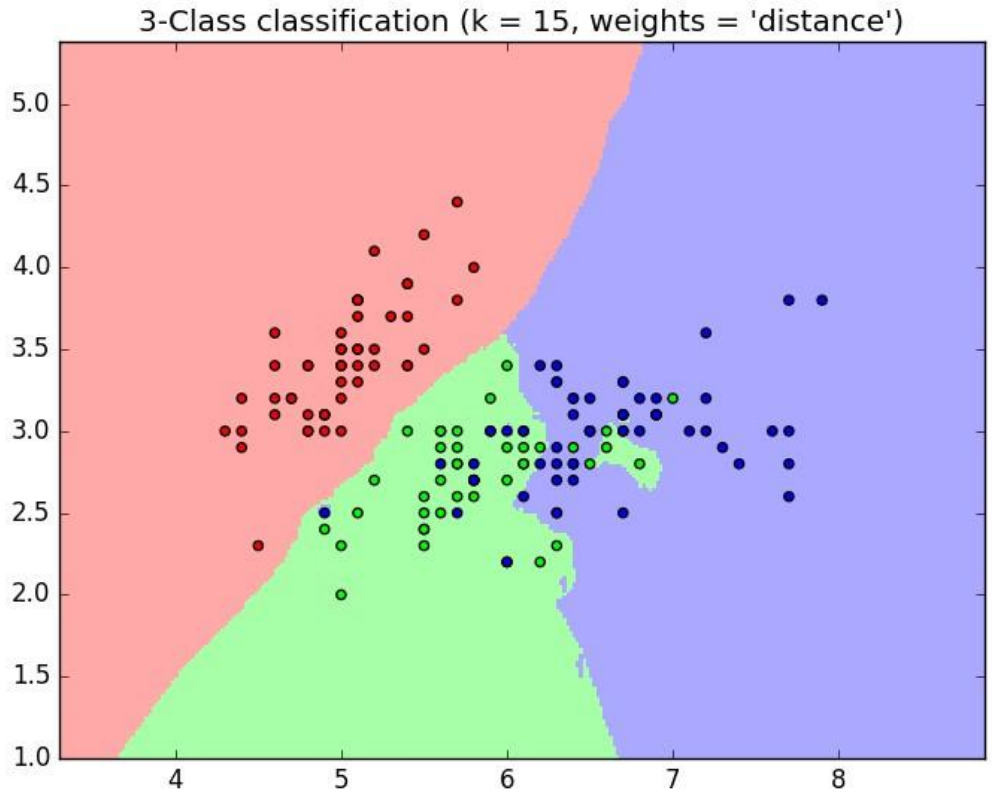
- El algoritmo de clasificación KNN es un método teóricamente maduro y uno de los más simples de al área del *Machine Learning*.
- Según este método, si la mayoría de *k* muestras más similares a una muestra (los vecinos más cercanos en el espacio de los datos) pertenecen a una categoría específica, esta muestra también pertenece a esta categoría.



La categoría desconocida del punto ***?*** variará según el número de vecinos más cercanos definidos.

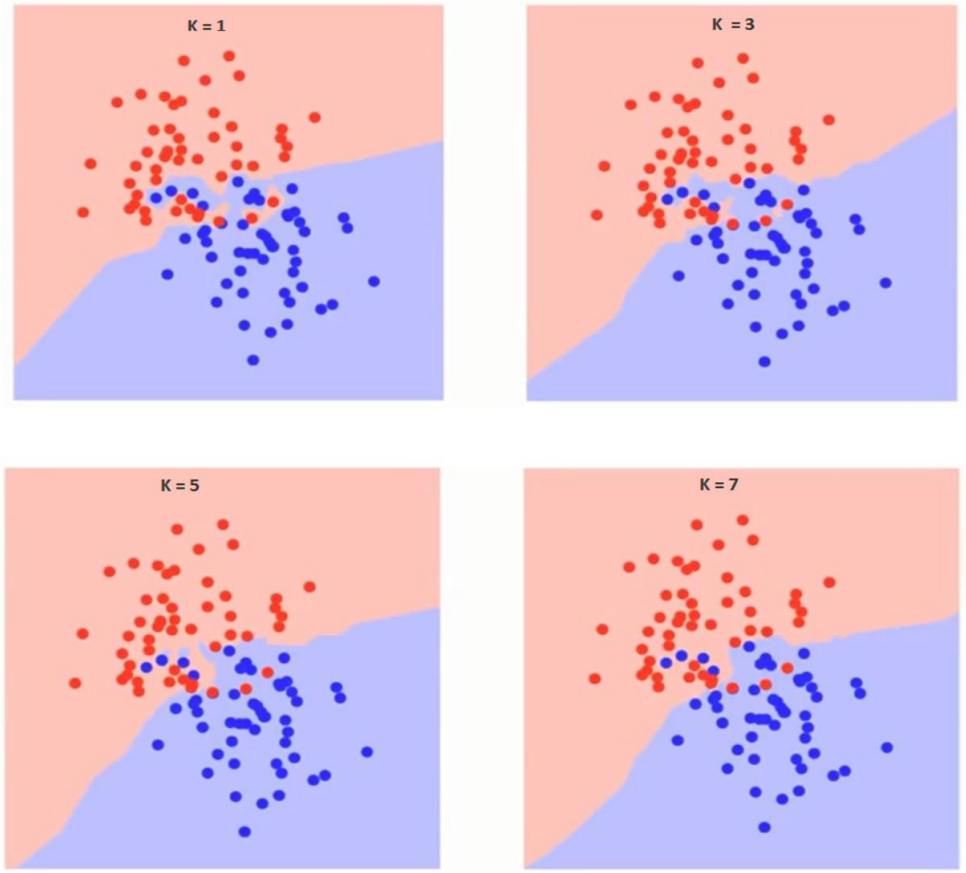
2. K-Vecinos más Cercanos (KNN, *K-Nearest Neighbors*) (cont.)

- Dado que el resultado de predicción se determina en base al número y/o peso dado a los vecinos en el conjunto de entrenamiento, el algoritmo KNN tiene una lógica simple.
- KNN es un método no paramétrico que normalmente se utiliza en conjuntos de datos con fronteras de decisión irregulares.
  - El algoritmo KNN generalmente adopta un método de votación mayoritario en caso de problemas de clasificación y el valor medio para problemas de regresión.
- KNN requiere un gran número de cálculos.
- Es un algoritmo “memorión”.



2. K-Vecinos más Cercanos (KNN, *K-Nearest Neighbors*) (cont.)

- Generalmente, un valor k mayor reduce el impacto del ruido en la clasificación, pero obscurece el límite entre las clases.
  - Un valor k más grande significa una mayor probabilidad de subajuste debido a que la segmentación es demasiado suave.
  - Un valor k menor significa una mayor probabilidad de sobreajuste debido a que la segmentación es demasiado refinada.

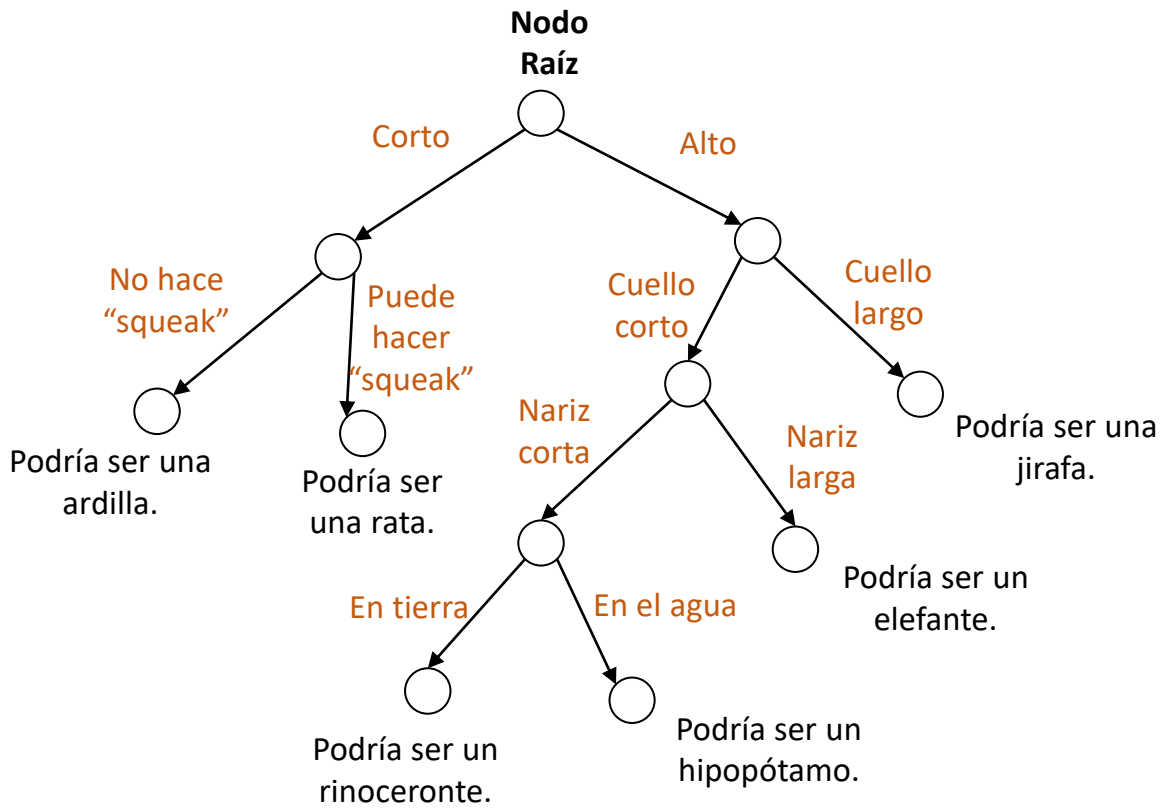


- El límite se vuelve más suave a medida que aumenta el valor de k.
- A medida que el valor de k aumenta hasta el infinito, todos los puntos de datos eventualmente se volverán azul o rojo.

3. Árboles de decisión

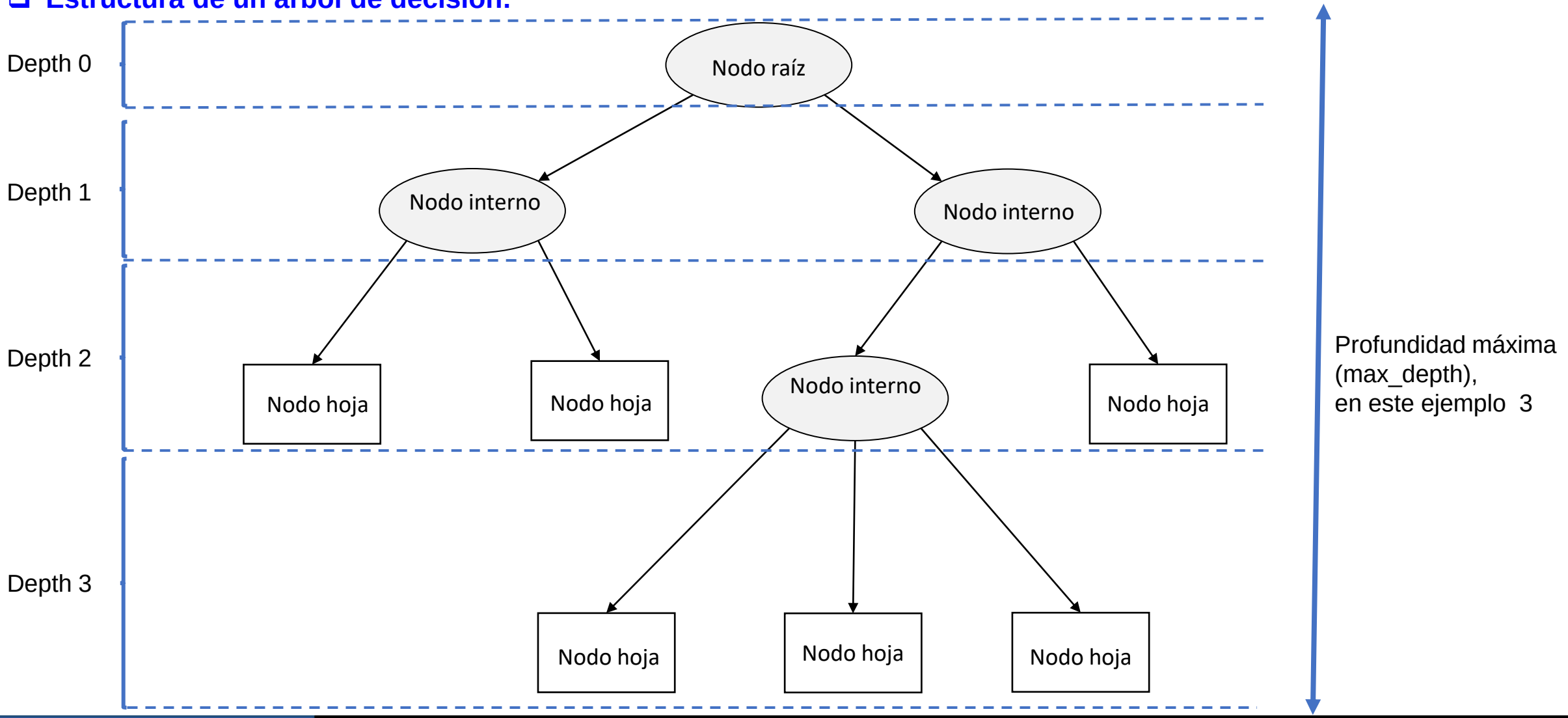
3. Un árbol de decisiones es una estructura de árbol (un árbol binario o un árbol no binario).

- **Cada nodo no-hoja representa una prueba/comparación sobre un atributo**.
- Cada rama representa la salida de un atributo en un cierto rango de valores, y **cada nodo hoja almacena una categoría**.
- Para utilizar el árbol de decisiones, iniciar desde el nodo raíz, probar los atributos de las muestras que se van a clasificar, seleccionar las ramas de salida y utilizar la categoría almacenada en el nodo hoja como resultado final.



3. Árboles de decisión (cont.)

❑ **Estructura de un árbol de decisión:**



3. Árboles de decisión (cont.): Construcción

- Para crear un árbol de decisión, necesitamos seleccionar atributos/variables y determinar la estructura del árbol para dichos atributos/variables. El paso clave de la construcción de un árbol de decisión es dividir los atributos de todos los datos, comparar los conjuntos resultantes en términos de 'pureza', y seleccionar así el atributo con la "pureza" más alta como el punto para la división del conjunto de datos.
- Las métricas para cuantificar la "pureza" incluyen la entropía de información y el índice GINI. La fórmula es la siguiente:

$$H(X) = -\sum_{k=1}^{K} p_k \log_2(p_k) \qquad Gini = 1 - \sum_{k=1}^{K} p_k^2$$

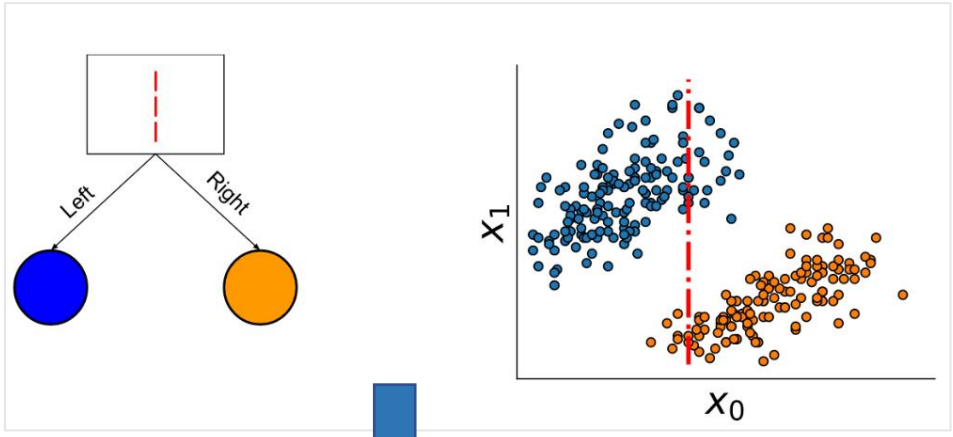
- donde p_k indica la probabilidad de que la muestra pertenezca a la clase k (hay K clases en total). Una mayor diferencia entre la pureza antes de la segmentación y después de la segmentación indica un mejor árbol de decisión.
- **Los algoritmos comunes para la creación de árboles de decisión incluyen ID3, C4.5 y CART (los nodos siempre tienen solo dos hijos).**

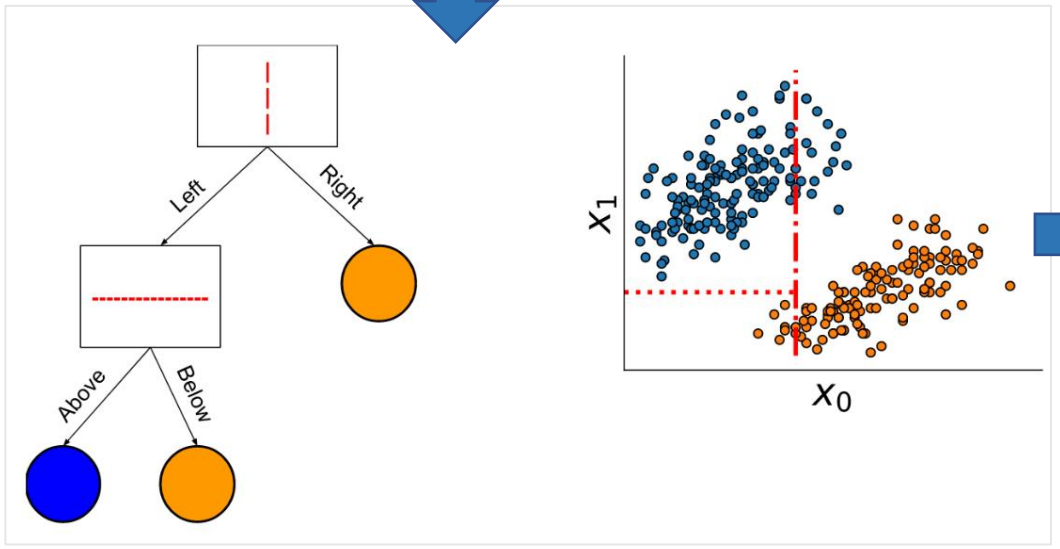
3. Árboles de decisión (cont.): Construcción

- **Selección de características/atributos:** Seleccionar una característica desde las contenidas en los datos como estándar de división del nodo actual. (Diferentes estándares generarán diferentes árboles de decisiones).
- **Generación de árbol de decisiones:** Generar los nodos internos desde la raíz hacia las hojas en función de las características seleccionadas y detenerse hasta que el conjunto de datos ya no pueda dividirse más.
- **Poda (pruning):** El árbol de decisiones puede caer en sobreajuste fácilmente, a menos que se realice una 'poda' necesaria (incluyendo una pre-poda y una post-poda) para reducir el tamaño del árbol y optimizar su estructura de nodos.

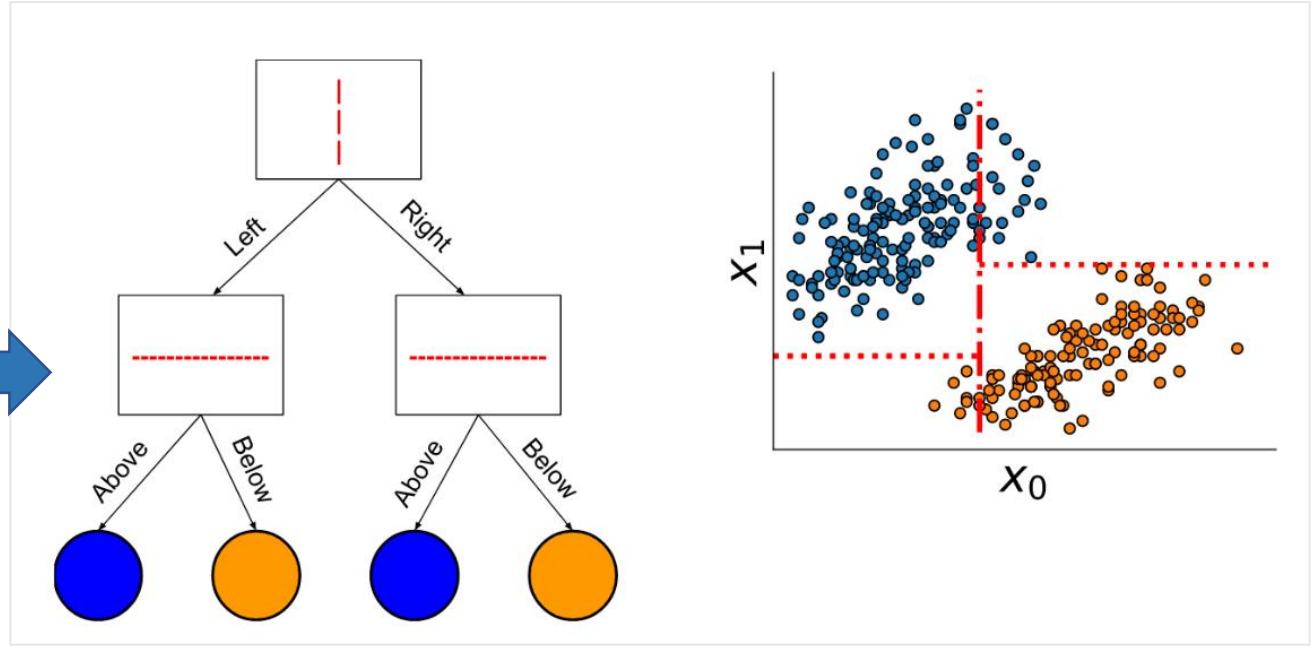
3. Árboles de decisión (cont.): Construcción

- **Intuición creación árbol de decisión para clasificación:**



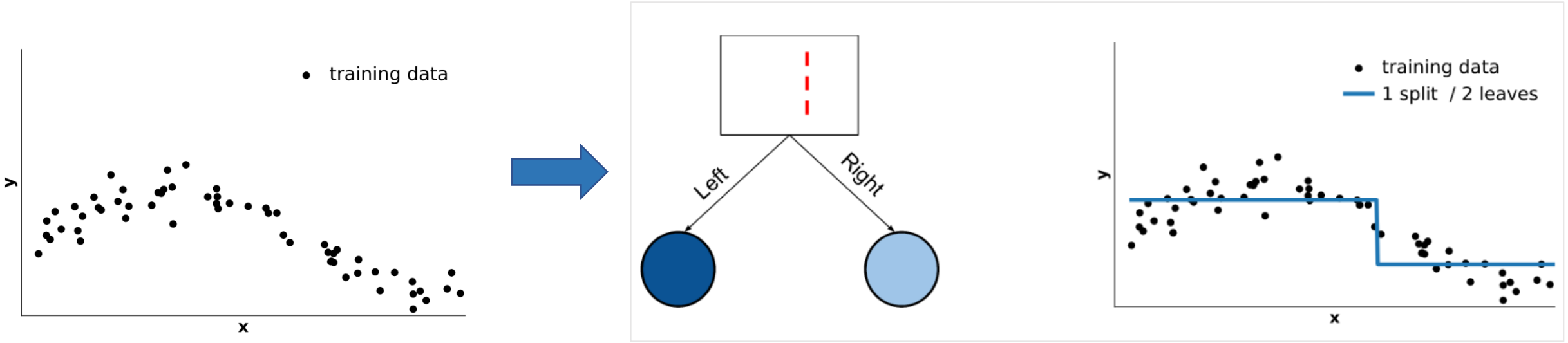


Se siguen las estrategias mencionadas para ir definiendo los umbrales de separación, asignando índices de pureza con las métricas de entropía o GINI.



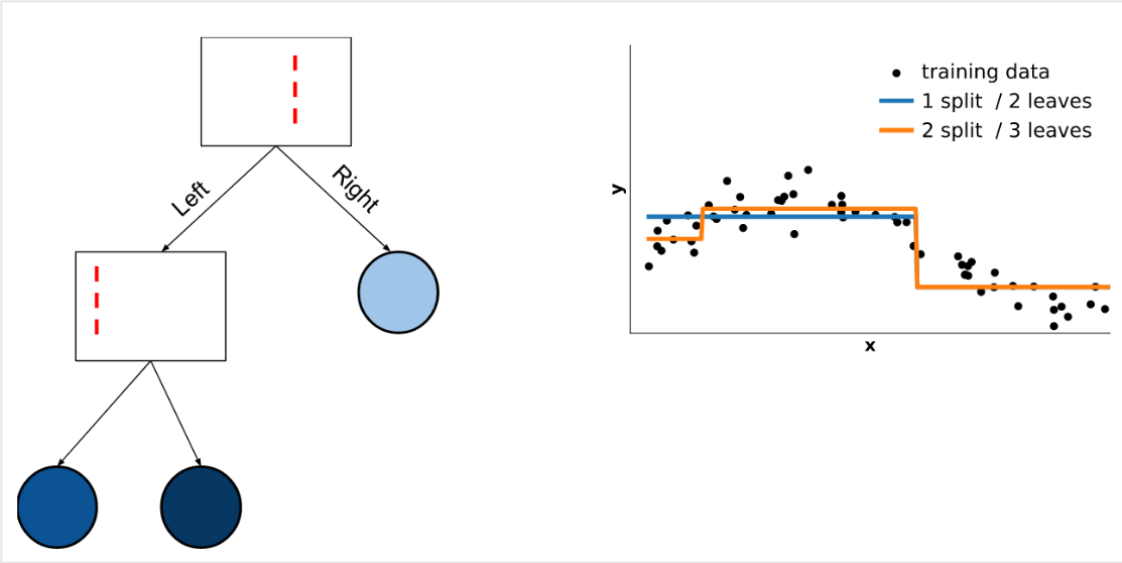
3. Árboles de decisión (cont.): Construcción

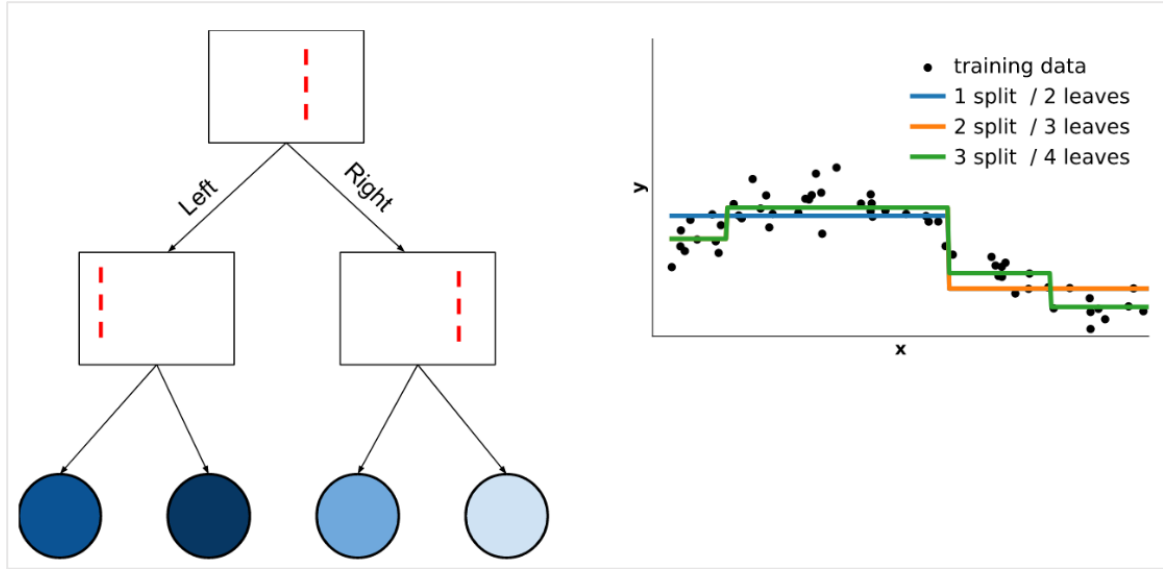
- **Intuición creación árbol de decisión para regresión:** Los árboles de decisión también se pueden ocupar para regresión.



3. Árboles de decisión (cont.): Construcción

❑ Intuición creación árbol de decisión para regresión (cont.)

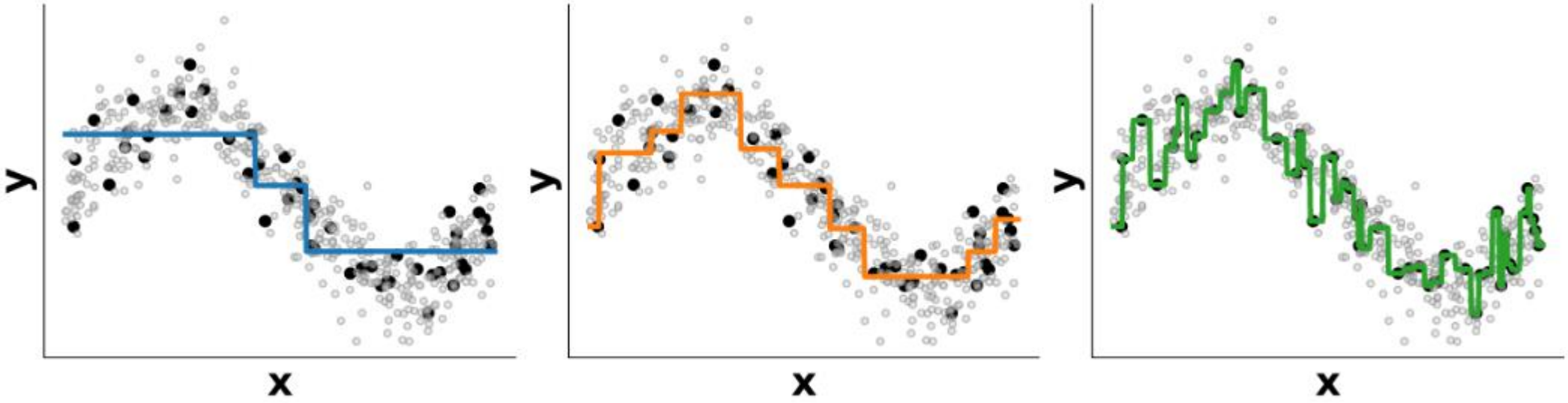




Imágenes tomadas desde: `https://inria.github.io/scikit-learn-mooc/trees/slides.html`

3. Árboles de decisión (cont.): Construcción

- Los árboles de decisión tienden a **sobreajustar** rápidamente a medida que incrementa la **profundidad** máxima con la que se definen.
- Ajustar este hiperparámetro con algún método visto (ej. búsqueda exhaustiva o aleatoria) para lograr un buen entrenamiento.

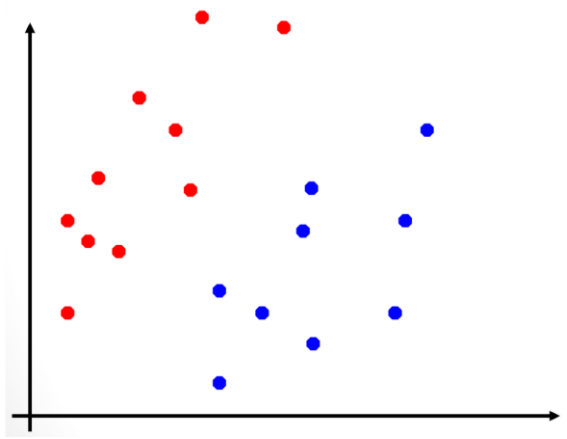


Incremento de la profundidad máxima, parámetro `max_depth` en scikit-learn.

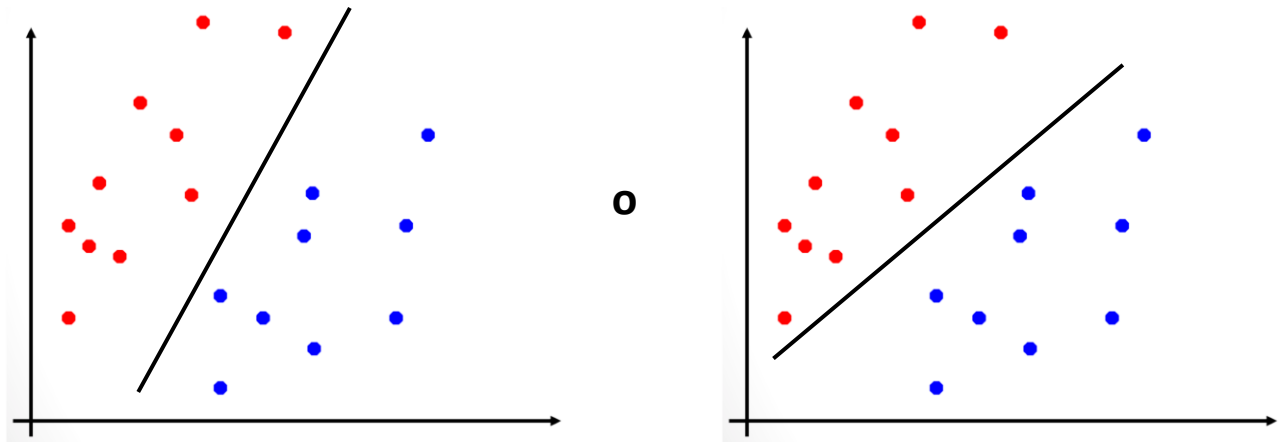
4. Máquinas de Soporte Vectorial (o SVM, *Support Vector Machines*)

4. Máquinas de Soporte Vectorial (o SVM, *Support Vector Machines*): Es un modelo de clasificación binaria cuyo **modelo básico es un clasificador lineal** definido en el espacio propio (*eigenspace*) con el intervalo más grande. Las **SVM también incluyen trucos de kernel que las convierten en clasificadores no lineales**. El algoritmo de aprendizaje de SVM es la solución óptima para la programación cuadrática convexa.

❑ ¿Cómo dividimos los conjuntos de datos rojo y azul por una línea recta?



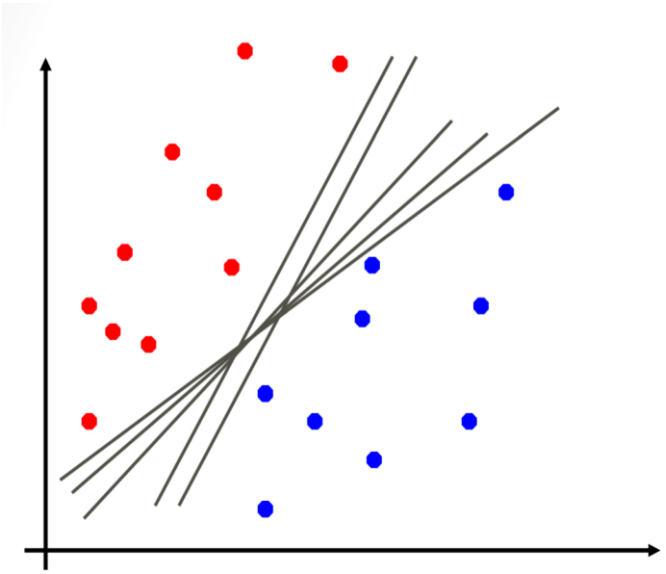
Con clasificación binaria
Conjunto de datos bidimensional

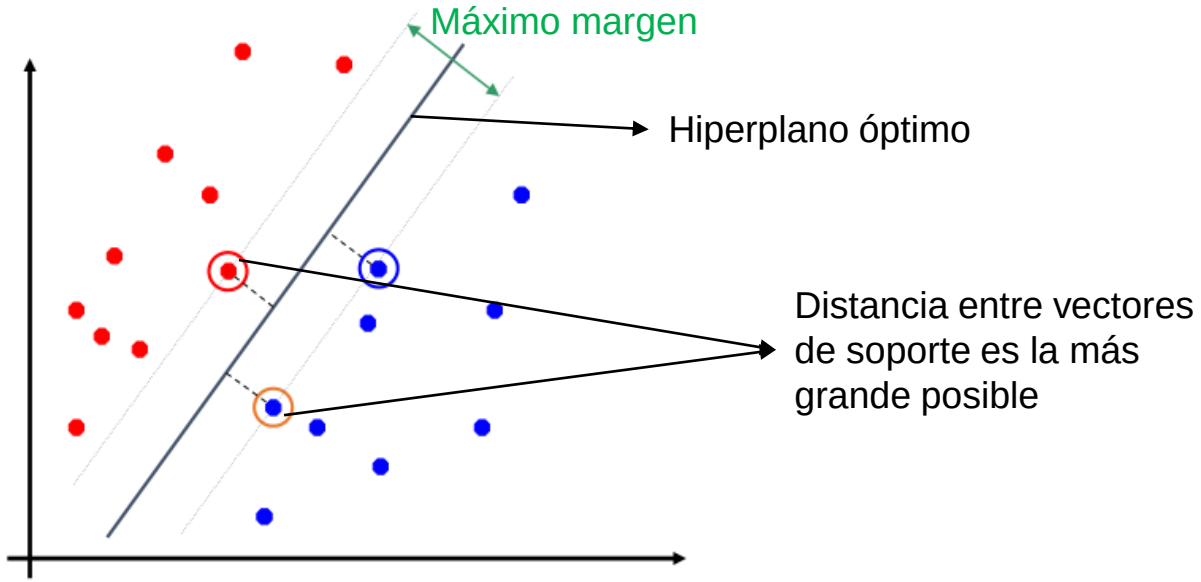


Tanto la recta de la izquierda como la de derecha se pueden utilizar para dividir los conjuntos de datos. ¿Cuál de ellas es el correcta?

4. Máquinas de Soporte Vectorial (o SVM, *Support Vector Machines*) (cont.)

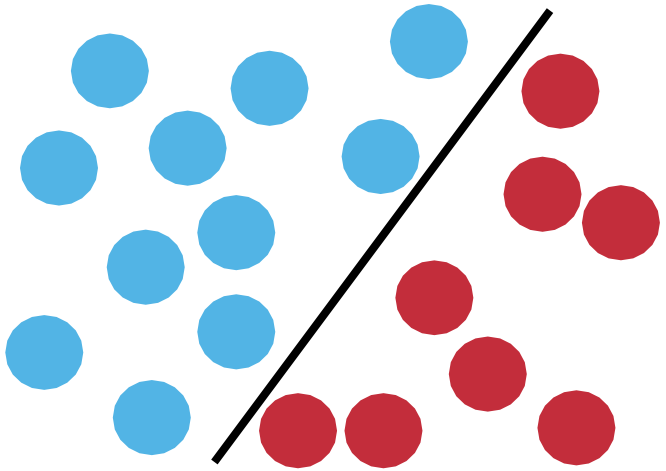
- Las líneas rectas se utilizan para dividir los datos en diferentes clases. En realidad, podemos usar varias líneas rectas para dividir los datos. La idea central de la SVM es encontrar una línea recta y mantener el punto más cercano de la línea recta lo más **lejos** posible. Esto puede permitir una fuerte capacidad de generalización del modelo. Estos puntos se llaman **vectores de apoyo o de soporte**.
- En el espacio bidimensional, usamos líneas rectas para la segmentación. En espacios de mayor dimensión, usamos **hiperplanos** para la segmentación.



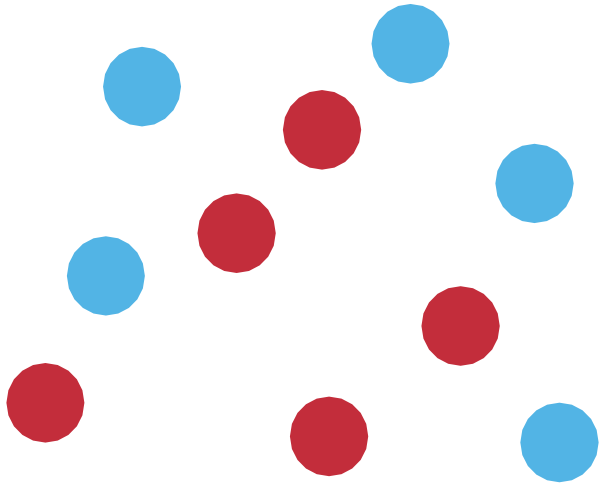


4. Máquinas de Soporte Vectorial (o SVM, *Support Vector Machines*) (cont.)

- ¿Cómo clasificamos un conjunto de datos separable no lineal?



La SVM lineal puede funcionar bien para conjuntos de datos lineales separables.

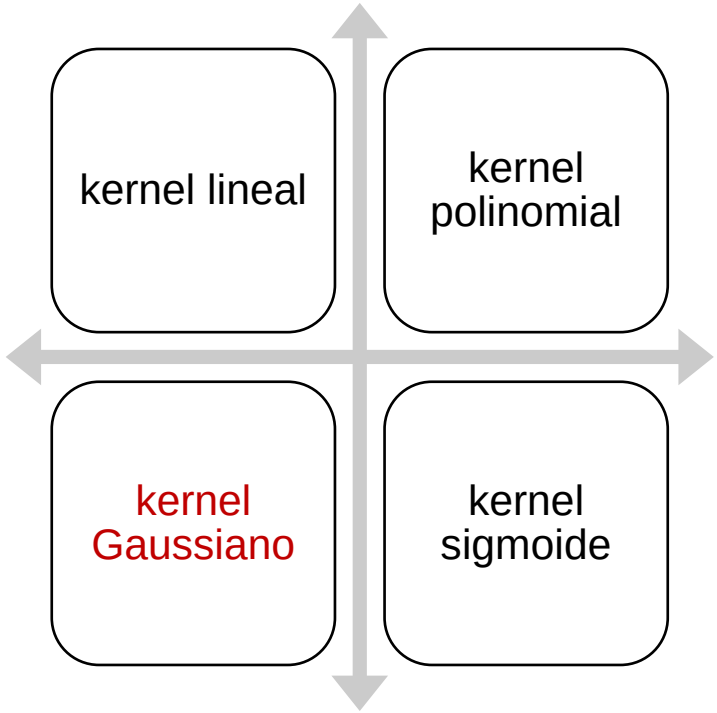


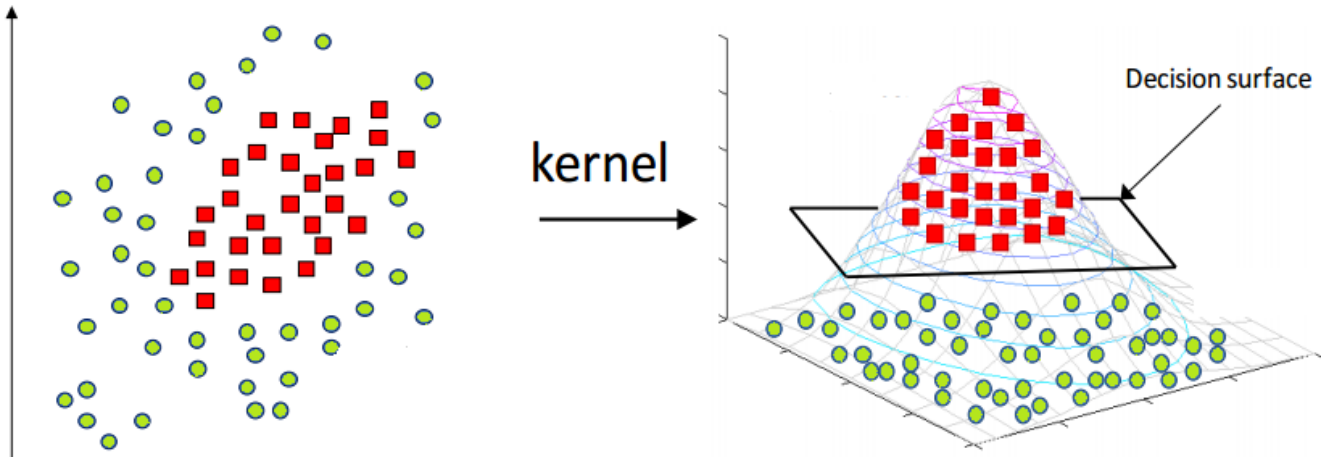
Los conjuntos de datos no lineales no se pueden dividir con una linea recta.

4. Máquinas de Soporte Vectorial (o SVM, *Support Vector Machines*) (cont.)

- Las funciones de kernel se utilizan para construir **SVM no lineales**.
- Estos kernel permiten que los algoritmos puedan ajustar un hiperplano en un espacio de características de alta dimensión transformado.

Típicas funciones de kernel





Espacio de entrada

Espacio de características de alta dimensión

5. Bayes Ingenuo (*Naive Bayes*)

5. Algoritmo Bayesiano Ingenuo: Un algoritmo simple de clasificación multi-clases basado en el teorema de Bayes. **Asume que las características son independientes entre sí**. Para una muestra determinada *X*, la probabilidad de que la muestra pertenezca a una categoría *C* es:

$$P(C_k \mid X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n \mid C_k) P(C_k)}{P(X_1, \ldots, X_n)}$$

- $X_1, \ldots, X_n$ son las características/*features* de las muestras de datos, que normalmente se describen mediante valores medidos de conjuntos de atributos m.
  - Por ejemplo, la característica de "color" puede tener tres atributos: rojo, amarillo y azul.
- C_k indica que los datos pertenecen a una categoría específica *k*.
- $P(C_k|X_1, \ldots, X_n)$ es una probabilidad posterior, o una probabilidad posterior de condición C_k.
- $P(C_k)$ es una probabilidad previa que es independiente de $X_1, \ldots, X_n$
- $P(X_1, \ldots, X_n)$ es la probabilidad priori de X.

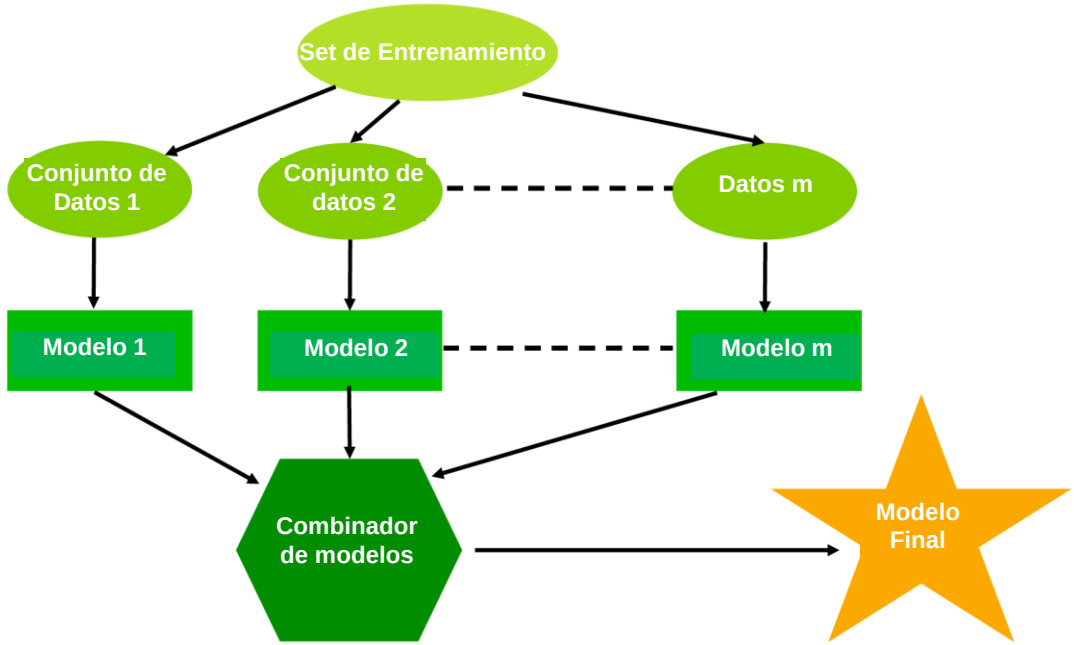
5. Bayes Ingenuo (*Naive Bayes*) (cont.)

- Asume independencia de las características.
  - Por ejemplo, si una fruta es roja, redonda y de unos 10 cm de diámetro, puede ser considerada una manzana.
  - Un clasificador de Bayes Ingenuo considera que cada característica contribuye independientemente a la probabilidad de que el fruto sea una manzana, independientemente de cualquier posible correlación entre el color, redondez y diámetro.

6. Métodos de Aprendizaje Combinado o en Conjunto (*Ensemble Learning*)

6. El aprendizaje en conjunto o combinado (*ensemble learning*) es un paradigma de *Machine Learning* en el que **se entrenan y combinan múltiples aprendices/alumnos para resolver el mismo problema**. Cuando se utilizan múltiples aprendices/alumnos, la capacidad de generalización integrada puede ser mucho más fuerte que la de un único aprendiz/alumno.

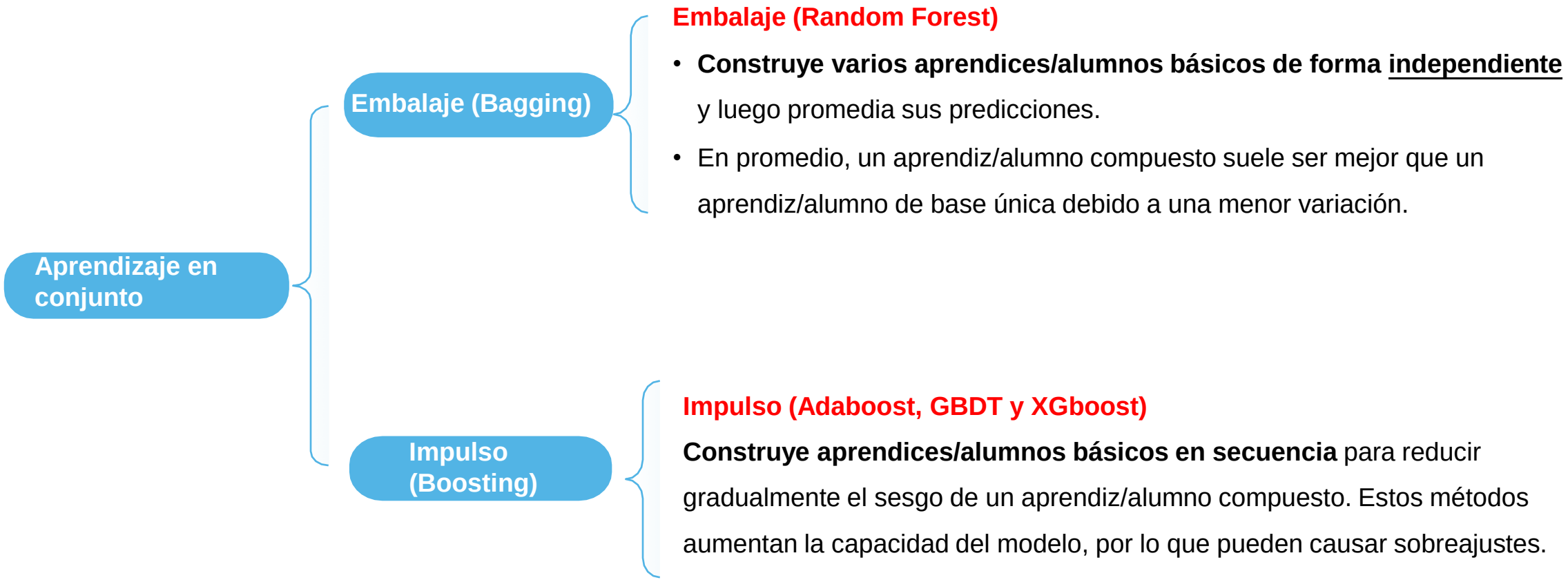
- Si se hace una pregunta compleja a miles de personas al azar y luego resume sus respuestas, la respuesta resumida es mejor que la respuesta de un experto en la mayoría de los casos. Esta es la sabiduría de las masas.



Ejemplo estrategia de aprendizaje combinado

6. Métodos de Aprendizaje Combinado o en Conjunto (*Ensemble Learning*) (cont.)

- Ejemplos de estas estrategias



Embalaje (Random Forest)

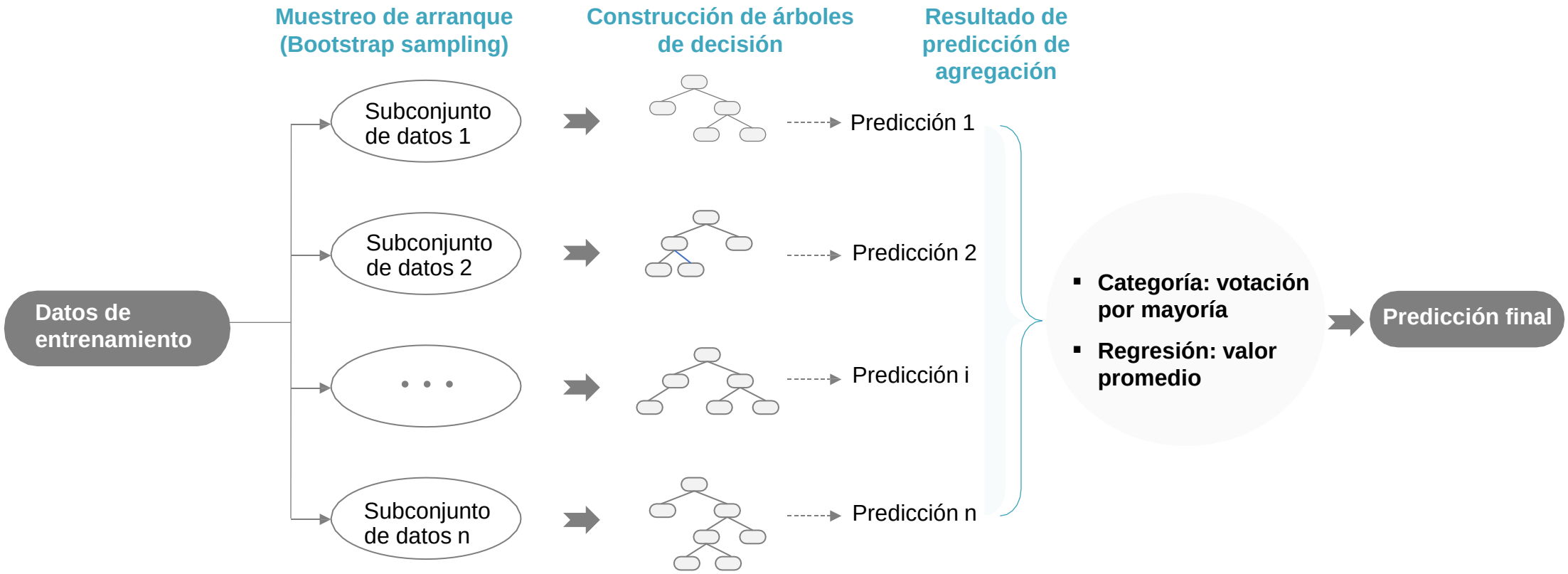
- **Construye varios aprendices/alumnos básicos de forma independiente** y luego promedia sus predicciones.
- En promedio, un aprendiz/alumno compuesto suele ser mejor que un aprendiz/alumno de base única debido a una menor variación.

Impulso (Adaboost, GBDT y XGboost)

Construye aprendices/alumnos básicos en secuencia para reducir gradualmente el sesgo de un aprendiz/alumno compuesto. Estos métodos aumentan la capacidad del modelo, por lo que pueden causar sobreajustes.

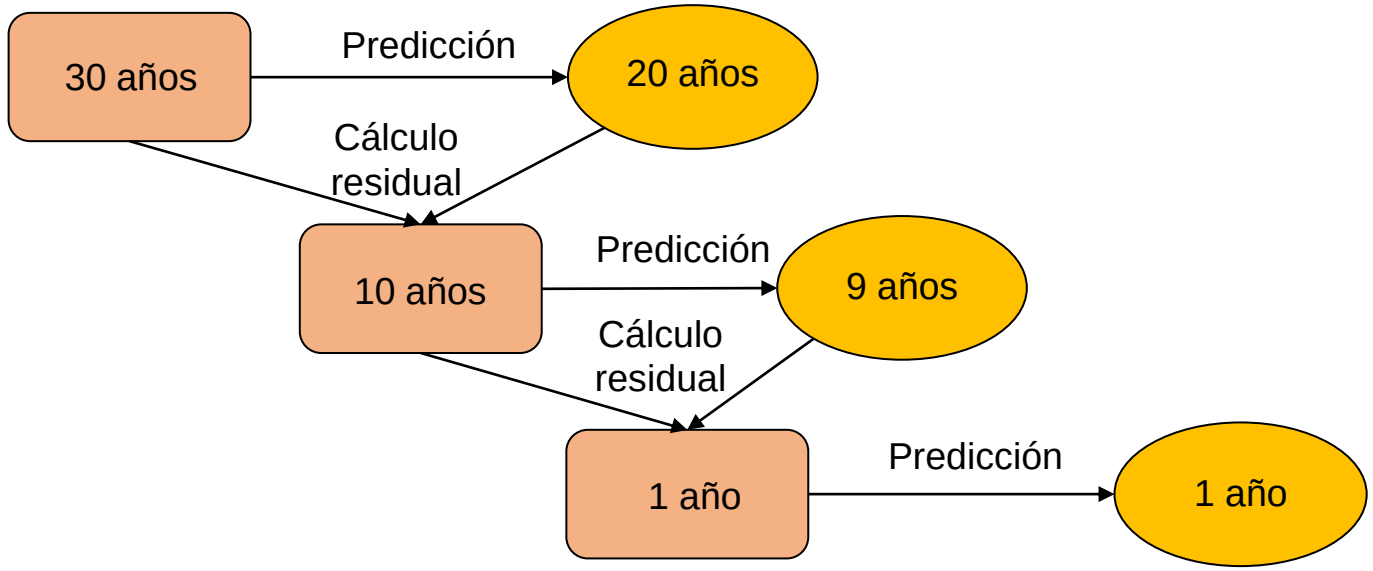
6. Métodos de Aprendizaje Combinado o en Conjunto (*Ensemble Learning*) (cont.)

- **Bosque aleatorio = Embalaje + árbol de decisiones CART**
- Los bosques aleatorios construyen múltiples árboles de decisión y los fusionan para hacer las predicciones más precisas y estables.
- Los bosques aleatorios pueden utilizarse tanto para problemas de clasificación como de regresión.



6. Métodos de Aprendizaje Combinado o en Conjunto (*Ensemble Learning*) (cont.)

- **GBDT (Gradient Boosting Decision Tree)** es un tipo de algoritmo de impulso (Boosting).
- El valor predicho es la suma de los resultados de todos los aprendices/alumnos básicos, en este caso árboles de decisión.
- La esencia de GBDT es usar nuevos árboles de decisión de forma secuencial y aprender desde los residuos de los árboles anteriores. Esto es, desde los errores entre los valores predichos y reales.
- Los datos de entrenamiento de cada árbol de decisión en GBDT dependen de la salida de los árboles de decisión anteriores.



Métricas de desempeño algoritmos de clasificación

Métricas de desempeño clasificadores

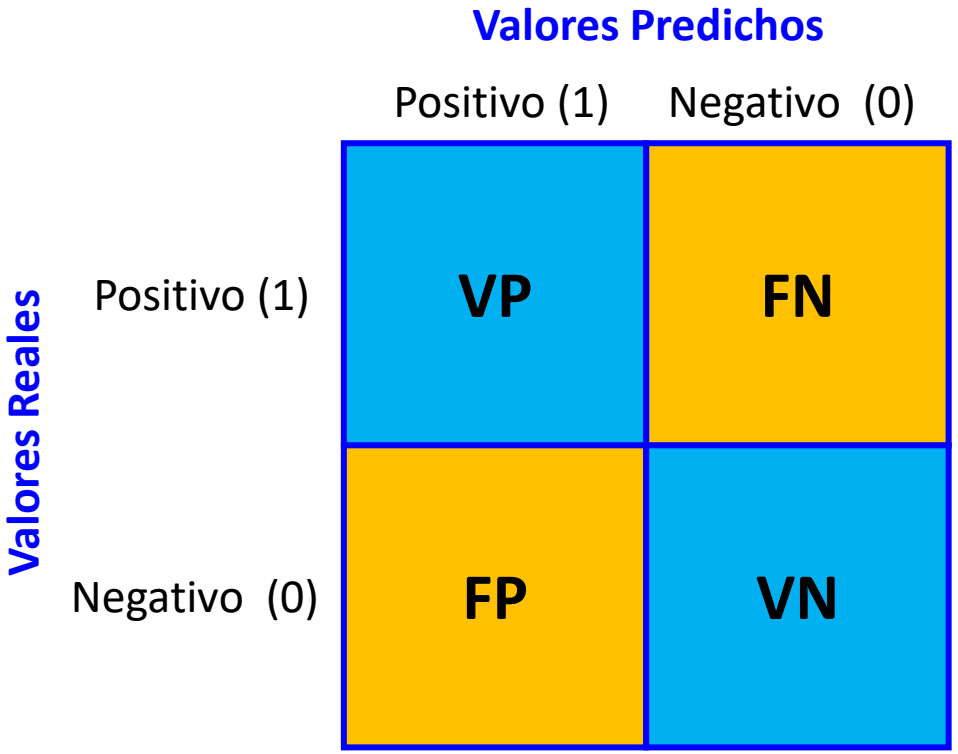
- Entre las estrategias para evaluar el desempeño de algoritmos de clasificación encontramos:

  1. Matriz de confusión
  2. Métricas de desempeño como: exactitud, precisión, otras.

Métricas de desempeño clasificadores: Términos y definiciones

- Antes de entrenar modelos de clasificación, cómo evaluaremos su efectividad?
- Partiremos por definir algunas métricas en función de datos obtenidos a partir de un problema de clasificación simple (binario) y del concepto de **matriz de confusión**:

- **P: Positivo**, número de casos reales positivos en los datos.
- **N: Negativo**, número de casos reales negativos en los datos
- **VP: Verdadero Positivo**, (positivo real), número de casos positivos correctamente clasificados.
- **VN: Verdadero Negativo**, (negativo real), número de casos negativos que son clasificados correctamente por el clasificador.
- **FP: Falso Positivo**, número de casos positivos que son clasificados incorrectamente por el clasificador (en la realidad eran negativos).
- **FN: Falso Negativo**, número de casos negativos que son clasificados incorrectamente por el clasificador (en realidad eran positivos).



Matriz de confusion 2x2, caso de clasificación binaria

Métricas de desempeño clasificadores: Términos y definiciones

❑ **Matriz de confusión**: es una tabla o matriz *CM* de al menos $m \times m$, donde m indica e número de clases involucrados. $CM_{i,j}$ indica el número de casos que realmente pertenecen a la clase i pero que están clasificados en la clase j por el clasificador.

Algunas métricas de desempeño típicas que aparecen:

- **Exactitud (Accuracy):**

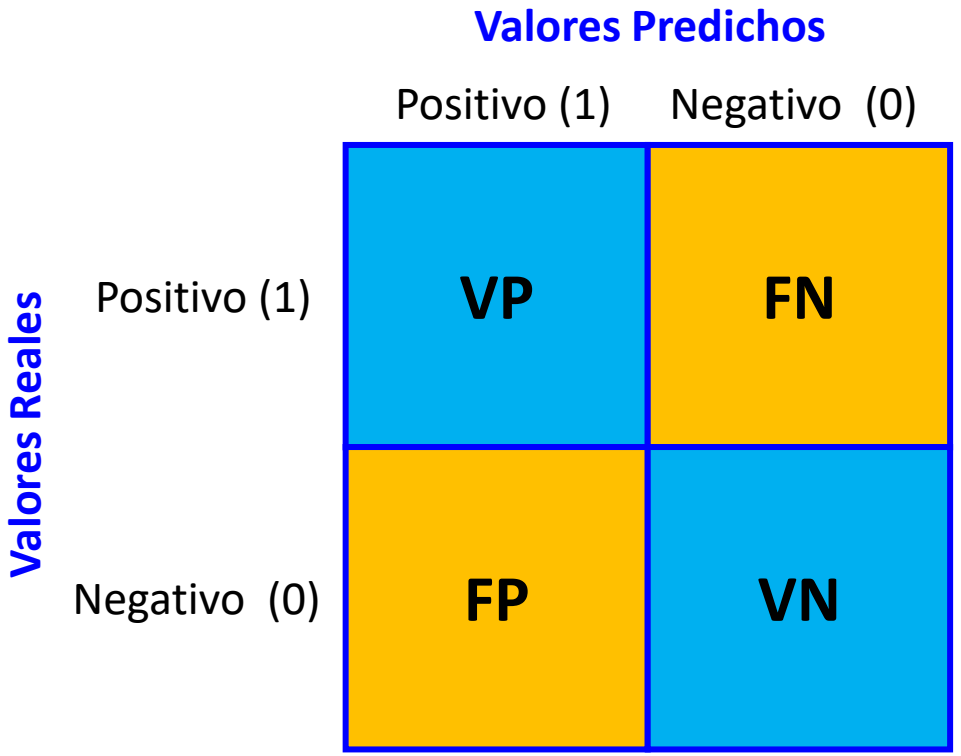
$$Exactitud = \frac{VP + VN}{VP + VN + FN + FP}$$

¿cuál es la proporción de predicciones correctas?

- **Error de clasificación o tasa de error:**

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} = 1 - ACC$$

¿cuál es la proporción de predicciones incorrectas?



Métricas de desempeño clasificadores: Términos y definiciones

Algunas otras métricas de desempeño típicas que aparecen:

- **Precisión**

$$Precisión = \frac{VP}{VP + FP}$$

¿qué proporción de predicciones positivas es correcta? **Apropiada cuando el foco es minimizar los FP**

- **Sensibilidad, tasa positiva real o Recall**

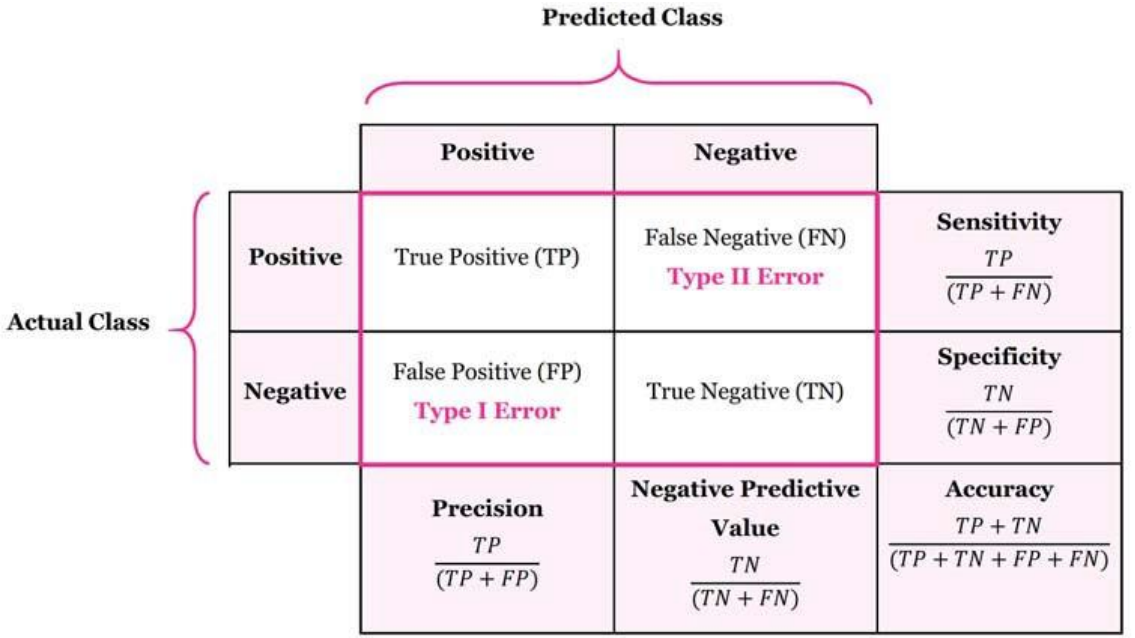
$$Sensibilidad = \frac{VP}{VP + FN}$$

¿qué proporción de los positivos reales se han predicho correctamente? **Apropiada cuando el objetivo es minimizar los FN**

- **Especificidad o tasa negativa real**

$$Especificidad = \frac{VN}{VN + FP}$$

¿qué proporción de los negativos reales se han predicho correctamente?



Fuente de imagen

Métricas de desempeño clasificadores

❑ F-1 Score o media armónica de la tasa de *Recall* y *Precisión*:

$$F{-}1\ score = 2 \cdot \frac{\text{Precisión} \cdot Recall}{Precisión + Recall}$$

- Sirve para resumir los valores de *Recall* y *Precisión* en una sola métrica, puede ayudar a comparar algoritmos en base a una métrica combinada cuando ni optimizar el *Recall* ni la *Precisión* son objetivos individuales. **Su mejor valor es 1 (mejor modelo) y 0 el peor caso**.
- No es tan intuitiva de entender como la exactitud pero es más útil cuando se tienen distribuciones de clases imbalanceadas (una clase tiene mucho más muestra que la otra). Si ambas clases son importantes, elegir el clasificador que entregue un valor más alto de F-1.
- Esta métrica se ha usado extensivamente en la literatura relacionada a procesamiento de lenguaje natural, aunque es un caso particular de la siguiente fórmula más general:

$$F{-}\beta\ score = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

Métricas de desempeño clasificadores

❑ Coeficiente de Correlación de Matthew (MCC):

- Fue formulado por primera vez en 1975 por Brian W. Matthews para evaluar el desempeño de predicciones de estructuras de proteínas secundarias.
- Puede ser entendido como un caso particular del coeficiente de correlación lineal (de Pearson) para un clasificador binario.
- Se considera especialmente útil en el caso de clases desbalanceadas.
- **Esta métrica toma valores entre -1 y 1. 0 (peor caso) indica predicciones aleatorias, +1 será el mejor caso y -1 indica una predicción inversa.**
- Es una métrica más completa que el F-1 score ya que también incluye las instancias negativas.

$$MCC = \frac{VP \cdot VN - FP \cdot FN}{\sqrt[2]{(VP + FP) \cdot (VP + FN) \cdot (VN + FP) \cdot (VN + FN)}}$$

Métricas de desempeño clasificadores

- **Ejemplo:** Hemos entrenado un modelo de M.L. para identificar si el objeto de una imagen es un gato. Ahora usamos 200 imágenes para verificar el desempeño del modelo. Entre las 200 imágenes, los objetos en 170 imágenes son gatos, mientras que otros no lo son.

- El resultado de identificación del modelo es que los objetos en 160 imágenes son gatos, mientras que otros no lo son. La matriz de confusión resume el resultado, luego algunas métricas que podemos calcular son:

Precisión: $P = \frac{VP}{VP+FP} = \frac{140}{140+20} = 87.5\%$

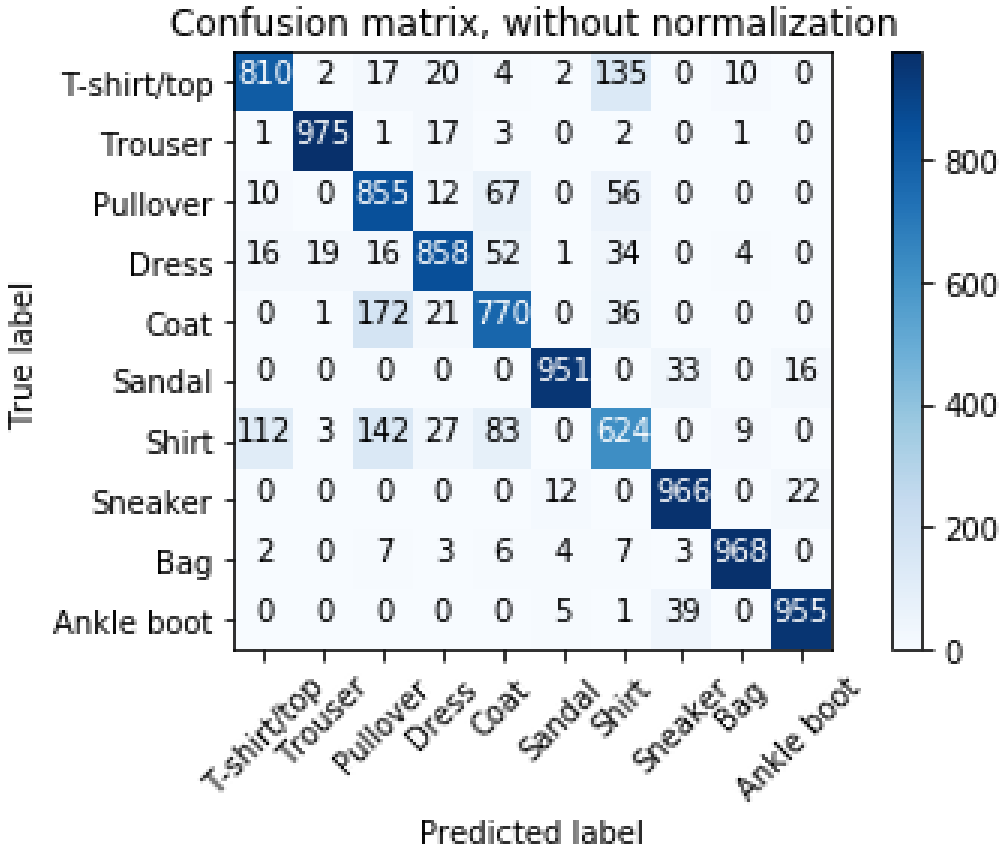
Recall: $R = \frac{VP}{P} = \frac{140}{170} = 82.4\%$

Exactitud: $ACC = \frac{VP+VN}{P+N} = \frac{140+10}{170+30} = 75\%$

Valor predicho / Valor real	Gatos	No Gatos	Total:
Gatos	140.	30.	170.
No Gatos	20.	10.	30.
Total:	160.	40	200.

Métricas de desempeño clasificadores

- **Idealmente, para un clasificador de alta exactitud, la mayoría de los valores de predicción deben ubicarse en la diagonal de $CM_{1,1}$ a $CM_{m,m}$ de la tabla**, mientras que los valores fuera de la diagonal son 0 o cerca de 0. Es decir, FP y FP son cerca de 0.
- Por ejemplo, la matriz de confusión de la derecha fue generada para un problema de clasificación de 10 tipos de vestimentas a partir de imágenes (problema clasificación multiclase).
- En esta forma de visualizar la matriz de confusión, la diagonal principal representa los valores correctamente clasificados, por lo tanto, en el mapeo de intensidades de colores, debiera ser más intenso.

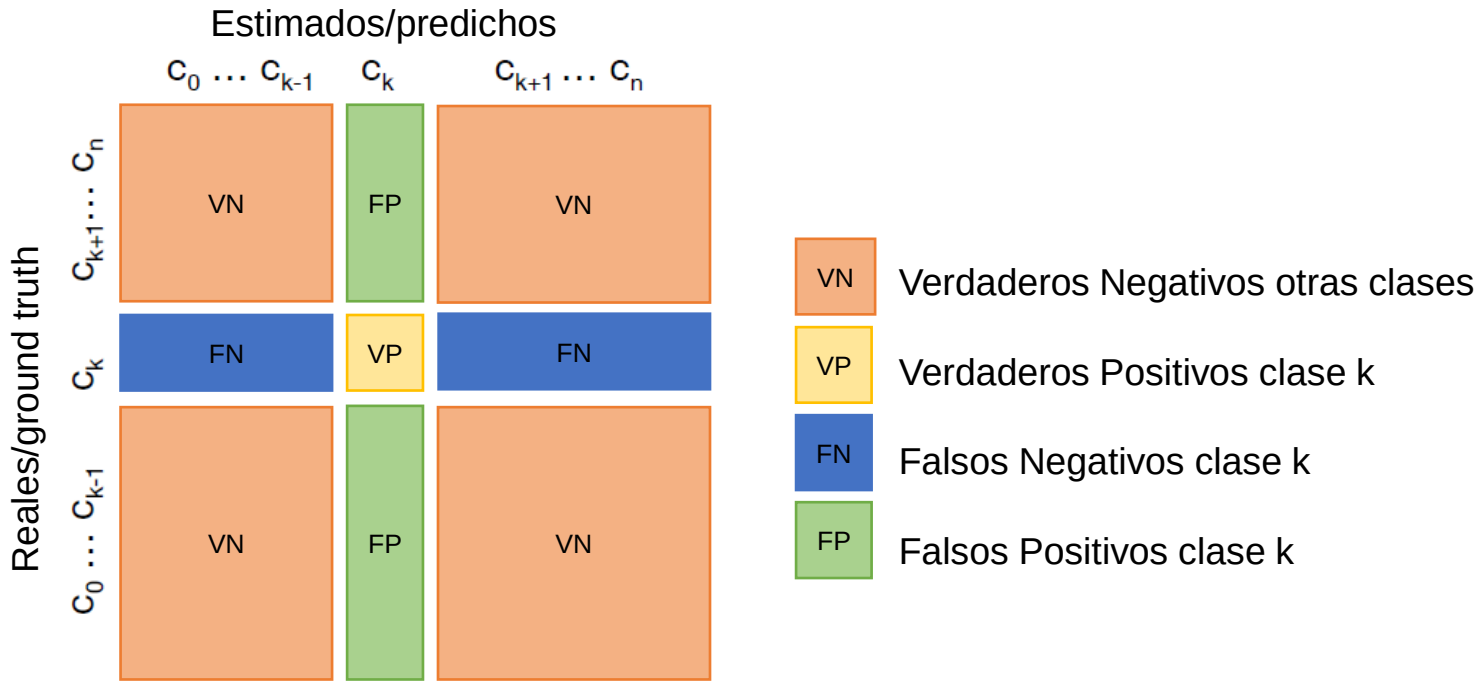


Este ejemplo lo veremos en prácticas posteriores.

Métricas de desempeño clasificadores

❑ Cómo podemos evaluar las métricas en el caso anterior de un problema de clasificación multiclase.

- La estrategia consiste en construir una matriz de confusión para cada clase para facilitar cálculos individuales para cada una y tener una mejor visualización.



Métricas de desempeño clasificadores

- **Luego, se puede resumir una métrica combinada a partir de los datos obtenidos siguiendo una estrategia de promedio micro o macro:**

- **Promedio micro (*micro average*):** Se obtiene usando los VP, VN, FP y FN obtenidos de cada clase sacados de las matrices de confusión separadas, se prefiere en caso de que las clases estén balanceadas:

$$PRE_{micro} = \frac{VP_1 + \cdots + VP_k + \cdots + VP_n}{VP_1 + \cdots + VP_n + FP_1 + \cdots + FP_n}$$

- **Promedio macro (*macro average*):** se promedian los valores de las métricas de desempeño obtenidas en cada clase, son un mejor indicio para el caso de datos con clases desbalanceadas. Ej. para la precisión sería:

$$PRE_{macro} = \frac{PRE_1 + \cdots + PRE_k + \cdots + PRE_n}{n}$$

- La exactitud (*accuracy*) se puede calcular directamente desde la matriz de confusión original o también por cada clase.

Métricas de desempeño clasificadores

- **Métrica promedio ponderada (*weighted average*)**: Similar a la métrica promedio pero incluyendo pesos que ponderan las métricas de cada clase, los pesos se calculan con la cantidad de datos que contiene cada clase dividido por el total de datos. ej. para la precisión sería:

$$PRE_{wa} = \frac{\#Datos\_C_1 \cdot PRE_1 + \cdots + \#Datos\_C_n \cdot PRE_n}{Total\ de\ Datos}$$

Métricas de desempeño clasificadores

❑ **Ejemplo,** la siguiente matriz de confusión muestra un problema de clasificación de 3 clases, calcular la precisión micro y macro.

		Valores predichos		
		Clase 0	Clase 1	Clase 2
Valores reales	Clase 0	8	10	1
	Clase 1	5	60	50
	Clase 2	3	30	200

II. Matriz de confusión resumida tomando en cuenta todos los VP, VN, FP, FN por clase:

		Predichos	
		Clase +	Clase -
Reales	Clase +	268	99
	Clase -	99	635

$$PRE_{micro} = \frac{268}{268+99} = 0.73$$

I. Primero dividamos la matriz de confusión en 3, una para cada clase, y obtengamos los valores de precisión individuales:

		Predichos	
		Clase 0	Otras
Reales	Clase 0	8	11
	Otras	8	340

$$PRE_1 = \frac{8}{8+8} = 0.5$$

		Predichos	
		Clase 1	Otras
Reales	Clase 1	60	55
	Otras	40	212

$$PRE_2 = \frac{60}{60+40} = 0.6$$

		Predichos	
		Clase 2	Otras
Reales	Clase 2	200	33
	Otras	51	83

$$PRE_3 = \frac{200}{200+51} = 0.80$$

$$PRE_{macro} = \frac{0.5+0.6+0.80}{3} = 0.63$$

Métricas de desempeño clasificadores

❑ **Ejemplo (cont.),** en scikit-learn, podemos ocupar la función `classification_report`, veamos un ejemplo:

```
[ ]  from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
     import matplotlib.pyplot as plt
```

Simulemos algunos datos que indiquen predicciones realizadas por un clasificador cualquiera y los datos reales disponibles

```
[ ]  # Classes
     C = "Cat"
     D = "Dog"
     W = "Wolf"
     target_names = [C,D,W]


     # Valores reales
     y_true = [C,C,D,D,D,C,W,W,C,D,C,C,C,C,D,C,W,C,D,C,D,D,W,D,C,W,W,W,D,C,D,W,D,W,W,D,C,D,C]

     # Valores predichos
     y_pred = [C,C,C,D,D,C,D,C,C,D,W,C,C,D,D,C,C,W,D,D,D,W,W,C,C,W,C,W,W,C,D,C,D,W,W,D,C,C,C]
```

Métricas de desempeño clasificadores

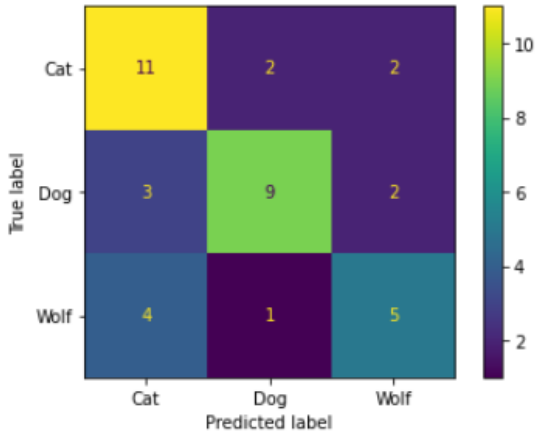
Matriz de confusión

```
[3]  cm = confusion_matrix(y_true,y_pred,labels = target_names)
     print('La matriz de confusión es: \n',cm)
```

```
La matriz de confusión es:
 [[11  2  2]
 [ 3  9  2]
 [ 4  1  5]]
```

También podemos desplegar la matriz de confusión usando matplotlib

```
[4]  disp = ConfusionMatrixDisplay(confusion_matrix=cm,  display_labels=target_names)
     disp.plot()
     plt.show()
```



Preguntas: qué hace la versión `ConfusionMatrixDisplay.from_estimator`, y el parámetro `normalize`

Métricas de desempeño clasificadores

Ahora mostremos un reporte de métricas para el ejemplo anterior:

```
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

         Cat       0.61      0.73      0.67        15
         Dog       0.75      0.64      0.69        14
        Wolf       0.56      0.50      0.53        10

    accuracy                           0.64        39
   macro avg       0.64      0.63      0.63        39
weighted avg       0.65      0.64      0.64        39
```

Las métricas micro no se despliegan dado que entregan los mismos resultados (comprobar), igualmente, si quisieramos estiamr las métricas por separado según se requiera, lo podemos hacer con las funciones individuales que proporciona `scikit-learn`:

```
[ ] from sklearn.metrics import precision_score, recall_score, f1_score
    print("Precision (micro): %f" % precision_score(y_true, y_pred, average='micro'))
    print("Recall (micro):    %f" % recall_score(y_true, y_pred, average='micro'))
    print("F1 score (micro):  %f" % f1_score(y_true, y_pred, average='micro'), end='\n\n')
    print("Precision (macro): %f" % precision_score(y_true, y_pred, average='macro'))
    print("Recall (macro):    %f" % recall_score(y_true, y_pred, average='macro'))
    print("F1 score (macro):  %f" % f1_score(y_true, y_pred, average='macro'), end='\n\n')
    print("Precision (weighted): %f" % precision_score(y_true, y_pred, average='weighted'))
    print("Recall (weighted):    %f" % recall_score(y_true, y_pred, average='weighted'))
    print("F1 score (weighted):  %f" % f1_score(y_true, y_pred, average='weighted'))
```

- Comprobar lo de arriba ejecutando la última porción de código.

Referencias

[1] T. Karras, S. Laine, T. Aila, A Style-Based Architecture for Generative Adversarial Networks, arXiv:1812.04948, 2019.

[2] Apuntes Curso: Convolutional Neural Networks for Visual Recognition, Standford University, Disponible Online: link

- Parte del material de este curso está basado en:
  - Certificación Huawei HCIA-AI
  - Documentación oficial Matlab, Scikit Learn, Tensorflow y Keras.