



ARBOLES DE DECISIÓN

APRENDIZAJE AUTOMÁTICO - CEIOT - FIUBA

Dr. Ing. Facundo Adrián Lucianna

REPASO CLASE ANTERIOR

- Clasificación
- Regresión logistica
- KNN
- Métricas de evaluación

CLASIFICACIÓN

Es mas común encontrarnos con problema de clasificación que de regresión:

- Una persona llega a una guardia con un set de síntomas atribuidos a una de tres condiciones medicas.
- Un servicio de banca online debe determinar si una transacción en el sitio es fraudulenta o no, usando como base la dirección IP, historia de transacciones, etc.
- En base a la secuencia de ADN de un numero de pacientes con y sin una enfermedad dada, un genetista debe determinar que mutaciones de ADN genera un efecto nocivo relacionado a la enfermedad o no.

REGRESIÓN LOGISTICA

En la gráfica se observa el problema de predecir usando regresión lineal. Dada la naturaleza de la función, hay valores en donde se obtienen $p(x) < 0$, o $p(x) > 1$. Esto va a ocurrir con cualquier regresión que de valores por fuera a 0 y 1.

Para evitar este problema, se debe modelar a $p(x)$ usando una función que nos asegure que siempre tendremos valores entre 0 y 1.

En **regresión logística**, esto lo resolvemos usando una función sigmoide:

$$p(x) = \frac{e^{w_0 + w_1 x_1}}{1 + e^{w_0 + w_1 x_1}}$$

REGRESIÓN LOGISTICA

En la gráfica se observa el problema de predecir usando regresión lineal. Dada la naturaleza de la función, hay valores en donde se obtienen $p(x) < 0$, o $p(x) > 1$. Esto va a ocurrir con cualquier regresión que de valores por fuera a 0 y 1.

Para evitar este problema, se debe modelar a $p(x)$ usando una función que nos asegure que siempre tendremos valores entre 0 y 1.

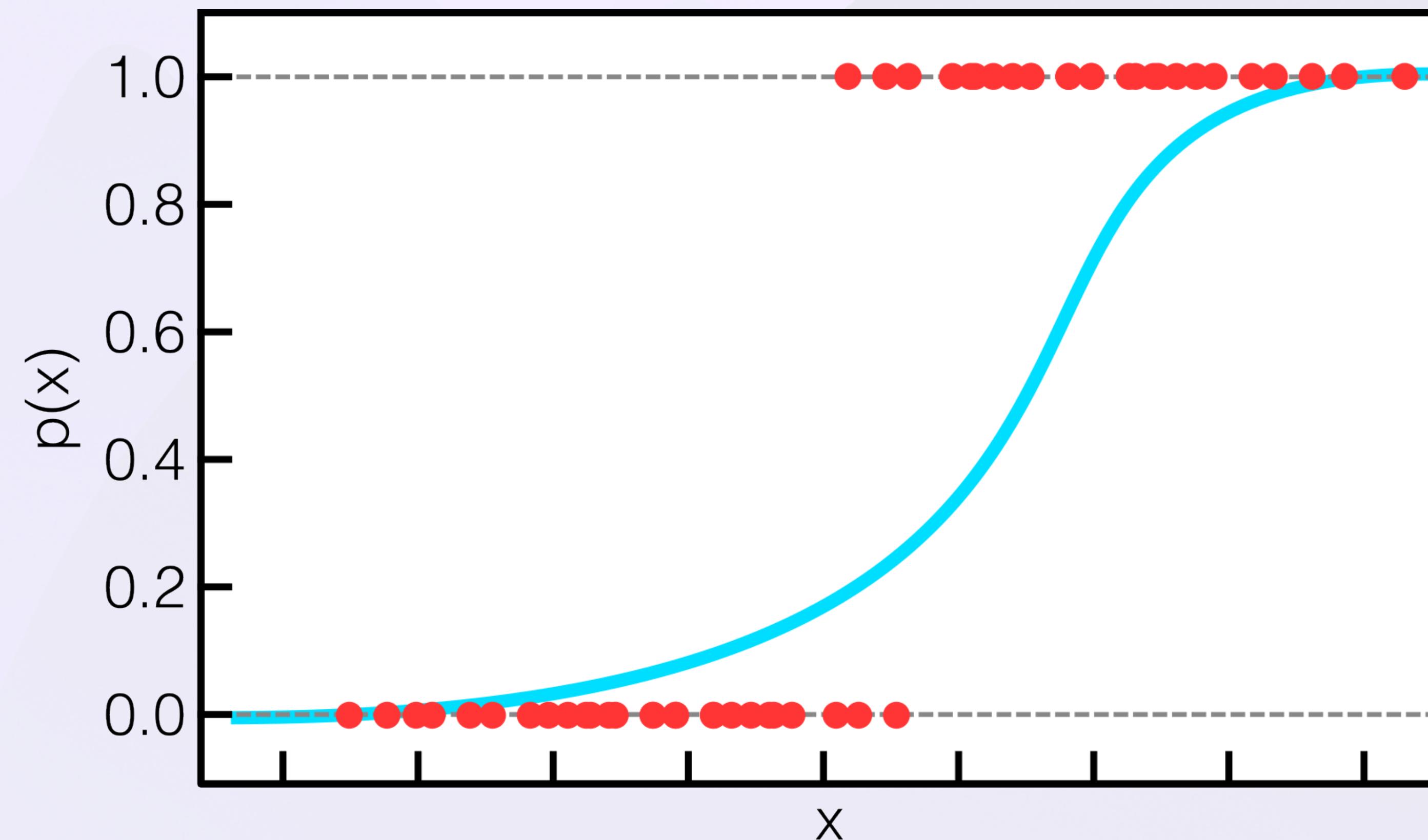
En **regresión logística**, esto lo resolvemos usando una función sigmoide:

$$p(\hat{X}) = \frac{e^{w_0 + w_1 x_1 + \dots + w_n x_n}}{1 + e^{w_0 + w_1 x_1 + \dots + w_n x_n}}$$

$$P(Y = k | X = x) = \frac{e^{w_{k0} + w_{k1} x_1 + \dots + w_{kn} x_n}}{1 + \sum_{l=1}^{K-1} e^{w_{l0} + w_{l1} x_1 + \dots + w_{ln} x_n}}$$

$$P(Y = K | X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{w_{l0} + w_{l1} x_1 + \dots + w_{ln} x_n}}$$

REGRESIÓN LOGISTICA



REGRESIÓN LOGISTICA - AJUSTE

Matemáticamente la **función de verosimilitud** es:

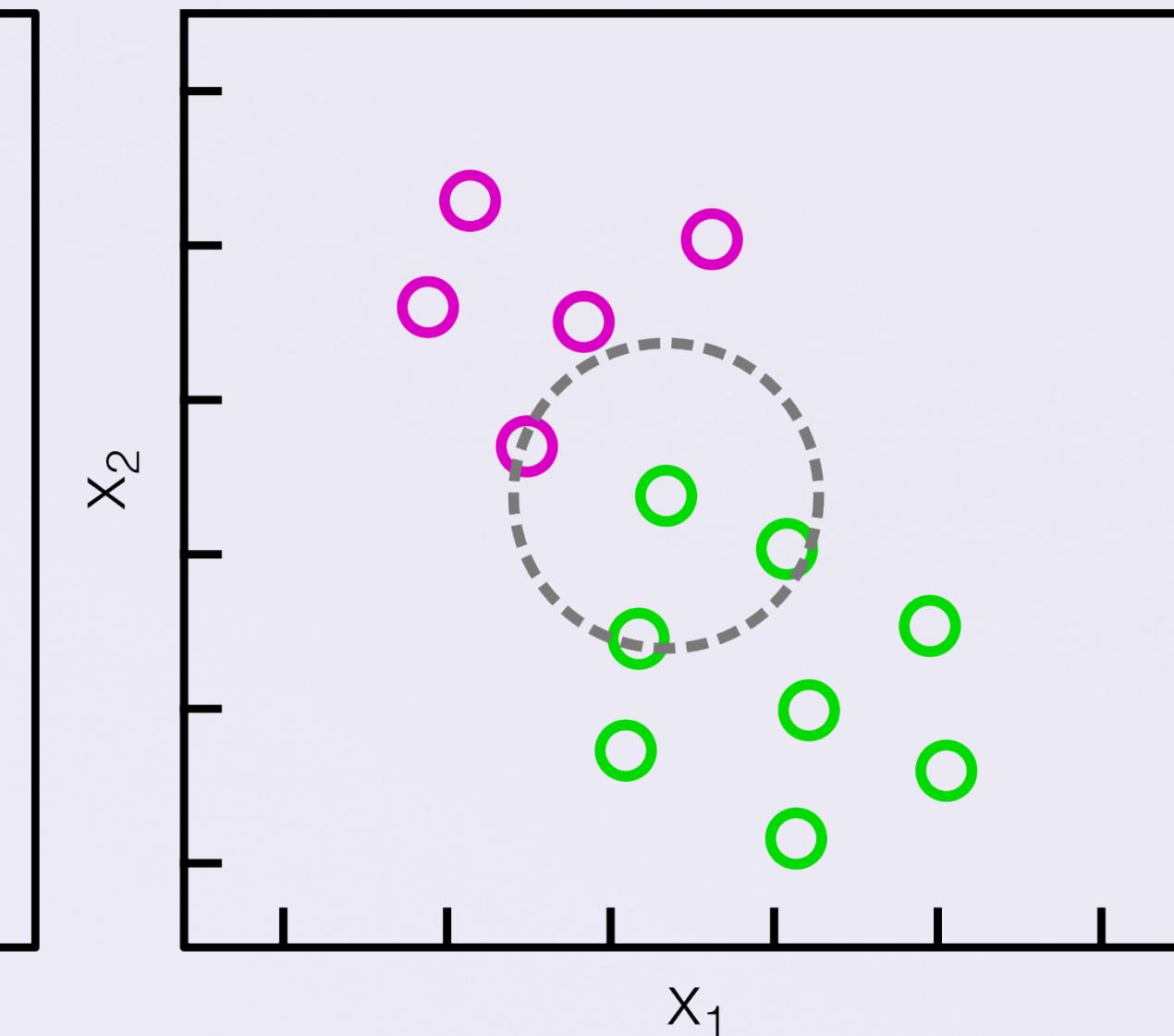
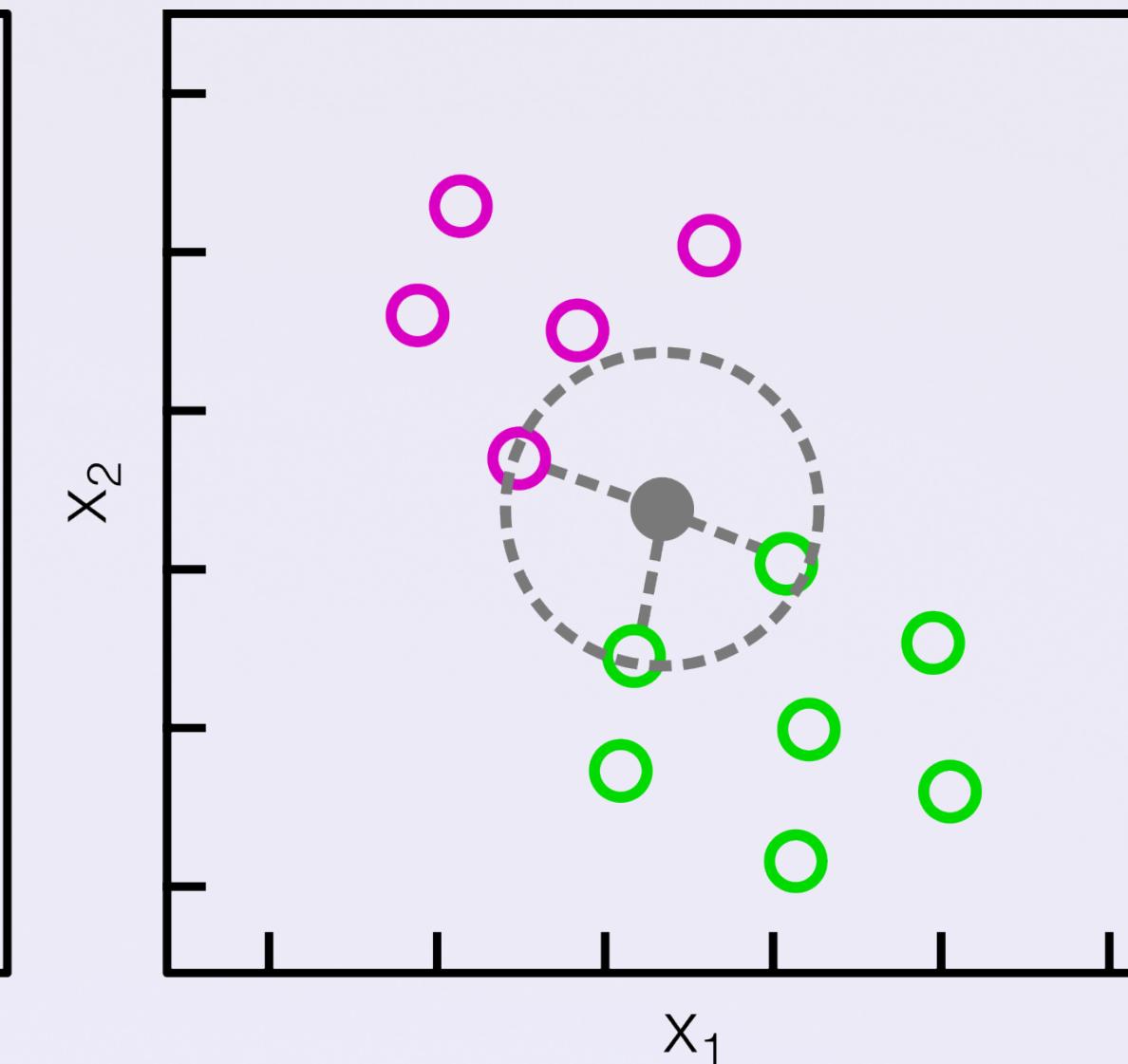
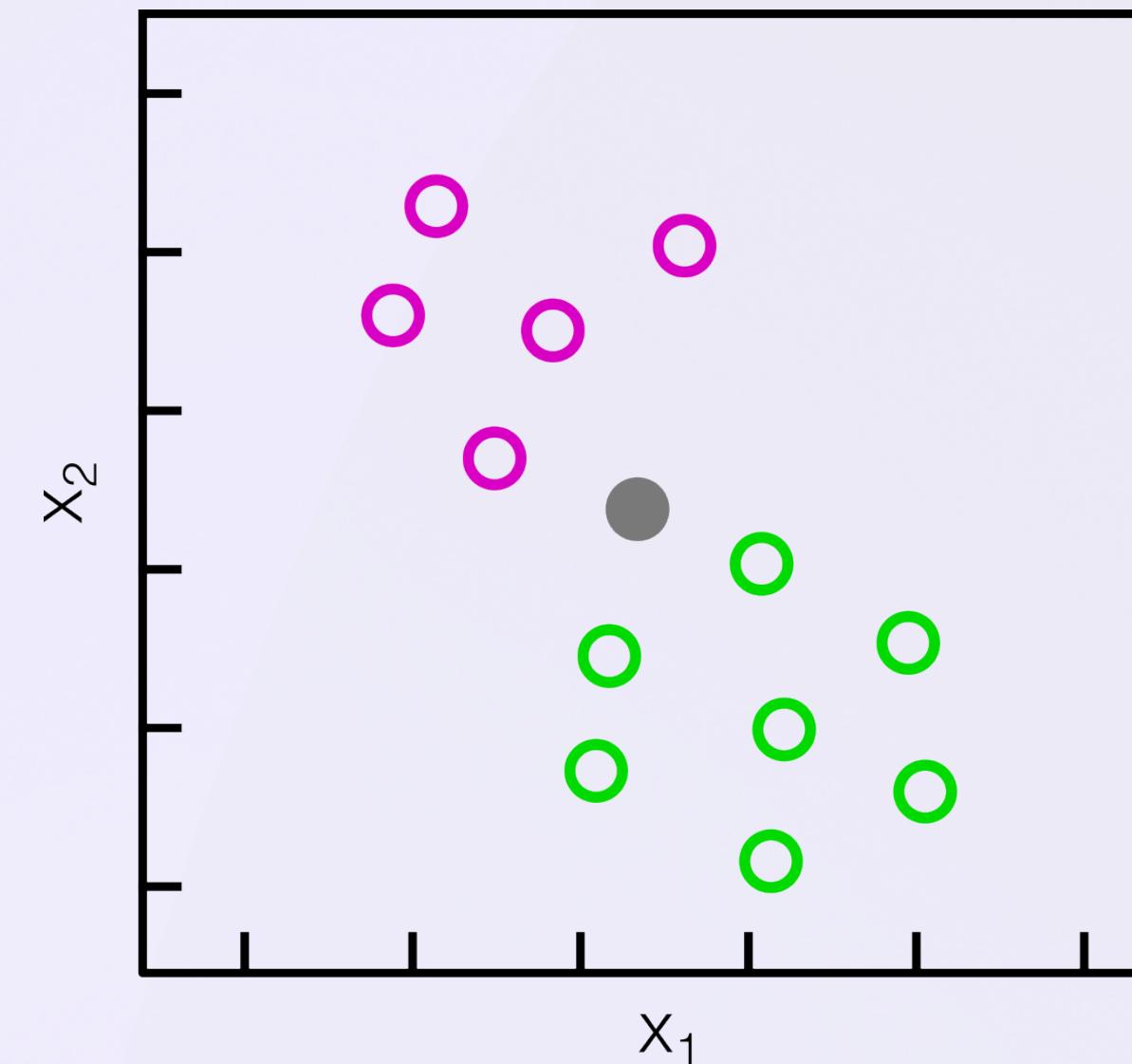
$$\ell(w_0, w_1) = \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=0} (1 - p(x_j))$$

Y buscamos encontrar los valores de w_0 y w_1 que hacen el valor máximo de esta función.

KNN

El clasificador de k vecinos más cercanos (KNN o k-NN), es un algoritmo que utiliza la proximidad de sus vecinos para hacer clasificaciones sobre la agrupación de un punto.

La idea se basa de la **suposición** de que se pueden encontrar puntos similares cerca uno del otro en base a votación de pluralidad (se elige la clase en función de la moda de la clase de sus vecinos).



MÉTRICAS DE EVALUACIÓN

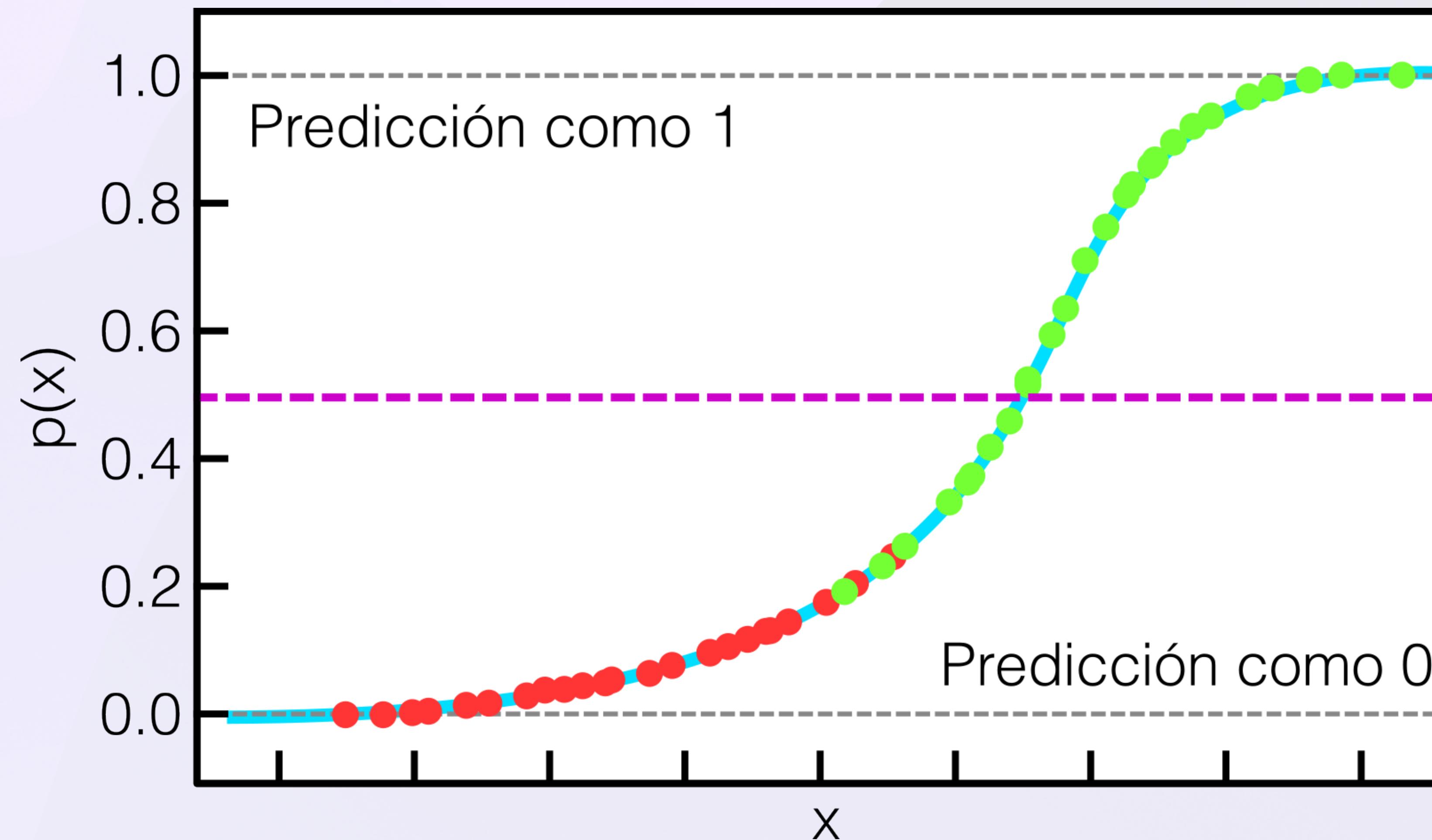
MATRIZ DE CONFUSIÓN

		Valores actuales	
		1	0
Predicción	1	Verdadero positivo (TP)	Falso positivo (FP)
	0	Falso negativo (FN)	Verdadero negativo (TN)

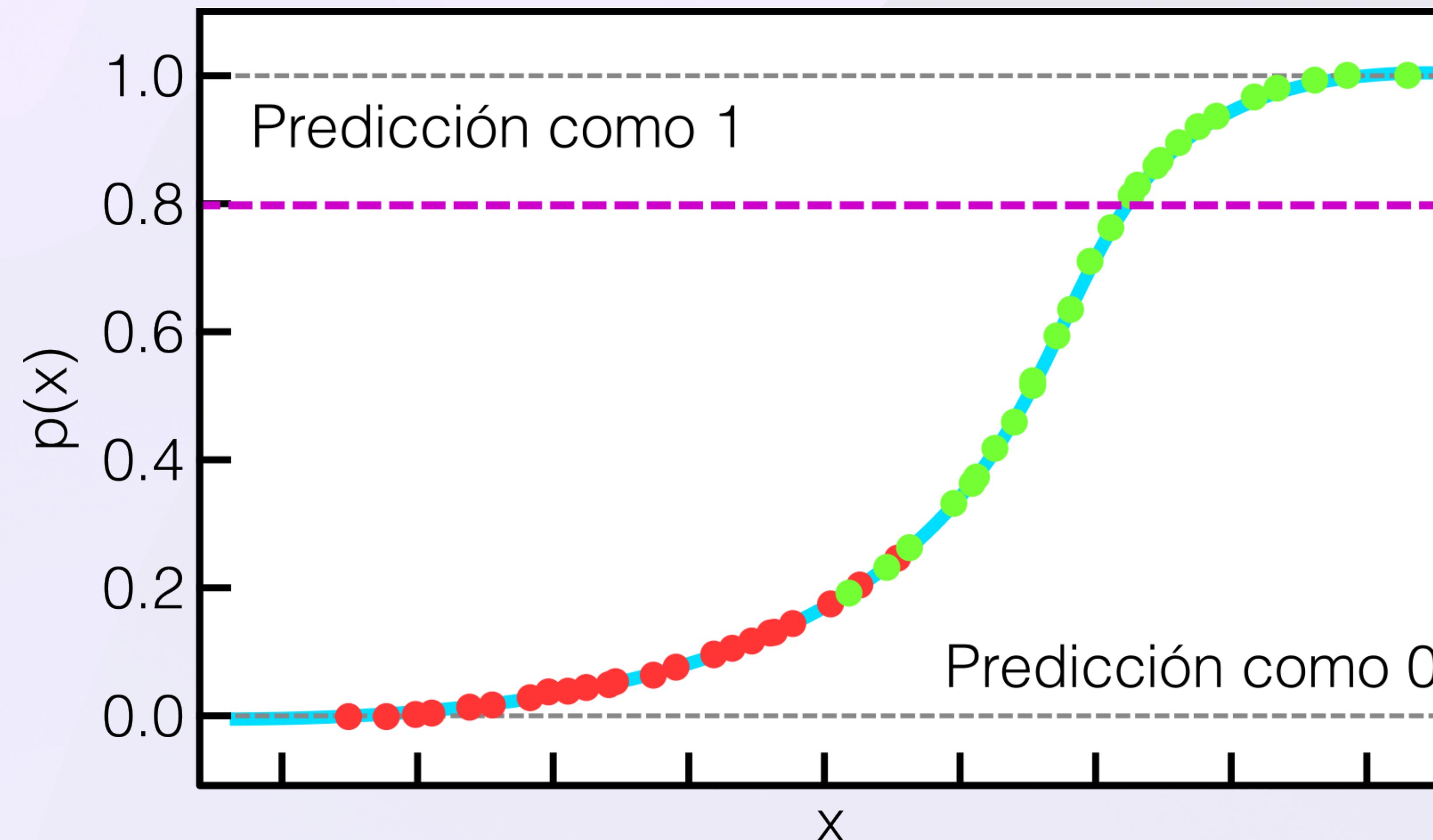
MÉTRICAS DE EVALUACIÓN

- **Sensibilidad:** $TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - FNR$
- **Especificidad:** $TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = 1 - FPR$
- **Exactitud:** $ACC = \frac{TP + TN}{P + N}$
- **Exactitud balanceada:** $BA = \frac{TPR + TNR}{2}$
- **Precisión:** $\text{Precision} = \frac{TP}{TP + FP}$
- **Recuperación:** $\text{Recall} = \frac{TP}{TP + FN}$
- **F1-score o F β -score:** $F_{\beta} = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$

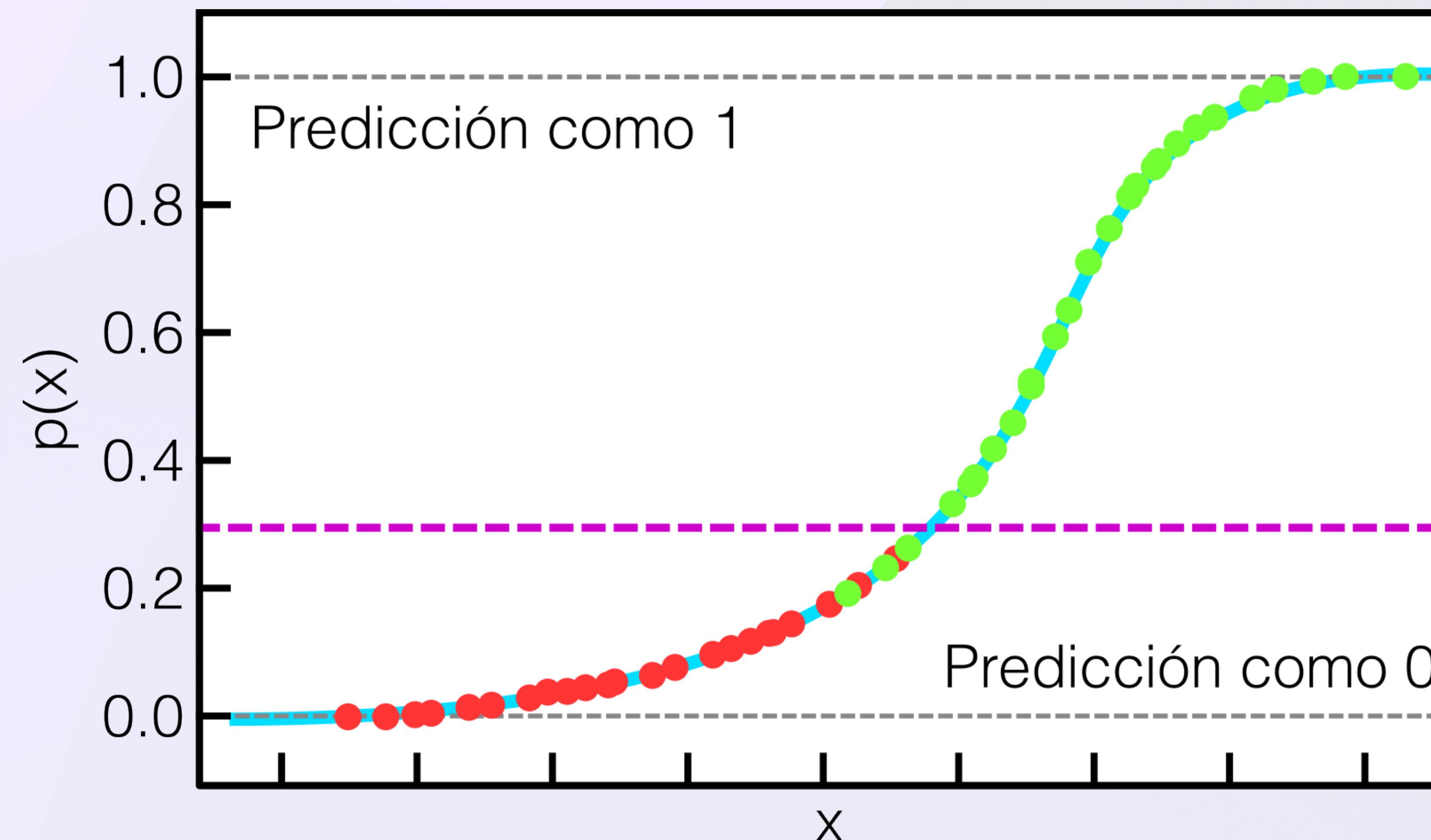
CURVA ROC



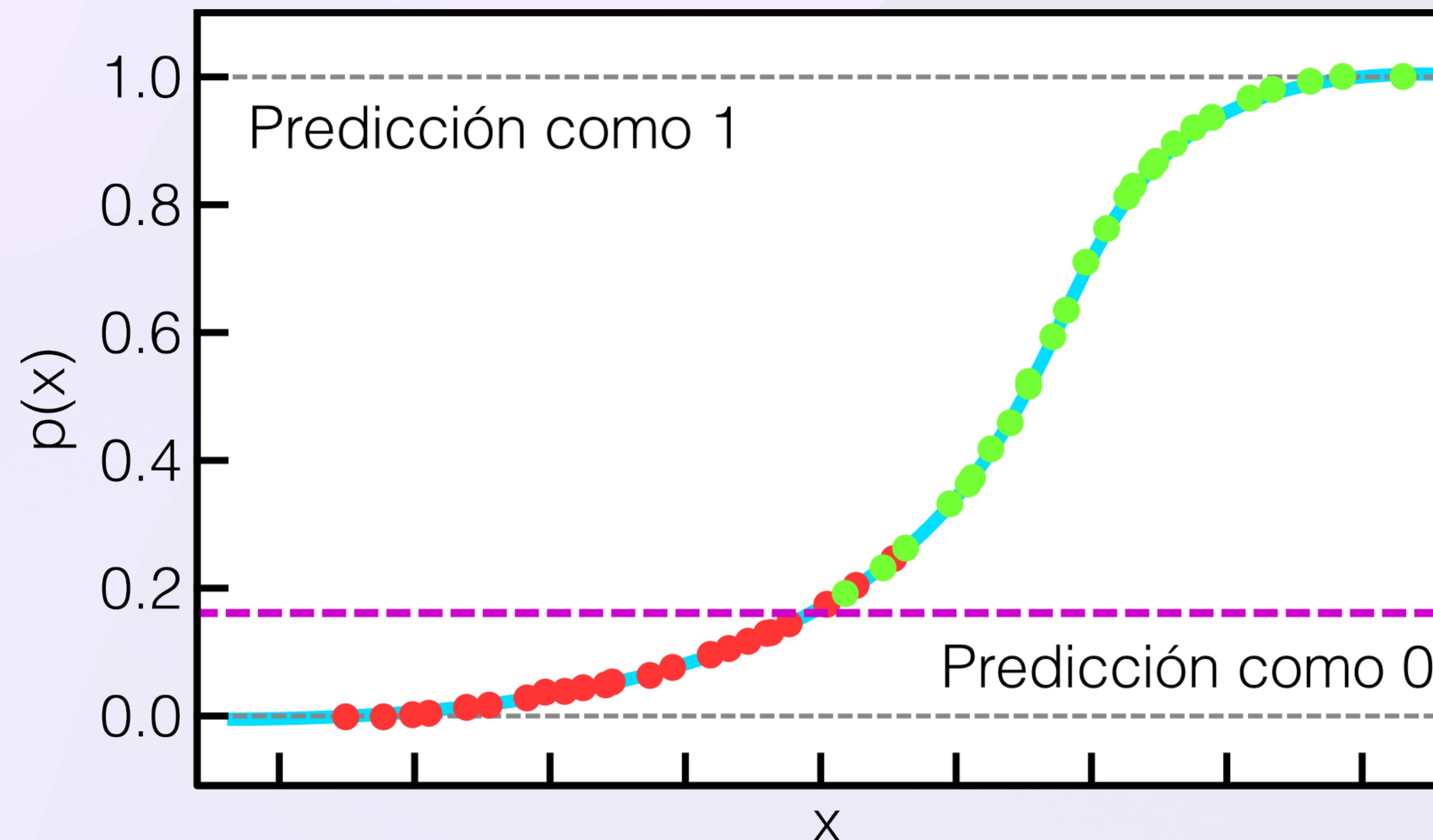
CURVA ROC



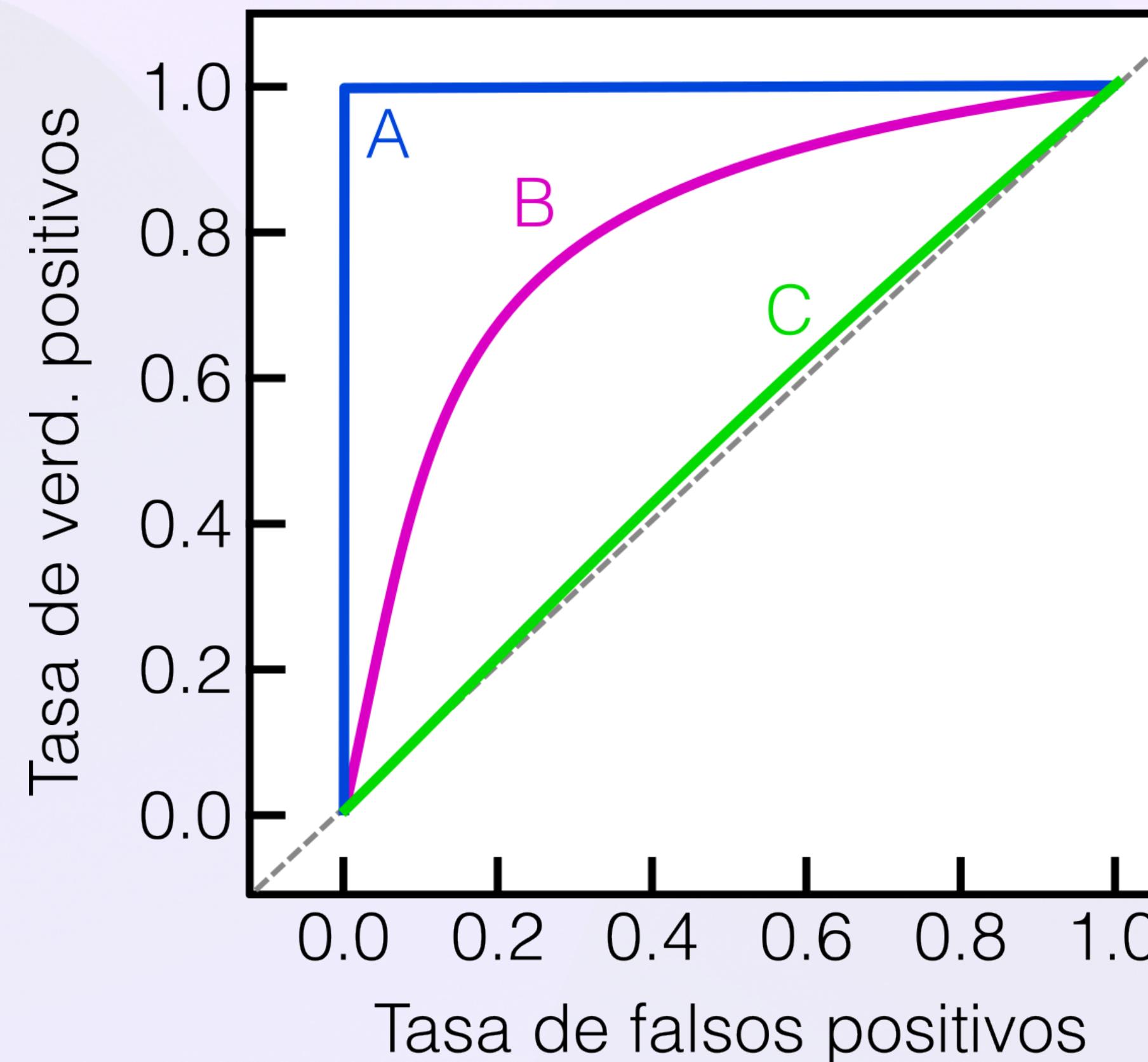
CURVA ROC



CURVA ROC



CURVA ROC



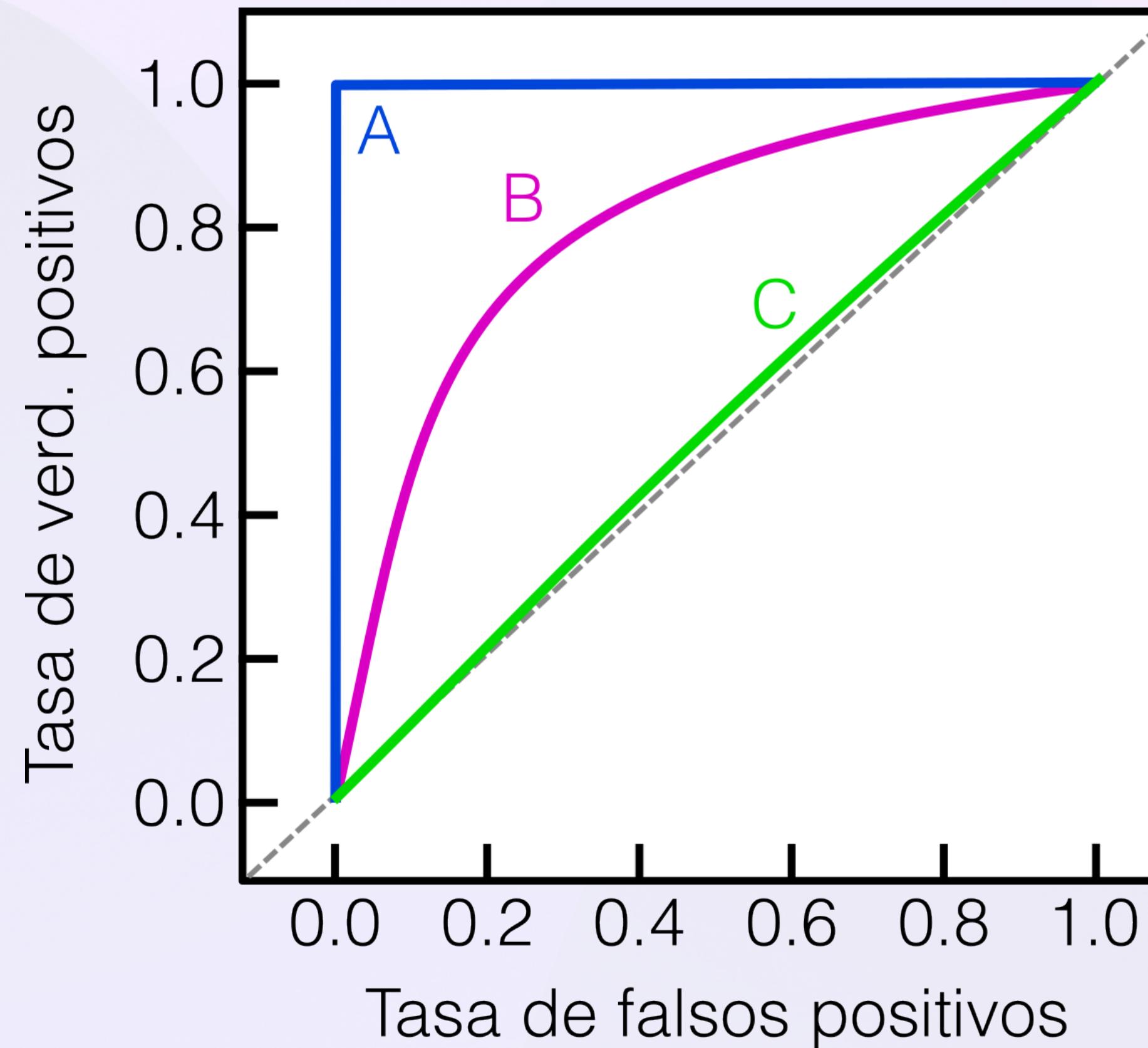
Siempre se arranca de umbral 1, donde la TPR es 0 y TFP es 0 y termina en 0 donde TVP es 1 y TFP es 1.

- **A** es la curva de un clasificador perfecto
- **B** es la curva de un clasificador estándar.
- **C** es la curva de un clasificador que adivina (el peor caso).

La curva ROC me permite encontrar el valor umbral que mejor resultado me dé.

Además me permite comparar clasificadores sin preocuparme del valor umbral elegido.

CURVA ROC



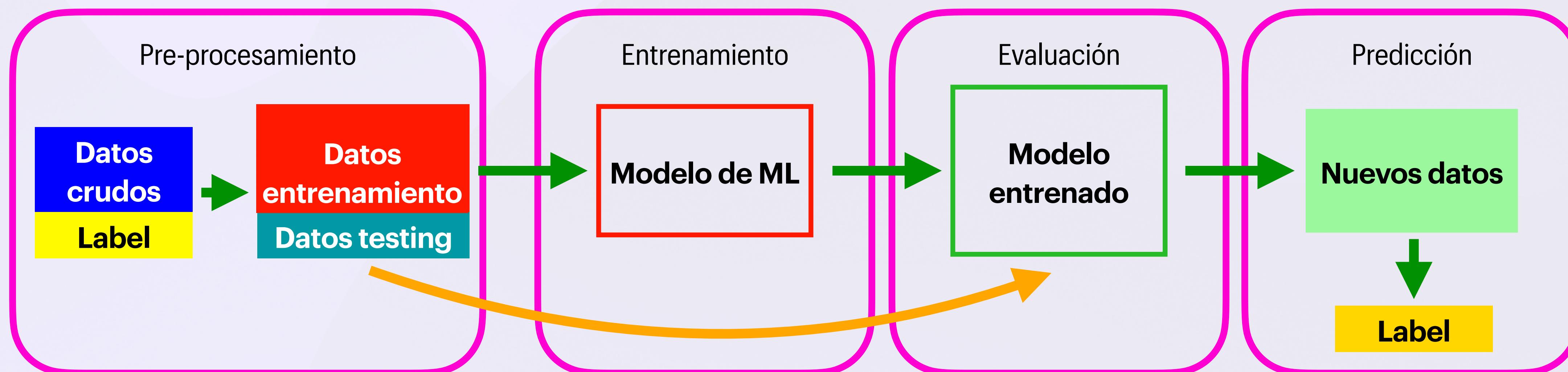
Si quiero bajar a una métrica a esta curva, podemos calcular el área bajo la curva (AUC).

- **A** tendrá un $AUC = 1$
- **B** tendrá un $0.5 < AUC < 1$
- **C** tendrá un $AUC = 0.5$

CONJUNTO DE VALIDACIÓN Y VALIDACIÓN CRUZADA

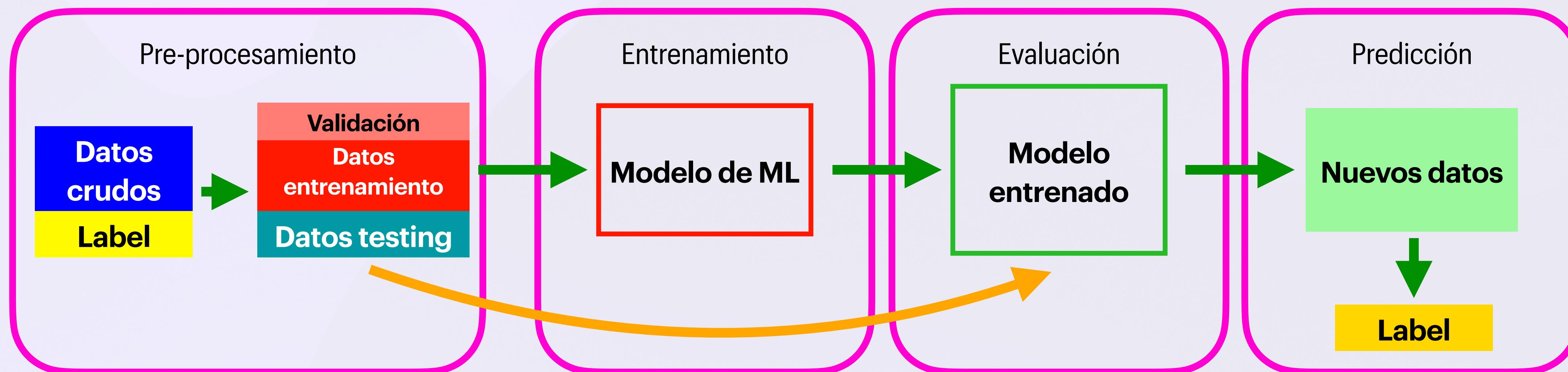
PROCESO DE MACHINE LEARNING

Cuando vimos esto, a propósito deje afuera un conjunto de datos...



PROCESO DE MACHINE LEARNING

Cuando vimos esto, a propósito deje afuera un conjunto de datos... **Conjunto de validación**



CONJUNTO DE VALIDACIÓN

Cuando vimos esto, a propósito deje afuera un conjunto de datos... [Conjunto de validación](#)

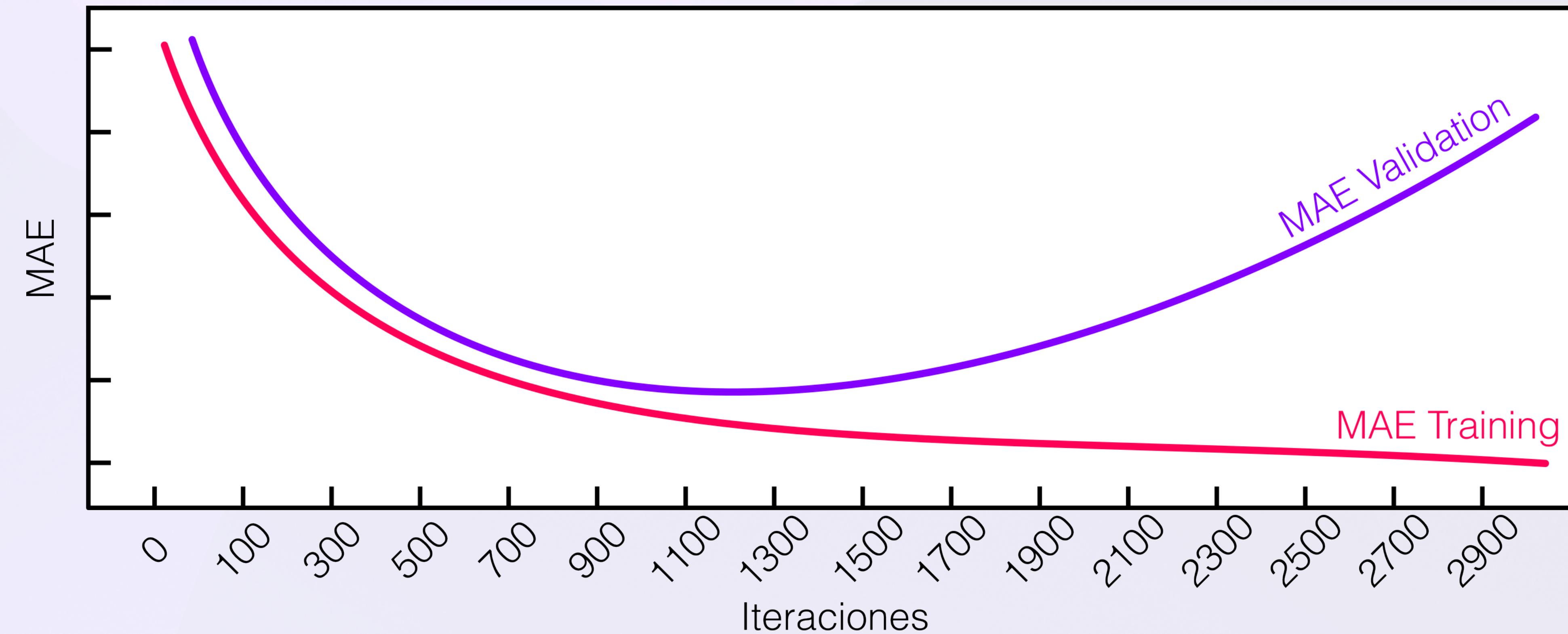
El conjunto de validación es un set que se usa en el entrenamiento para evaluar como se entrena el modelo pero sin usarlo para entrenar. En general, se usa aproximadamente un 10% del set total.

Supongamos que entrenamos un modelo que con cada paso mejora las métricas de entrenamiento, pero si lo dejamos mucho tiempo, llevará a un **sobreajuste**. Entonces, con el set de validación, cada cierto tiempo, evaluamos y vemos como performa.

Lo que va a pasar es que hay un punto donde la métrica de evaluación empezara a aumentar indicando **sobreajuste**, entonces en ese momento cortamos el entrenamiento.

La diferencia con el **conjunto de testeo** es que a este **nunca** lo usamos en ningún proceso de entrenamiento

CONJUNTO DE VALIDACIÓN



VALIDACIÓN CRUZADA

Este proceso que vimos mejora la generalización, pero es muy sensible a la selección del conjunto de validación. Además, eliminar datos que hubieran servido para entrenar es algo caro que pagamos.

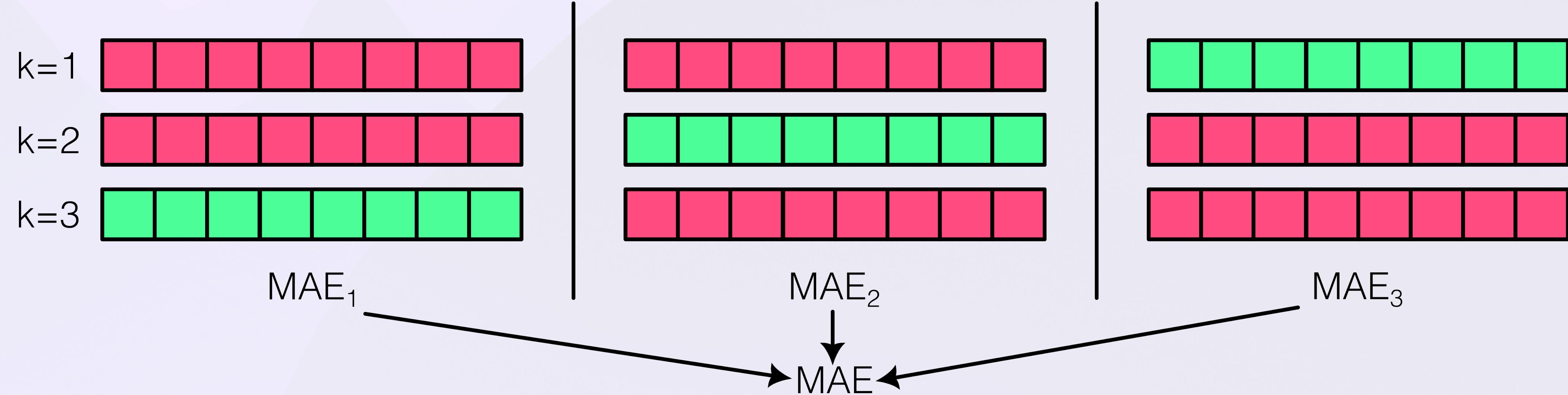
Buscando atacar esto, existe **K-Fold cross validation**.

Los datos de entrenamientos se dividen en **K** sub-conjuntos. Con esta separación, realizamos el proceso de validación que vimos previamente pero **K** veces:

Cada vez vamos eligiendo como set de validación a un conjunto **K** y a los restantes **K-1** como entrenamiento.

Una vez finalizado, los errores medidos se promedian para obtener la efectividad total del modelo.

VALIDACIÓN CRUZADA



VALIDACIÓN CRUZADA

Valores típicos usado es K=5 o 10.

Para que se usa esto?

El ejemplo de iteraciones nos da una pista...

VALIDACIÓN CRUZADA

Valores típicos usado es K=5 o 10.

Para que se usa esto?

El ejemplo de iteraciones nos da una pista...

Nos permite encontrar los **hiperparámetros**, ya que nos permite evaluar al menos una vez cada punto de entrenamiento, sin tener problemas de **sobreajuste** y sin tocar el conjunto de testeo.

Una vez que encontramos aquellos parámetros que mejoren las métricas de validación cruzada, ya se puede entrenar el modelo y testear con el conjunto de test para obtener las métricas del modelo.

VARIANTES DE LA VALIDACIÓN CRUZADA

Stratified K-Fold Cross Validation

Cuando tenemos un desbalance fuerte en la variable target, se usa esta variante. Lo que se busca es que cada uno de los K conjuntos, se eligen de forma que mantenga mas o menos la misma proporciones, para mantener la distribuciones generales.

Esto también se aplica a **conjunto de testeo!**

VARIANTES DE LA VALIDACIÓN CRUZADA

Leave-P-out Cross Validation

En este caso, dado n datos de entrenamiento, se eligen p datos, se entrena con $n-p$ y se valida con los p datos. Esto se va repitiendo para todas las combinaciones posibles.

Es la versión mas extrema de validación cruzada

Un caso particular es cuando $p=1$, llamada **Leave-1-out Cross Validation**



ARBOLES DE DECISIÓN

ARBOLES DE DECISIÓN

Los árboles de clasificación y regresión, conocidos como **CART** (Classification and Regression Trees), son una poderosa técnica de aprendizaje automático que se utiliza ampliamente para resolver problemas tanto de clasificación como de regresión.

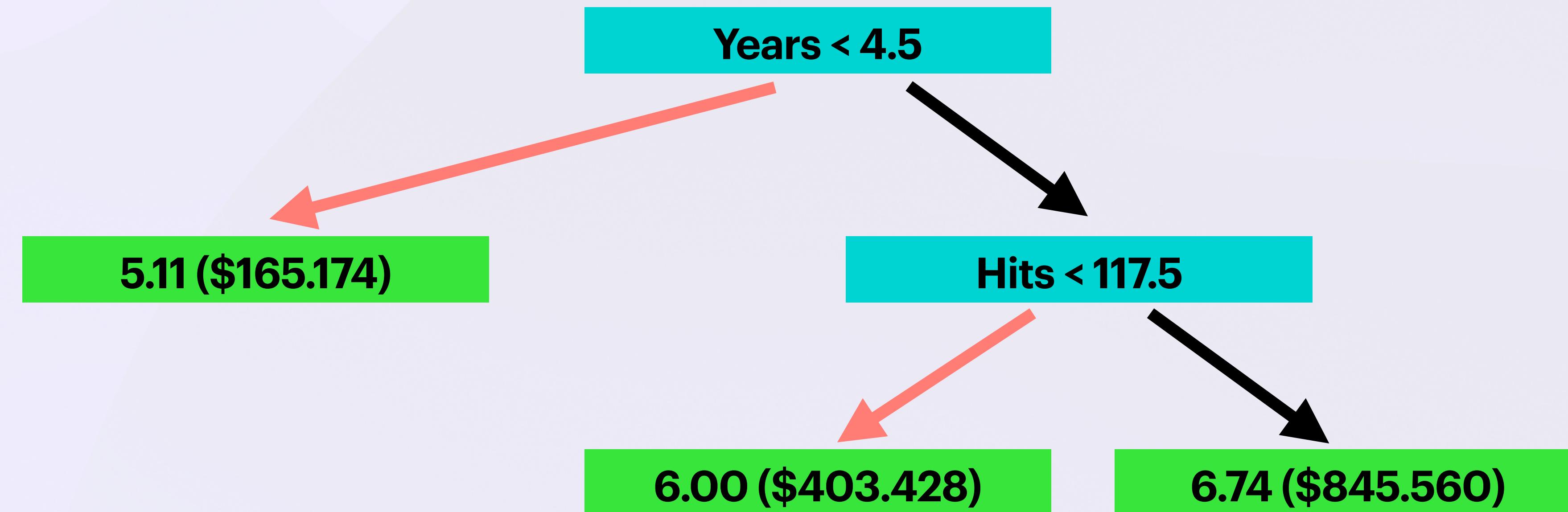
Los árboles CART son modelos de decisión que utilizan una estructura de árbol para realizar predicciones basadas en reglas **lógicas sencillas y fáciles de interpretar**.

[Arboles de clasificación](#)

[Arboles de regresión](#)

ARBOLES DE REGRESIÓN

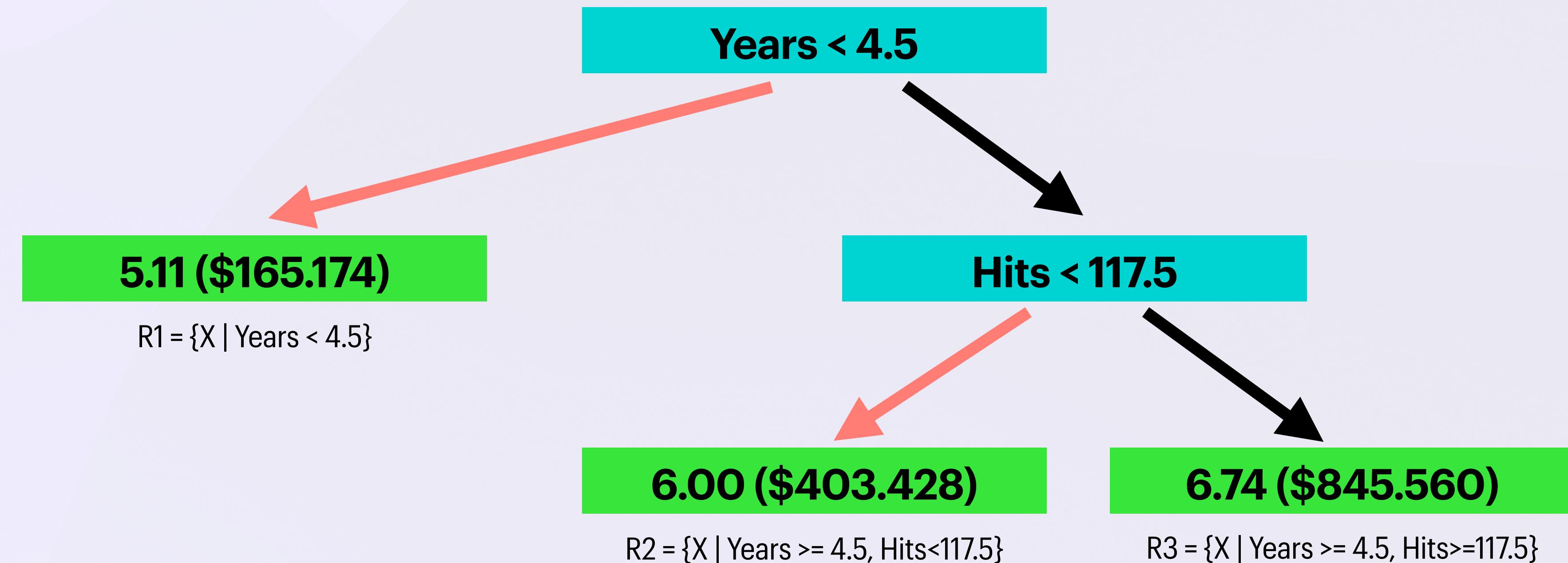
Empecemos con un ejemplo sencillo, usando el dataset [Hitters](#) (Datos de salarios de jugadores de Beisbol de 1987 y estadísticas deportivas de 1986)



Se predice el logaritmo del salario (tiene una distribución mas de campana). **Years** es años jugando en las ligas. **Hits** es el número de hits que realizó el año pasado.

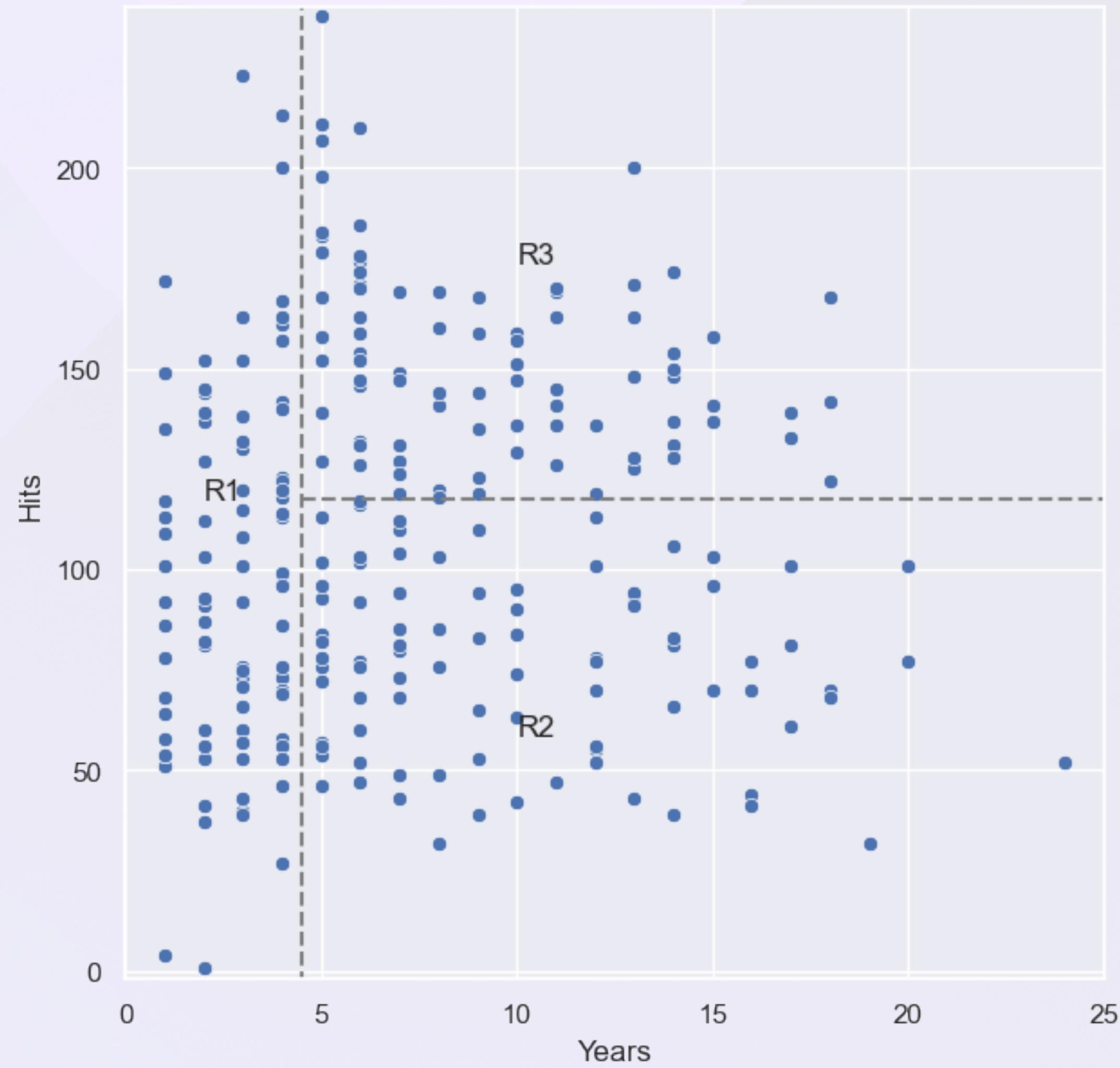
ARBOLES DE REGRESIÓN

Empecemos con un ejemplo sencillo, usando el dataset [Hitters](#) (Dato de salarios de jugadores de Beisbol de 1987 y estadísticas deportivas de 1986)



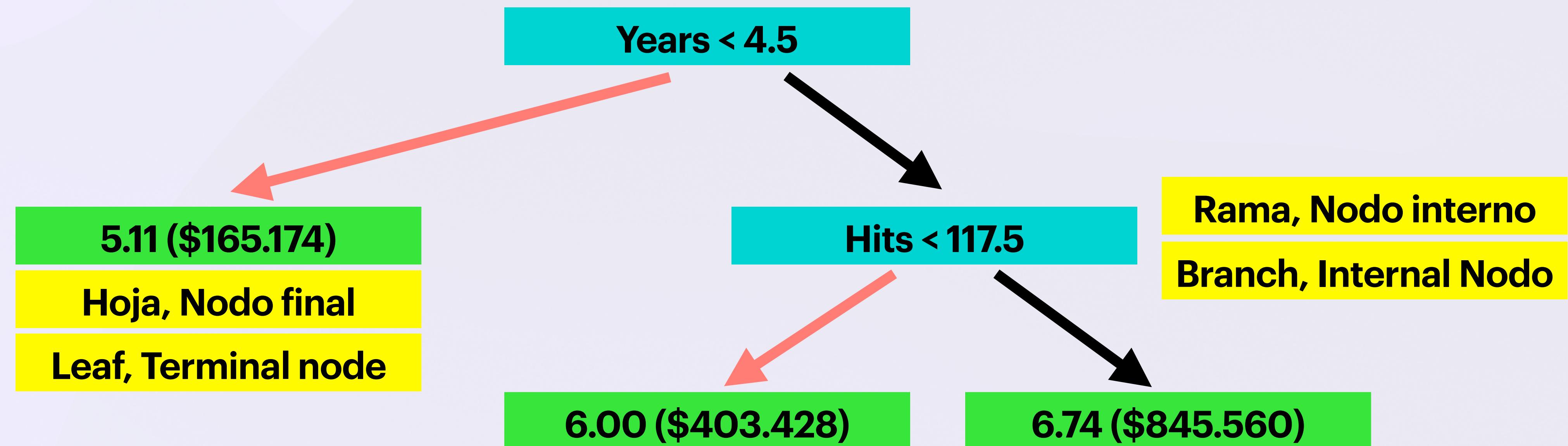
Se predice el logaritmos del salario (tiene una distribución mas de campana). **Years** es años jugando en las ligas. **Hits** es el numero de hits que realizó el año pasado.

ARBOLES DE REGRESIÓN



ARBOLES DE REGRESIÓN

Empecemos con un ejemplo sencillo, usando el dataset [Hitters](#) (Datos de salarios de jugadores de Beisbol de 1987 y estadísticas deportivas de 1986)



Este árbol de regresión es una sobre simplificación de la verdadero valores de regresión entre **Salary**, **Years** y **Hits**. Sin embargo, tiene sus ventaja porque es mas fácil entender y tienen mejor representación gráfica.

ARBOLES DE REGRESIÓN

Como construimos el proceso de construcción del árbol de regresión:

1. Dividimos el espacio de observaciones, que son el set de los valores posibles X_1, X_2, \dots, X_p , en J regiones distintas y que no se solapan R_1, R_2, \dots, R_J .
2. Para cada observación que cae en una región R_j , hacemos la misma predicción, la cual es simplemente la media de la respuesta de los valores de entrenamiento que están en R_j . Podemos usar otra métrica de medición de posición central.

ARBOLES DE REGRESIÓN

Cómo dividimos el espacio de observaciones?

En teoría, el espacio lo podríamos dividir en cualquier tipo de regiones, pero se elige espacios “rectangulares” para simplificar el modelo.

El objetivo es encontrar cajas R_1, \dots, R_J que minimice la suma al cuadrado de los residuos, dado por:

$$RSS = \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

ARBOLES DE REGRESIÓN

Cómo dividimos el espacio de observaciones?

En teoría, el espacio lo podríamos dividir en cualquier tipo de regiones, pero se elige espacios “rectangulares” para simplificar el modelo.

El objetivo es encontrar cajas R_1, \dots, R_J que minimice la suma al cuadrado de los residuos, dado por:

$$RSS = \sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Es la media de \mathbf{y} en la región R_j

ARBOLES DE REGRESIÓN

Cómo dividimos el espacio de observaciones?

Es imposible buscar todas las combinaciones posibles de valores para encontrar la minimizar a RSS.

Tenemos que usar algún algoritmo de optimización. Para ello tomamos un algoritmo **top-down greedy** que es conocido como **Recursive binary splitting**.

- **Top-down:** Arrancamos desde el tronco del árbol y vamos bajando.
- **Greedy:** En cada paso, se busca la mejor bifurcación en ese paso particular.

ARBOLES DE REGRESIÓN

Recursive binary splitting

Se elige un X_j y el punto de corte s de tal forma que bifurca el espacio de features en dos regiones $\{X|X_j < s\}$ y $\{X|X_j \geq s\}$ que lleve a la mayor reducción de **RSS**.

Es decir, para cada valor de j y cada valor de s :

$$R_1(j, s) = \{X|X_j < s\} \quad R_2(j, s) = \{X|X_j \geq s\}$$

Y buscamos el valor de j y s que minimice esta ecuación:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

ARBOLES DE REGRESIÓN

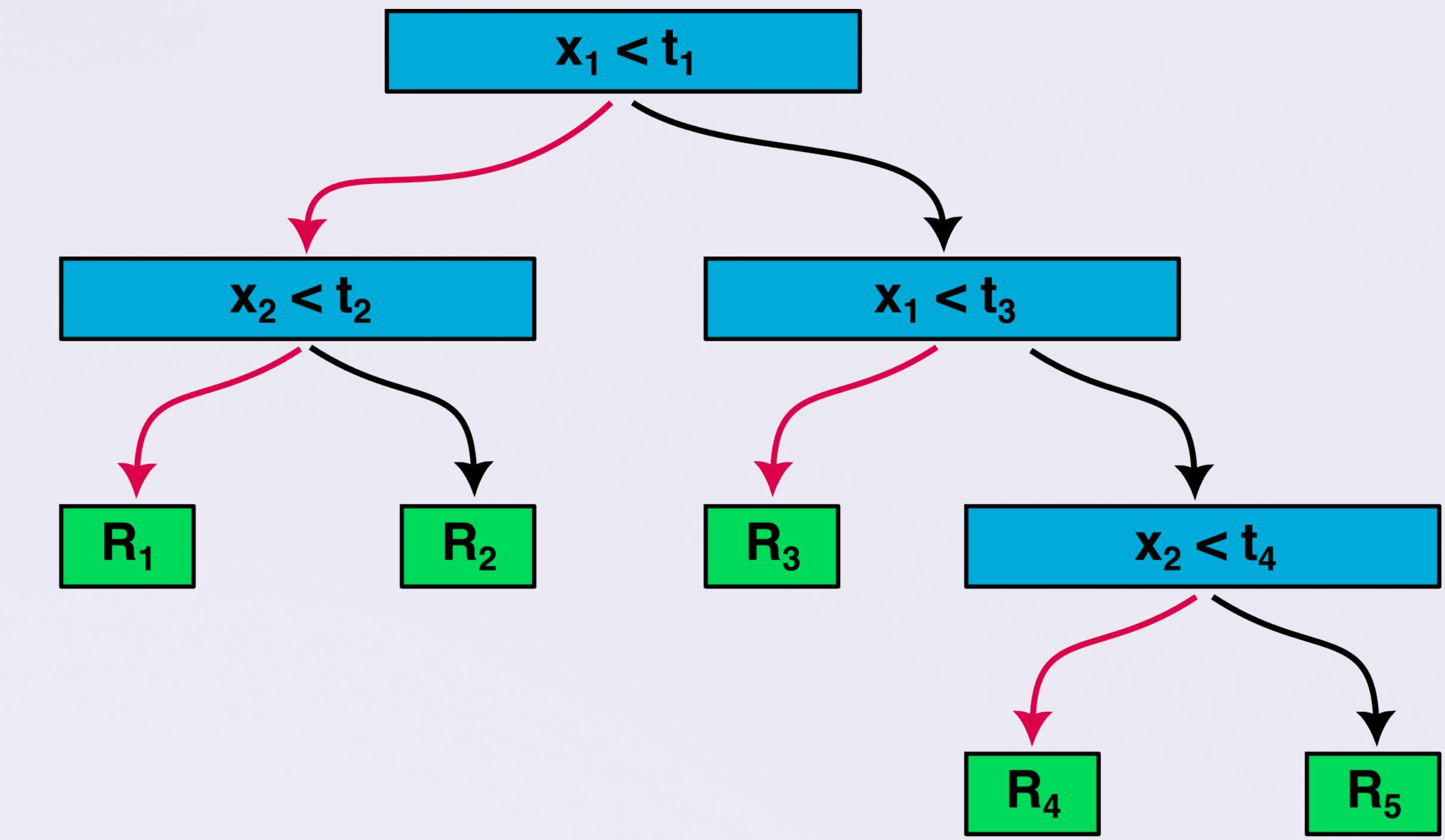
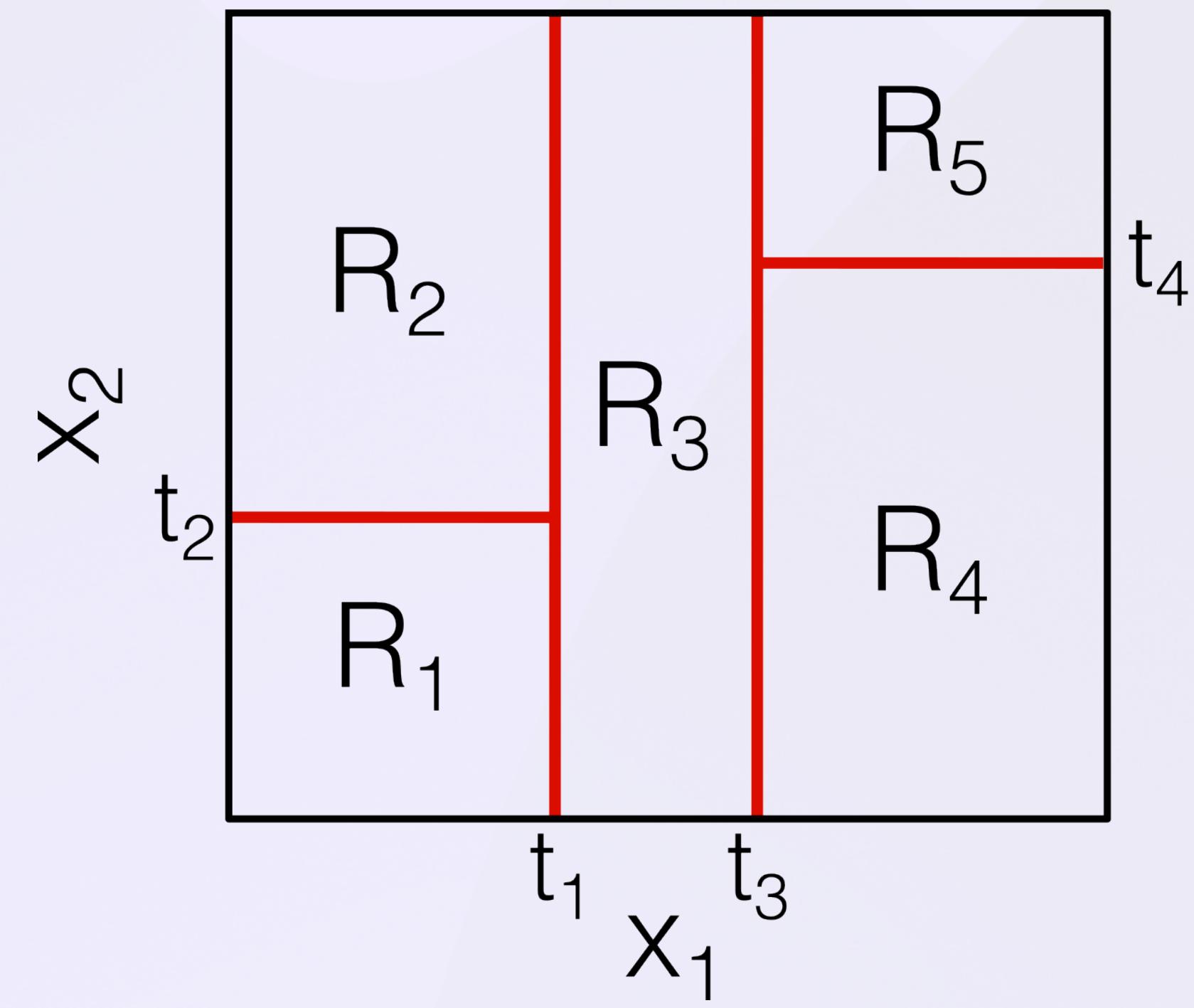
Recursive binary splitting

Y recursivamente repetimos esto para los segmentos que se generan, pero ahora tomando a las regiones formadas y aplicando este proceso, partimos en nuevas regiones.

Este proceso continua hasta que llegamos a un **criterio de corte**.

ARBOLES DE REGRESIÓN

Recursive binary splitting



ARBOLES DE REGRESIÓN

Podando los arboles

El proceso que se describió puede producir buenas predicciones del set de entrenamiento, pero muy fácilmente puede generar **overfitting**, haciendo que se desempeñe muy mal en el set de validación.

El caso mas extremo es un árbol con una hoja por cada punto del set de entrenamiento.

Esto se debe a que el árbol es muy complejo. Un árbol mas pequeño con menor regiones, puede llevar a menos **varianza** y mejor interpretación a expensa de un poco de **sesgo**.

ARBOLES DE REGRESIÓN

Podando los arboles

La estrategia mas obvia es construir el árbol sólo mientras la disminución en el RSS sea mayor a un valor umbral (relativamente alto). Esta estrategia dará como resultado árboles más pequeños, pero es demasiado cortoplacista,

Una división aparentemente sin valor en las primeras etapas del árbol podría ser seguida por una división muy buena, es decir, una división que conduzca a una gran reducción del RSS más adelante.

ARBOLES DE REGRESIÓN

Podando los arboles

Una mejor estrategia es llevar un paso de eliminación hacia atrás. Construimos un árbol enorme T_0 , y luego vamos podando para obtener un **sub-árbol**.

Como hacemos? Intuitivamente es elegir un sub-árbol que disminuya el error de validación.

Pero de nuevo, **estamos ante un problema demasiado complejo de iterar**.

ARBOLES DE REGRESIÓN

Podando los arboles

Vamos a introducir un valor de penalización α a nuestra formula de RSS. Dado un valor de α , existe un sub-árbol T perteneciente a T_0 que:

$$\sum_{m=1}^L \sum_{i \in R_j} (y_i - \hat{y}_{R_m})^2 + \alpha L$$

...es minimo.

L indica el numero de hojas del sub-árbol.

α presenta un trade-off entre complejidad del árbol y su capacidad de ajustar a los datos. Si $\alpha=0$ el árbol elegido es T_0 . Cuando mas grande es α , el precio a pagar por el tamaño de árbol cada vez es mayor. α nunca es negativo.

ARBOLES DE REGRESIÓN

Podando los arboles

Algo interesante es que a medida que aumentamos a desde cero, las ramas se podan del árbol de una manera anidada y predecible, por lo que es fácil obtener la secuencia completa de subárboles en función de a .

Podemos seleccionar un valor de a usando un **conjunto de validación** o usando **validación cruzada**.

Luego volvemos al conjunto de datos completo y obtenemos el sub-árbol correspondiente a a .

ARBOLES DE REGRESIÓN

Algoritmo total

1. Usando **Recursive binary splitting** se crea el árbol mas grande con el dataset de entrenamiento, terminando el entrenamiento cuando cada hoja tenga menos de un numero determinado de observaciones.
2. Se aplica la técnica de podado de árbol para obtener un set de sub-arboles como función de a . Usando validación cruzada para elegir a .
3. Elija el sub-árbol del paso 2 que corresponde al valor de a que disminuya el error.

The background features a minimalist design with abstract, wavy shapes in shades of purple and dark blue. These shapes are layered and overlap, creating a sense of depth and movement across the entire frame.

VAMOS A PRÁCTICAR UN POCO...

ARBOL DE CLASIFICACIÓN

ARBOL DE CLASIFICACIÓN

Un árbol de clasificación es muy similar a uno de regresión, pero ahora se usa para predecir una **variable cualitativa**.

En el caso de regresión, al llegar la hoja, obteníamos el valor con el promedio de los valores en la hoja. Ahora, obtenemos la clase en base a la **clase que más ocurre** en las muestras que están en la hoja.

Al interpretar los resultados de un **árbol de clasificación**, a menudo estamos interesados no sólo en la predicción de clase correspondiente a una región de nodo terminal particular, sino también en las **proporciones de clase entre las observaciones de entrenamiento** que caen en esa región.

ARBOL DE CLASIFICACIÓN

La forma en que se crea un árbol de clasificación es muy parecida al árbol de regresión con la estrategia **top-down** y **greedy**, pero no contamos con el error cuadrático.

Una primera métrica que podemos usar es la **tasa de error de clasificación**:

$$E = 1 - \max_k(\hat{p}_{mk})$$

Es la fracción de las observaciones de entrenamiento en esa región que no pertenecen a la clase más común.

\hat{p}_{mk} representa la proporción de las observaciones de entrenamiento en la región **m** que son de la clase **k**.

Ojo, el error de clasificación **no es suficientemente sensible para crecer a los árboles** y, en la práctica, son preferibles otras dos medidas.

ARBOL DE CLASIFICACIÓN

El **índice de Gini**, el cual es una medida de la desigualdad usada inicialmente para medir la desigualdad de los países:

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Como una medida de la varianza a travez de todas las clases K. El cual se observa que el **índice de Gini** toma un valor pequeño si todas las \hat{p}_{mk} son cercanas a cero o uno.

Por esto, se dice que el índice de Gini es una medida de la pureza de un nodo terminal

ARBOL DE CLASIFICACIÓN

En un árbol de clasificación, queremos encontrar una rama de decisión que de mucha información.

Si en una rama, $x < 0$, hace que todos las observaciones de una clase vaya para un lado y todas las otras observaciones vayan para la otra, **va a ser una excelente rama a elegir**.

En cambio, el caso que no afecta a como se mueven las clases, probablemente no es una buena opción.

La forma que capturamos esta noción de cuanta información transmite es con **Entropía**. O también como medida de desorden.

ARBOL DE CLASIFICACIÓN

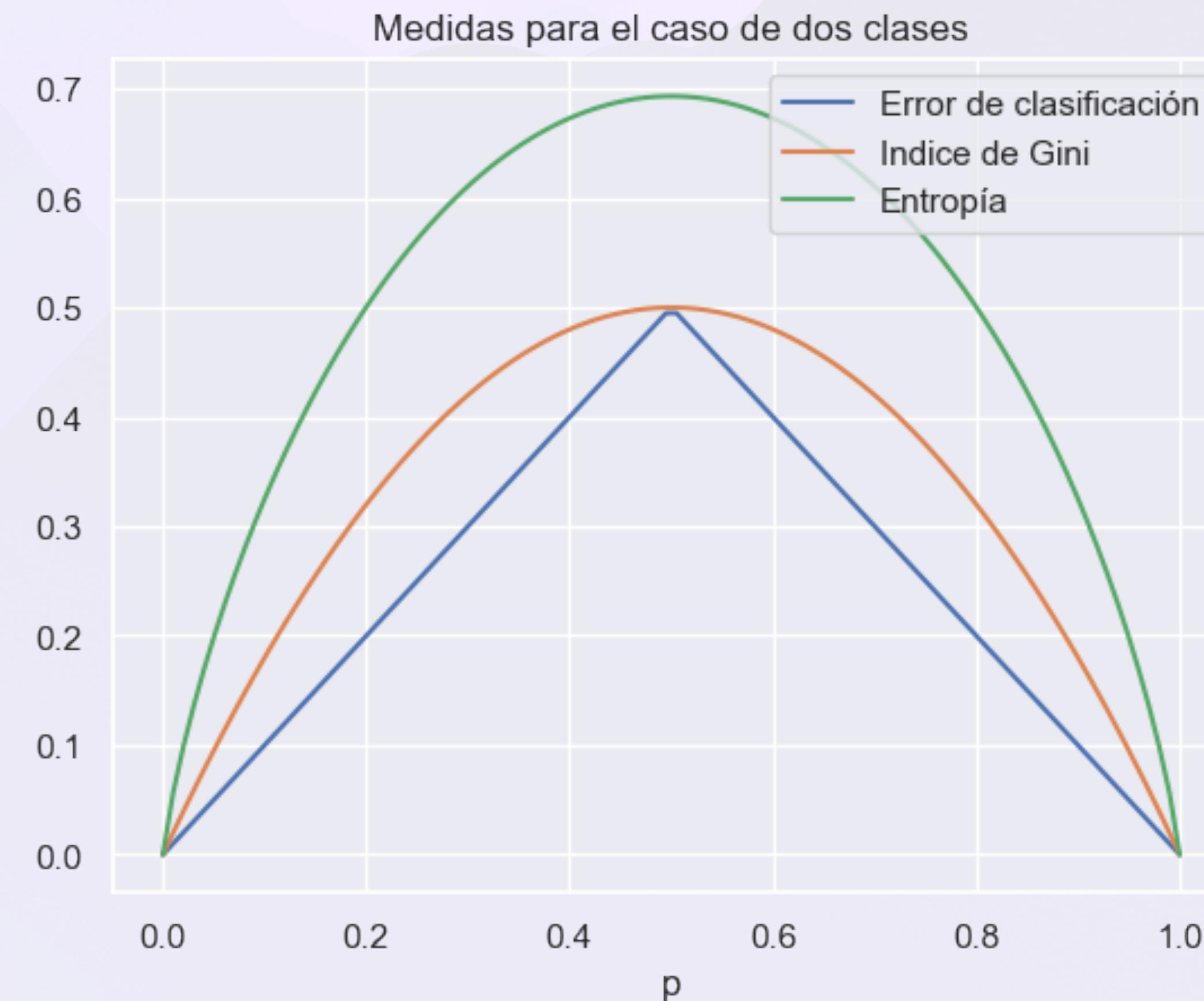
Si en una hoja, todas las observaciones de entrenamiento son de una sola clase, la entropía es cero.

En cambio, si las clases están desperdigadas de forma uniforme entre la clase, la entropía es grande.

La definición de **entropía** es: $D = - \sum_{k=1}^K \hat{p}_{mk} \log(\hat{p}_{mk})$

Vemos que si \hat{p}_{mk} son cercanos a cero o a uno, la entropía es un valor pequeño. Si las proporciones son similares entre si, este valor va a ser grande.

ARBOL DE CLASIFICACIÓN



ARBOL DE CLASIFICACIÓN

Todo el resto es igual al árbol de regresión

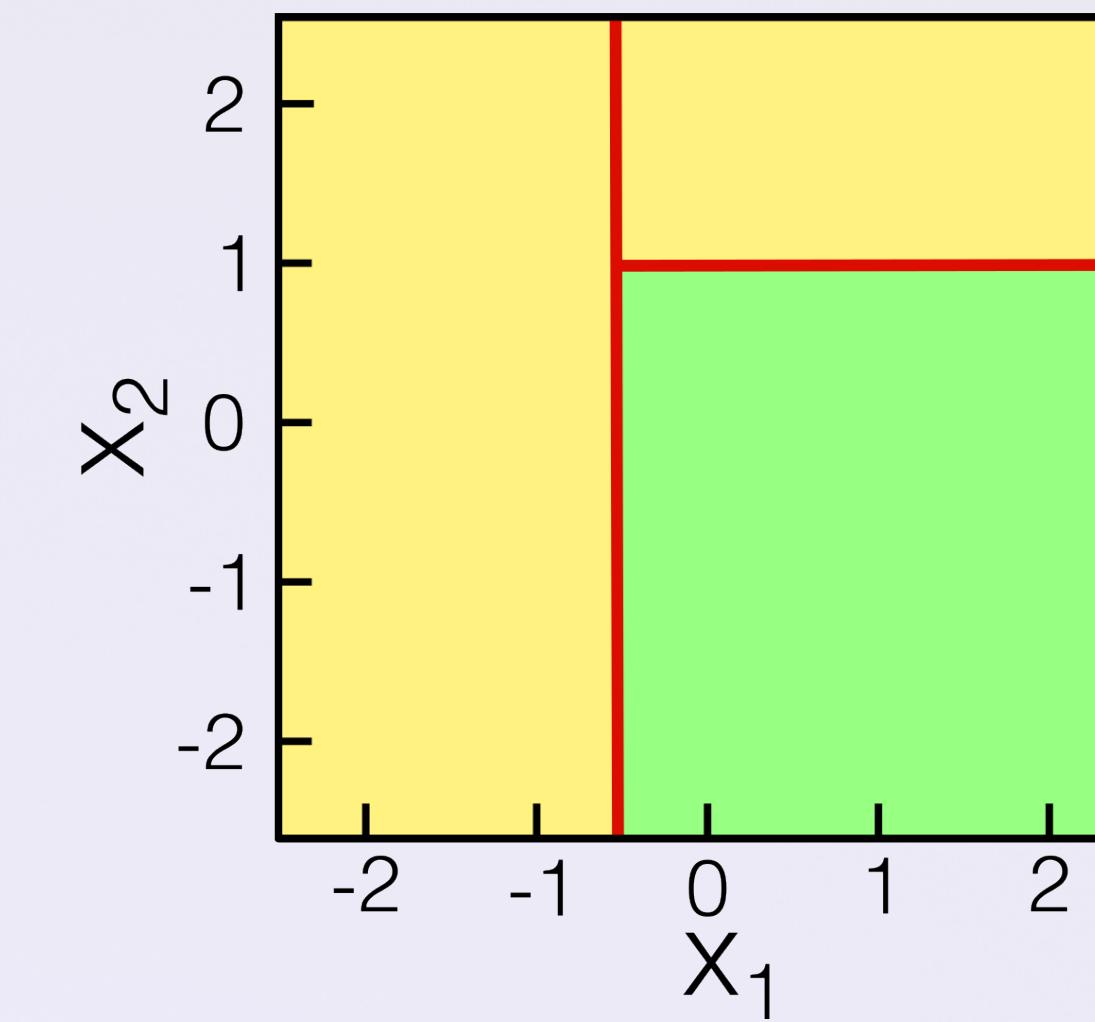
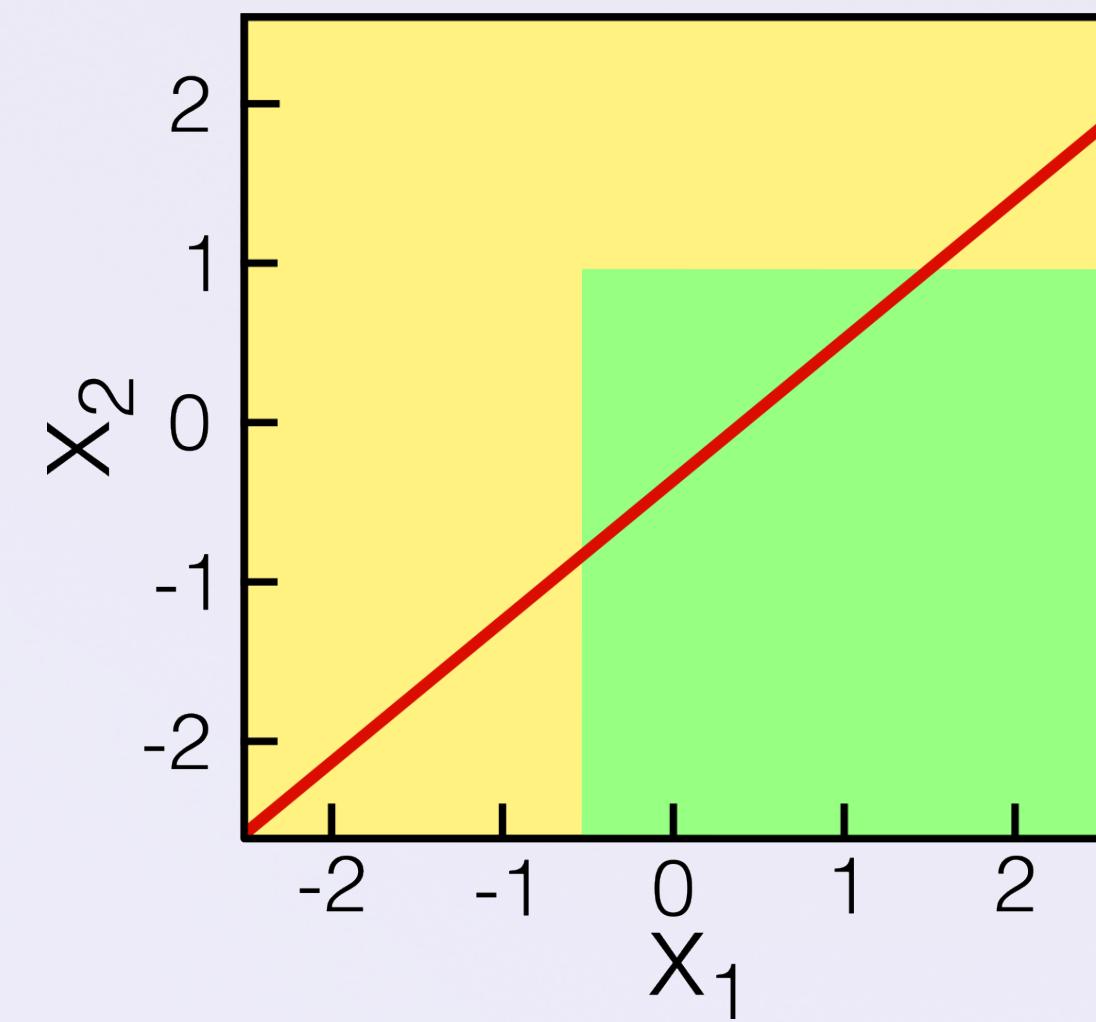
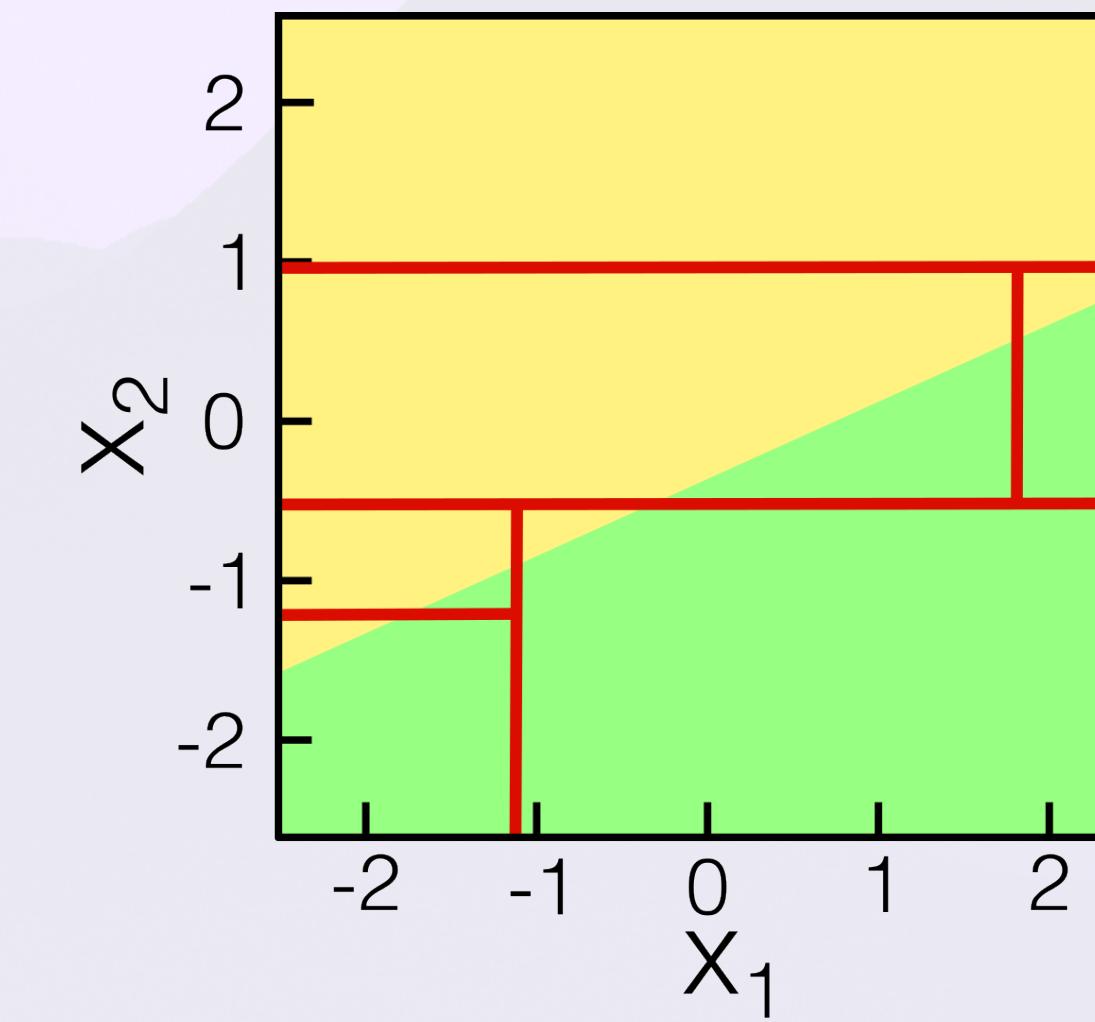
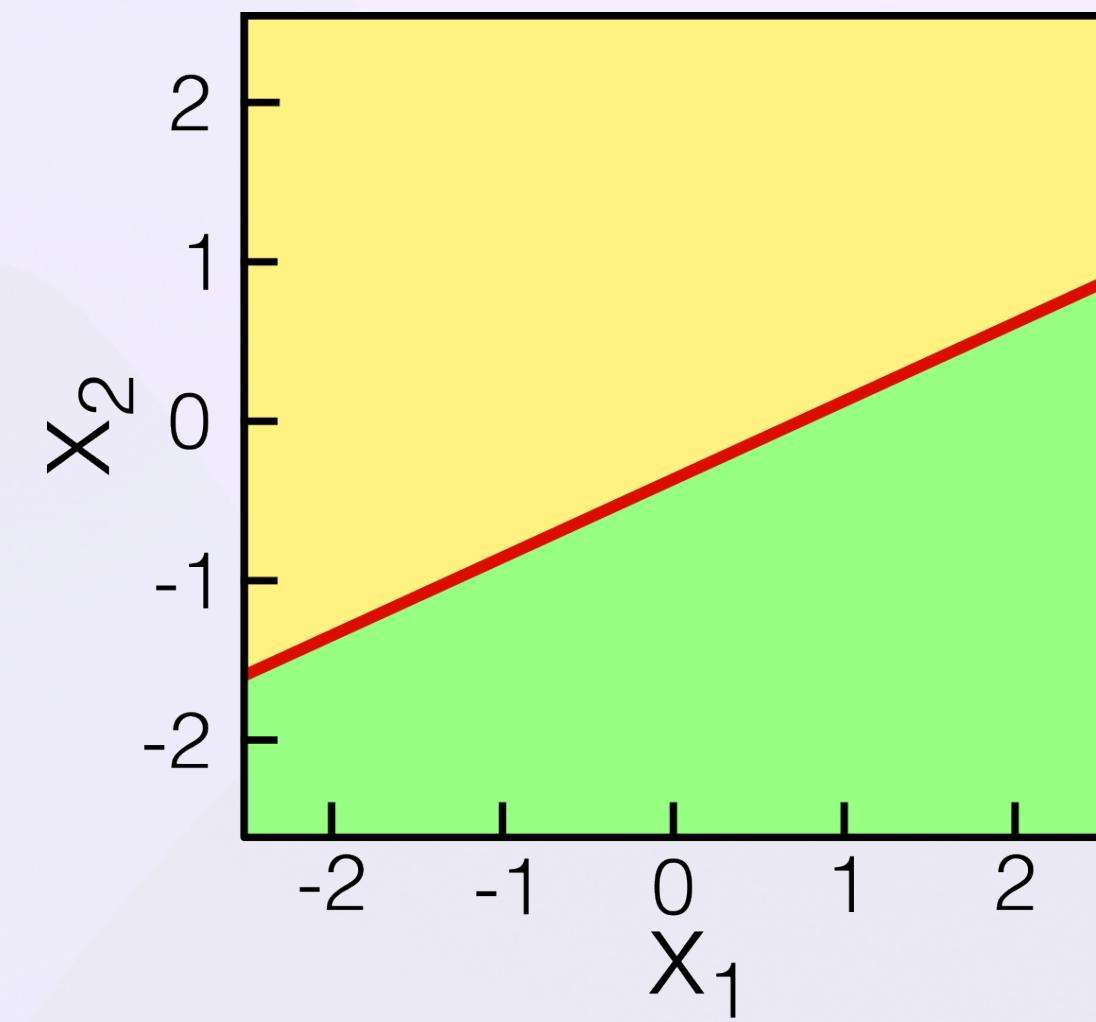
The background features a minimalist design with abstract, wavy shapes in shades of purple and dark blue. These shapes are layered and overlap, creating a sense of depth and movement across the entire frame.

VAMOS A PRÁCTICAR UN POCO...

VENTAJAS Y DESVENTAJAS DE LOS ARBOLES

- ✓ Son fáciles de explicar a las personas, mas inclusive que a la regresión lineal.
- ✓ Se hipotetiza que el árbol de decisión se acerca más a la forma que un humano piensa.
- ✓ Se puede representar graficamente.
- ✓ Los arboles puede manejar fácilmente variables cualitativas sin necesidad de crear variables dummy.
- ✗ No tienen el mismo nivel de exactitud de predicción que otro modelos
- ✗ No son robustos, pequeños cambios en los datos pueden cambiar grandes cambios en la predicción.

VENTAJAS Y DESVENTAJAS DE LOS ARBOLES





BOSQUES ALEATORIOS

BOSQUES ALEATORIOS

Es común que los arboles sobreajusten, dado que tan exacto tienden a adaptarse a los datos de entrenamiento.,

Una forma de evitar esto es mediante bosques aleatorios, en el cual se construyen múltiples arboles de decisión y combinar sus salidas.

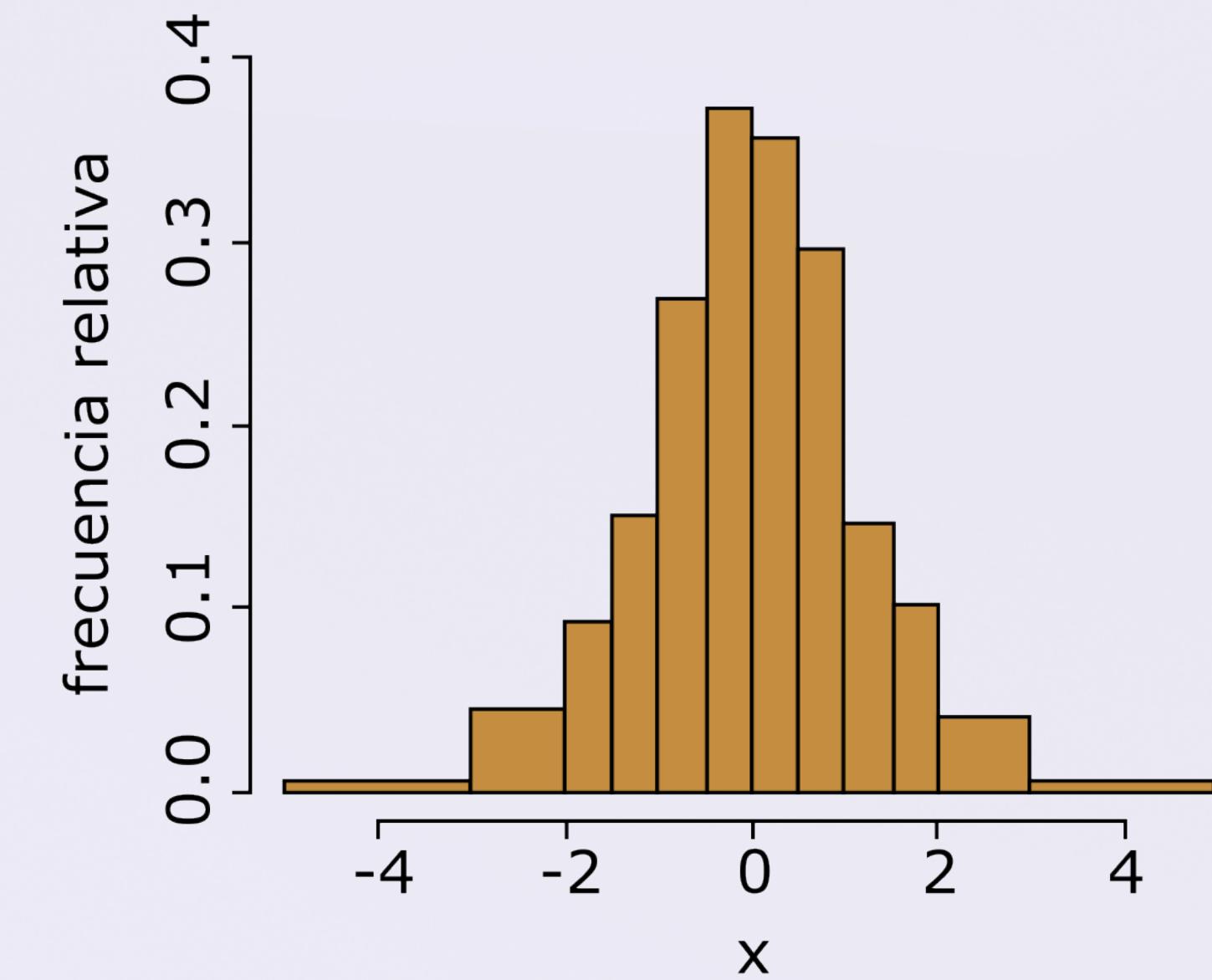
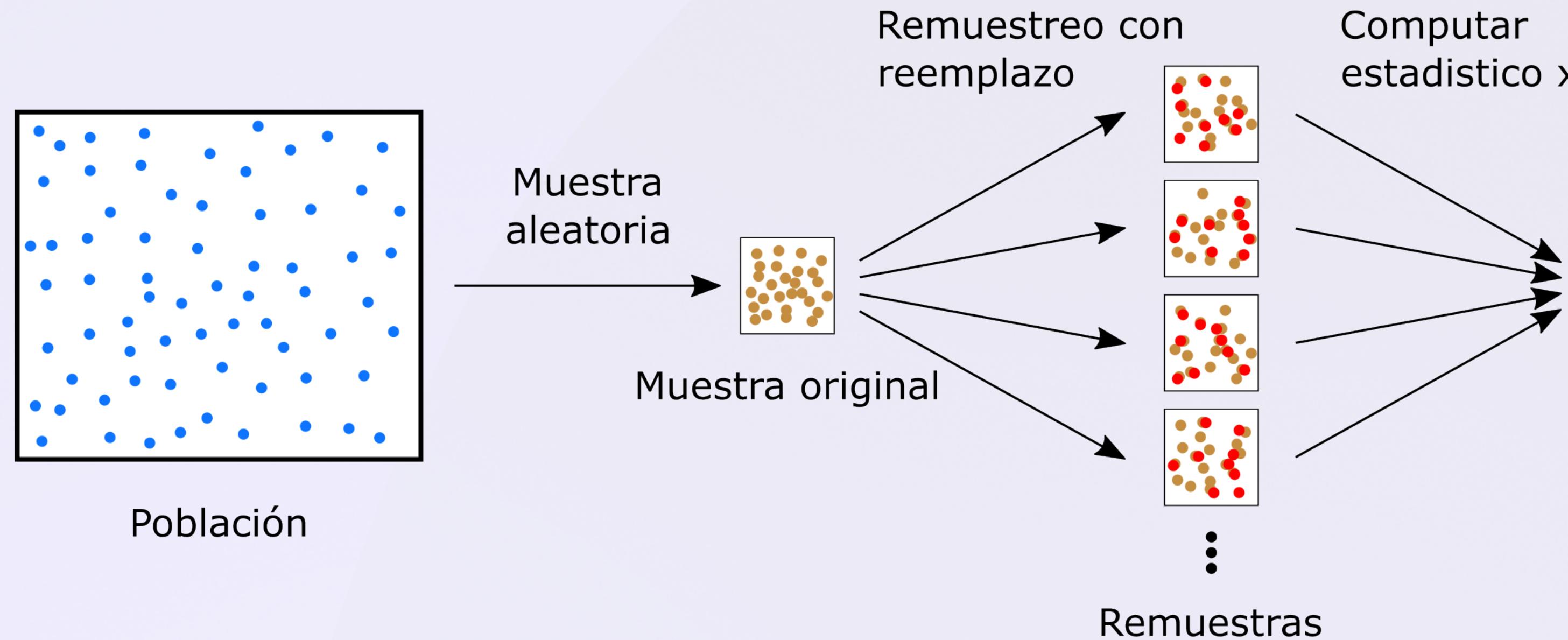
Si son de **clasificación**, estos arboles votan la clase.

Si son de **regresión**, se promedia las predicciones.

BOSQUES ALEATORIOS

Como se construyen aleatoriamente estos arboles?

Una forma es usando **bootstrapping**



BOSQUES ALEATORIOS

Como se construyen aleatoriamente estos arboles?

Bagging

En el caso de los arboles, en vez de entrenar a los arboles con todas las observaciones del set de entrenamiento, se arma un nuevo set para cada árbol, usando **Bootstrapping**.

Dado que cada árbol es entrenado diferente, va a ser diferente de los demás arboles.

En general estos arboles son profundos, es decir gran **varianza** pero poco **sesgo**. Al promediar las salidas, en regresión, reducimos la varianza.

Bagging ha mejorar sustancialmente los resultados cuando se lleva a cientos o miles de arboles.

BOSQUES ALEATORIOS

Los **bosques aleatorios** son una mejora de los obtenidos mediante **bagging** únicamente.

Los **bosques aleatorios** hace lo mismo que mediante **bagging**, pero además se usa una cantidad aleatoria de atributos. El valor de cuantos atributos a usar se elige aproximadamente la raíz cuadrada de la cantidad de atributos totales.

Por ejemplo, si tenemos $p=13$ atributos, se usarán $m=4$, y en cada árbol será una combinación al azar diferente.

BOSQUES ALEATORIOS

El razonamiento de esto es:

Supongamos que hay un atributo que es muy fuerte predictor, junto a otros atributos moderadamente fuertes. Si aplicamos **Bagging**, todos estos arboles siempre tenderán a usar el atributo fuerte en la bifurcación inicial. Por consiguiente, todos serán parecidos y habrá una fuerte correlación entre arboles, **quitando la posibilidad de reducir la varianza**.

Bosques aleatorios, al separar los atributos, en promedio $(p-m)/p$ de las particiones no van a considerar al atributo fuerte, quitando la correlación.

BOOSTING

Por ultimo, nos queda **Boosting**

La idea es similar a la de Bagging, pero en vez de construir arboles aleatoriamente dado por el proceso de Bootsraping, los nuevos arboles que se construyen usando la información de arboles anteriores.

El truco está en que se entrena un árbol, se calcula los residuos, y esos residuos pasan a ser la variable predictora del siguiente árbol. Los arboles elegidos son chicos con unos pocos nodos.

Esto es un proceso de aprendizaje lento que depende de arboles anteriores. Un proceso lento de aprendizaje, estadísticamente lleva a buenos entrenamiento.

Un algoritmo derivado de esto es el famoso **XGBoost**

The background features a minimalist design with abstract, wavy shapes in shades of purple and dark blue. These shapes are layered and overlap, creating a sense of depth and movement across the entire frame.

VAMOS A PRÁCTICAR UN POCO...