Machine Learning aplicado a las Ciencias Sociales

Clase 6. Análisis supervisado. CART

Repaso: Métricas de evaluación para variables continuas

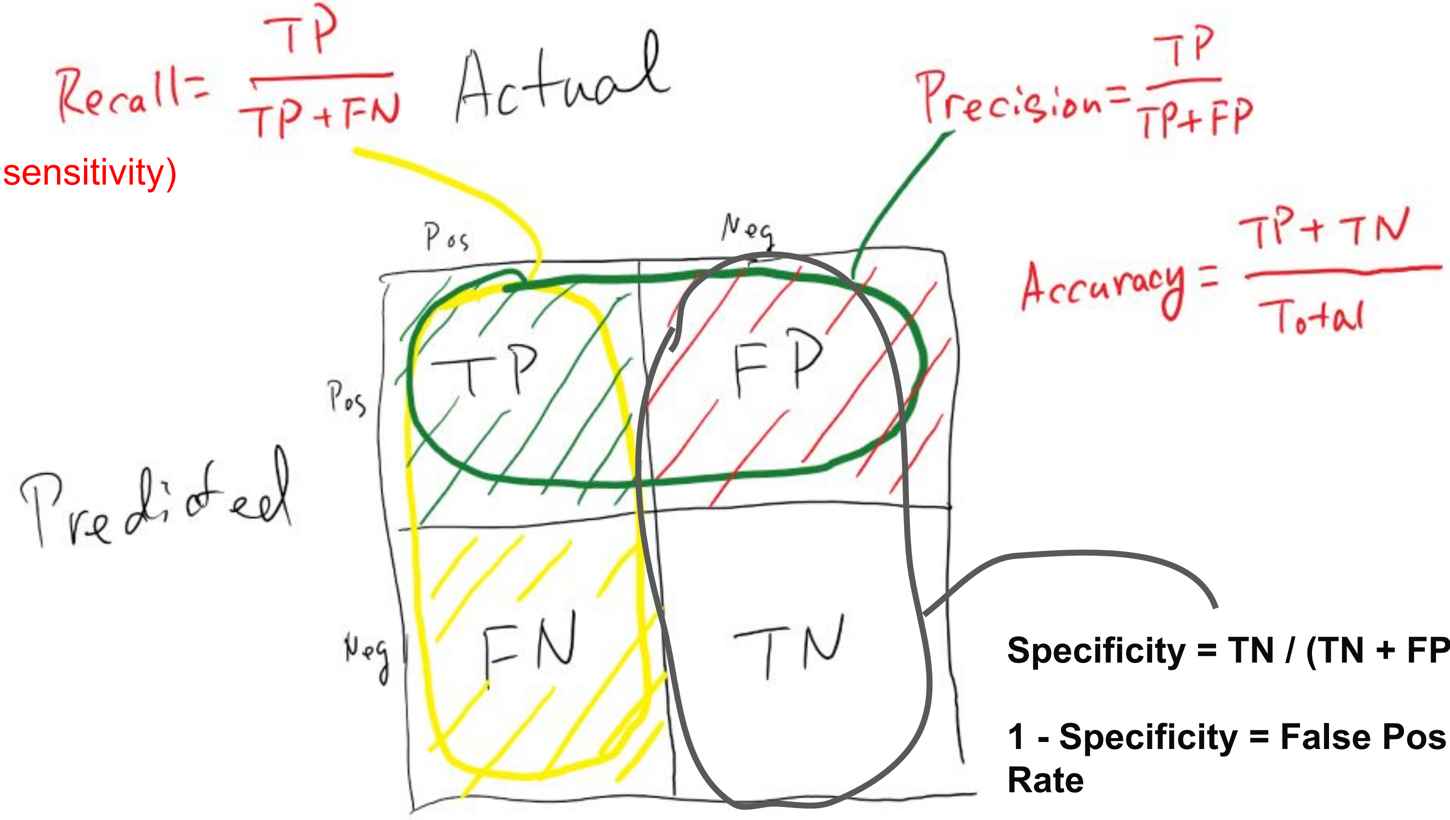
Medida	Tipo de modelo	Explicación
Root-mean-square deviation (RMSE)	Regresión	Raíz del error cuadrático medio, muestra la diferencia promedio entre los valores predichos y los reales.
Mean Absolute Error (MAE)	Regresión	Diferencia entre los valores predichos por el modelo y los valores reales, expresado en la unidad de la variable a predecir.

Repaso: Métricas de evaluación para variables categóricas

Medida	Tipo de modelo	Explicación
Recall	Clasificación	La proporción de positivos identificados correctamente sobre el total de positivos reales: cuántos positivos detecta.
Precision	Clasificación	La proporción de positivos identificados correctamente sobre el total de positivos predichos: cuán preciso es el modelo al clasificar positivos.
Accuracy	Clasificación	La proporción de casos clasificados correctamente sobre el total de las observaciones.
F1	Clasificación	Combina las métricas de recall y precisión, hace la media armónica entre ambas medidas.
Curva ROC	Clasificación	Expresa la relación entre la tasa de verdaderos positivos y falsos positivos en distintos umbrales de clasificación.

Matriz de confusión

(O sensitivity)

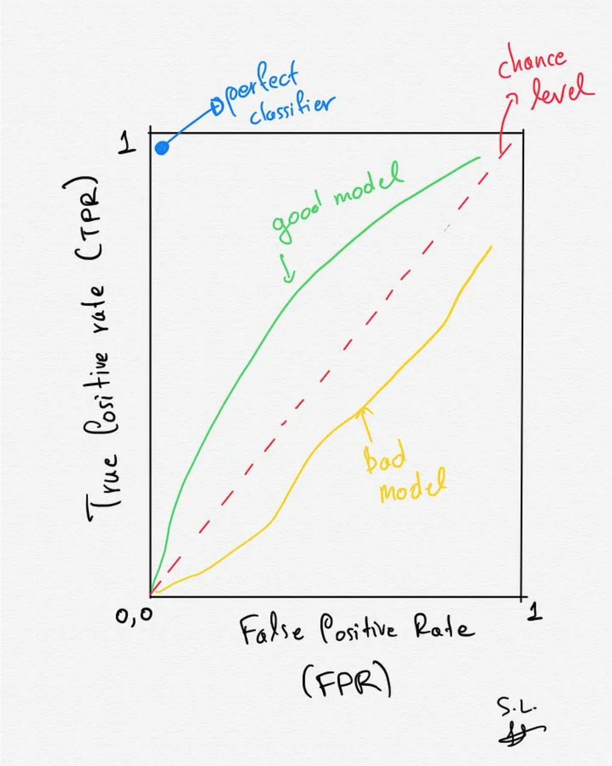


Specificity = TN / (TN + FP)

1 - Specificity = False Positive Rate

ROC - Métricas de evaluación

- En el eje y tenemos la TPR (también conocida como sensitivity o ya mencionado recall) y en el x los falsos positivos (probabilidad de falsa alarma, 1-specificity).
- La curva nos muestra para cada valor del umbral de decisión la relación entre estos dos valores.
- Nos dice qué tan bien el modelo puede distinguir los casos.
- La medida normalizada de esta curva es el área bajo la curva (más grande, mejor).



Evaluando modelos de clasificación

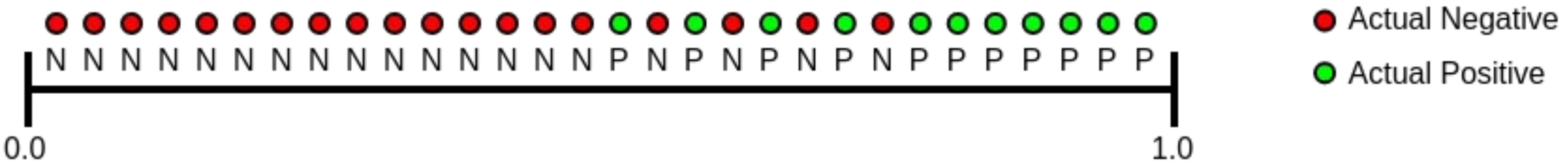
- Ahora… ¿cómo decide un método si la predicción es positiva o negativa?

- Regresión logística (pero vale para otros)
  - La predicción “nativa” de una regresión logística es una probabilidad
  - Se aplica una regla de decisión para asignarle un valor positivo o negativo

- Si $p(y \mid X) > 0.5$ => el caso es positivo
- Si $p(y \mid X) \le 0.5$ => el caso es negativo

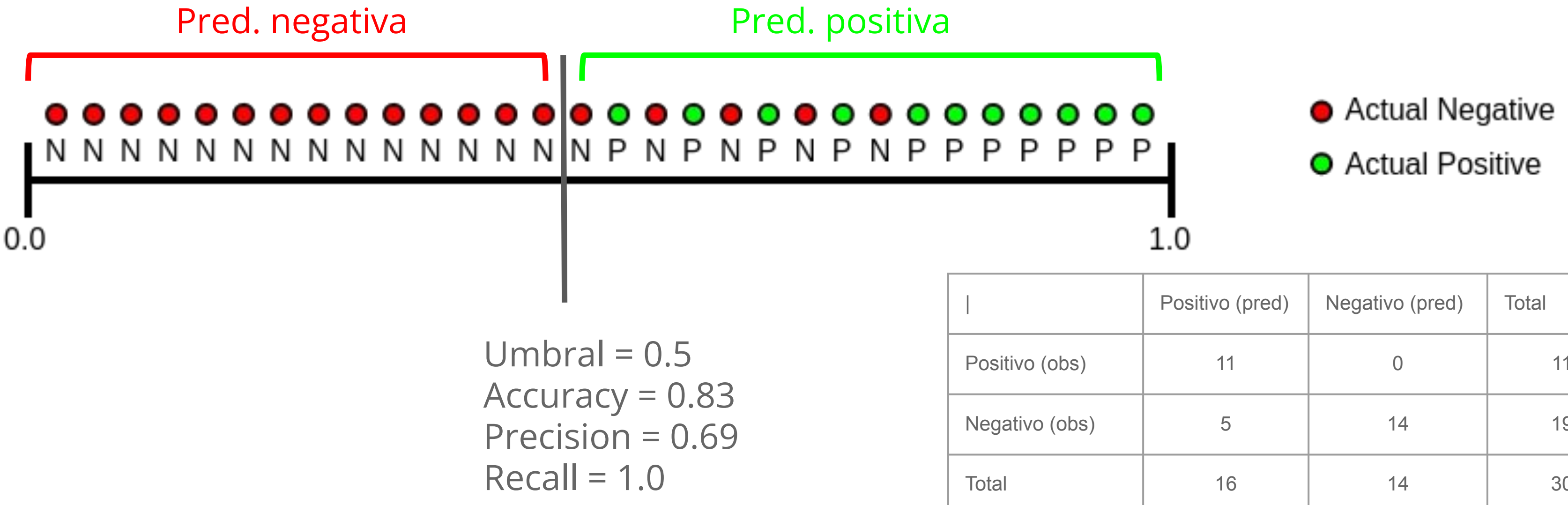
Evaluando modelos de clasificación: ROC

- Variando ese umbral (0.5), puedo hacer hacer variar las métricas basadas en la matriz de confusión



Evaluando modelos de clasificación: ROC

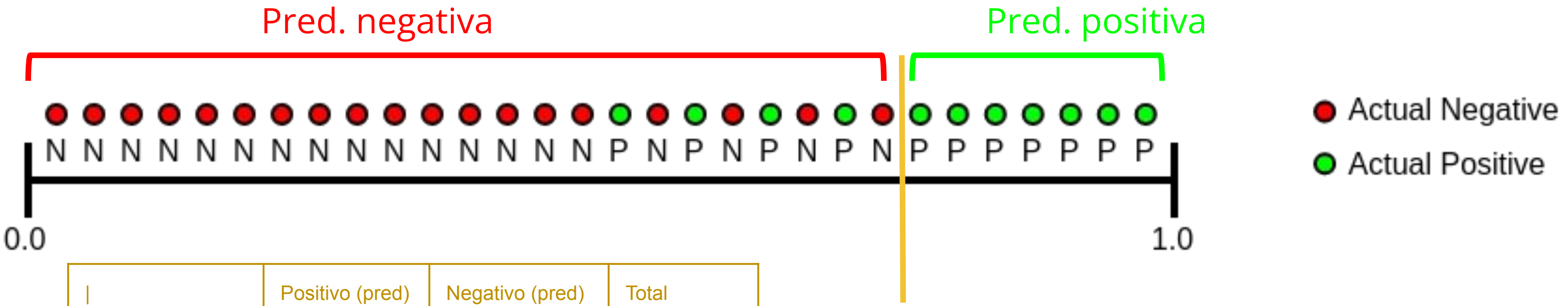
- Variando ese umbral (0.5), puedo hacer hacer variar las métricas basadas en la matriz de confusión



|	Positivo (pred)	Negativo (pred)	Total
Positivo (obs)	11	0	11
Negativo (obs)	5	14	19
Total	16	14	30

Evaluando modelos de clasificación: ROC

- Variando ese umbral (0.5), puedo hacer hacer variar las métricas basadas en la matriz de confusión

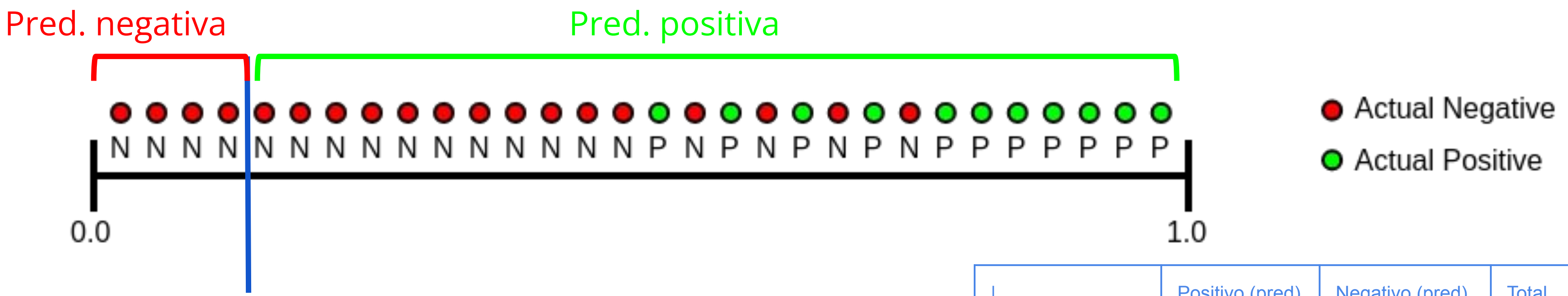


|	Positivo (pred)	Negativo (pred)	Total
Positivo (obs)	7	4	11
Negativo (obs)	0	19	19
Total	7	23	30

Umbral = 0.76
Accuracy = 0.87
Precision = 1.0
Recall = 0.64

Evaluando modelos de clasificación: ROC

- Variando ese umbral (0.5), puedo hacer hacer variar las métricas basadas en la matriz de confusión

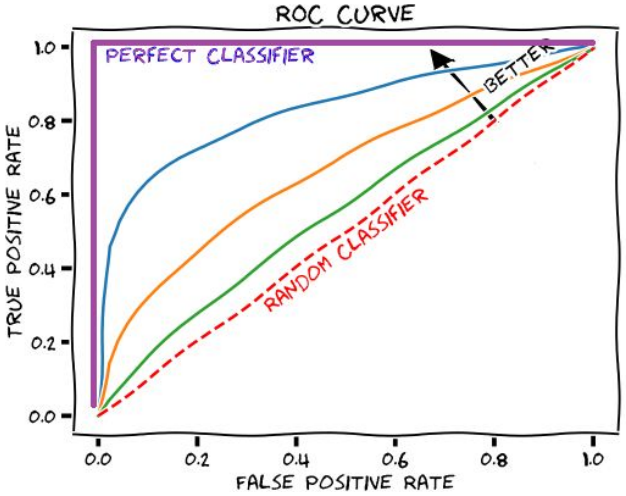


Umbral = 0.13
Accuracy = ?
Precision = ?
Recall = ?

|	Positivo (pred)	Negativo (pred)	Total
Positivo (obs)	11	0	11
Negativo (obs)	15	4	19
Total	26	4	30

Evaluando modelos de clasificación: ROC

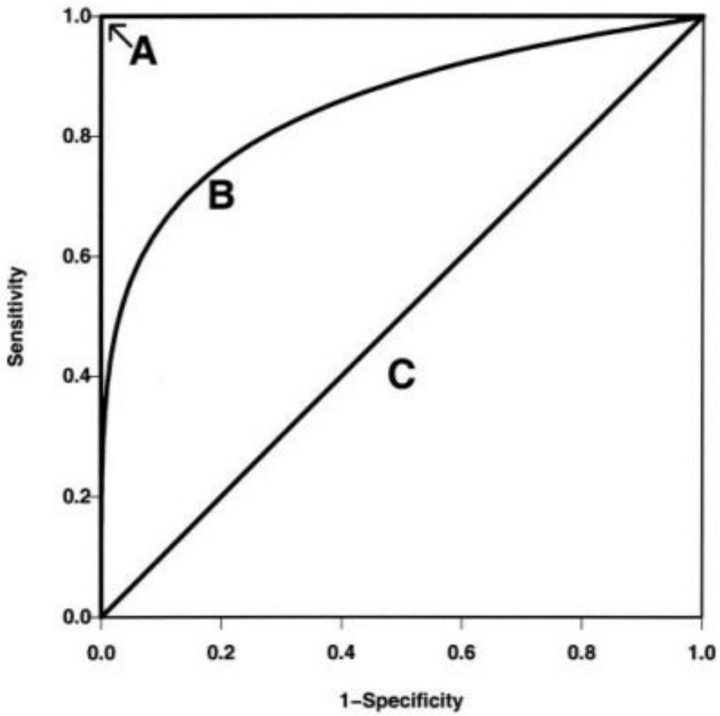
- Curva ROC y métrica AUC: cuantificar la performance agregada de diferentes clasificadores para todos los umbrales posibles
- Eje X: tasa de falsos positivos (1 - especificidad)
- Eje Y: tasa de verdaderos positivos (sensibilidad)
- Querríamos clasificadores que scoreen alto en Y y bajo en X

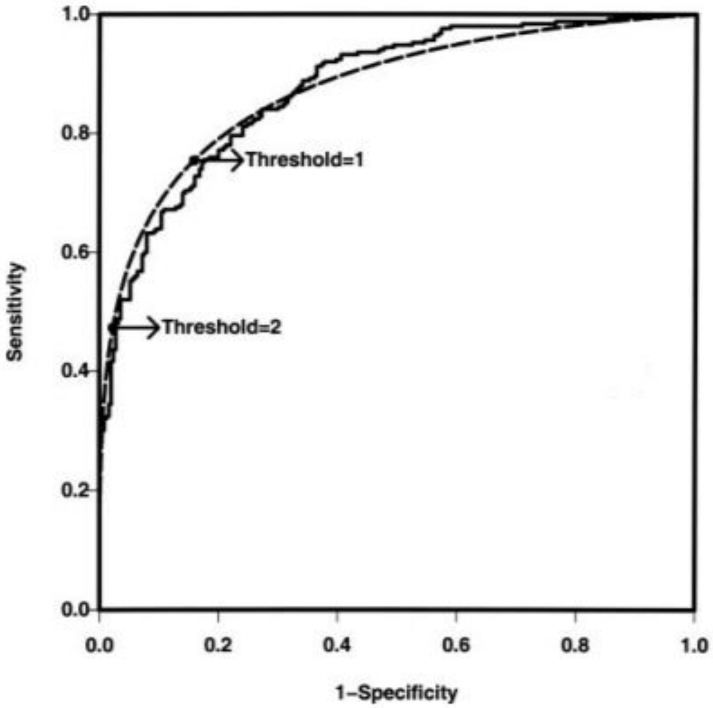


Evaluando modelos de clasificación: ROC

True positive rate (o recall)

$$\frac{tp}{tp + fn}$$

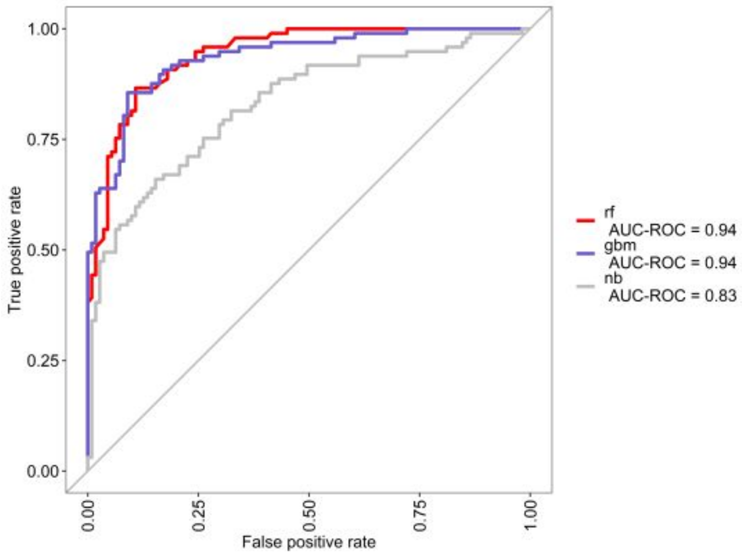




False positive rate

$$\frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

Evaluando modelos de clasificación: ROC



- En este ejemplo tenemos ploteados las curvas ROC de tres modelos de clasificación
- ¿Cuál es mejor? ¿Por qué?

Ahora, vamos a un modelo concreto…

Árboles de decisión

- Modelos de aprendizaje supervisado para la
  - regresión de variables numéricas
  - clasificación de variables categóricas
- Premisa básica: **dividir el espacio de predictores en segmentos más simples**
- Son fáciles de interpretar y simples, pero esto último también hace que tengan baja capacidad predictiva

Árbol de decisión intuitivo

Vamos a construir “a mano” el árbol de decisión para poder saber si podemos jugar o no al fútbol...

El set de entrenamiento es el de la tabla de la derecha.

Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Nublado	Alta	Alta	Débil	Si
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Baja	Normal	Fuerte	No
Nublado	Baja	Normal	Fuerte	Si
Soleado	Media	Alta	Débil	No
Soleado	Baja	Normal	Débil	Si
LLuvia	Media	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si
Nublado	Media	Alta	Fuerte	Si
Nublado	Alta	Normal	Débil	Si
LLuvia	Media	Alta	Fuerte	No

Árbol de decisión intuitivo

El modelo parte desde un nodo hoja inicial.

En primer lugar, deberíamos verificar si todos los registros pertenecen a la misma clase, ya que en ese caso deberíamos construir un nodo hoja con esa clase como etiqueta. Como no es el caso, particionamos por la variable Pronóstico.

Por cada partición creamos una arista (split) y un nodo hijo (internal node)

Ahora, **aplicamos recursivamente** el algoritmo. En primer lugar analizamos la **partición** correspondiente a **Pronóstico == "Nublado"**

Pronóstico	Temperatura	Humedad	Viento	Jugar
Nublado	Alta	Alta	Débil	Si
Nublado	Baja	Normal	Fuerte	Si
Nublado	Media	Alta	Fuerte	Si
Nublado	Alta	Normal	Débil	Si

Al analizar a qué clase de la variable que queremos predecir pertenecen los registros, vemos que **todos corresponden a "Sí".** Por lo tanto, **este será un nodo terminal con la etiqueta "Sí".**

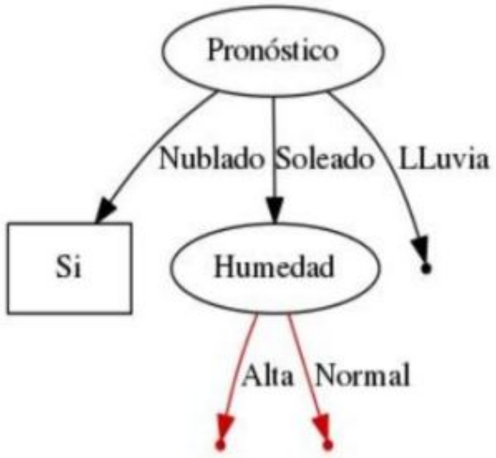
¿Qué pasa en la partición de **Pronóstico == "Soleado"**?

Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

Los registros pertenecen a **distintas clases**, por lo que vamos a tener que hacer una **nueva partición**

¿Por qué variable lo hacemos (Temperatura, Humedad o Viento)?

Si vemos los datos, nos conviene **abrir el nodo por Humedad**. Ya que si fuese por Viento o Temperatura, deberíamos hacer una partición más.



Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

Para la **sub-partición Humedad == "Alta", tenemos que todos los casos dan "No".**

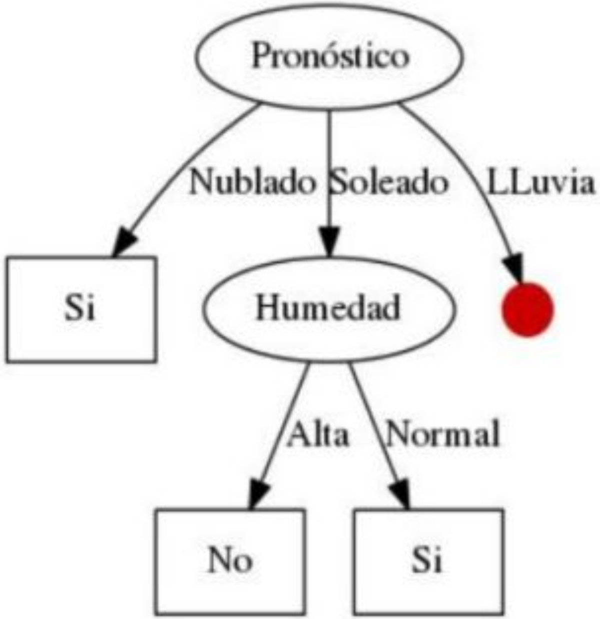
Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No

De manera que será un nodo terminal con la etiqueta "No".

Para la **sub-partición Humedad == “Normal”, todos los casos dan “Sí”.**

Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

Ahora solo resta analizar la partición correspondiente al **Pronóstico == "Lluvia"**



Pronóstico	Temperatura	Humedad	Viento	Jugar
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Baja	Normal	Fuerte	No
LLuvia	Media	Normal	Débil	Si
LLuvia	Media	Alta	Fuerte	No

De nuevo, como los registros pertenecen a **distintas clases**, vamos a tener que hacer una **nueva partición**

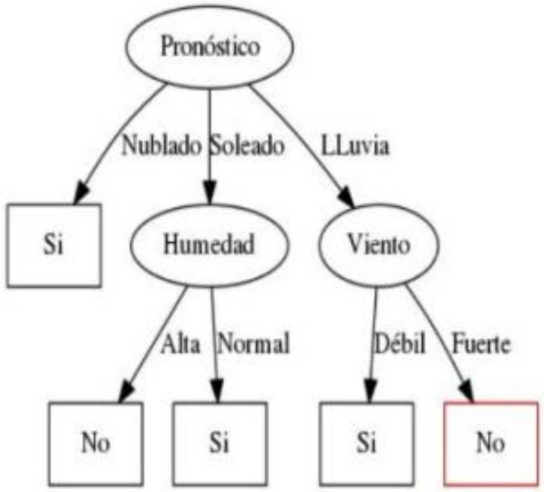
¿Por qué variable lo hacemos (Temperatura, Humedad o Viento)?

Siguiendo la lógica que veníamos trabajando, nos conviene usar como criterio de partición la variable **“Viento”**

Pronóstico	Temperatura	Humedad	Viento	Jugar
LLuvia	Media	Alta	Débil	Si
LLuvia	Baja	Normal	Débil	Si
LLuvia	Baja	Normal	Fuerte	No
LLuvia	Media	Normal	Débil	Si
LLuvia	Media	Alta	Fuerte	No

Hacemos las sub-particiones y definimos los dos nodos terminales según:

- Si el viento es débil → Sí
- Si el viento es fuerte → No



En base a lo aprendido en clases anteriores, **¿qué tipo de árbol de decisión era este?**

¿Un árbol de **regresión** o de **clasificación**?

Algoritmo general

Sea Dt el conjunto Tr-Set en un nodo t

- Si Dt contiene registros que pertenecen todos a la misma clase yt, luego t es un nodo hoja rotulado como yt
- Si Dt es un conjunto vacío, luego t es un nodo hoja rotulado por la clase default, yd
- Si Dt contiene registros que pertenecen a más de una clase, usar un test de atributo para separar los datos en subconjuntos más pequeños.
- Recursivamente aplicar el procedimiento a cada subconjunto

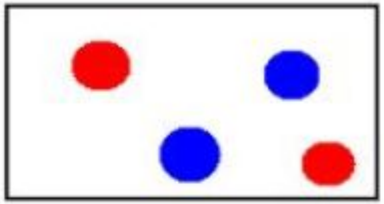
Árboles de clasificación

- Se predice que cada observación pertenezca a la clase que tiene la moda en las observaciones del set de entrenamiento.
- El primer problema fundamental aquí es: **¿Cómo elegir entre las particiones candidatas?**
    - Tenemos splits binarios
    - Splits pluricotómicos
    - Variables cuantitativas
    - Variables cualitativas
    - …

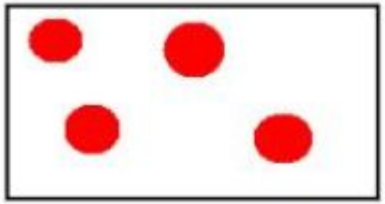
- El segundo problema: ¿cuándo dejamos de hacer crecer el árbol?

Ganancia / Pérdida de impureza

- **¿Cómo se elige entre particiones candidatas?**
  - Medidas de impureza del nodo → queremos que el nodo que separó la clasificación tenga impureza mínima



Impureza Máxima



Impureza Mínima

Ganancia / Pérdida de impureza

- Definamos p(i|t) como la probabilidad de la clase i en el nodo t (por ejemplo, la fracción de registros con la etiqueta i en el nodo t)

- Para un problema de clasificación binaria (0/1), la distribución máxima de impureza, donde ambas clases están presentes de igual manera, viene dada por la distribución:

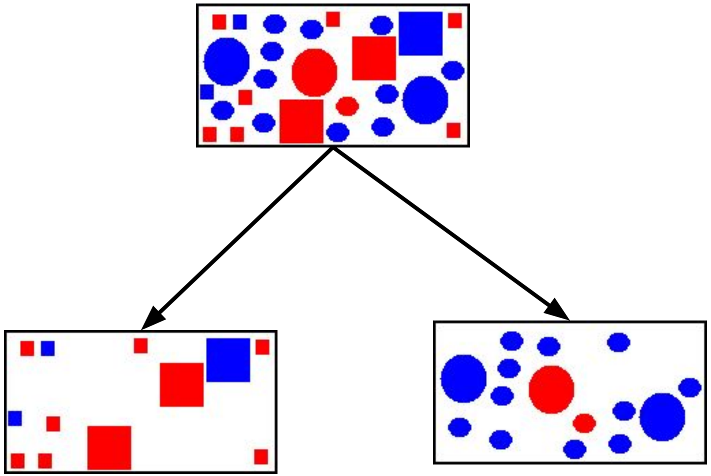
$$p(0|t) = p(1|t) = 0.5$$

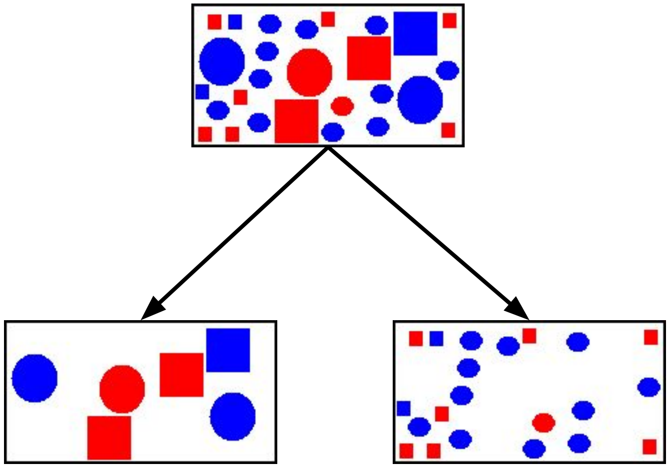
- La mínima de impu se obtiene cuando está presente sólo una clase, es decir:

$$p(0|t) = 1 - p(1|t) = 0 \vee 1$$

Ganancia / Pérdida de impureza

Siguiendo con el ejemplo de la clasificación según el color de las figuras. Si podemos seleccionar la forma y el tamaño, ¿qué partición conviene realizar?





Ganancia / Pérdida de impureza

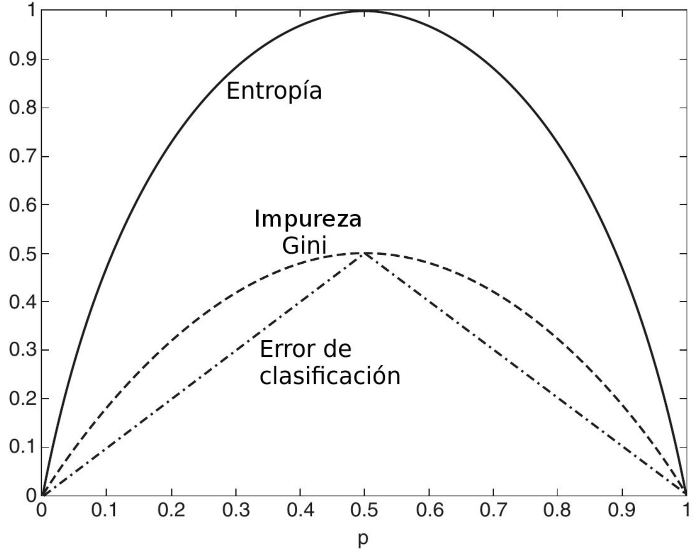
- **¿Cómo se elige entre particiones candidatas?**
  - Medidas de impureza del nodo → lo medimos con **funciones de pérdida**
    - **Classification error rate** → fracción de las observaciones del set de entrenamiento que no pertenecen a la clase con más ocurrencias
    - **Índice de Gini / Pureza de los nodos**→ un valor bajo indica que el nodo contiene de manera predominante observaciones de una única clase
    - **Entropía** → un valor bajo indica que el nodo contiene de manera predominante observaciones de una única clase

Ganancia / Pérdida de impureza

$$\text{Entropía}(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t)$$

$$\text{Impureza Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$

$$\text{Error de clasificación}(t) = 1 - \max_i [p(i|t)]$$



Ganancia / Pérdida de impureza

- Las medidas de impureza, pero por sí solas, no son suficientes para decirnos cómo funcionará la división. Todavía tenemos que mirar la impureza antes y después de la división. Podemos hacer esta comparación usando la ganancia:

$$\Delta = I(\text{padre}) - \sum_{j \in \text{hijos}} \frac{N_j}{N} I(\text{hijo}_j)$$

- Donde I es la medida de impureza, Nj es el número de registros en el nodo hijo j y N es el número de registros en el nodo padre.

- Cuando I es la entropía, esta cantidad se llama ganancia de información.

Un último ejemplo

PARTICION 1

P (Nodo padre)	
C1	6
C2	6
n	12
GINI	0,5

K1 (hijo 1)	
C1	5
C2	2
n_{k1}	7
$P(C1\|n_{k1})^2$	0,51
$P(C2\|n_{k1})^2$	0,08
SUMA	0,59
$GINI_{k1}$	0,41
$n_{k1}/n*GINI_{k1}$	0,24

K2 (hijo 2)	
C1	1
C2	4
n_{k2}	5
$P(C1\|n_{k2})^2$	0,04
$P(C2\|n_{k2})^2$	0,64
SUMA	0,68
$GINI_{k2}$	0,32
$n_{k2}/n*GINI_{k2}$	0,13333333

$GINI_{part1}$	0,371

PARTICION 2

P (Nodo padre)	
C1	6
C2	6
n	12
GINI	0,5

K1 (hijo 1)	
C1	7
C2	1
n_{k1}	8
$P(C1\|n_{k2})^2$	0,77
$P(C2\|n_{k2})^2$	0,02
SUMA	0,78
$GINI_{k1}$	0,22
$n_{k1}/n*GINI_{k1}$	0,15

K2 (hijo 2)	
C1	1
C2	3
n_{k2}	4
$P(C1\|n_{k2})^2$	0,0625
$P(C2\|n_{k2})^2$	0,5625
SUMA	0,625
$GINI_{k2}$	0,375
$n_{k2}/n*GINI_{k2}$	0,125

$GINI_{part2}$	0,271

Early Stopping / Pre-podado / Overfitting

- Podemos detenernos cuando todos los registros pertenecen a la misma clase, o cuando todos los registros tienen los mismos atributos. Esto es correcto en principio, pero probablemente conduciría a un sobreajuste.

- Prepoda: establecer un umbral mínimo en la ganancia, y detenerse cuando ninguna división logra una ganancia por encima de este umbral. Esto evita el sobreajuste, pero es difícil de calibrar en la práctica.

Early Stopping / Pre-podado / Overfitting

- Post-poda: construir el árbol completo, y luego realizar una poda
- Para podar un árbol, examinamos los nodos desde abajo hacia arriba y simplificamos las ramas del árbol (de acuerdo con algunos criterios).
- Los subárboles complicados pueden ser reemplazados con un solo nodo o con un subárbol más simple (secundario).

Árboles de regresión

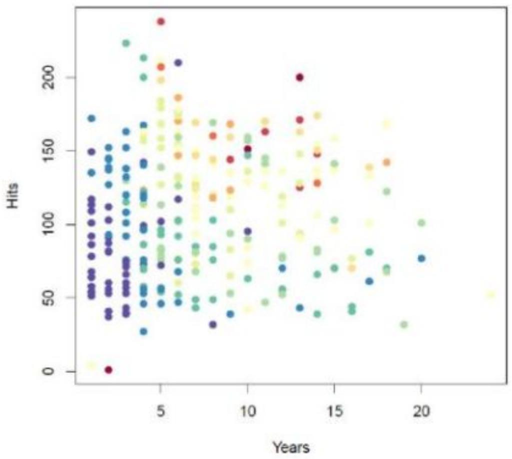
- Variable de resultado es cuantitativa
- La varianza de dicho valor en un nodo nos dá una medida de impureza de dicho valor.
- Error Cuadrático Medio como medida de impureza y la función a maximizar seguirá siendo la ganancia.

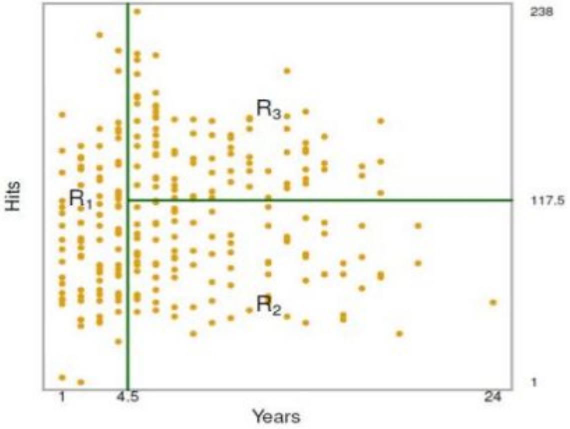
$$\Delta = ECM(\text{padre}) - \sum_{j \in \text{hijos}} \frac{N_j}{N} ECM(\text{hijo}_j)$$

- Objetivo: máxima ganancia, donde ECM es el Error Cuadrático Medio, Nj es el número de registros en el nodo hijo j y N es el número de registros en el nodo padre.

Árboles de regresión

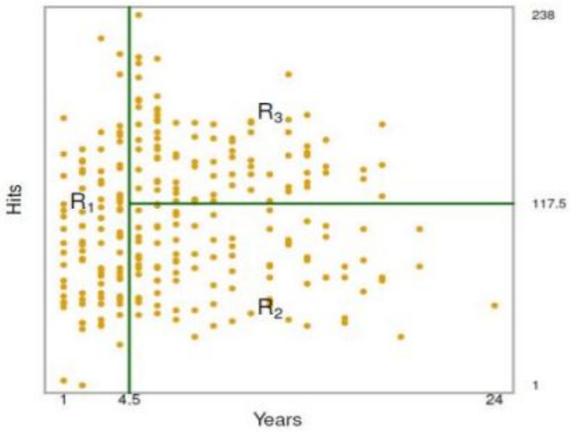
- Funcionan de manera parecida a los árboles de clasificación
- *Peeero* acá no trabajamos con moda, si no con la media
- Partimos el espacio predictor (es decir, los posibles valores de x) en una cantidad determinada de regiones (R1, R2, R3 (...))

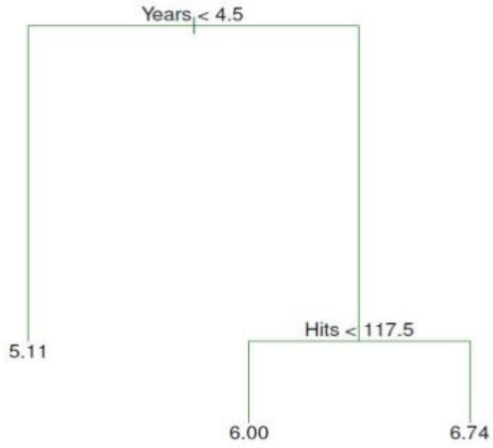




Árboles de regresión

- Para cada observación que cae en Rj, hacemos la misma predicción, lo que quiere decir que hacemos la media de valores predichos para las observaciones del training set en Rj





¿Cómo se construyen las regiones?

- En teoría las regiones R1,...,RJ podrían tener cualquier forma.
- Elegimos dividir el espacio de predictores en rectángulos o cajas en varias dimensiones por simplicidad y facilidad de interpretación del modelo predictivo resultante. El objetivo es encontrar cajas R1,...,RJ que minimizan la suma de **residuos al cuadrado (RSS)** dada por

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

¿Cómo se construyen las regiones?

- Problema: no es factible computacionalmente considerar todas las posibles particiones del espacio de atributos en J cajas.
- Enfoque de arriba hacia abajo “greedy” que es conocido como recursive binary splitting.
- **Recursive binary splitting** comienza en la parte de arriba del árbol (donde todas las observaciones pertenecen a una sola región) y sucesivamente particiona el espacio de predictores.
- **Greedy** porque en cada paso de la construcción del árbol se busca la mejor división en ese punto en particular (**óptimo local**) en lugar de mirar hacia adelante y elegir una división que llevaría a un mejor árbol en un paso futuro.

Overfit

- En general, un árbol de decisión muy complejo que optimice las métricas puede terminar en sobreajuste (por ejemplo: una división con un solo resultado por registro).
  - Restringir el algoritmo a únicamente divisiones binarias (CART).
  - Utilizar un criterio de división que penalice explícitamente el número de resultados

Overfit

- Divisiones que penalicen explícitamente el número de resultados
    - **Prepoda:** se definen hiperparámetros que van a detener el crecimiento del árbol antes de que alcance su máxima profundidad. En cada partición del árbol, se evalúa por cross-validation la ganancia. Cuando se llega a un punto donde las particiones no mejoran, se detiene el crecimiento del árbol.
    - **Post-poda:** construir el árbol completo, y luego realizar una poda. Examinamos los nodos desde abajo hacia arriba y simplificamos las ramas del árbol (de acuerdo con algunos criterios).

¿Cómo se elige el nivel de complejidad?

- Tenemos que ver las diferencias entre las métricas de train y test con **cross-validation**
  - Cantidad de particiones
  - De nodos terminales
  - Umbral para la métrica de impureza
- Vamos a ver esto en la parte práctica, con R y tidymodels

¿Cuáles son los pros y contras de los árboles?

Pros

- Son muy interpretables y fáciles de explicar
- Podemos manejar variables cualitativas sin tener que hacer variables dummy (0/1)

Contras

- No tienen muy buena capacidad predictiva
- Son poco robustos. Esto quiere decir que un cambio muy pequeño en los datos de entrenamiento, puede cambiar radicalmente el modelo final