

Intro a ML

+

Árboles de decisión

M4 - DCSCyHD

¿Qué es Machine Learning?

- ¿Podría una computadora ir más allá de “lo que sea que sepamos decirle que haga” y realmente “aprender” por su cuenta como realizar una determinada tarea?
- ¿Podría ser posible el aprendizaje automático de estas reglas a partir de los datos?
- Spoiler → sí, en cierta medida
- Podemos empezar diciendo que ML implica usar ciertas herramientas computacionales y estadísticas, para abordar ciertos problemas de aprendizaje automático...

¿Qué problemas se pueden abordar?

- Riesgo de cáncer prostático
- Predicción de ataques cardíacos en función de variables demográficas, orgánicas, médicas, etc.
- Generar un detector de spam automático para tu cuenta de correo
- Reconocer dígitos en números manuscritos
- Establecer relaciones entre el ingreso y variables demográficas
- Clasificar imágenes satelitales
- Identificar tópicos en corpus textuales
- Clasificar textos según su sentimiento -positivos o negativos

¿Qué es Machine Learning?

- Las herramientas de machine learning pueden clasificarse en dos grandes grupos:
 - **aprendizaje supervisado:**
 - tenemos una variable “objetivo” que queremos predecir (output o Y) en base a una o más variables (inputs o X)
 - en problemas de regresión, Y es cuantitativa
 - en problemas de clasificación, Y es cualitativa
 - Algunos ejemplos: KNN, Naive Bayes (M3), Árboles de decisión (Empezamos hoy!)

¿Qué es Machine Learning?

- Las herramientas de machine learning pueden clasificarse en dos grandes grupos:
 - **aprendizaje no supervisado:**
 - no buscamos predecir una variable Y, el objetivo es descubrir patrones en los datos
 - detectar subgrupos homogéneos entre las observaciones
 - segmentar los registros según sus features
 - Algunos ejemplos: Clustering, PCA

Flujo de trabajo ML en aprendizaje supervisado

- Si bien existen distintos modelos de ML, en líneas generales el workflow siempre es:
 - Hago una partición de entrenamiento/validación de la base con la que voy a trabajar
 - Entreno el modelo con el dataset de entrenamiento
 - Evalúo y pruebo distintos hiperparámetros (ya vamos a ver qué son)
 - Fiteo el modelo a los datos
 - Predigo sobre el set de validación
 - Evalúo la performance del modelo en el set de validación → con distintas métricas

Métricas de evaluación para variables continuas

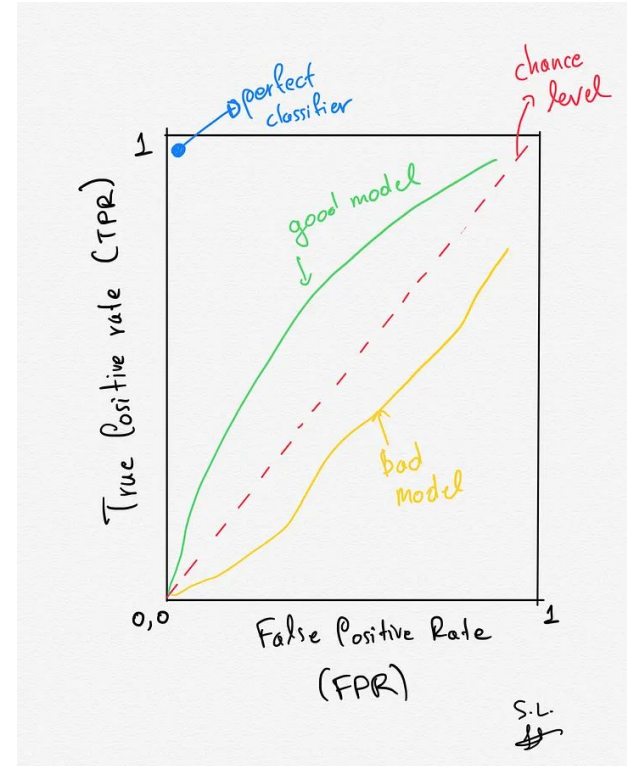
Medida	Tipo de modelo	Explicación
Root-mean-square deviation (RMSE)	Regresión	Raíz del error cuadrático medio, muestra la diferencia promedio entre los valores predichos y los reales.
Mean Absolute Error (MAE)	Regresión	Diferencia entre los valores predichos por el modelo y los valores reales, expresado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

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.

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).



Resumen

- Machine Learning
 - Aprendizaje supervisado
 - Uso variable(s) X para predecir Y
 - Divido mi base en train/test, armo el modelo en base a train y veo qué tan bien funciona prediciendo sobre tes
 - Aprendizaje no supervisado
 - No busco predecir, busco encontrar patrones interesantes en los datos

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 golf.

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”?

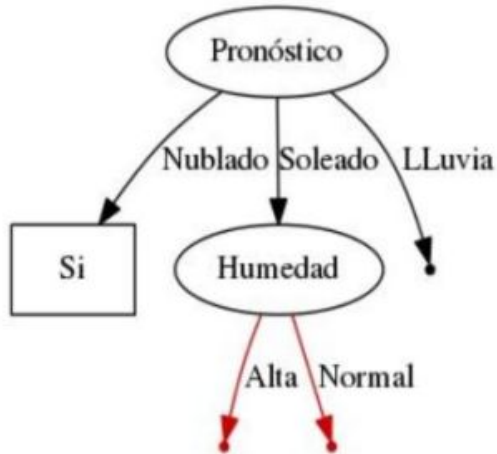


Pronóstico	Temperatura	Humedad	Viento	Jugar
Soleado	Alta	Alta	Débil	No
Soleado	Alta	Alta	Fuerte	No
Soleado	Media	Alta	Débil	No
Soleado	Baja	Normal	Débil	Si
Soleado	Media	Normal	Fuerte	Si

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 <input type="radio"/>	Débil	No <input type="radio"/>
Soleado	Alta	Alta <input type="radio"/>	Fuerte	No <input type="radio"/>
Soleado	Media	Alta <input type="radio"/>	Débil	No <input type="radio"/>
Soleado	Baja	Normal <input checked="" type="radio"/>	Débil	Si <input checked="" type="radio"/>
Soleado	Media	Normal <input checked="" type="radio"/>	Fuerte	Si <input checked="" type="radio"/>

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**

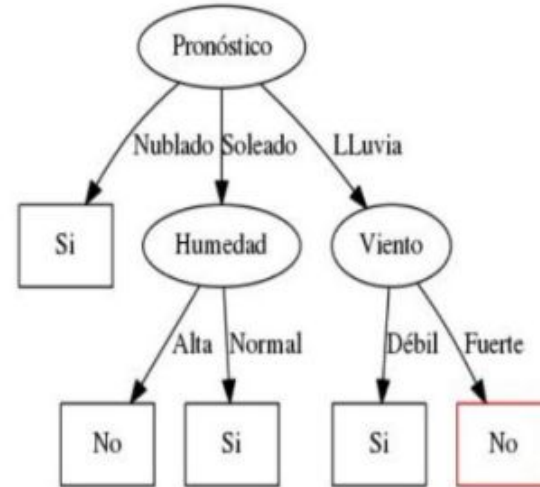
¿Por qué variable lo hacemos (Temperatura, Humedad o Viento)?

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 <input type="radio"/>	Si <input type="radio"/>
LLuvia	Baja	Normal	Débil <input type="radio"/>	Si <input type="radio"/>
LLuvia	Baja	Normal	Fuerte <input checked="" type="radio"/>	No <input checked="" type="radio"/>
LLuvia	Media	Normal	Débil <input type="radio"/>	Si <input type="radio"/>
LLuvia	Media	Alta	Fuerte <input checked="" type="radio"/>	No <input checked="" type="radio"/>

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?**

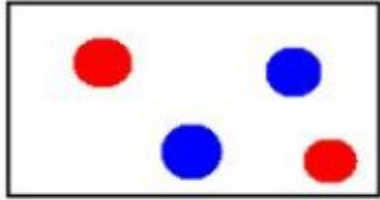
¿Un árbol de **regresión** o de **clasificación**?

Árboles de clasificación

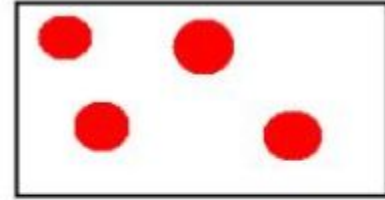
- Se predice que cada observación pertenezca a la clase que tiene la moda en las observaciones del set de entrenamiento.
- El problema fundamental aquí es: **¿Cómo elegir entre las particiones candidatas?**
 - Tenemos splits binarios
 - Splits pluricotómicos
 - Variables cuantitativas
 - Variables cualitativas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

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



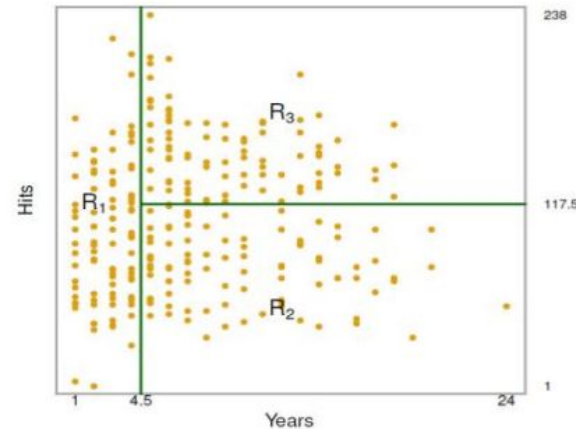
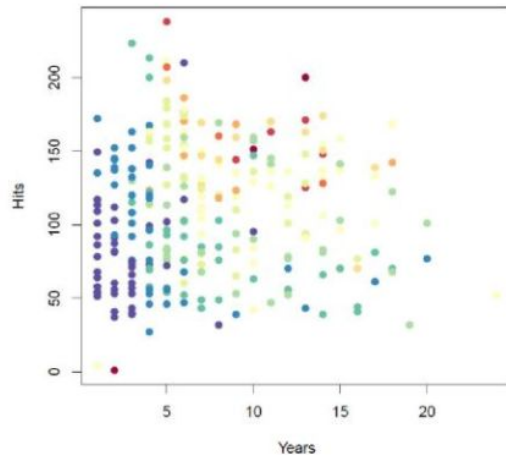
Impureza Mínima

Ganancia / Pérdida de impureza

- ¿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

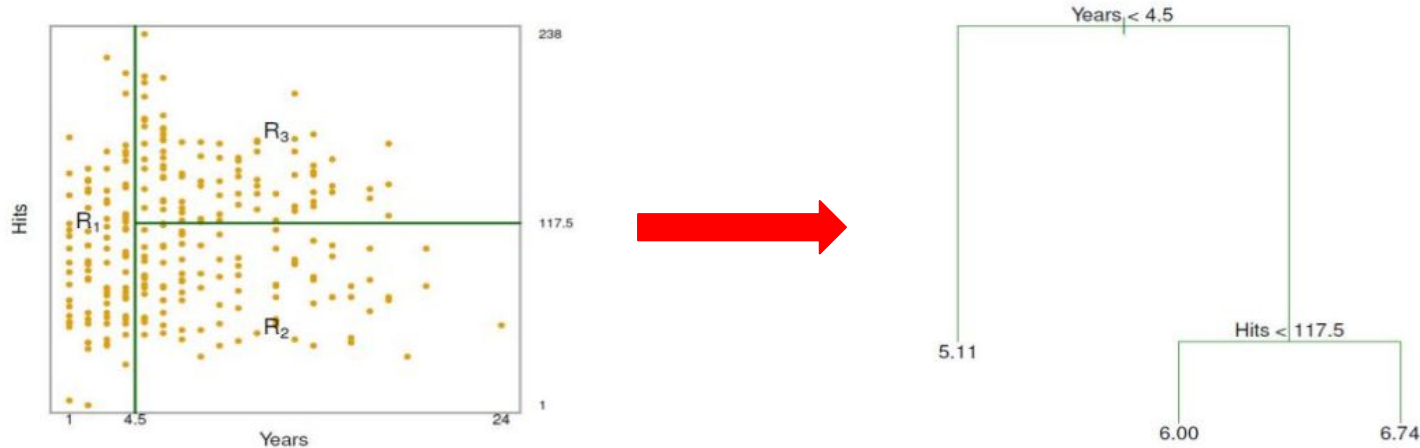
Árboles de regresión

- Funcionan de manera parecida a los árboles de clasificación
- *Pero* 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 (R_1, R_2, R_3, \dots)



Árboles de regresión

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



¿Cómo se construyen las regiones?

- En teoría las regiones R_1, \dots, R_J 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 R_1, \dots, R_J 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