

Árboles de decisión

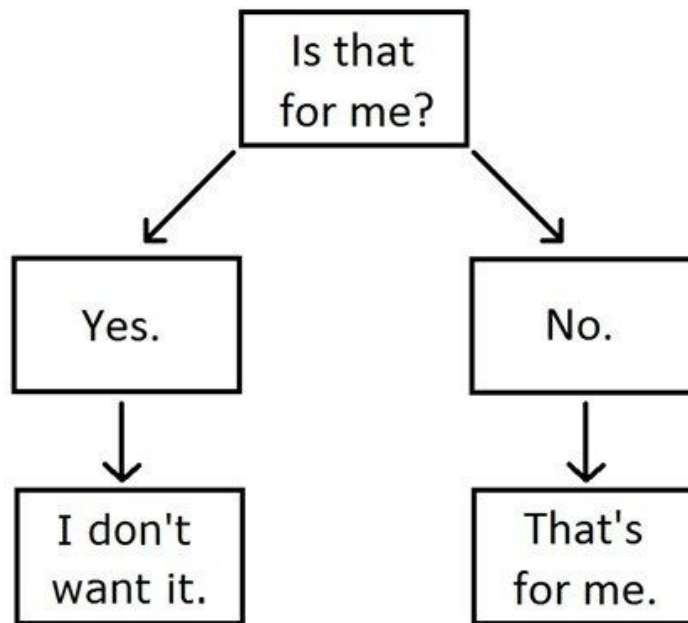
Verano 2024
Paula Pérez Bianchi



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

My Cat's Decision-Making Tree.

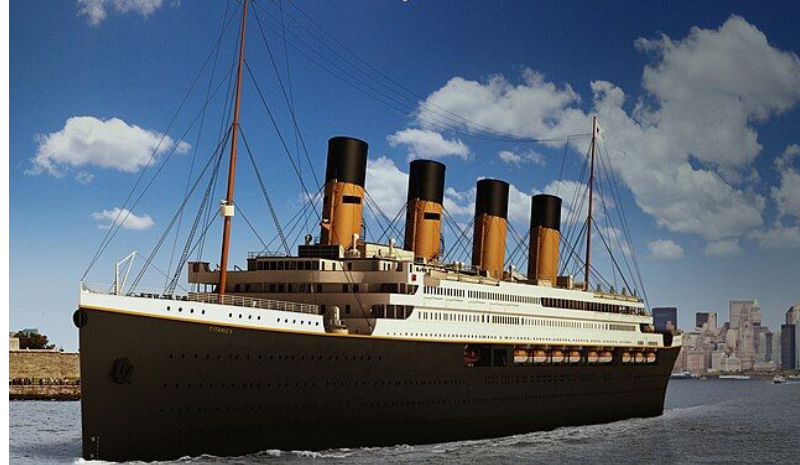


Fuente de datos: **Titanic**

El Titanic se hundió en Abril de 1912.

Tenía la fama de ser “Inhundible” pero chocó con un iceberg y no le alcanzó con la fama.

Como no había suficientes salvavidas 1502 de 2224 pasajeros y personal de a bordo murieron.



Dataset en:

<https://www.kaggle.com/competitions/titanic>

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare
	1	2	1 female	38.0	1	0	71.2833
	3	4	1 female	35.0	1	0	53.1000
	6	7	1 male	54.0	0	0	51.8625
	10	11	3 female	4.0	1	1	16.7000
	11	12	1 female	58.0	0	0	26.5500

Objetivo: ¿Podemos predecir quienes sobrevivieron al titanic?

Manos a los datos!

Titanic - ¿Qué características tenían los pasajeros que sobrevivieron?



Consigna 1 - ¡Exploren los datos y escribanlas en un papel (o grafíquenlas)! (10 min)

```
X.head()
```

✓ 0.4s

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare
1	2	1	female	38.0	1	0	71.2833
3	4	1	female	35.0	1	0	53.1000
6	7	1	male	54.0	0	0	51.8625
10	11	3	female	4.0	1	1	16.7000
11	12	1	female	58.0	0	0	26.5500

Algunas preguntas...

- ¿Cuántos pasajeros de primera clase había? ¿Y de segunda y tercera?
- ¿Cuál era la proporción de niños?
- ¿Quiénes les parece que tenían prioridad en los botes de rescate?

Se nos perdió una etiqueta, ¿la podemos deducir?

Tenemos los datos de una pasajera,
¿me pueden decir si sobrevivió o no?



	Pclass	Sex	Age	SibSp	Parch	Fare	Survived
184	1	female	27.0	1	1	247.5208	???

ATTENTION!

No vale buscar en el dataset
(además no está)

*¿Adivinaron o usaron reglas? ¿Cuáles? ¿Se pueden
generalizar?*

!!!Competencia!!!

Ahora les voy a dar los datos de **10 pasajeros**,
¿Pueden responder cuáles sobrevivieron?

Consigna

- 1) Armen un conjunto de reglas generales y generen etiquetas para estos 10 pasajeros.

```
def clasificador_naive_instance(x):  
    if REGLA:  
        return True  
    return False
```

- 2) Subir sus predicciones y el código que usaron para sus reglas a un google forms.

<https://forms.gle/UYfUge87zwcXyWTt9>



Tienen 15 min!

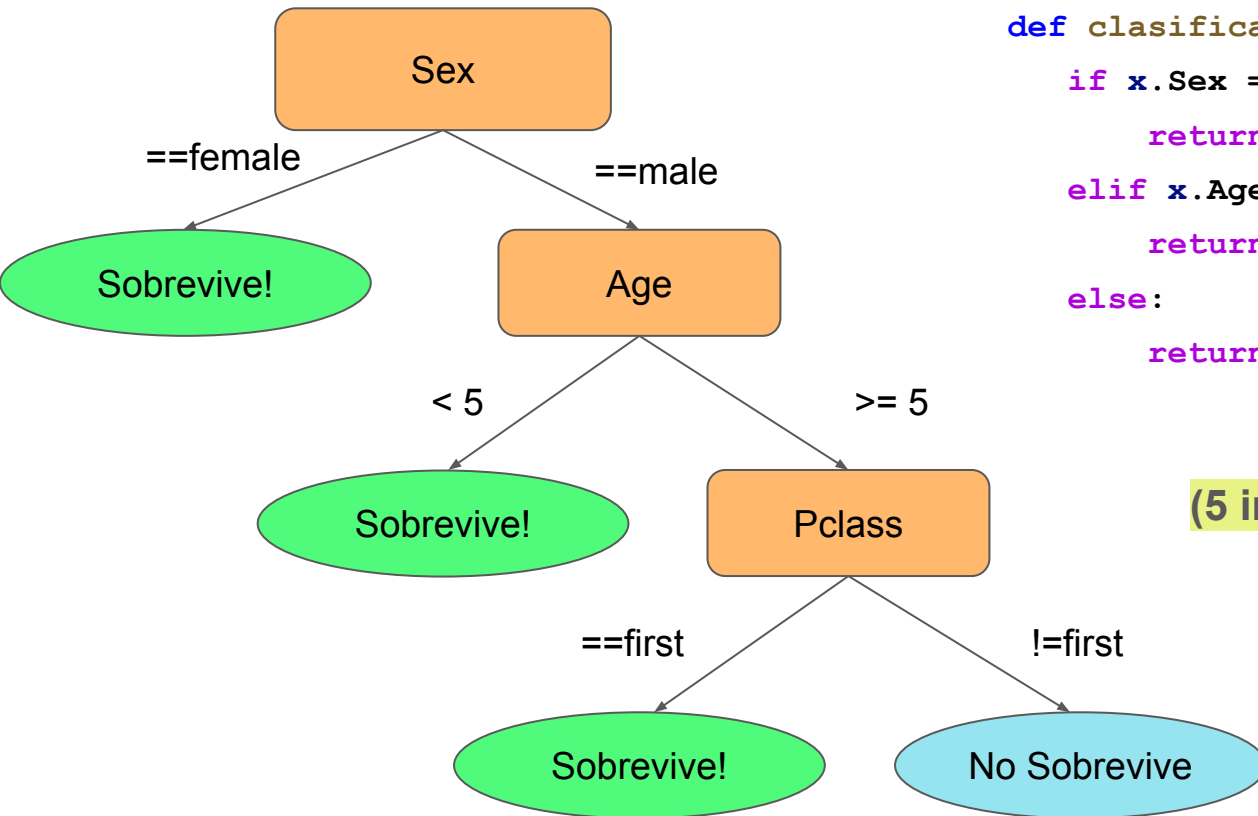
<https://www.online-stopwatch.com/timer/15minute/>



→ Sobrevivió (True)



→ No sobrevivió (False)



```
def clasificador_naive_instance(x):  
    if x.Sex == "female":  
        return True  
    elif x.Age < 5 or x.Pclass == 1 :  
        return True  
    else:  
        return False
```

SCORE : 50%
(5 instancias bien clasificadas)

*Cada nodo interno evalúa un atributo discreto a Cada
rama corresponde a **un valor/umbral** para a
Cada hoja predice un valor de **Y***

Esto que hicimos a ojo se puede formalizar como un
método de Aprendizaje Supervisado

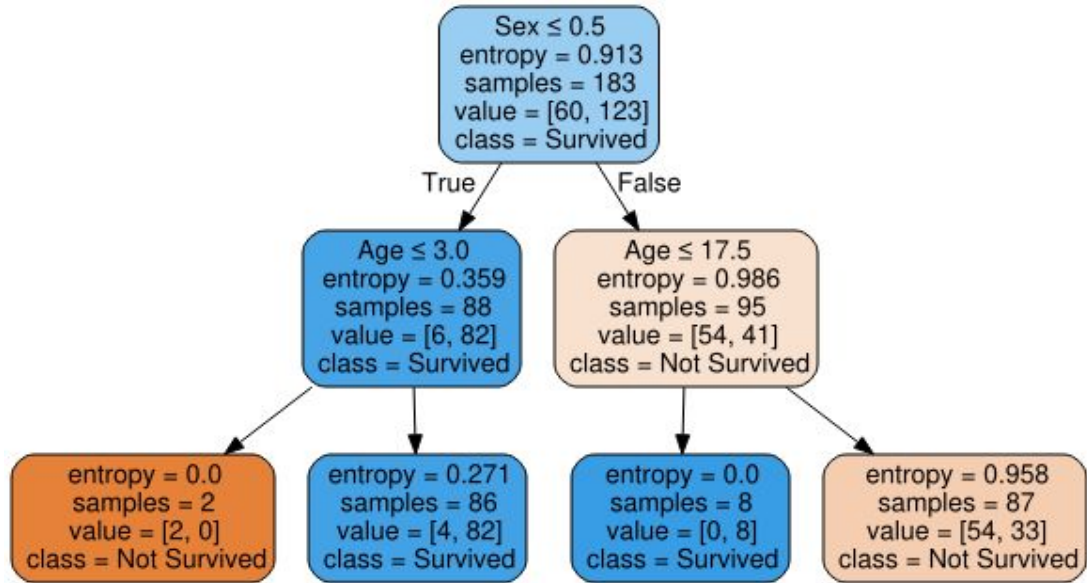
Árboles de decisión

Es un modelo de AA utilizado que puede ser utilizado para **clasificación**.

Se basa en el armado de una **jerarquía de reglas**. Estas las podemos expresar usando una fórmula lógica de ANDs y ORs. Es decir que el árbol representa una **disyunción de conjunciones sobre valores de atributos**. Podemos pensar el armado del árbol como jugar al ¿Quién es quién?

En el ejemplo anterior la fórmula quedaría:
 $(\text{Sex} == \text{Female}) \vee (\text{Age} < 5) \vee (\text{Age} \geq 5 \wedge \text{Pclass} == 1)$

Los árboles son un **modelo altamente interpretable**. Es decir que dada una predicción particular podemos entender por qué el modelo la generó. Sólo hay que mirar la rama de la hoja correspondiente a la predicción.



¿Cómo se hace esto en Python?



```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)
```

[\[source\]](#)

```
from sklearn.tree import DecisionTreeClassifier, plot_tree,
export_graphviz
```

```
arbol = DecisionTreeClassifier(criterion="entropy",
                               max_depth= 2)
```

```
arbol.fit(X, y) #Entrenamiento
```

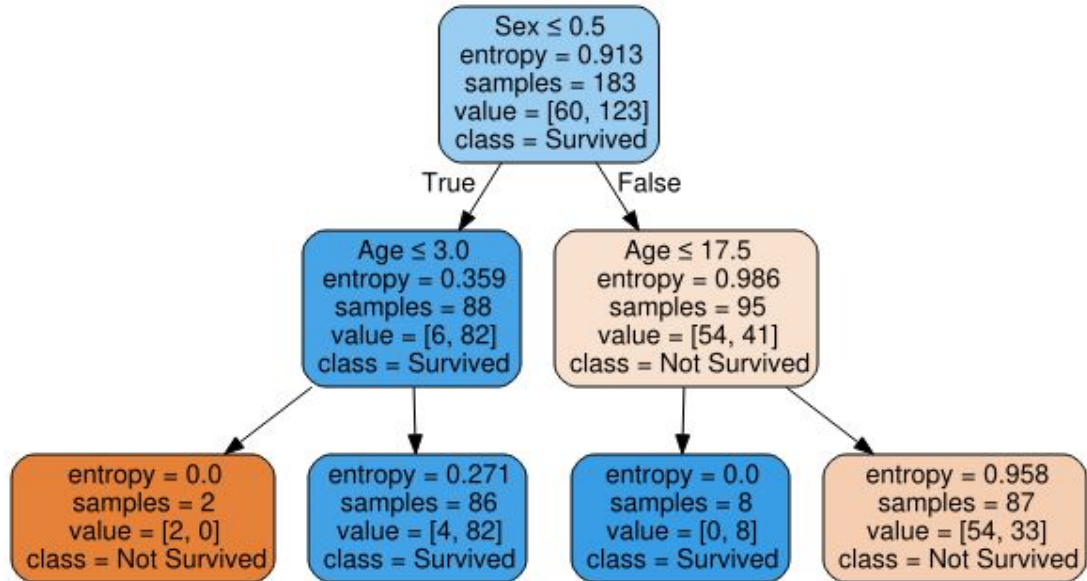
```
prediction = arbol.predict(X) #Generamos las predicciones
# Generamos el grafo del árbol
```

Hiperparámetros

→ Son los parámetros que nos permiten configurar el modelo. Por ejemplo, la altura máx del árbol de decisión.

¿Cómo queda el árbol?

- Miremos la información que aparece en cada nodo
- ¿Qué expresa **value**? (0 == Not Survived, 1 == Survived)



```
import graphviz
```

```
dot_data = export_graphviz(arbol, out_file=None, feature_names= X.columns,  
                           class_names= ["Not Survived", "Survived"],  
                           filled=True, rounded=True,  
                           special_characters=True)
```

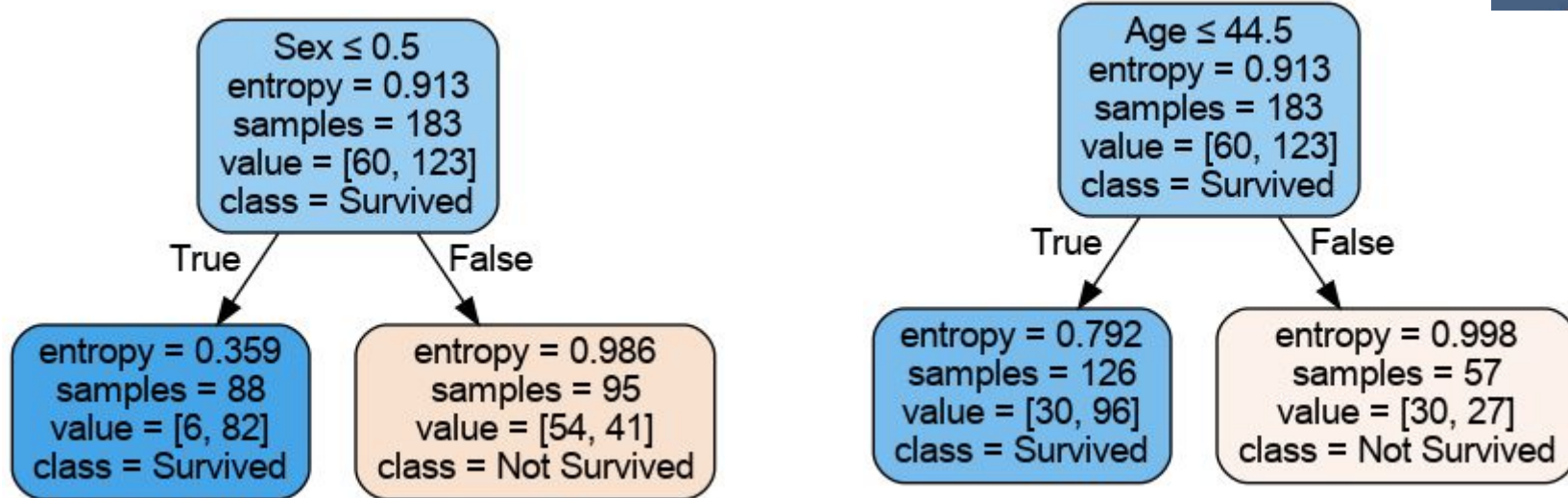
```
graph = graphviz.Source(dot_data) #Armamos el grafo
```

```
graph.render("titanic", format= "png") #Guardar la imagen
```

¿Cómo armamos el árbol?

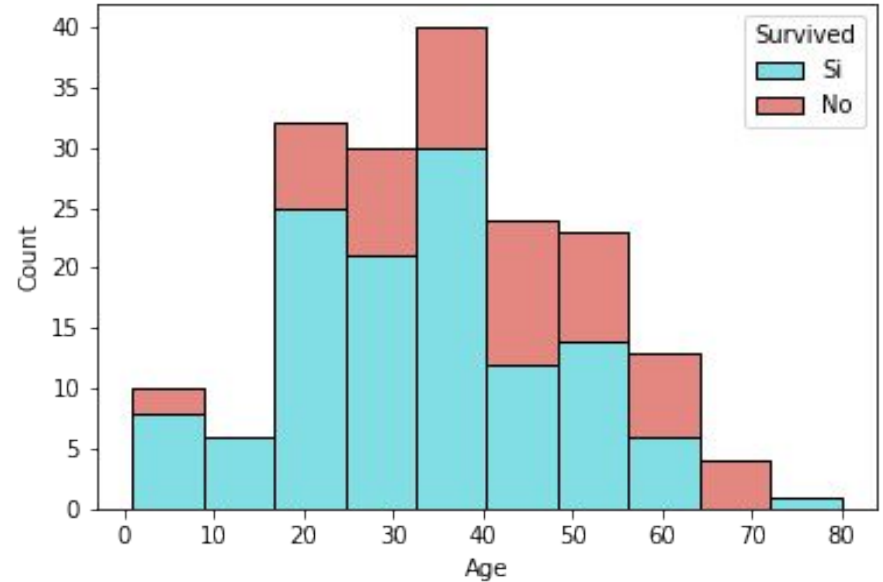
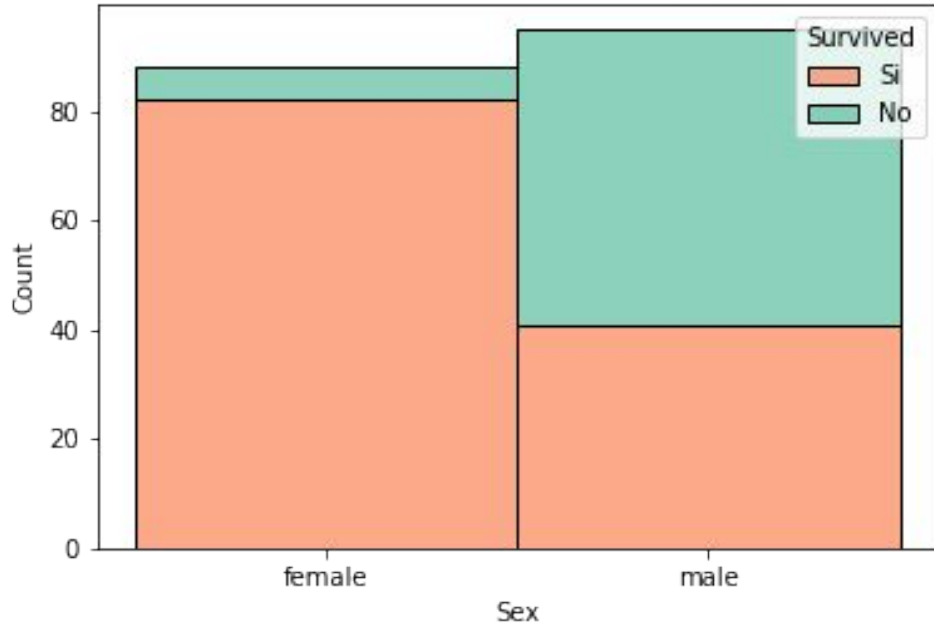
¿Qué atributo usamos para el primer corte?

Comparemos los árboles luego de cortar por género y edad. ¿Cuál es mejor?



Discutan 5 minutos!

¿Cómo se ven esos cortes?
¿Dónde los podemos visualizar?



Inducción Top-Down para árboles de decisión

(versión simplificada de ID-3 y C4.5 (Quinlan))

Input: S un conjunto de instancias con atributos A .



- 1) Elegimos $a \in A$, el atributo que produce el “mejor corte” de S para el **nodo_actual**
- 2) Para cada valor v_i posible de a , crear un nuevo hijo del **nodo_actual**
- 3) Clasificar (repartir) las instancias en los nuevos nodos, según el valor de a .
$$S_i \leftarrow \{ x \mid x \in S \wedge x[a] = v_i \}$$
- 4) Si las instancias están bien clasificadas o se cumple algún *criterio de corte*: **TERMINAR**
Si no: Iterar sobre los hijos del nodo actual con $A_i \leftarrow A - \{a\}$.

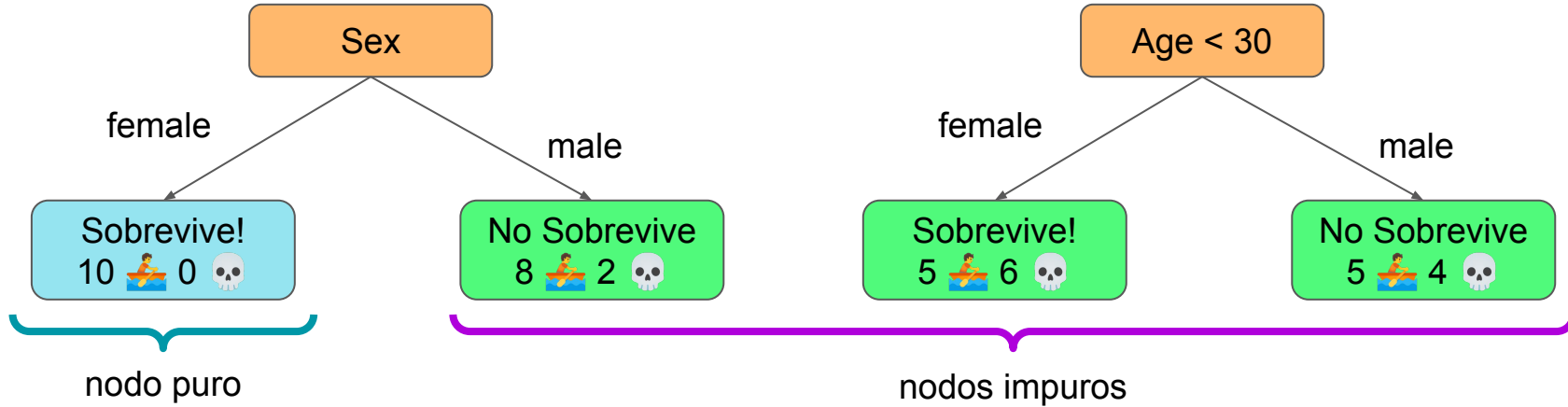
¿Cómo definimos el mejor corte?
→ Necesitamos métricas!!



Para pensar: ¿Cómo hacemos el paso 2) para atributos continuos? ¿Cómo se definen las regiones del paso 3) ¿en ese caso?

Árboles de decisión - Medidas de impureza

 → Sobrevivió
 → No sobrevivió



⇒ Luego, el primer árbol tiene un mejor corte que el segundo. Pero ... *¿Qué lo hace mejor?*
¿Cómo medirían esto? (Piensen 5min)

$$\Delta M(S, c) = M(S) - (Prop_{c, \leq} * M(S_{c, \leq}) + Prop_{c, >} * M(S_{c, >}))$$

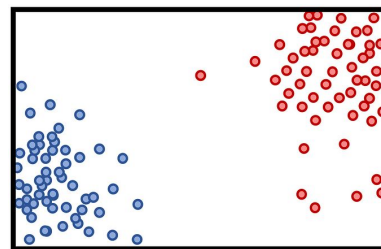
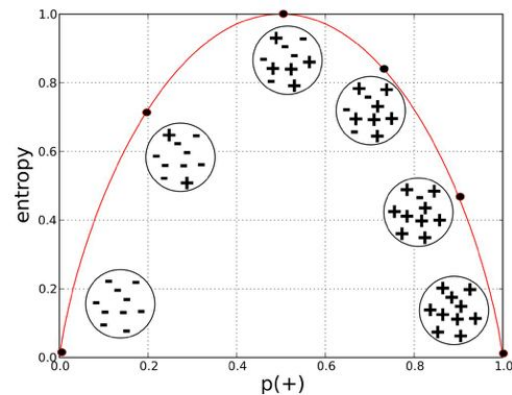
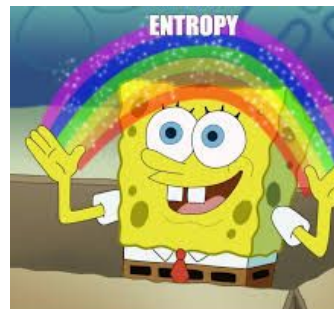
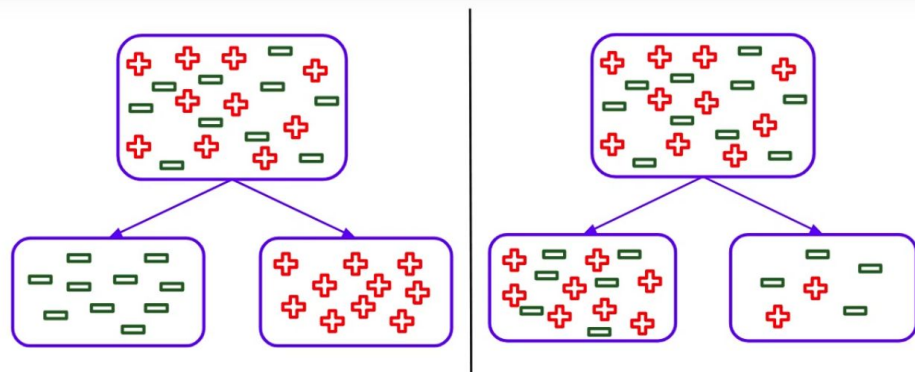
*“cuánto gano si divido a **S** en regiones **S_{c, ≤}** y **S_{c, >}** según la medida **M**.” (caso binario)*

Árboles de decisión - Medidas de impureza

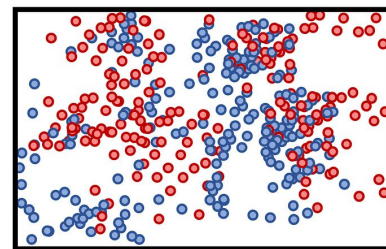
Entropía

Mide el **nivel promedio** de "información", "sorpresa" o "incertidumbre" inherente a los posibles resultados de una variable aleatoria.

$$H(S) = - \sum_{k \in \text{clases}(S)} p_s(k) \log_2 p_s(k) = \mathbb{E}[-\log(p_s(k))]$$



Low Entropy



High Entropy

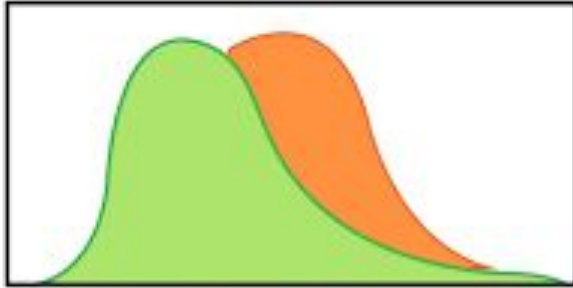
Árboles de decisión - Medidas de impureza

Info Gain (Ganancia de Información)

Cuánta entropía removemos al hacer un corte.

$$InfoGain(S, \langle a, c \rangle) = H(S) - (Prop_{\leq} \cdot H(S_{\leq}) + Prop_{>} \cdot H(S_{>}))$$

$$H(S) = - \sum_{k \in \text{clases}(S)} p_s(k) \log_2 p_s(k)$$



Low information gain High entropy



High information gain Low entropy

Árboles de decisión - Medidas de impureza

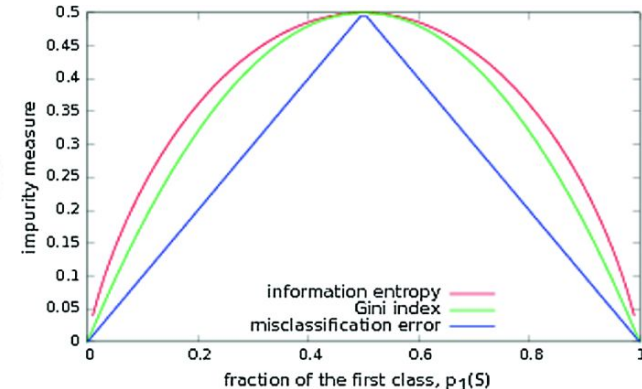
Gini Gain

La Impureza de Gini mide la *probabilidad de que una instancia particular sea clasificada erróneamente* si esta fuese etiquetada aleatoriamente de acuerdo con la distribución de clases dentro de la región



$$Gini(S) = 1 - \sum_{k \in \text{clases}(K)} (p_s(k))^2$$

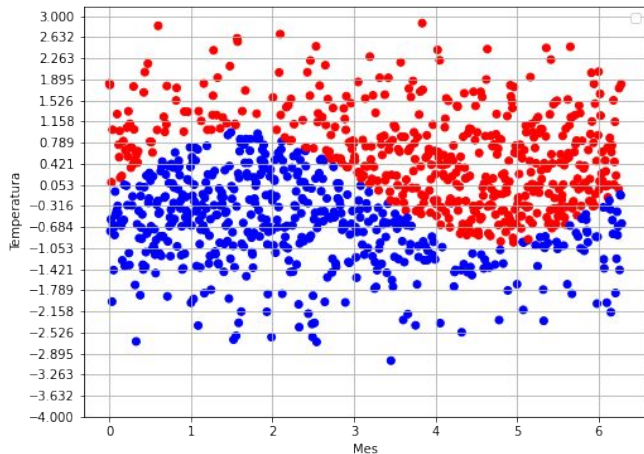
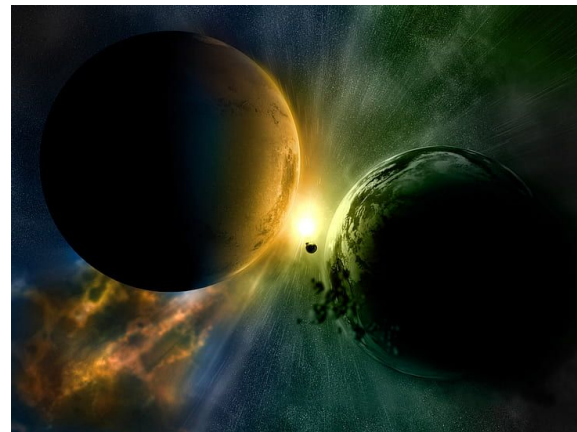
$$Gini_Gain(S, \langle a, c \rangle) = Gini(S) - (Prop_{\leq} \cdot Gini(S_{\leq}) + Prop_{>} \cdot Gini(S_{>}))$$





Miremos estos datos

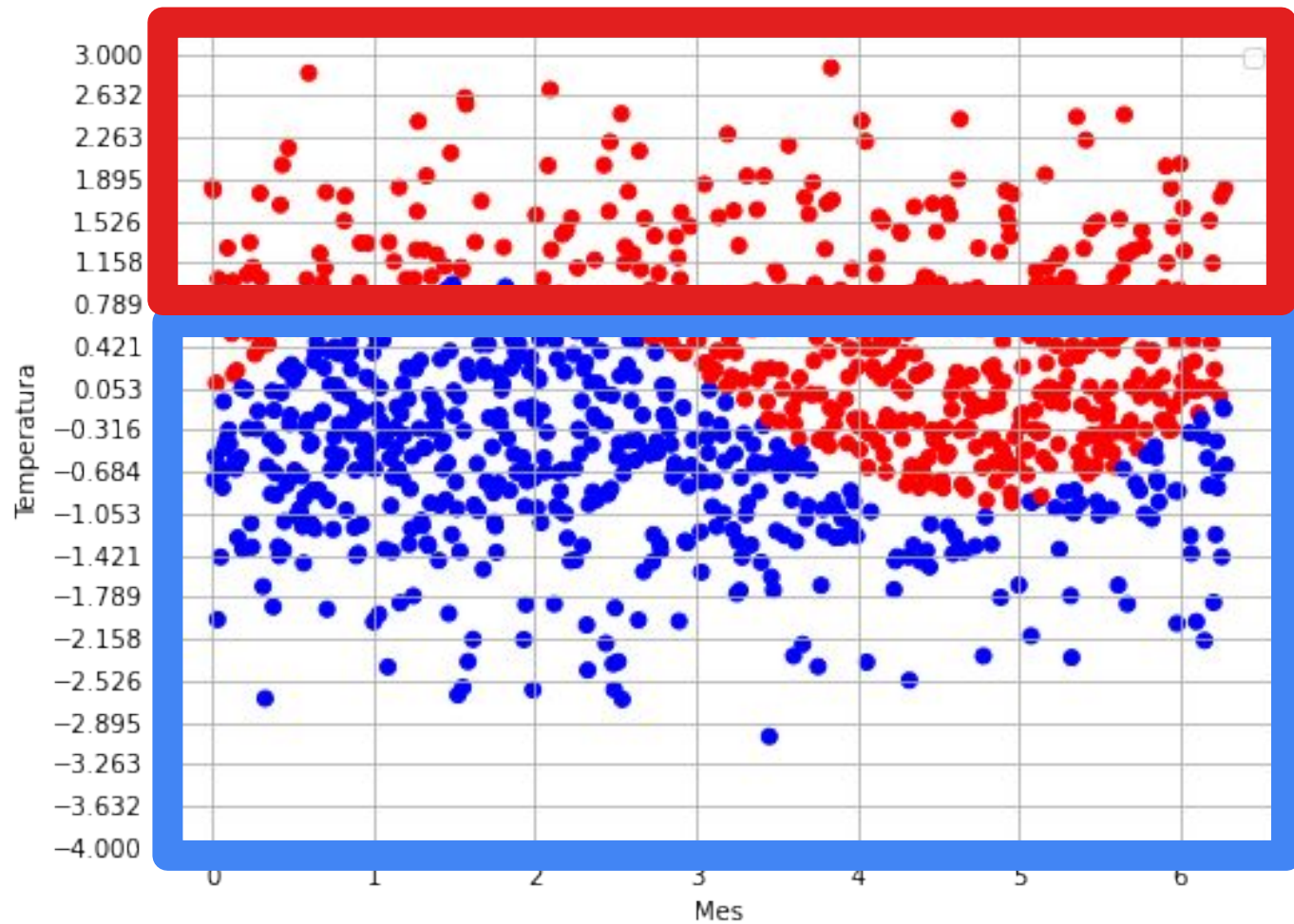
En el año 2.100 descubrimos dos nuevos planetas en el sistema solar. Los dos muy lejanos pero ambos tardan el mismo tiempo en dar una vuelta al sol. Enviamos astronautas a ambos planetas y cada día ellos nos envían sus mediciones de temperatura. Desde el planeta Tierra recibimos una señal con **(Temperatura, Planeta)**. Luego de algunos meses (pocos pero suficientes para observar una vuelta al sol) por alguna razón las señales de ambos planetas llegan cortadas, por lo que sólo recibimos el dato de la temperatura pero no a qué planeta corresponde. ¿Podremos a partir de los datos que tenemos predecir de qué planeta provienen los datos de temperatura recibidos?

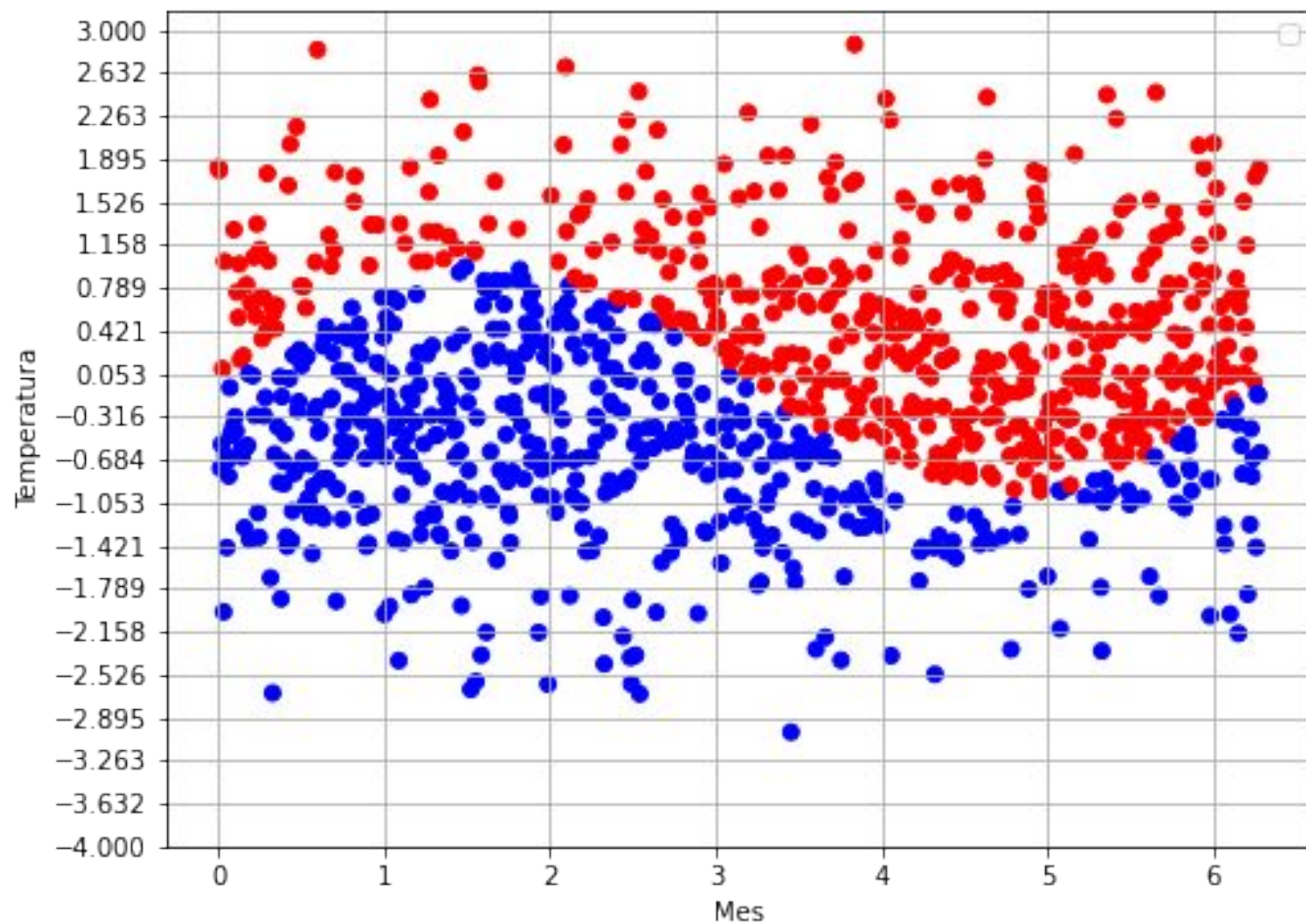


- ¿Cómo serían los cortes de un árbol de decisión?
- ¿Les parece que estos datos son separables con un árbol de altura 2?

Consigna: (5min)

- Traten de separar estos datos cortando la región como lo haría un árbol de decisión. ¿En qué valor cortaron?

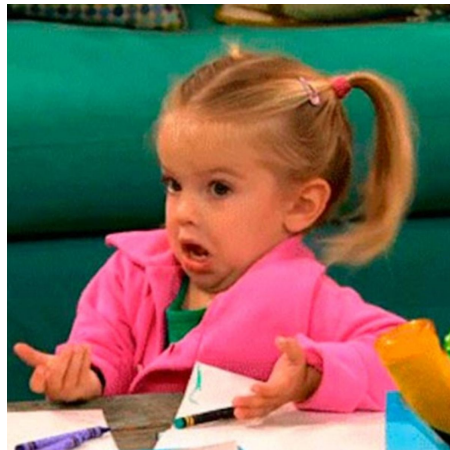




Les parece que podemos separar bien esos datos usando árboles de decisión?

¿Y de quién es la culpa?

¿De los datos? ¿Del modelo?



No todos los modelos sirven para todos los datos.

Podemos pensar que existe una función **F** que puede predecir las clases dado un dato. Luego los modelos quieren aprender esa **F** pero siempre van a terminar aprendiendo una aproximación.

Sesgo inductivo

Conjunto de **supuestos** y **conocimientos previos** que el algoritmo de aprendizaje utiliza para hacer predicciones sobre nuevos datos no vistos, basándose en los datos de entrenamiento que ha visto.

Estos supuestos pueden provenir de una variedad de fuentes, incluyendo:

- Estructura del modelo
- La función objetivo que el algoritmo esté optimizando
- Características de funcionamiento del algoritmo (cómo recorre el espacio de hipótesis hasta elegir un único modelo, la semilla utilizada)
- etc

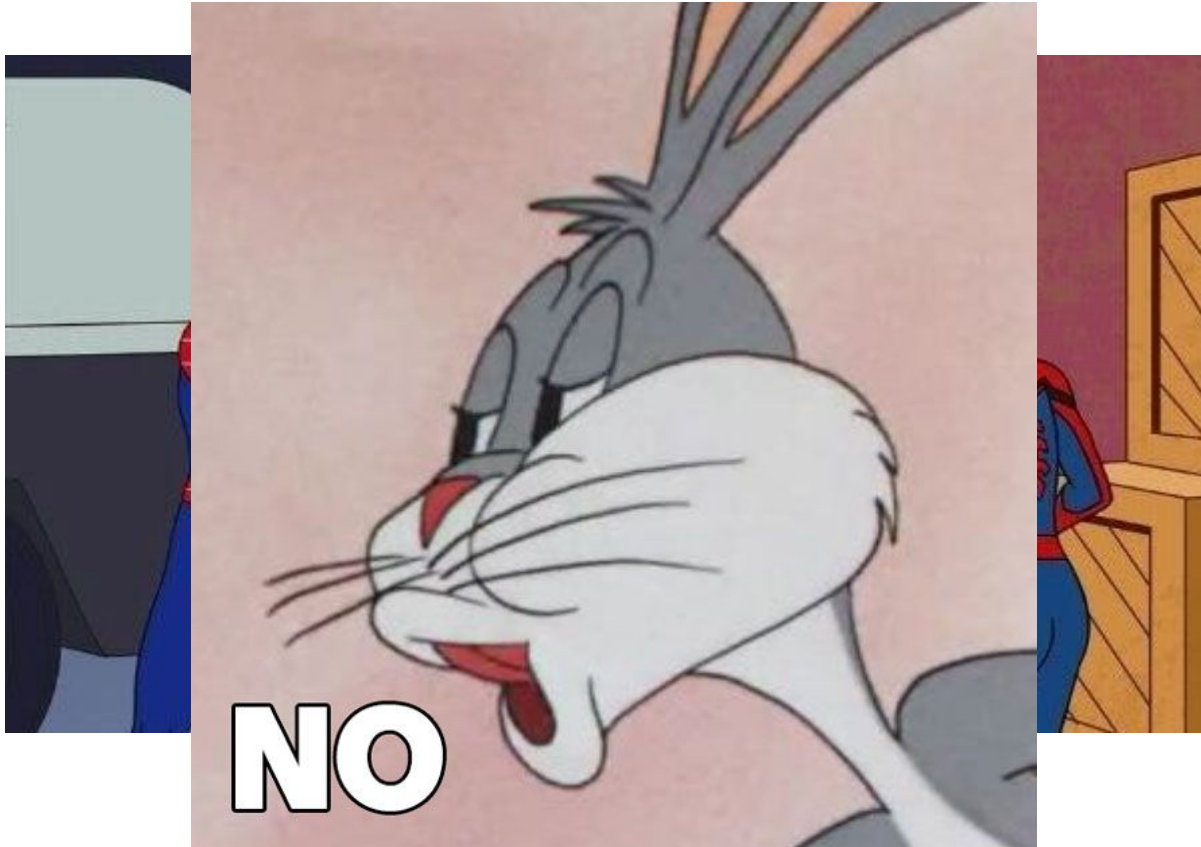
El **sesgo inductivo** determina los **tipos de funciones** que el algoritmo puede aprender y los **tipos de errores** que se espera que cometa.

¿Cual es el sesgo inductivo de los árboles?

- Las regiones son rectangulares
- Los atributos más discriminativos están cerca de la raíz (por cómo se toma el mejor corte).
- Árboles más pequeños y menos complejos en términos de su estructura de acuerdo al criterio de parada establecido.

Esto se puede formalizar pero no lo veremos acá. Pueden verlo en la Sec 2.7 del Mitchell

¿Todos los árboles son iguales?



Un detalle no arbolado

¿En la vida real, sobre qué datos queremos usar nuestro clasificador?

¿Esta bien medir cuán bueno es nuestro clasificador usando los datos que usamos para entrenar?



Dataset

Training

Testing

Holdout Method

Altura del árbol

Armen dos modelos de árboles de decisión y grafiquen los árboles resultantes.

Modelo 1: Altura máx = 2

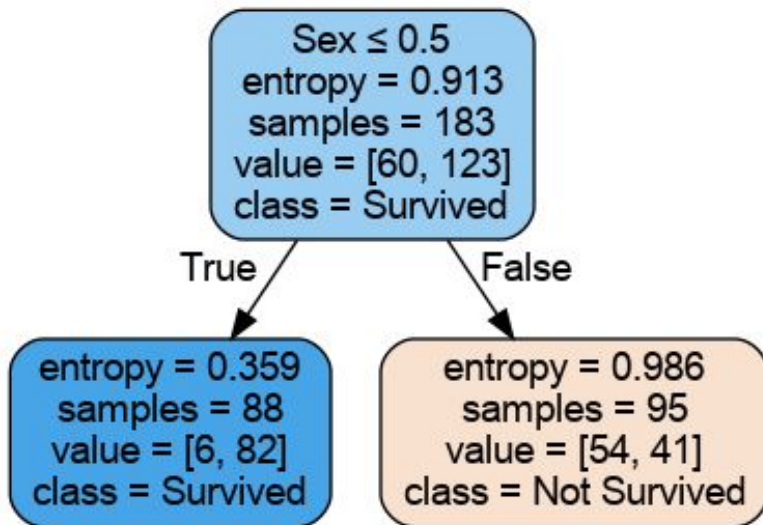
Modelo 2: Altura Máx = infinito (*¿Qué significa altura infinita?*)



Competencia entre árboles

Usemos ambos modelos para predecir las instancias del archivo de test.



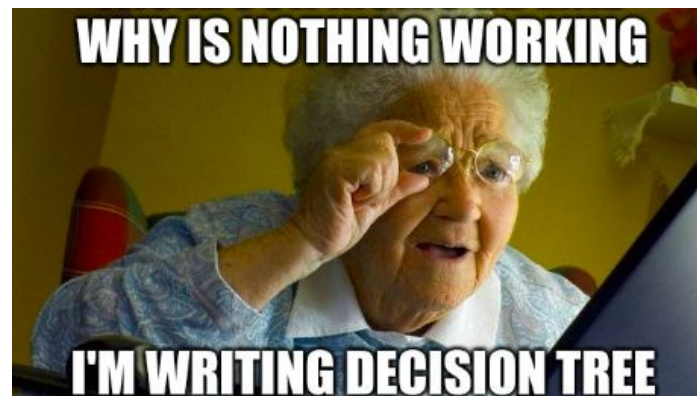


EN
TRAINING

SCORE: 136 de 186

EN TEST

SCORE: 11 de 20





**EN
TRAINING**

SCORE: 186 de 186

EN TEST

SCORE: 13 de 20



Para pensar:

Miren la cantidad de instancias en cada hoja.

¿Qué les parece que está aprendiendo el modelo?

Generalización

Queremos un modelo lo **suficientemente rico** para captar ideas complejas. Pero sin caer en:

Subajuste (underfitting)

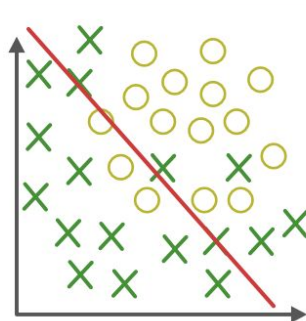
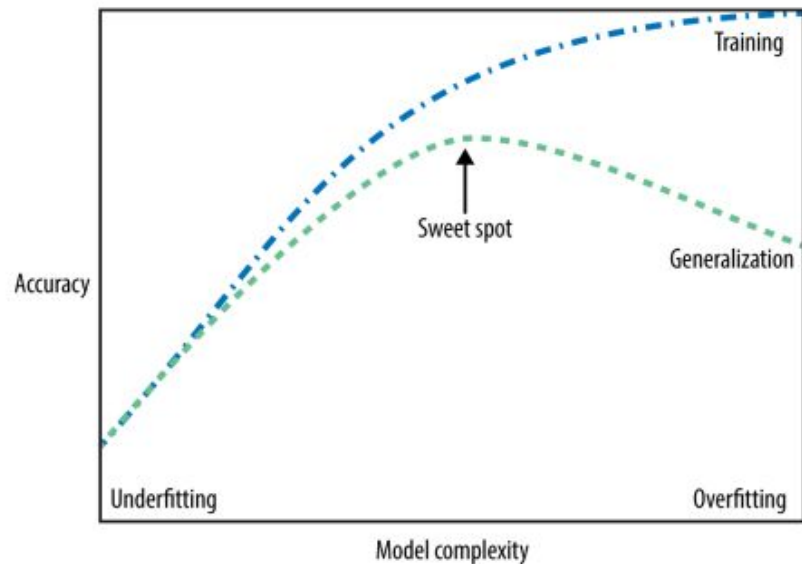
Construcción de un modelo **demasiado simple** que no capture la información disponible.

Sobreajuste (overfitting)

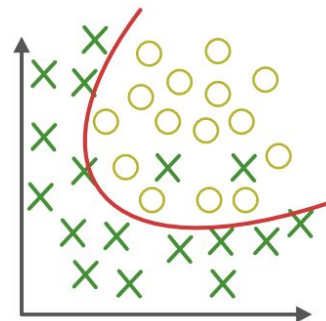
Construcción de un modelo **demasiado complejo** para la cantidad de información que disponemos.

No hay que perder de vista el objetivo.
¿Para qué queremos usar el modelo?

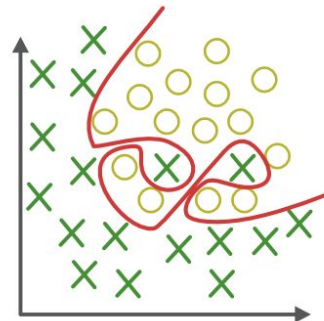
En el ejemplo de Titanic,
¿Cuál árbol subajusta y cuál sobreajusta?



Under-fitting
(too simple to explain the variance)



Appropriate-fitting



Over-fitting
(forcefitting--too good to be true)

Modelos Simples

Una de las formas de evitar el sobreajuste es simplificar los modelos. En general queremos que nuestros modelos cumplan:

Navaja de Ockham (un principio metodológico y filosófico).

Cuando tenemos dos modelos que compiten y producen las mismas predicciones, debemos elegir el más simple. Un modelo más simple es aquel que tiene menos parámetros o menor complejidad estructural.

¿Por qué? En la práctica se ve que ayuda a:

- Evitar el sobreajuste y por lo tanto **mejorar el rendimiento en datos no vistos**.
- Mejora la **interpretación**. Los modelos más simples son más fáciles de interpretar y entender.
- Reduce los **costos computacionales**.



Ockham chooses a razor

En el caso de los árboles, ¿cuál es el menos complejo?

Resumen

- ★ Árboles de decisión
- ★ Métricas de calidad de regiones: Entropía, Gini Gain, Info Gain
- ★ Sesgo Inductivo
- ★ Generalización: Overfitting vs Underfitting
- ★ Navaja de Ockham



Bibliografía + Créditos

- Mitchell, "Machine Learning", McGraw-Hill (1997)
(en particular la sección 2.7)
- Müller & Guido, "Introduction to Machine Learning with Python", O'Reilly (2016).
(ver código y capítulo 2)
- James, Witten, Hastie & Tibshirani, "An Introduction to Statistical Learning with Applications in R", 6th ed, Springer (2015)
(ver Capítulo 8)

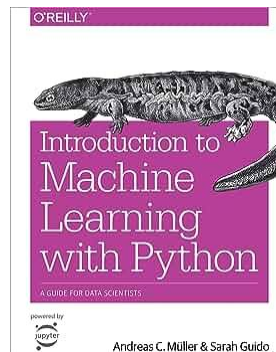
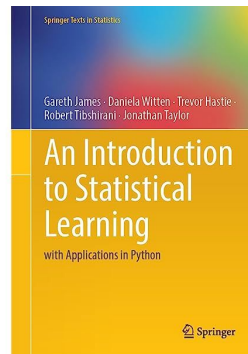
Otros recursos

<https://scikit-learn.org/stable/modules/tree.html#decision-trees>

https://www.youtube.com/watch?v=_L39rN6gz7Y (video sobre decision trees)

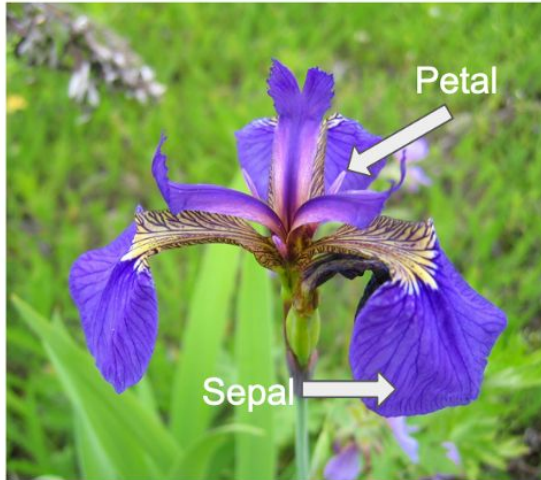
Créditos

- Clases de la materia de AA dictada por Pablo Brusco en 1c2023
- Clase correspondiente a este tema del cuatri anterior dictada por Vivi Cotik
- Autores de los magníficos memes e imágenes.

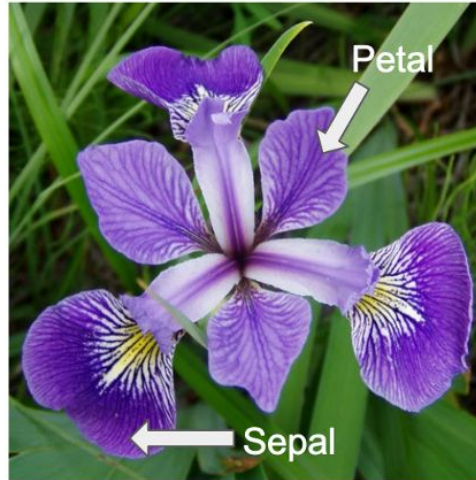


Actividades

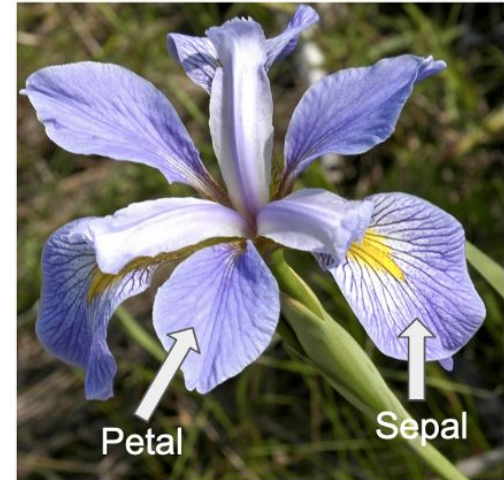
Iris setosa



Iris versicolor



Iris virginica



50 muestras de cada una de tres especies de flores Iris: setosa, versicolor y virginica. De cada flor se midieron 4 atributos: largo y ancho del sépalo y del pétalo.

Actividades

- Clasificar flores iris en 3 clases: setosa, versicolor y virginica

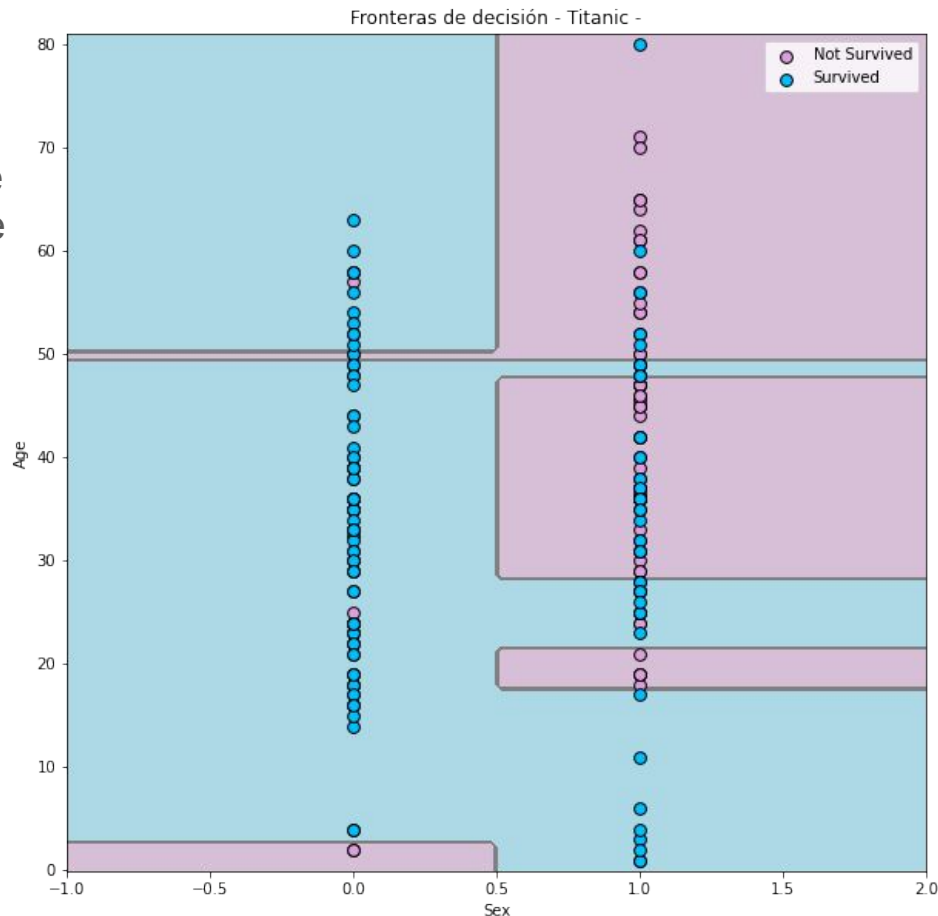
- 1) Exploren los datos. Hay algún atributo que separa bien las clases?
- 2) Construyan un clasificador usando árboles de decisión de **scikit-learn**.
- 3) Para el clasificador anterior, visualice el gráfico del árbol. Cuál es el atributo utilizado en el primer corte?
- 4) Comparen árboles que utilizan distintos criterios de corte. Cambia el atributo utilizado en el primer corte?

Fronteras de decisión

Fronteras de decisión: Regiones del espacio de características en un problema de clasificación donde un modelo de aprendizaje automático toma decisiones de clasificación diferentes.

En árboles:

- Forma jerárquica.
- Rectangular.
- (en binarios) Cada nodo interno divide el espacio de características en 2 regiones.
- Cada hoja representa una región del espacio de características donde se asigna una etiqueta de clase.

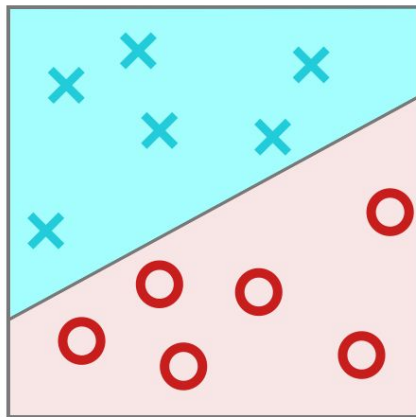


Repaso - Tipos de problemas

Queremos **aprender a aproximar** una **función desconocida** pero de la cual tenemos ejemplos.

Clasificación $\text{output}(i) :: \text{Bool} \mid \text{Enum}$

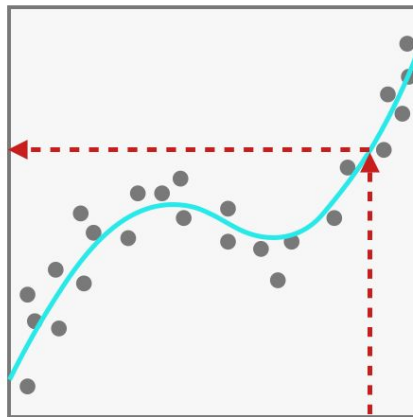
Lo que intentamos predecir son un **conjunto de etiquetas, sin orden.**



Classification

Regresión $\text{output}(i) :: \mathbb{Z} \mid \mathbb{N} \mid \mathbb{R}$

Lo que intentamos predecir son **valores cuantitativos**



Regression

KNN - *K Vecinos Más Cercanos*

Método muy sencillo para clasificar instancias.

Cómo clasifico a una instancia nueva?

Input: set de entrenamiento X , $K \in \mathbf{N}$, instancia x

- 1) Calculo las **distancias** de x a todas las instancias en el set de entrenamiento X .
- 2) Tomamos las K instancias más cercanas y veo sus clases.
- 3) Devuelvo como predicción la clase **más frecuente**.

Cómo elegimos el K ?

Qué medida de distancia usamos?

Para qué valores de K overfittea y underfittea?

