

# EST-25134: Aprendizaje Estadístico

**Profesor:** Alfredo Garbuno Iñigo — Primavera, 2022 — Arboles de decisión.

**Objetivo:** Después de haber estudiado los modelos basados en *splines* es natural considerar modelos que puedan lidiar con varios atributos y que también ajusten modelos por regiones. En esta sección estudiaremos árboles de decisión como un modelo que logra ambos objetivos.

**Lectura recomendada:** Capítulo 8 de [2].

## 1. INTRODUCCIÓN

En esta sección estudiaremos modelos basados en **árboles de decisión**. Los cuales es-tratifican el espacio de los atributos en regiones sencillas para efectuar predicciones. Reciben su nombre pues la segmentación que se utiliza para realizar predicciones es una secuencia de **decisiones binarias**.

Los árboles de decisión se utilizan para tareas de **regresión** y **clasificación**. Son modelos sencillos que pueden interpretarse fácilmente, aunque pueden tener una capacidad predictiva limitada. Después de esta sección estudiaremos cómo se pueden utilizar como base para modelos mas complejos.

### 1.1. Ejemplo: Baseball

Consideremos los datos de salarios de jugadores profesionales de *baseball*. Lo que queremos es predecir el Sueldo en funcións de las características de carrera de cada jugador.

	Hits	Years	Salary
1	1	81	14
2	2	130	3
3	3	141	11
4	4	87	2
5	5	169	11
6	6	37	70

```
1 Salary
2   226 when Years < 4.5
3   465 when Years ≥ 4.5 & Hits < 118
4   949 when Years ≥ 4.5 & Hits ≥ 118
```

LISTING 1. Segmentación de datos utilizando un árbol de decisión.

La representación gráfica del árbol anterior tiene dos **nodos internos** (dónde se toman las decisiones binarias) y tres **nodos terminales**. El número en cada nodo representa la respuesta y el porcentaje es el número de observaciones del conjunto de entrenamiento que se han decantado en cada nodo.

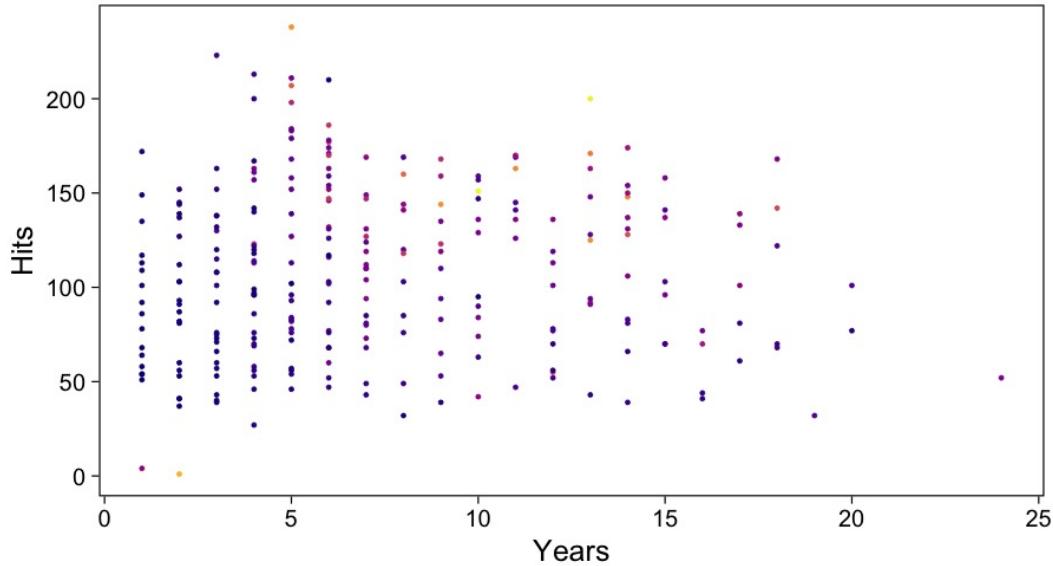


FIGURA 1. Salario codificado por color: salarios bajos (azul, morado) y salarios altos (naranja, amarillo).

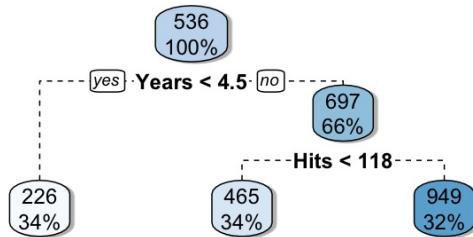


FIGURA 2. Representación gráfica de un árbol de decisión.

## 1.2. Terminología

- Las regiones las denotamos por  $R_i$ , y son conocidas como los **nodos terminales**.
- La representación gráfica usualmente está al revés. Las **hojas** están en el fondo del gráfico.
- Las decisiones donde se cortan las regiones se denominan **nodos internos**.
- Representación alterna: **years < 4.5** o **hits < 118**.

## 1.3. Interpretación del árbol

- La variable **years** es el factor mas importante en determinar la respuesta del modelo para **salary**.
- Si nos fijamos en los jugadores con poco experiencia el número de **hits** no influye en la determinación de **salary**.
- Para jugadores con mas de 5 años de experiencia, el número de **hits** realizados el año anterior determina el nivel del salario. Para aquellos con mas de 5 años jugando, a mayor número de **hits** se les asocia mayor **salary**.

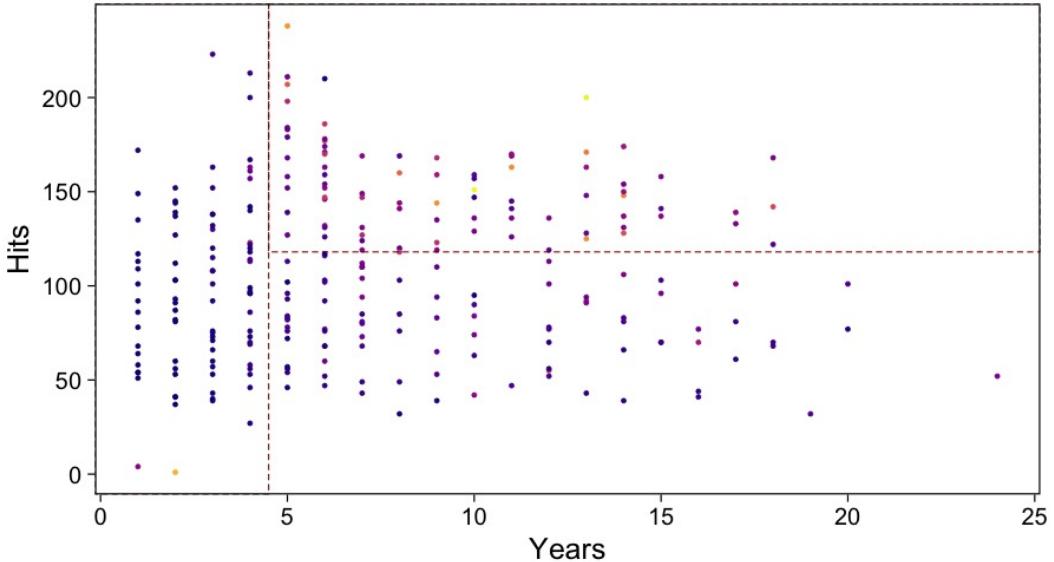


FIGURA 3. Salario codificado por color: salarios bajos (azul, morado) y salarios altos (naranja, amarillo). Las líneas punteadas representan las decisiones sobre los atributos.

## 2. CONSTRUCCIÓN DE UN ÁRBOL DE DECISIÓN

1. Dividimos el espacio de predictores en  $J$  regiones mutuamente excluyentes  $R_i$  con  $i = 1, \dots, J$  donde  $R_i \cap R_j = \emptyset$  para toda pareja  $i, j$ .
2. Cada observación se asigna una región  $R_j$  donde hacemos la misma predicción para todas las observaciones en dicha región. Esto lo hacemos con promediando la variable respuesta.

### 2.1. Mas detalles

- En principio las regiones las podemos construir de cualquier manera. Por simplicidad utilizamos regiones rectangulares con cortes perpendiculares a los ejes.
- El objetivo es encontrar las regiones  $R_1, \dots, R_J$  que minimizan el error cuadrático

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

donde  $\hat{y}_{R_j}$  es la respuesta en la  $j$  éSIMA región.

- Por supuesto, el problema de considerar todas las posibles particiones en  $J$  cajas es un **problema combinatorio**.
- Por lo tanto, tomamos una estrategia **voraz** (miope, *greedy*) **secuencial** (creciente, *top-down*).
- Es *secuencial* (creciente) pues construye el árbol tomando las decisiones mas amplias. Es decir, cortando en las regiones mas grandes para después refinárlas.
- Es *miope* (voraz) pues cada decisión de corte se toma en cada uno de los pasos sin considerar los subsecuentes.
- Consideraremos utilizar el predictor  $X_j$  y utilizar el punto de corte  $s$  de tal manera que resulten las regiones

$$R_1(j, s) = \{X | X_j < s\}, \quad R_2(j, s) = \{X | X_j \geq s\}, \quad (2)$$

que tengan la máxima reducción de RSS.

- Consideramos dentro de cada región otra decisión de selección de variable y decisión de corte para refinar el espacio de los atributos.
- El procedimiento continua hasta que se satisface un criterio de terminación. Por ejemplo, que todas las regiones tengan a lo más 5 observaciones o se alcance una profundidad máxima del árbol.

## 2.2. Predicciones

Las predicciones se realizan tomando el promedio de las respuestas en cada una de las regiones. Por lo tanto, para predecir la respuesta en un punto tenemos que evaluar en dónde se encuentra dicho punto y luego tomar el promedio de los datos de entrenamiento en dicha región.

*2.2.1. Para pensar:* En la Fig. 4 ¿qué partición del espacio resulta de un árbol de decisión?

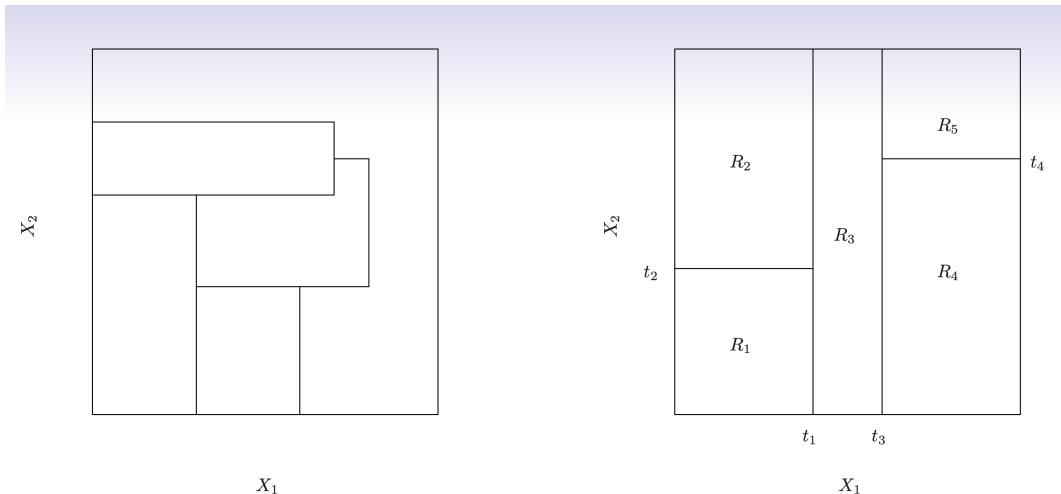


FIGURA 4. Dos particiones del espacio de atributos. Imagen tomada de [2].

En la Fig. 5, dos representaciones gráficas del mismo árbol de decisión.

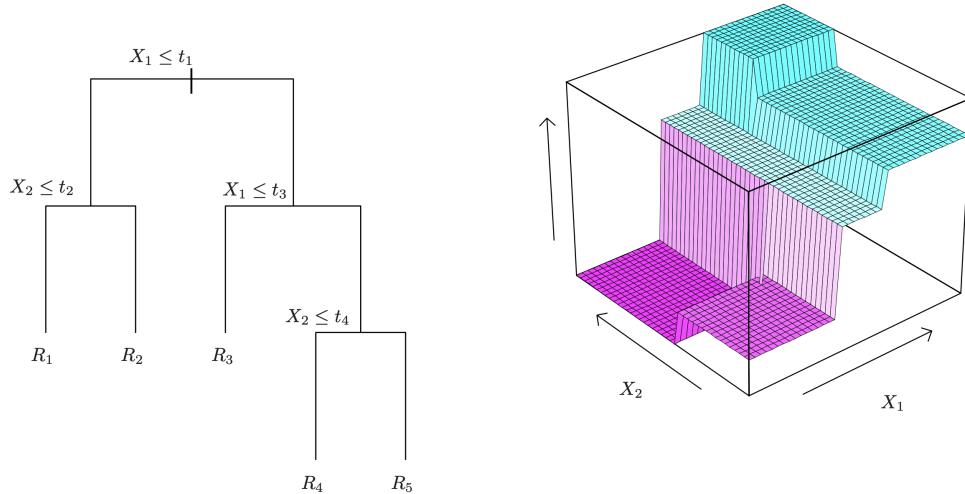


FIGURA 5. Imagen tomada de [2].

### 2.3. Error de generalización

- Si ajustamos un árbol de decisión (*descrito anteriormente*) podemos sobre-ajustar fácilmente los datos de entrenamiento (¿por qué?).
  - Un árbol mas pequeño puede tener **menor varianza** al costo de tener **mas sesgo**.
  - Podríamos considerar cortes que sólo tengan una mejora de  $x\%$  puntos en el RSS.
  - Pero nos podríamos quedar cortos, un mal corte inmediato podría ayudar a refinar el árbol en el largo plazo.

## 2.4. Proceso de poda

- Podemos construir un árbol muy grande  $T_0$ , y podarlo para encontrar un sub-árbol con buenas capacidades predictivas.
  - El método de poda que se utiliza es por medio de una medida de complejidad (*cost complexity pruning, weakest link pruning*).
  - Consideramos una secuencia de árboles\* indexados por un parámetro  $\alpha > 0$ . Para cada valor de  $\alpha$  tenemos un sub-árbol  $T \subset T_0$  tal que

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|, \quad (3)$$

es lo mas pequeño posible.

- En esta notación  $|T|$  denota el número de nodos terminales (regiones) del árbol.

Utilizar un validación cruzada implicaría evaluar la capacidad predictiva de cada sub-árbol posible. Lo cual se traduce en un costo computacional alto.

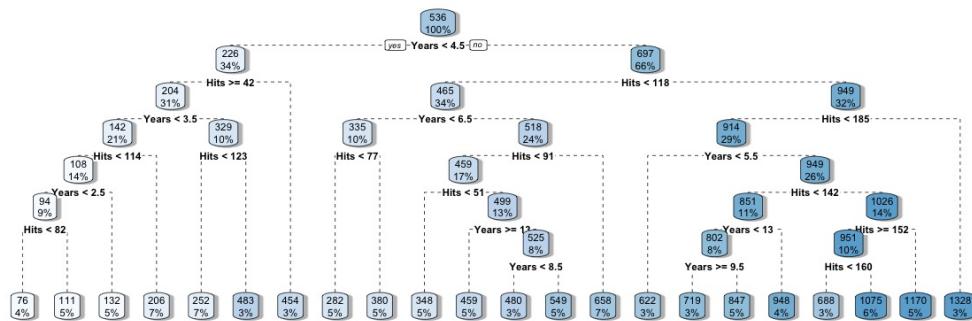
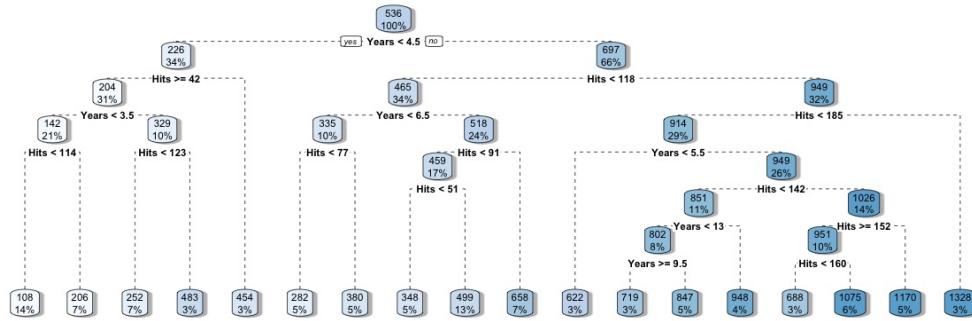
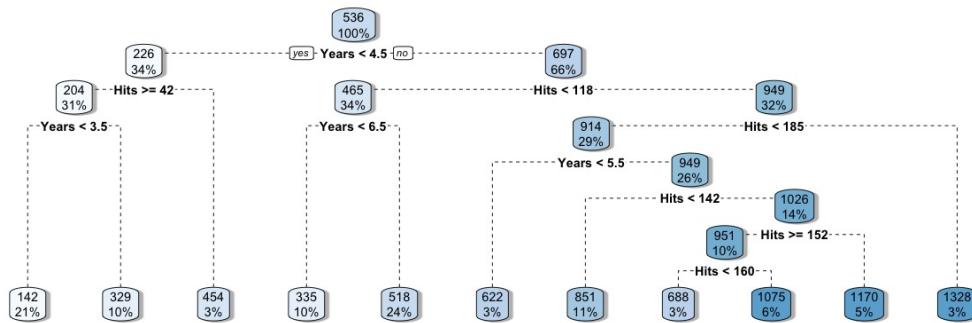


FIGURA 6. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-6}$ .

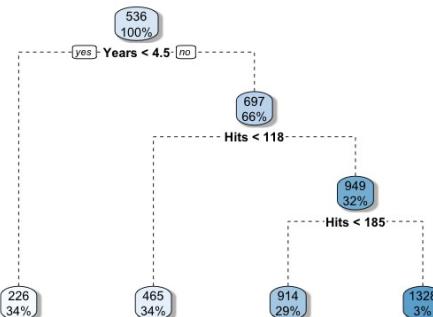
## 2.5. Selección del mejor sub-árbol

- El parámetro  $\alpha$  controla el compromiso entre complejidad y ajuste al conjunto de entrenamiento.
  - Para cada valor de  $\alpha$  existe un árbol asociado  $T_\alpha$ . Bajo de una secuencia  $\alpha_1 < \alpha_2 < \dots$  tenemos una sucesión de árboles en donde cada árbol es óptimo. La prueba la encuentran en ([1, 3]).
  - Usamos un valor óptimo de  $\hat{\alpha}$  por medio de ...
  - Después, ajustamos el árbol utilizando  $\hat{\alpha}$  y el conjunto de datos completo.

FIGURA 7. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-3}$ .FIGURA 8. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-2}$ .

## 2.6. Resumen

- Usamos el conjunto de entrenamiento para ajustar un árbol de decisión. Utilizamos un criterio de paro de acuerdo al número de observaciones en los nodos terminales.
- Usamos poda de árboles considerando una penalización por complejidad y obtenemos una secuencia de árboles indexados por  $\alpha$ .
- Usamos validación cruzada con  $K$  bloques para escoger  $\alpha$ .
- Reajustamos utilizando todo el conjunto de datos utilizando la  $\hat{\alpha}$  que encontramos en el procedimiento de validación.

FIGURA 9. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-1}$ .

## 2.7. Ejemplo:

Consideremos los datos descritos en este [caso de estudio](#) por Julia Silge (autora del libro *tidymodels*). El objetivo es poder predecir la capacidad de las turbinas de viento en Canada por medio de cierta colección de descriptores. Puedes seguir [la liga](#) para una descripción mas detallada de los datos.



FIGURA 10. Imagen tomada de la documentación de los datos [caso de estudio](#).

¿Cómo se relacionan las características como año de producción o tamaño de la turbina con su capacidad energética?

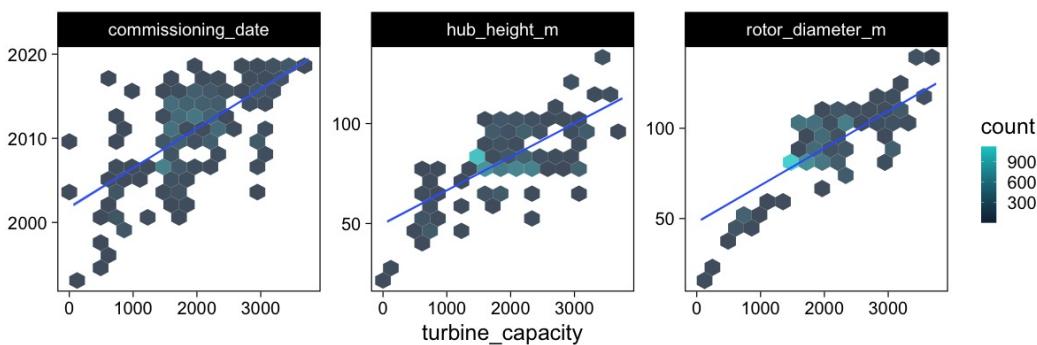


FIGURA 11. Gráficos de densidad entre la variable objetivo (eje horizontal) y atributo marcado en el panel.

Dividimos el conjunto de datos en 50 % entrenamiento y 50 % prueba. Creamos la especificación del modelo, considerando que tenemos el parámetro  $\alpha$  como un parámetro especificado por el usuario.

```
1 tree_spec <- decision_tree(
```

```

2   cost_complexity = tune(),
3   ) %>%
4   set_engine("rpart") %>%
5   set_mode("regression")
6
7 tree_spec

```

Definimos la rejilla donde queremos explorar  $\alpha$ :

```

1 tree_grid <- grid_regular(cost_complexity(), levels = 10)
2 tree_grid

```

Ajustamos el modelo utilizando validación cruzada y la rejilla

```

1 doParallel::registerDoParallel()
2
3 set.seed(345)
4 tree_rs <- tune_grid(
5   tree_spec,
6   turbine_capacity ~.,
7   resamples = wind_folds,
8   grid = tree_grid,
9   metrics = metric_set(rmse)
10 )
11
12 tree_rs

```

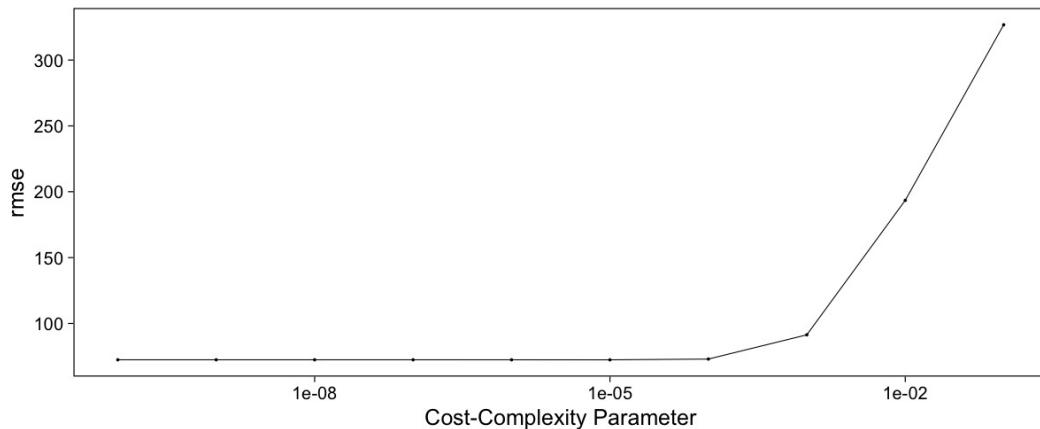


FIGURA 12. Error de validación evaluado por validación cruzada para distintos valores de  $\alpha$ .

Podemos escoger el mejor modelo de acuerdo a la métrica que definamos:

```

1 final_tree <- finalize_model(tree_spec, select_best(tree_rs, "rmse"))
2 final_tree

```

Podemos ajustar el mejor modelo a los datos de entrenamiento o pedirle que ajuste con la separación inicial.

```

1 final_fit <- fit(final_tree, turbine_capacity ~ ., wind_train)
2 final_rs <- last_fit(final_tree, turbine_capacity ~ ., wind_split)

```

### 3 ÁRBOLES DE CLASIFICACIÓN

Por supuesto, no podemos visualizar la respuesta como un modelo de  $\mathbb{R}^p \rightarrow \mathbb{R}$ . Pero podemos escoger las variables mas informativas para la predicción (mas adelante discutimos esto):

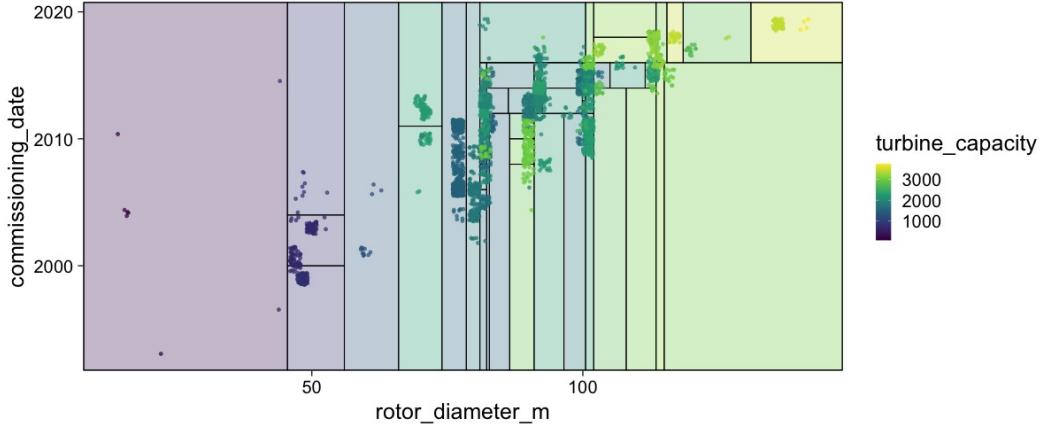


FIGURA 13. Superficie de respuesta para un modelo simplificado con la configuración encontrada por validación cruzada.

El modelo ajustado es bastante complejo. Por ejemplo, podemos visualizar el árbol y las decisiones:

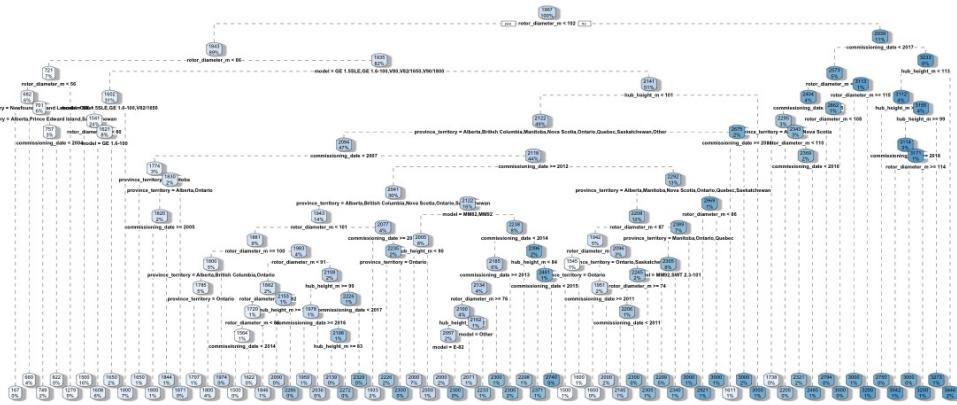


FIGURA 14. Representación gráfica del árbol de decisión.

### 3. ÁRBOLES DE CLASIFICACIÓN

- La construcción es muy similar a la construcción en el ámbito de regresión.
- Por supuesto, no podemos utilizar el RSS como métrica de ajuste.
- Podríamos utilizar el **error de clasificación** para generar el árbol.
- Pero, el error de clasificación **no** es lo suficientemente sensible para ajustar un árbol.

#### 3.1. Métricas de ajuste: el índice de Gini y devianza

- El **índice de Gini** está definido por

$$G(m) = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}), \quad (4)$$

donde la suma es a través de todas las clases y  $\hat{p}_{mk}$  es la probabilidad de la  $k$  éSIMA clase en la región  $m$ .

- Toma valores pequeños si todas las  $\hat{p}_{mk}$  son pequeñas o cercanas a 1.
- Por esta, razón, el índice de Gini también se denomina un **índice de pureza**. Pues nos indica si en un nodo, tenemos una clase **predominante**.
- Una métrica alternativa es la **entropía cruzada** o devianza

$$D(m) = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}. \quad (5)$$

- La cual tienen valores similares en la práctica.

### 3.2. Ejemplo: Scooby-Doo

```
1 monster_real    n
2 1      FALSE 404
3 2      TRUE 112
```

Utilizaremos el año en que salió el episodio y el *rating* que tuvo ese episodio para predecir si el monstruo era real al final del episodio o no. Especificamos el modelo

```
1 tree_spec <-
2   decision_tree(
3     cost_complexity = tune(),
4     tree_depth = tune(),
5     min_n = tune()
6   ) %>%
7   set_mode("classification") %>%
8   set_engine("rpart")
```

Especificamos la rejilla de búsqueda

```
1 tree_grid <- grid_regular(cost_complexity(), tree_depth(), min_n(), levels = 4)
2 tree_grid %>% head() %>% as.data.frame()
```

	cost_complexity	tree_depth	min_n
1	1e-10	1	2
2	1e-07	1	2
3	1e-04	1	2
4	1e-01	1	2
5	1e-10	5	2
6	1e-07	5	2

Realizamos el ajuste en cada bloque con cada especificación del modelo.

```
1 set.seed(345)
2 tree_rs <-
3   tune_grid(
4     tree_spec,
5     monster_real ~ year aired + imdb,
6     resamples = scooby_folds,
7     grid = tree_grid,
8     metrics = metric_set(accuracy, roc_auc, sensitivity, specificity)
9   )
```

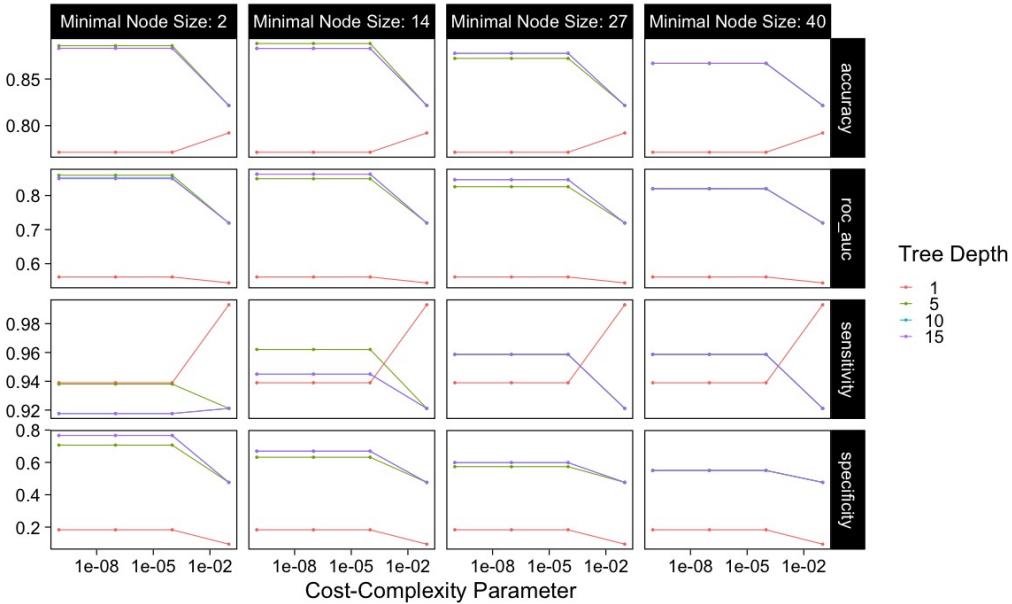


FIGURA 15. Resultados de validación cruzada para la configuración de tres parámetros en el modelo de árbol: profundidad, mínimo de observaciones en nodos y penalización por complejidad.

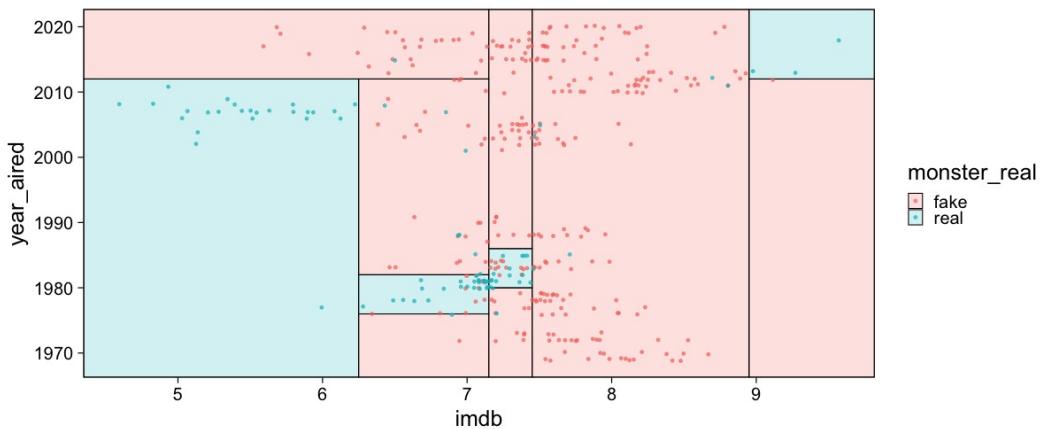


FIGURA 16. Superficie de respuesta por el árbol de decisión.

#### 4. CONCLUSIONES

- Los árboles de decisión son fáciles de interpretar y explicar.
- Algunos piensan que los árboles de decisión refleja el patrón de toma de decisiones de las personas.
- Son fáciles de visualizar, incluso si hay muchos predictores.
- Son difíciles de ajustar cuando las relaciones son lineales.
- Las predicciones numéricas pueden ser poco precisas.
- Son sensibles al conjunto de datos.

#### REFERENCIAS

- [1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification And Regression Trees*. Routledge, New York, oct 2017. ISBN 978-1-315-13947-0. . 5

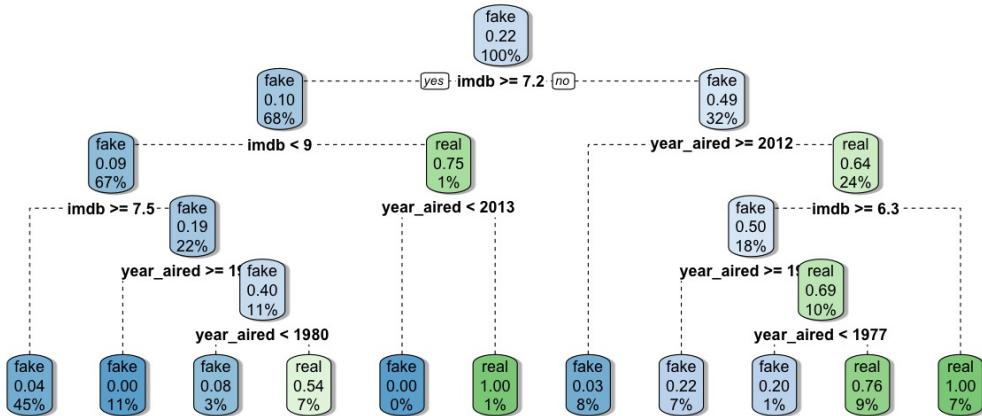


FIGURA 17. Representación gráfica del árbol de decisión.

- [2] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer US, New York, NY, 2021. ISBN 978-1-07-161417-4 978-1-07-161418-1. . 1, 4
- [3] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996. ISBN 978-0-521-71770-0. . 5