

# EST-25134: Aprendizaje Estadístico

**Profesor:** Alfredo Garbuno Iñigo — Primavera, 2023 — Árboles 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 estratifican 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. Motivación

¿Cómo identificarías un correo electrónico que debería de ser marcado como *spam*?

Para resolver esta pregunta podrías pensar en identificar palabras clave en el mensaje o el remitente. Intuitivamente, esto tiene sentido. ¿Pero cómo construirías un modelo predictivo para esta tarea? Un modelo de regresión, como veremos mas adelante, tienes distintas limitantes para resolver la tarea.

```
1 data(spam, package = "kernlab")
2 spam <- spam %> as_tibble() %> mutate(type = relevel(type, ref = "spam"))
```

```
1 spam %>
2   group_by(type) %>
3   tally()
```

```
1 # A tibble: 2 × 2
2   type      n
3   <fct>    <int>
4   1 spam     1813
5   2 nonspam  2788
```

```

1 logistic_spec <- logistic_reg(penalty = .0003, mixture = 1) ▷
2   set_engine("glmnet")
3
4 logistic_recipe <- recipe(type ~ ., spam_train)
5
6 logistic_wf <- workflow() ▷
7   add_recipe(logistic_recipe) ▷
8   add_model(logistic_spec)

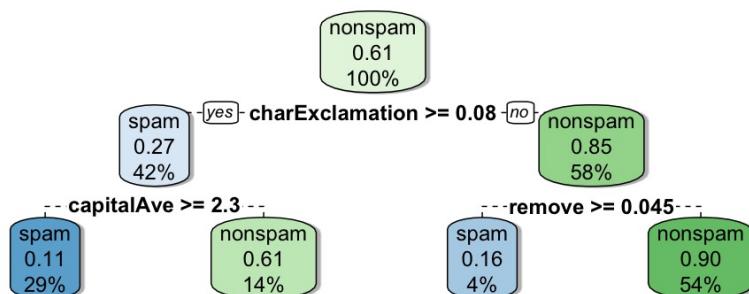
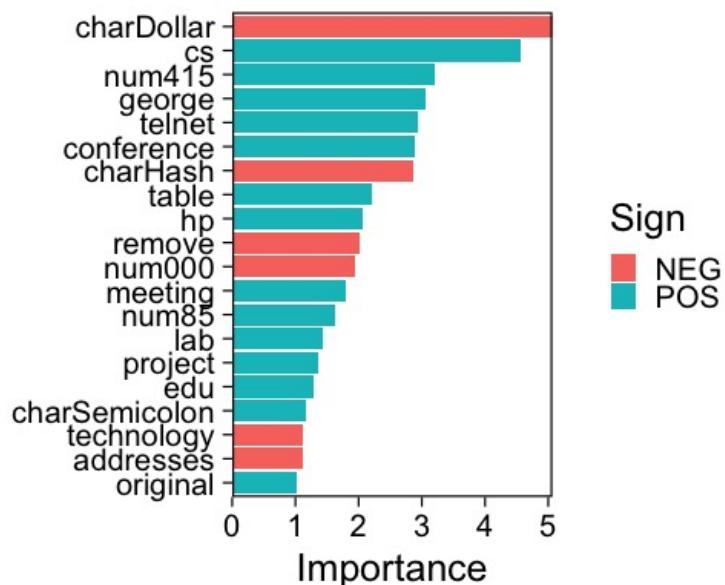
```

```

1 fit(logistic_wf, spam_train) ▷
2   augment(new_data = spam_test) ▷
3   conf_mat(type, .pred_class)

```

		Truth
Prediction	spam	nonspam
spam	405	26
nonspam	49	671



```

1 class_tree_fit ▷
2   augment(new_data = spam_test) ▷
3   conf_mat(type, .pred_class)

```

```

1      Truth
2 Prediction spam nonspam
3   spam      311      49
4 nonspam    143    648

```

## 1.2. Ejemplo: Baseball

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

```

1 # A tibble: 263 × 3
2   Hits  Years Salary
3   <int> <int>  <dbl>
4 1     81     14  475
5 2    130      3  480
6 3    141     11  500
7 4     87      2  91.5
8 5    169     11  750
9 # ... with 258 more rows
10 # Use 'print(n = ...)' to see more rows

```

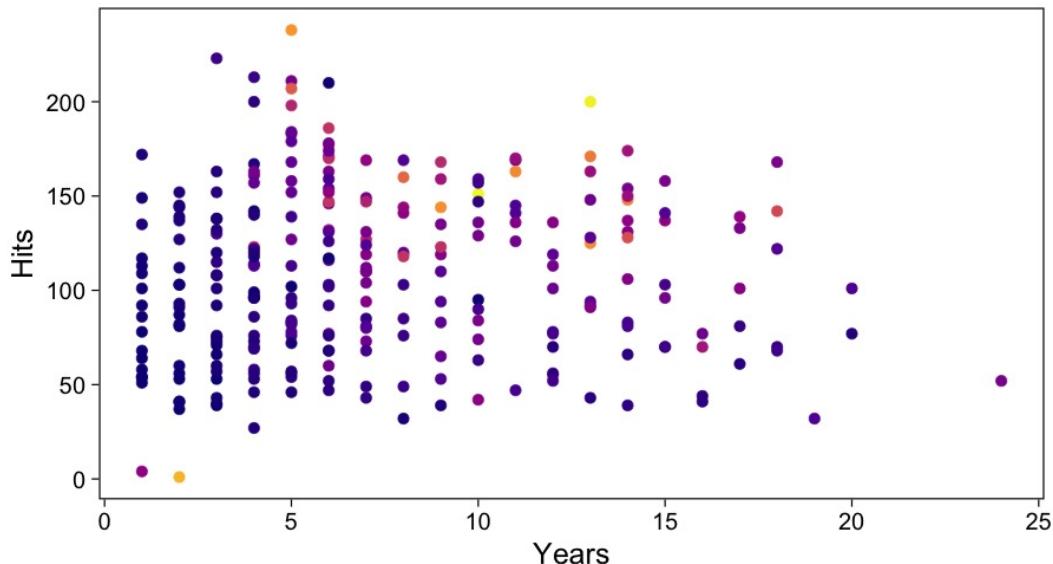


FIGURA 1. Salario codificado por color: salarios bajos (azul, morado) y salarios altos (naranja, amarillo).

Un árbol de decisión nos permitirá hacer predicciones con reglas como se muestra a continuación.

```

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.

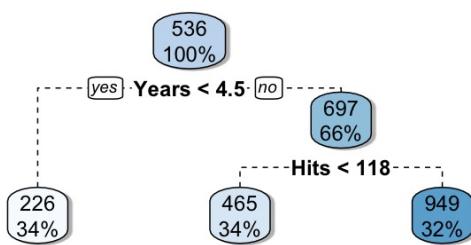


FIGURA 2. Representación grafica de un árbol de decisión.

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.

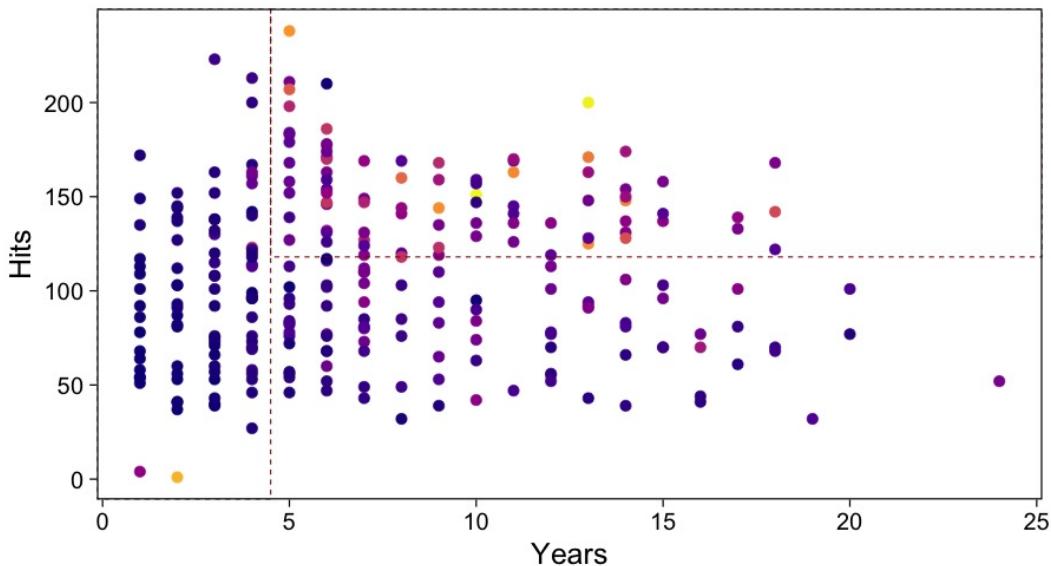


FIGURA 3. Salario codificado por color: salarios bajos (azul, morado) y salarios altos (naranja, amarillo). Las líneas punteadas representan las decisiones sobre los atributos.

### 1.3. 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**.
- Los nodos internos están representados por las decisiones: **years < 4.5** o **hits < 118**.

### 1.4. Interpretación del árbol

- La variable **years** es el factor mas importante\* en determinar la respuesta del modelo para **salary**.

## 2 CONSTRUCCIÓN DE UN ÁRBOL DE DECISIÓN

---

- Si nos fijamos en los jugadores con poca 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`.

### 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 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**.

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

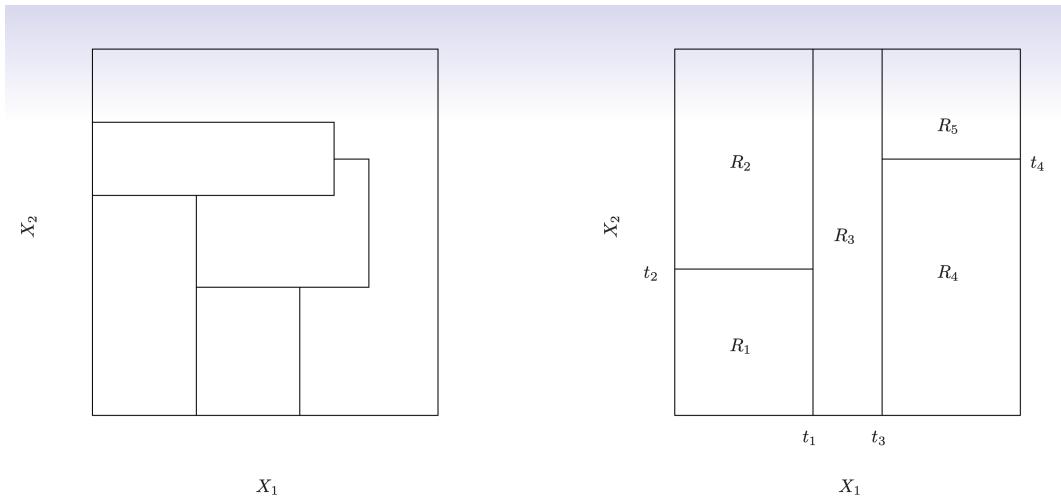


FIGURA 4. Dos particiones del espacio de atributos. Imagen tomada de [2].

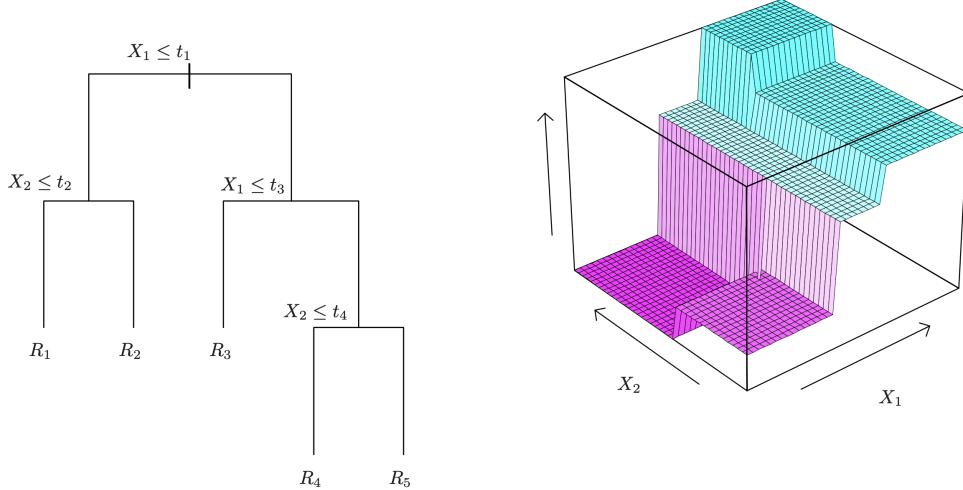


FIGURA 5. Imagen tomada de [2].

### 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 más 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*).
- Consideraremos una secuencia de árboles\* indexados por un parámetro  $\alpha > 0$ . Para cada

valor de  $\alpha$  tenemos un sub-árbol  $T(\alpha) \subset T_0$  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.

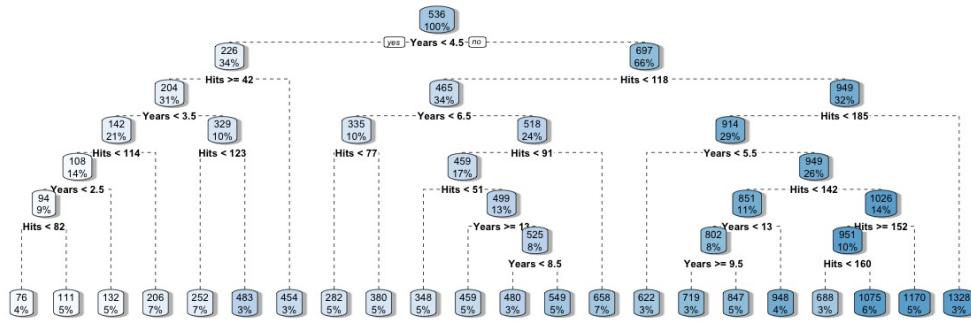


FIGURA 6. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-6}$ .

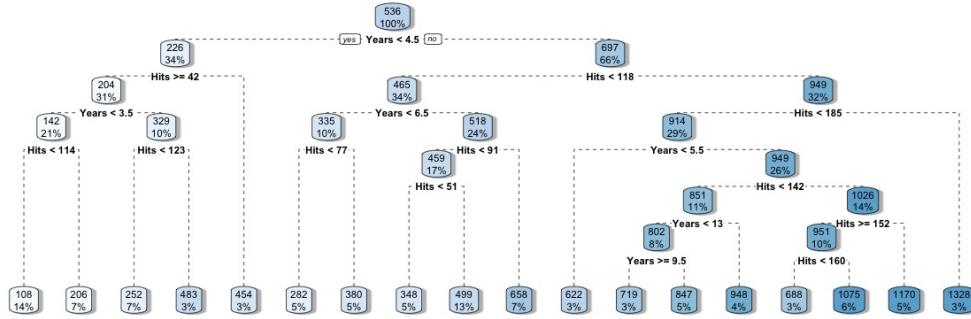


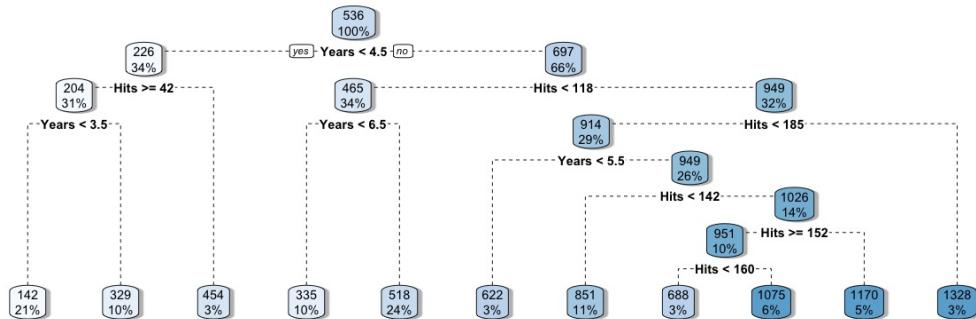
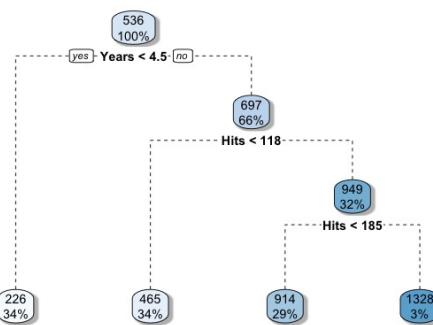
FIGURA 7. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-3}$ .

## 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 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.

## 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.

FIGURA 8. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-2}$ .FIGURA 9. Representación grafica de un árbol de decisión. Penalización  $\alpha = 10^{-1}$ .

- 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.

## 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 Canadá por medio de cierta colección de descriptores. Puedes seguir [la liga](#) para una descripción mas detallada de los datos.

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

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

```



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

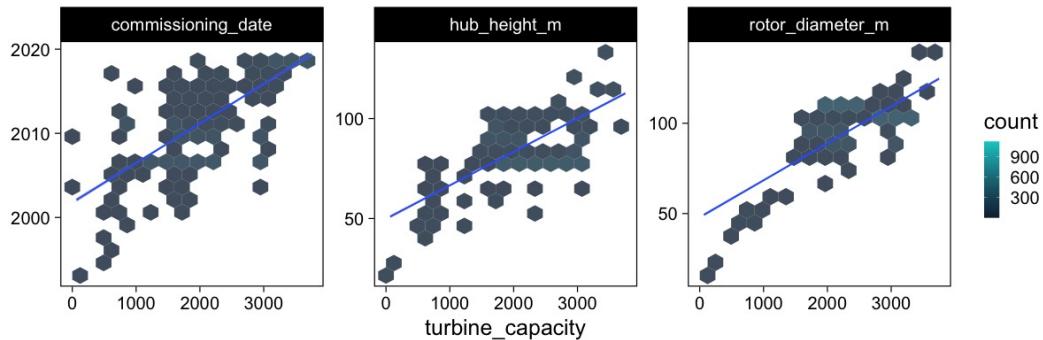


FIGURA 11. Gráficos de densidad entre la variable objetivo (eje horizontal) y atributo marcado en el panel.

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 set.seed(345)
3 tree_rs <- tune_grid(
4   tree_spec,
5   turbine_capacity ~ .,
6   resamples = wind_folds,
7   grid = tree_grid,
8   metrics = metric_set(rmse)
```

```

9 )
10 tree_rs

```

Usando validación cruzada podemos cuantificar el error de generalización para cada valor de  $\alpha$ . Recuerda que de acuerdo a lo que discutimos antes  $\alpha$  tiene una incidencia directa en el tamaño del árbol.

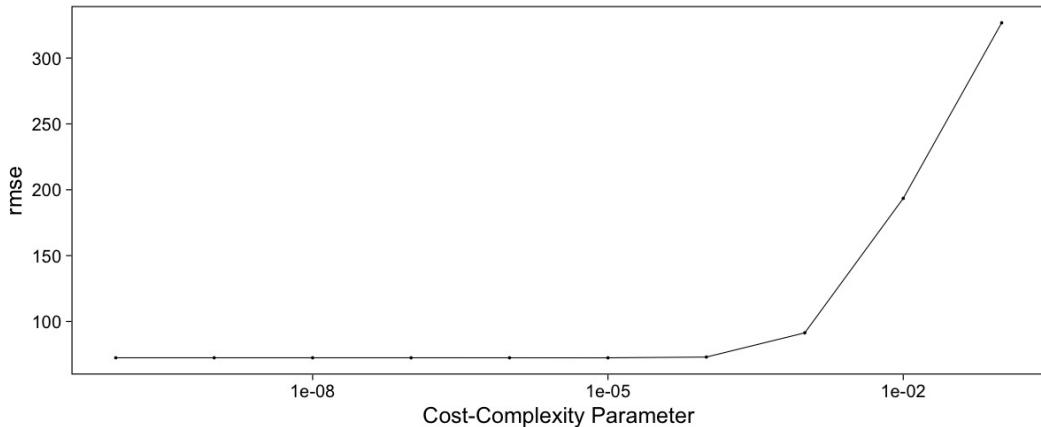


FIGURA 12. Error de validación evaluado por validación cruzada para distintos valores de  $\alpha$ .

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)

```

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):

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

### 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

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

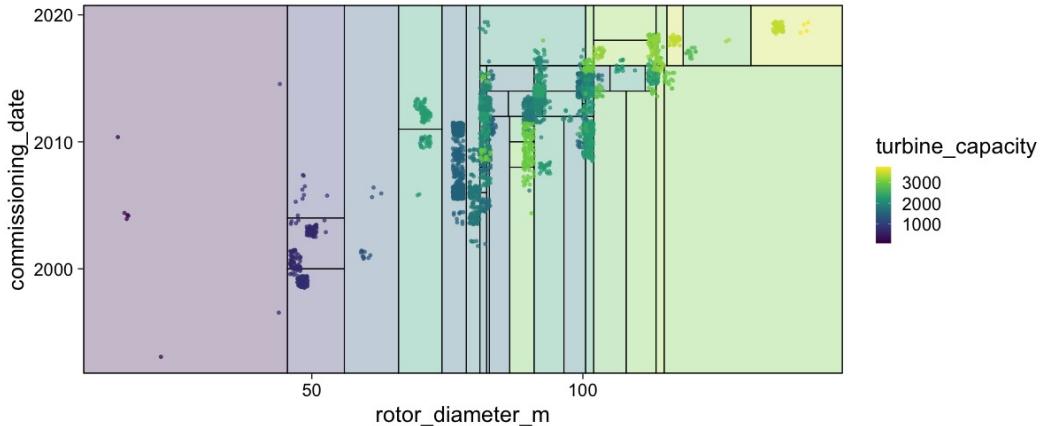


FIGURA 13. Superficie de respuesta para un modelo simplificado con la configuración encontrada por validación cruzada.

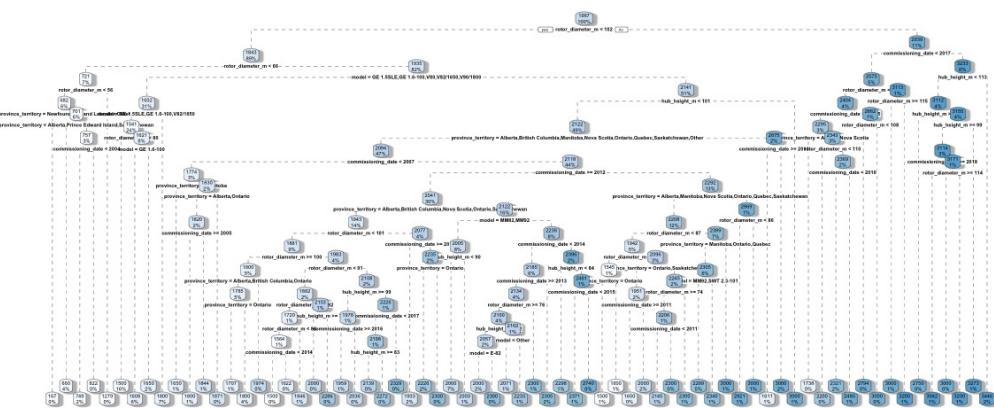


FIGURA 14. Representación gráfica del árbol de decisión.

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 incide en decisiones similares al índice de Gini en la práctica.

### 3.2. Aplicación: Episodios de Scooby-Doo

Tomamos el [siguiente caso](#) de Julia Silge para ejemplificar un problema de clasificación con árboles de decisión. El objetivo es predecir si al final del capítulo el monstruo era un monstruo real o era un disfraz.

```

1 # A tibble: 2 × 2
2   monster_real     n
3   <chr>           <int>
4 1 FALSE            404
5 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 ▷ print(n = 5)

```

```

1 # A tibble: 64 × 3
2   cost_complexity tree_depth min_n
3   <dbl>           <int> <int>
4 1 0.0000000001    1      2
5 2 0.0000001       1      2
6 3 0.0001          1      2
7 4 0.1             1      2
8 5 0.0000000001    5      2
9 # ... with 59 more rows
10 # Use 'print(n = ...)' to see more rows

```

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   )

```

## 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 que se utilizaron para entrenarlos.

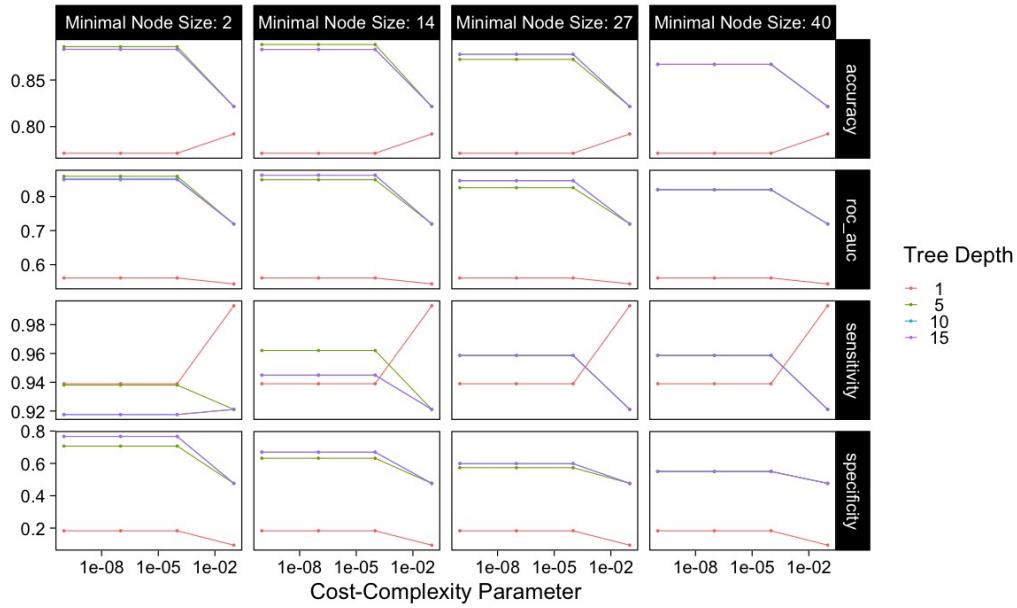


FIGURA 15. Resultados de validación cruzada para la configuración de tres parámetros en el modelo de árbol: profundidad, mínimo de observaciones en nodos y penalización por complejidad.

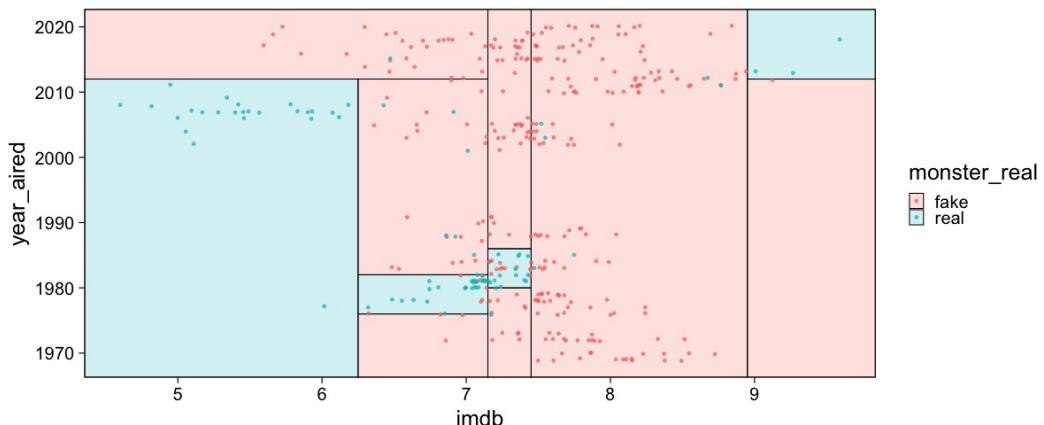


FIGURA 16. Superficie de respuesta por el árbol de decisión.

## 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. . 7
- [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. 1, 6
- [3] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996. ISBN 978-0-521-71770-0. . 7

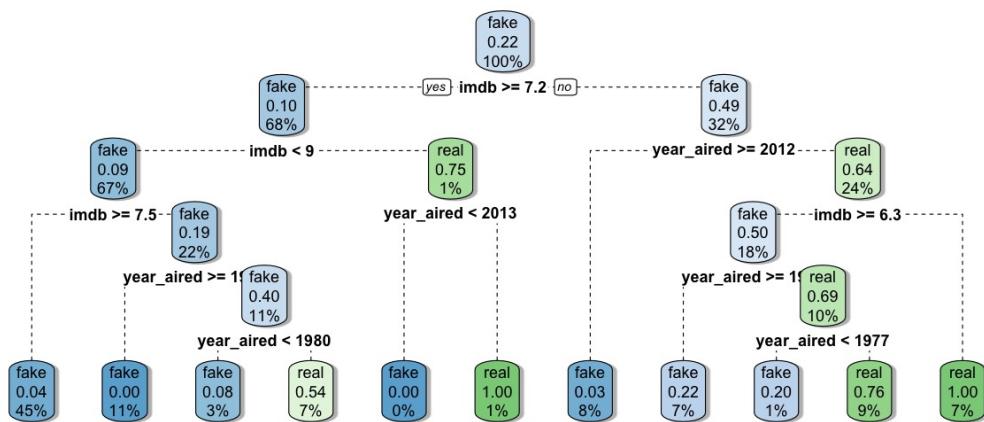


FIGURA 17. Representación gráfica del árbol de decisión.