

Modelos Estadísticos para la Regresión y la Clasificación

Práctico 6 - Árboles - Métodos de Agregación

Micaela Long

Instituto de Matemática y Estadística Prof. Rafael Laguardia (IMERL)
Facultad de Ingeniería, Universidad de la República, Uruguay

9 de octubre de 2024

Material para hacer práctico 6 (disponible en EVA):

- Teóricos Árboles de clasificación y regresión y métodos de agregación basados en árboles (Tema 7)

Será de utilidad tener a mano el libro :

“An Introduction to Statistical Learning with Applications in R”

o su versión en Python:

“An Introduction to Statistical Learning with Applications in Python”

Ambos pueden ser descargados aquí: <https://www.statlearning.com/>

Ejercicio 1

Demostrar que si X es categórica con m clases entonces hay $2^{m-1} - 1$ posibles divisiones.

Sugerencia: usar teorema del binomio

$$(a + b)^m = \sum_{k=0}^m \binom{m}{k} a^{m-k} b^k$$

donde $\binom{m}{k}$ es el coeficiente binomial, que se calcula como:

$$\binom{m}{k} = \frac{m!}{k!(m-k)!}$$

Ejercicio 2:

Demostrar que las tres expresiones del índice de Gini son iguales.

Queremos demostrar

$$\phi(p) = 1 - \sum_{k=1}^K p_k^2 = \sum_{k=1}^K p_k(1 - p_k) = 2 \sum_{k \neq k'}^K p_k p_{k'}$$

Recordar

$$\sum_{k=1}^K p_k = 1$$

Ejercicio 3:

Considerar el siguiente conjunto de datos:

x_1		a	a	b	a	a	b	b	b
x_2		b	a	a	a	a	b	b	b
y		1	1	1	1	-1	-1	-1	-1

- (a) Construir un árbol de clasificación, con el índice de Gini, a partir de esta muestra.
- (b) Comparar con el árbol obtenido en R. ¿Qué argumento de la función `rpart` debería cambiar para obtener el mismo árbol?

En Python la función es `DecisionTreeClassifier`.

Sea $p = (p_1, p_2, \dots, p_k)$, con p_i proporción de la clase i .

Error de clasificación:

$$\phi(p) = 1 - \max(p_1, p_2, \dots, p_k)$$

Índice de Gini:

$$\phi(p) = 1 - \sum_{i=1}^k p_i^2$$

Entropía:

$$\phi(p) = \sum_{i=1}^k -p_i \log(p_i)$$

Si $p_i(t)$ es la proporción de la clase i en el nodo t , definimos:

$$i(t) = \phi(p_1(t), p_2(t), \dots, p_K(t))$$

y la variación de la impureza del nodo t respecto a sus nodos hijos t_L y t_R es

$$\Delta i(t, s) = i(t) - p_L i(t_L) - p_R i(t_R)$$

donde $i(t)$ es la impureza del nodo t , p_L la proporción de muestras en el nodo t_L , $i(t_L)$ impureza del nodo t_L .

Ejercicio 3

Parte b

Link a nootebook de Python:

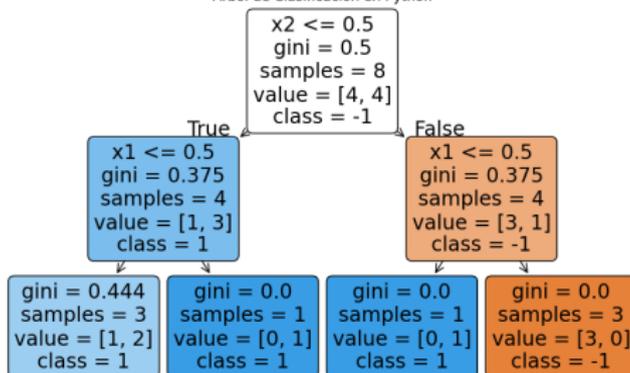
<https://colab.research.google.com/drive/1m4KH14SPSS8yq31AtCYesk1AReo551HA#scrollTo=R1nuWKuqTGDX>

RScript en EVA.

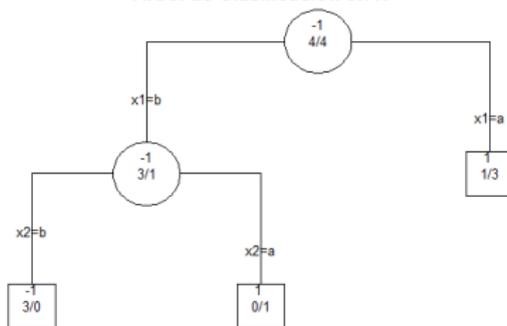
Ejercicio 3

Parte b

Árbol de Clasificación en Python

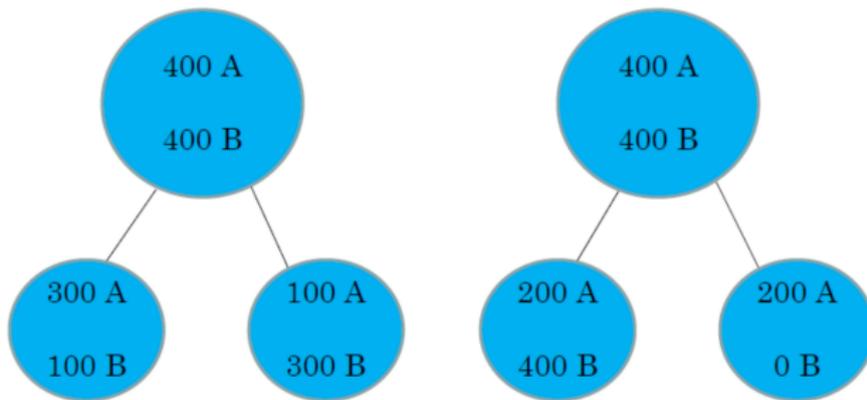


Árbol de Clasificación en R



Ejercicio 4:

Calcular $\Delta i(t, s)$ para estas dos particiones utilizando el error de clasificación, el índice de Gini y la entropía.



Bootstrap:

- Creación de distintos conjuntos de datos a través del muestreo con reposición del conjunto de datos original.

Bagging (Bootstrap Aggregating):

- Método de agregación que utiliza bootstrap para generar múltiples conjuntos de datos a partir del conjunto original y entrena un modelo en cada conjunto.

Random Forest:

- Combina predicciones de muchos árboles obtenidos de muestras bootstrap.
- En cada nodo se elige un subconjunto de variables para determinar la mejor partición.

Boosting:

- Método de agregación que ajusta modelos secuencialmente.
- Cada nuevo modelo corrige los errores del modelo anterior, combinando sus predicciones de manera ponderada.
- Ej.: AdaBoost (Adaptive Boosting).

Ejercicio 8:

Consideramos el conjunto de datos Carseats de la biblioteca rpart.

- 1 Dividir el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba (2/3 -1/3) con `set.seed(2024)`.
- 2 Ajustar un árbol de regresión al conjunto de entrenamiento. Dibujar el árbol e interpretar los resultados. ¿Qué tasas de error sobre la muestra de entrenamiento y sobre la muestra test se obtienen?
- 3 Utilizar la validación cruzada para determinar el nivel óptimo de complejidad del árbol. ¿La poda del árbol mejora la tasa de error de la prueba?
- 4 Utilizar el método Bagging para analizar estos datos. ¿Qué tasa de error sobre la muestra test se obtiene?
- 5 Utilice Random Forest para analizar estos datos. ¿Qué tasa de error sobre la muestra test se obtiene? Utilice la función `importance()` para determinar qué variables son las más importantes. Describa el efecto de `m`, el número de variables consideradas en cada división, sobre la tasa de error obtenida.
- 6 Contestar las mismas preguntas si la variable Sales se discretiza de la siguiente manera: 1 si la variable Sales es superior a 8, 0 en caso contrario.

Ejercicio 8

Algunas consideraciones

Para este ejercicio van a tener que trabajar con el conjunto de datos Carseats, del libro *“An Introduction to Statistical Learning”*.

- Relacionado con las ventas de asientos de automóvil en 400 tiendas.
- Contiene información sobre factores que podrían influir en las ventas de estos productos en diferentes ubicaciones.
- Los datos incluyen tanto variables categóricas como numéricas.

Pueden cargarlo en R usando:

```
# Instalamos el paquete ISLR
install.packages("ISLR")

# Cargar el paquete
library(ISLR)

# Cargar el dataset Carseats
data("Carseats")
```

o en Python:

```
# Instalamos paquete ISLP
!pip install ISLP

# Importamos función para cargar datos
from ISLP import load_data

# Cargamos los datos
Carseats = load_data("Carseats")
```

Ejercicio 8

Algunas consideraciones

Carseats

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No
...
395	12.57	138	108	17	203	128	Good	33	14	Yes	Yes
396	6.14	139	23	3	37	120	Medium	55	11	No	Yes
397	7.41	162	26	12	368	159	Medium	40	18	Yes	Yes
398	5.94	100	79	7	284	95	Bad	50	12	Yes	Yes
399	9.71	134	37	0	27	120	Good	49	16	Yes	Yes

400 rows x 11 columns

- Tenemos 400 observaciones (filas de la matriz) y 11 variables (columnas).
- Nos interesa predecir la variable Sales.

Ejercicio 8

Algunas consideraciones

Fijaremos una semilla. En R lo hacemos así:

```
# Para reproducibilidad, fijamos semilla  
set.seed(2024)
```

y en Python así:

```
# Para reproducibilidad, fijamos semilla  
import random  
random.seed(2024)
```

Esto inicializa el generador de números aleatorios, y garantiza la reproducibilidad de los resultados.