

---

# **Aplicaciones matemáticas para predicción de afecciones coronarias mediante inteligencia artificial**

---

Alberth Yesid Pinco Bravo



Universidad del Valle  
Departamento de Matemáticas  
Santiago de Cali, Colombia  
Septiembre, 2025

---

# **Aplicaciones matemáticas para predicción de afecciones coronarias mediante inteligencia artificial**

---

*Director*

*Autor* Andrés Valderrama, M.Sc

Alberth Yesid Pinco Bravo *Co-director*

Heliana Arias, Ph.D

Trabajo de Grado

Universidad del Valle  
Departamento de Matemáticas  
Santiago de Cali, Colombia  
Septiembre, 2025

# Tabla de contenidos.

<b>0. Introducción</b>	<b>2</b>
<b>1. Marco teórico</b>	<b>3</b>
1.1. Análisis de componentes principales (PCA) . . . . .	3
1.1.1. Cálculo de las componentes principales . . . . .	7
1.1.2. Relación con la descomposición en valores singulares (SVD) . . . . .	14
1.1.3. Ejemplo práctico . . . . .	17
1.2. Árboles de decisión . . . . .	24
1.2.1. Algoritmo de Hunt . . . . .	25
1.2.2. Estructura de un árbol de decisión . . . . .	26
1.2.3. Criterios de partición . . . . .	29
1.2.4. Ejemplo . . . . .	36
1.3. eXtreme Gradient Boosting (XGBoost) . . . . .	40
1.4. Métricas de desempeño . . . . .	53
<b>2. Metodología</b>	<b>56</b>
<b>3. Resultados</b>	<b>57</b>
<b>4. Conclusiones</b>	<b>67</b>
<b>5. Limitaciones</b>	<b>67</b>
<b>6. Anexos</b>	<b>68</b>

## 0. Introducción

Según la Organización Mundial de la Salud (OMS), las enfermedades cardiovasculares son la principal causa de muerte en el mundo; se estima que 17,9 millones de personas fallecen al año por esta causa<sup>1</sup>. Frente a este desafío, la Inteligencia Artificial (IA) tiene la capacidad de identificar riesgos potenciales de manera oportuna, permitiendo intervenciones antes del desarrollo de un infarto cardíaco. La IA es capaz de identificar patrones que incluso los especialistas más experimentados podrían no detectar, representando un avance revolucionario en la práctica cardiológica.

En este contexto, este trabajo aborda el problema de predecir la presencia de síndrome coronario agudo (SCA) a partir de datos estructurados provenientes de la Clínica de Occidente de la ciudad de Cali, con una muestra de 859 pacientes y 24 variables clínicas y paraclínicas. El objetivo es construir un modelo de machine learning capaz de clasificar pacientes con SCA y, a la vez, exponer con rigor los fundamentos matemáticos de los métodos empleados que lo sustentan.

La estrategia metodológica combina: (i) el análisis de componentes principales (PCA) como método de preprocesamiento para filtrar información relevante y eliminar el ruido; (ii) los árboles de decisión como predictores base, los cuales se basan en decisiones secuenciales que imitan el razonamiento lógico humano; y (iii) extreme gradient boosting (XGBoost), entendido como método de ensamble que integra un conjunto de árboles mediante una optimización secuencial de manera que cada nuevo árbol corrige las predicciones de los árboles anteriores, aumentando la precisión y la robustez del modelo.

La sinergia entre PCA y XGBoost mejoró el desempeño respecto del mismo modelo sin PCA, lo que respalda la pertinencia de la estrategia implementada en este conjunto de datos. Además, en comparación con modelos de regresión logística y Random Forest, **XGBoost+PCA** obtuvo los mejores resultados en exactitud (accuracy), sensibilidad (recall) y F1-score, perfilándose como candidato a modelo para desplegar y contribuir de esta manera al desarrollo de estrategias de intervención y prevención del síndrome coronario agudo.

---

<sup>1</sup>[www.who.int](http://www.who.int)

# 1. Marco teórico

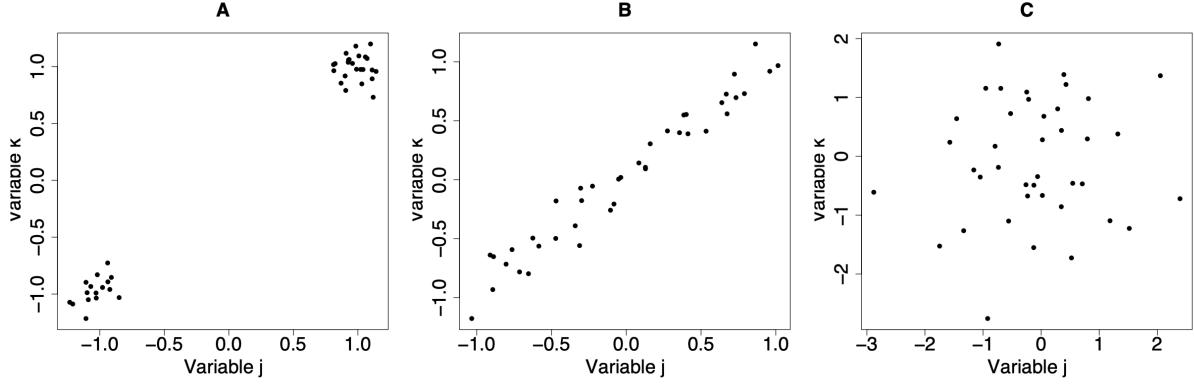
En esta sección se presentan los fundamentos matemáticos de los métodos que sostienen el modelo predictivo —el análisis de componentes principales (PCA) y el extreme gradient boosting (XGBoost)—, junto con las métricas de evaluación comúnmente utilizadas para medir el desempeño de un modelo. Dado que XGBoost es un método de ensamble que construye y optimiza árboles de decisión de forma aditiva, resulta imprescindible exponer primero la teoría de los árboles para comprender a profundidad su funcionamiento. Por ello, este capítulo se estructura en el siguiente orden: PCA → árboles de decisión → XGBoost → métricas de desempeño.

Para la sección de PCA se usaron como referencias [13, 16, 6, 4, 11, 2, 17]; para la sección de árboles de decisión [10, 24, 12, 18, 1, 11, 15, 23, 5, 9, 21, 3, 22]; para la de XGBoost [7, 8, 11, 9, 4]; y para la de métricas de desempeño [14, 18, 24].

## 1.1. Análisis de componentes principales (PCA)

El análisis de componentes principales es un método estadístico multidimensional utilizado para reducir la dimensión de un conjunto de datos. Se dice multidimensional por 2 razones: la primera es que cada observación, muestra, o individuo que se tiene está representado por varias variables; por ejemplo, un paciente de un conjunto de datos clínicos, el cual está caracterizado por su edad, sexo, peso, estatura, tipo de sangre, entre otros. La segunda razón es porque este método permite analizar simultáneamente las relaciones entre las variables y las observaciones, capturando así la estructura general del conjunto de datos. Dando como resultado un nuevo conjunto de variables sintéticas, conocidas como componentes principales.

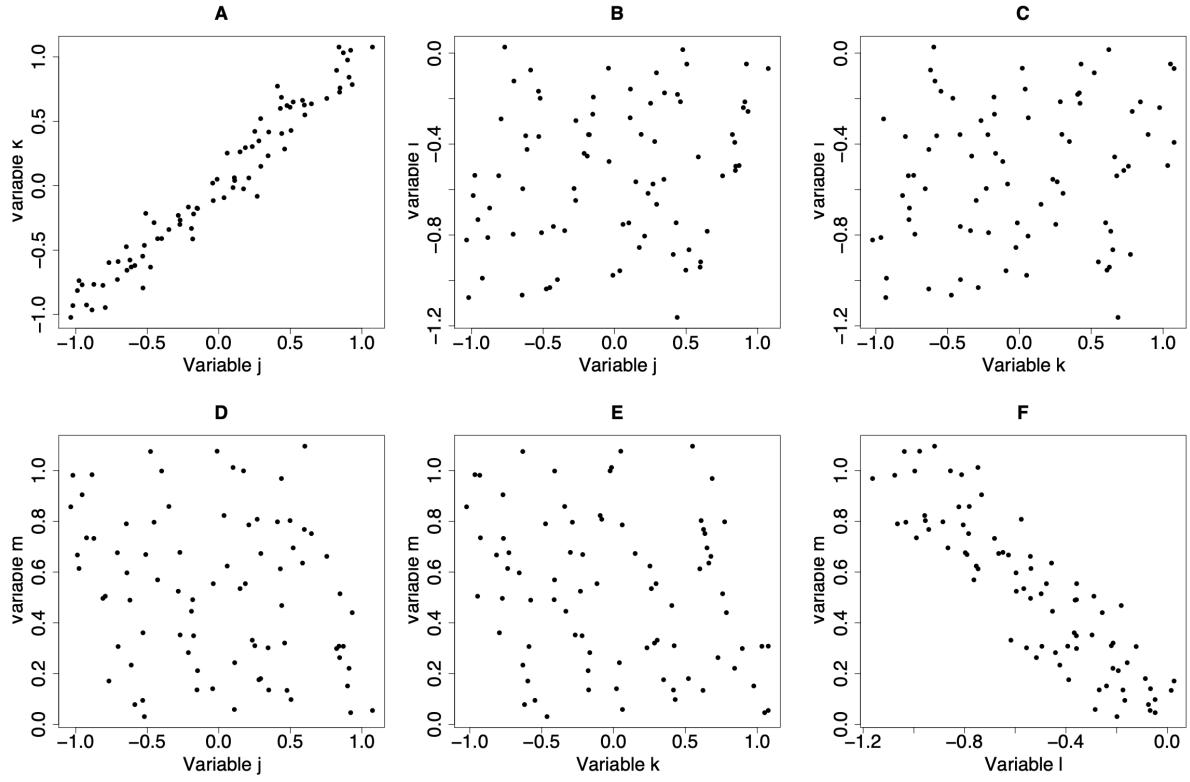
Supongamos que tenemos 3 conjuntos de datos distintos, cada uno con 40 individuos y descrito por 2 variables ( $j$  y  $k$ ), como se aprecia en la [Figura 1](#). En el gráfico A se observa una clara separación de los individuos en 2 grupos, mientras que en el gráfico B se observa una correlación lineal entre las variables; a medida que aumenta la variable  $j$ , aumenta la variable  $k$ , es decir, son directamente proporcionales. Por otro lado, en el gráfico C no se evidencia ningún tipo de patrón o estructura.



**Figura 1.** 40 individuos descritos por dos variables ( $j$  y  $k$ ). Fuente: tomada de [13]

Ejemplos como estos son fáciles de ver cuando se trabaja con 2 dimensiones. Sin embargo, en la práctica un conjunto de datos suele estar descrito por una cantidad significativa de variables; por ejemplo, un conjunto de imágenes, donde cada imagen es un individuo, y cada pixel es una variable. Así, una sola imagen con un tamaño de  $50 \times 50$  estaría representada por 2500 variables. Por lo tanto, si quisiéramos entrenar un modelo de machine learning con todas estas variables sería bastante tedioso e ineficiente ya que conllevaría un alto costo computacional y podría dar lugar a problemas como overfitting, en el cual el modelo se sobreajusta a los datos de entrenamiento, ocasionando que este no generalice bien cuando se le pasen nuevos datos. Sin embargo, este tipo de métodos no solo son valiosos cuando se cuenta con un número elevado de variables. También se usan en datasets más pequeños, de por ejemplo solo 10 variables, porque precisamente el objetivo es buscar relaciones lineales entre estas. Así, si se cuenta con 2 o 3 variables que están correlacionadas, podemos evitar esa redundancia representando dicho conjunto de variables con una sola, una nueva variable sintética.

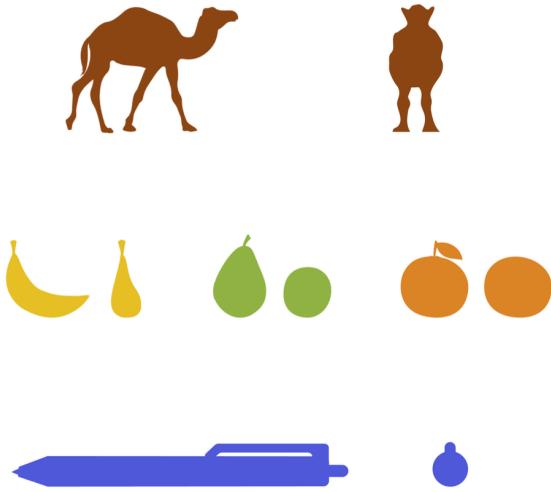
En la [Figura 2](#) se realiza una comparación dos a dos de las variables ( $j, k, l, m$ ) de un conjunto de datos. En el cual se observa que hay una correlación positiva entre las variables  $j$  y  $k$  (gráfico A), y una correlación negativa entre las variables  $l$  y  $m$  (gráfico F). Por lo que tendría mucho sentido dividir estas variables en 2 grupos,  $(j, k)$  y  $(l, m)$ , y representar a cada grupo con una sola variable. Esto puesto que aquellas variables pertenecientes a un mismo grupo están muy correlacionadas, mientras que de un grupo a otro no existe correlación.



**Figura 2.** Relaciones entre las variables ( $j, k, l$  y  $m$ ). Fuente: tomada de [13]

Es claro que si se cuenta con un número reducido de variables, bastaría con observar las nubes de puntos o analizar la matriz de correlación. Sin embargo, en el caso de tener, por ejemplo, 15 variables, esto implicaría examinar 105 coeficientes de correlación, lo que resulta poco práctico. Por esta razón, nos vemos en la necesidad de utilizar un método que permita sintetizar el conjunto de datos independientemente del número de variables que se tengan. No obstante, dicha reducción debe realizarse de tal manera que se conserve la mayor cantidad de información posible. Es decir, buscaremos la proyección de la nube de puntos que maximice la representación de las características esenciales del conjunto de datos.

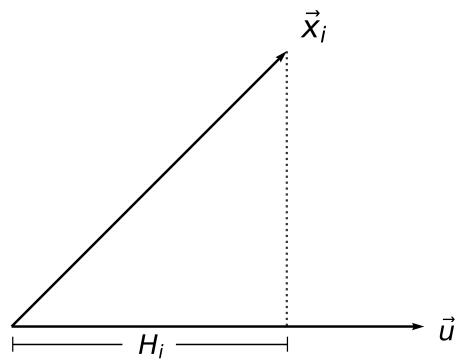
Este concepto se ejemplifica muy bien en la [Figura 3](#), donde se muestra cómo, al proyectar ciertos objetos tridimensionales sobre un plano, siempre existe un ángulo ideal que proporciona la mejor representación de las relaciones entre las variables, aquella que deforma lo menos posible la nube de puntos. Consiguiendo así, que las distancias entre los puntos



**Figura 3.** Proyecciones de distintos objetos tridimensionales sobre un plano

proyectados sean tan próximas como las distancias entre los puntos originales.

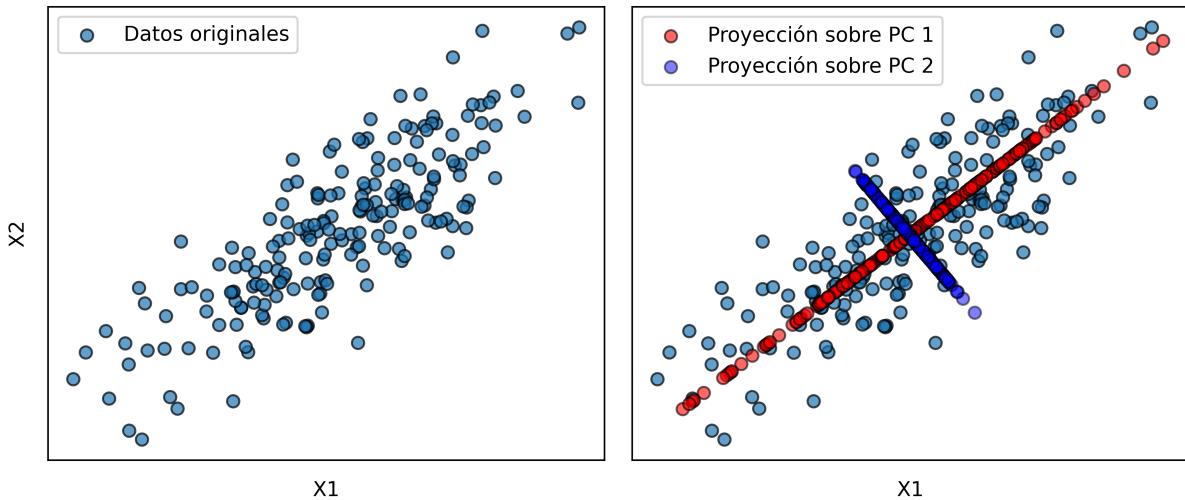
De este modo, dado que la proyección de un vector a lo más podrá tener la misma magnitud que dicho vector, el objetivo será maximizar las magnitudes de estas proyecciones. Esto es equivalente a determinar un vector  $u$  que maximice  $\sum_{i=1}^I H_i^2$ , donde  $H_i$  representa la proyección del individuo  $x_i$  sobre el vector  $u$  ([Figura 4](#)). Así,  $u$  define la dirección en la que el conjunto de datos está mas disperso, y por lo tanto en la que se concentra la información mas relevante.



**Figura 4.** Proyección de  $x_i$  sobre  $u$ .

### 1.1.1. Cálculo de las componentes principales

Supongamos que nuestro conjunto de datos está representado por una matriz  $X \in \mathbb{R}^{n \times p}$ , donde  $n$  corresponde al número de observaciones (individuos) y  $p$  al número de características (variables). Además, para el desarrollo de este análisis asumiremos que  $X$  ha sido previamente centrada, es decir, que cada variable  $X_j$  ( $j = 1, 2, \dots, p$ ) tiene media  $\bar{X}_j = \frac{1}{n} \sum_{i=1}^n x_{ij} = 0$ . Esta condición implica que el vector promedio de las filas de  $X$  es el origen en  $\mathbb{R}^p$ . Más adelante entenderemos porqué esta transformación es fundamental para el análisis.



**Figura 5.** Ejemplo PCA en 2 variables: datos ( $X_1, X_2$ ) y proyecciones sobre  $PC_1$  y  $PC_2$ .

Consideremos ahora el vector unitario  $u \in \mathbb{R}^p$ , que representa una dirección en el espacio de las características. La proyección de un individuo  $x_i = (x_{i1}, x_{i2}, \dots, x_{ip}) \in \mathbb{R}^p$  sobre la dirección  $u$  está dada por el producto escalar  $x_i \cdot u$ . Por lo tanto, la proyección de todos los individuos en esta dirección puede representarse por el vector  $Xu \in \mathbb{R}^n$ ,

$$\begin{array}{ccc}
 X & u & Xu \\
 \left[ \begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{array} \right] & \left[ \begin{array}{c} u_1 \\ u_2 \\ \vdots \\ u_p \end{array} \right] & = \left[ \begin{array}{c} x_1 \cdot u \\ x_2 \cdot u \\ \vdots \\ x_n \cdot u \end{array} \right].
 \end{array}$$

Luego, para determinar la dirección en la que se encuentra una mayor dispersión de los puntos proyectados, debemos maximizar la varianza de estas proyecciones, es decir, maximizar  $Var(Xu)$ . Por definición, la varianza se calcula como

$$Var(Xu) = \frac{1}{n} \sum_{i=1}^n (x_i \cdot u - \mu_u)^2,$$

donde  $\mu_u = \frac{1}{n} \sum_{i=1}^n x_i \cdot u$  es la media de las proyecciones en la dirección  $u$ .

Ahora bien, expandiendo  $\mu_u$  podemos observar que

$$\begin{aligned} \mu_u &= \frac{1}{n} (x_1 + x_2 + \cdots + x_n) \cdot u, \\ &= \frac{1}{n} \left( \sum_{i=1}^n x_{i1}, \sum_{i=1}^n x_{i2}, \dots, \sum_{i=1}^n x_{ip} \right) \cdot u, \\ &= \left( \frac{1}{n} \sum_{i=1}^n x_{i1}, \frac{1}{n} \sum_{i=1}^n x_{i2}, \dots, \frac{1}{n} \sum_{i=1}^n x_{ip} \right) \cdot u, \\ &= (\bar{X}_1, \bar{X}_2, \dots, \bar{X}_p) \cdot u, \\ &= 0 \cdot u, \\ &= 0, \end{aligned}$$

pues cada variable  $X_j$  tiene media  $\bar{X}_j = 0$  para todo  $j = 1, \dots, p$ .

Esto implica que la varianza de estas proyecciones se reduce a

$$\begin{aligned} Var(Xu) &= \frac{1}{n} \sum_{i=1}^n (x_i \cdot u - 0)^2, \\ &= \frac{1}{n} \sum_{i=1}^n (x_i \cdot u)^2, \\ &= \frac{1}{n} (Xu)^\top Xu, \\ &= \frac{1}{n} u^\top X^\top Xu. \end{aligned}$$

Sin embargo, note que el producto  $X^\top X$  se puede expresar de la siguiente forma

$$\begin{aligned}
 & \begin{bmatrix} X^\top \\ X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix} \begin{bmatrix} X \\ | & | & | \\ X_1 & X_2 & \cdots & X_p \end{bmatrix} = \begin{bmatrix} X_1^\top X \\ X_2^\top X \\ \vdots \\ X_p^\top X \end{bmatrix} \\
 & = n \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix} = n\Sigma.
 \end{aligned}$$

Aquí,  $\sigma_{ij} = Cov(X_i, X_j)$  representa la covarianza entre las variables  $X_i$  y  $X_j$ . Por lo tanto, la matriz de covarianza  $\Sigma$  puede expresarse como

$$\Sigma = \frac{1}{n} X^\top X.$$

Más aun, si las variables además de estar centradas, han sido estandarizadas, es decir, cada variable  $X_j$  tiene desviación estándar  $\sigma_j = 1$ , entonces los elementos  $\sigma_{ij}$  corresponden directamente a los coeficientes de correlación  $\rho_{ij}$ , ya que por definición

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j},$$

y por lo tanto  $\Sigma = \frac{1}{n} X^\top X$  se convierte en la matriz de correlación.

Es por esta razón que el PCA exige estandarizar (o al menos centrar) el conjunto de datos. De lo contrario, el producto  $\frac{1}{n} X^\top X$  no representaría la matriz de correlación (o covarianza).

Se denomina PCA no normado cuando las variables han sido únicamente centradas, y PCA normado cuando, además de estar centradas, han sido escaladas por su desviación estándar [13]. Este ultimo enfoque es el más común, ya que en la mayoría de los casos las variables tienen diferentes unidades de medida (kg, cm, °C, etc.), lo que puede generar sesgos.

Al estandarizar, se garantiza que todas las variables contribuyan de manera equitativa, evitando la dominancia de aquellas con mayor magnitud. De este modo nos aseguramos de que las proyecciones reflejen la verdadera dispersión de los datos, independientemente de las magnitudes de las variables originales. Por lo tanto, a partir de ahora asumiremos que el conjunto de datos  $X$  ha sido previamente estandarizado.

Bajo esta suposición, la varianza de las proyecciones que definimos antes como  $Var(Xu)$ , puede reescribirse de la siguiente manera:

$$\begin{aligned} Var(Xu) &= \frac{1}{n} u^\top X^\top Xu, \\ &= \frac{1}{n} u^\top (n\Sigma)u, \\ &= u^\top \Sigma u. \end{aligned}$$

De este modo, nuestro objetivo será encontrar un vector  $u$  unitario ( $u^\top u = 1$ ) tal que maximice  $u^\top \Sigma u$ . En otras palabras, queremos maximizar

$$f(u) = u^\top \Sigma u$$

sujeto a la condición

$$g(u) = u^\top u - 1 = 0.$$

Por lo tanto, haciendo uso de los multiplicadores de Lagrange tenemos que

$$\begin{aligned} \mathcal{L}(u, \lambda) &= u^\top \Sigma u - \lambda(u^\top u - 1), \\ &= \sum_{i,j=1}^p u_i \sigma_{ij} u_j - \lambda \left( \sum_{i=1}^p u_i^2 - 1 \right). \end{aligned}$$

De lo cual, derivando con respecto a cada componente de  $u$  obtenemos

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial u_k} &= 2 \sum_{i=1}^p \sigma_{ki} u_i - 2\lambda u_k, \\ \frac{\partial \mathcal{L}}{\partial u} &= 2\Sigma u - 2\lambda u. \end{aligned}$$

Por consiguiente, al igualar a cero para encontrar el máximo tendremos que

$$\begin{aligned} 2\Sigma u - 2\lambda u &= 0, \\ \Sigma u - \lambda u &= 0, \\ \Sigma u &= \lambda u. \end{aligned}$$

Esto nos indica que la dirección que estamos buscando, la cual maximiza la varianza de las proyecciones, corresponde a un vector propio de la matriz de correlación  $\Sigma$ . En particular, al multiplicar por  $u^\top$  a ambos lados de la ecuación obtenemos

$$\begin{aligned} u^\top \Sigma u &= u^\top \lambda u, \\ &= \lambda(u^\top u), \\ &= \lambda \cdot 1, \\ Var(Xu) &= \lambda. \end{aligned}$$

Así, maximizar  $Var(Xu)$  equivale a encontrar el mayor valor propio de la matriz de correlación. Y por lo tanto, si ordenamos los valores propios de  $\Sigma$  como  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$ , entonces el vector propio  $u = v_1$  asociado a  $\lambda_1$  es la dirección en la que se encuentra la mayor dispersión de los datos proyectados [2].

De esta manera construimos lo que se conoce como la primera componente principal

$$\psi_1 = Xv_1.$$

Dicha componente, perteneciente a  $\mathbb{R}^n$ , es una combinación lineal de las variables originales, pues se puede expresar como

$$\psi_1 = v_{11}X_1 + v_{12}X_2 + \dots + v_{1p}X_p,$$

donde  $v_{1j}$  representa el peso asociado a la variable  $X_j$ . Esto implica que, cuanto mayor sea  $|v_{1j}|$ , más contribuirá  $X_j$  a la dirección de  $\psi_1$ , lo que indica una mayor relevancia de esta variable en la primera componente principal.

Suponiendo por ejemplo que  $X$  es un conjunto de datos que describe a cuatro individuos mediante tres variables  $(x, y, z)$ , la proyección de estos individuos sobre la dirección de  $v_1$

se puede expresar como

$$X \begin{bmatrix} v_1 \\ v_{11} \\ v_{12} \\ v_{13} \end{bmatrix} = \begin{bmatrix} \psi_1 \\ x_1 v_{11} + y_1 v_{12} + z_1 v_{13} \\ x_2 v_{11} + y_2 v_{12} + z_2 v_{13} \\ x_3 v_{11} + y_3 v_{12} + z_3 v_{13} \\ x_4 v_{11} + y_4 v_{12} + z_4 v_{13} \end{bmatrix}.$$

De esta manera, hemos reducido el conjunto de datos original a una sola variable ( $\psi_1$ ), obteniendo la mejor representación unidimensional posible. Sin embargo, esta representación no siempre es suficiente para capturar toda la información relevante. Lo que nos lleva a buscar una segunda dirección que capture la mayor varianza de esa información restante que no ha sido representada por la primera componente principal.

Por lo tanto, de manera similar a como se realizó anteriormente, nuestro objetivo ahora es encontrar otro vector unitario  $u$  que maximice  $Var(Xu) = u^\top \Sigma u$ , sujeto a la condición adicional de que  $Xu$  no comparta información con  $\psi_1 = Xv_1$ , es decir, que no exista una correlación lineal entre estas.

En este contexto, la falta de correlación se expresa como

$$Cov(Xu, \psi_1) = E[(Xu)^\top \psi_1] - E[Xu]E[\psi_1] = 0.$$

Luego, dado que  $Xu$  y  $\psi_1$  son combinaciones lineales de las variables originales, las cuales están centradas en el origen, sus medias son cero. Por lo tanto, la ecuación se reduce a

$$\begin{aligned} E[(Xu)^\top \psi_1] &= 0, \\ E[u^\top X^\top X v_1] &= 0, \\ E[u^\top n \Sigma v_1] &= 0, \\ nE[u^\top \Sigma v_1] &= 0, \\ E[u^\top \Sigma v_1] &= 0, \\ u^\top \Sigma v_1 &= 0. \end{aligned}$$

Recuerde que  $\Sigma v_1 = \lambda_1 v_1$ , lo cual implica que  $\lambda_1 u^\top v_1 = 0$  y por lo tanto  $u^\top v_1 = 0$  ya que

$\lambda_1 \neq 0$  (pues es el mayor valor propio de  $\Sigma$ , una matriz semidefinida positiva y no nula).

Por lo tanto, haciendo uso nuevamente de los multiplicadores de Lagrange, la función a optimizar será

$$\mathcal{L}(u, \lambda, \delta) = u^\top \Sigma u - \lambda(u^\top u - 1) - 2\delta u^\top \Sigma v_1,$$

por lo que al derivar con respecto a  $u$  e igualar a cero tendremos que

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial u} &= 2\Sigma u - 2\lambda u - 2\delta \Sigma v_1 = 0, \\ \Sigma u - \lambda u - \delta \Sigma v_1 &= 0. \end{aligned}$$

Luego, multiplicando a izquierda por  $v_1^\top$  obtenemos

$$\begin{aligned} v_1^\top \Sigma u - \lambda v_1^\top u - \delta v_1^\top \Sigma v_1 &= 0, \\ -\delta v_1^\top \Sigma v_1 &= 0, \\ -\delta v_1^\top \lambda_1 v_1 &= 0, \\ -\delta \lambda_1 &= 0, \end{aligned}$$

lo que implica que  $\delta = 0$  (pues  $\lambda_1 \neq 0$ ), y por lo tanto,

$$\begin{aligned} \Sigma u - \lambda u &= 0, \\ \Sigma u &= \lambda u, \\ u^\top \Sigma u &= \lambda, \\ Var(Xu) &= \lambda. \end{aligned}$$

Es decir, maximizar  $Var(Xu)$  nuevamente equivale a encontrar el mayor de los valores propios de  $\Sigma$  bajo la condición de que el vector propio asociado  $u$  sea ortogonal al eje definido por  $v_1$ , que corresponde a la primera componente principal. En otras palabras, buscamos que  $u^\top v_1 = 0$ , lo cual se cumple siempre, ya que los vectores propios asociados a valores propios distintos son ortogonales siempre que la matriz es simétrica.

Por lo tanto, si  $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_p$ , tomaremos  $u$  como  $v_2$ , el vector propio asociado a  $\lambda_2$ . Así, la componente principal  $\psi_2 = Xv_2$  capturará la mayor varianza posible que no ha sido capturada por  $\psi_1$ . En consecuencia,  $\psi_1$  y  $\psi_2$  no comparten información y no hay redundancia.

En el caso de que  $\lambda_1$  sea igual a  $\lambda_2$ , es decir, que este valor propio tenga multiplicidad algebraica  $m_a(\lambda_1) \geq 2$ , no representa ningún problema. Pues sabemos que para cualquier matriz simétrica con entradas reales, la dimensión del espacio propio asociado a un valor propio  $\lambda$  es igual a su multiplicidad algebraica. Por lo tanto, existe un conjunto de vectores propios linealmente independientes asociados a  $\lambda$ . Luego, a partir de dicho conjunto, utilizando el método de Gram-Schmidt, podemos obtener un conjunto de vectores propios ortogonales asociados a  $\lambda$ .

De forma iterativa, se construyen las componentes principales restantes bajo la condición de que cada componente principal  $\psi_k = Xv_k$  no esté correlacionada con ninguna de las anteriores. Lo que es equivalente a  $v_k^\top v_j = 0$  para cada  $j = 1, \dots, k-1$ . (Una demostración mas detallada se encuentra en el texto *An Introduction to Multivariate Statistical Analysis* de T. W. Anderson [2]).

De este modo, construimos la matriz  $V$ , cuyas columnas corresponden al conjunto de vectores propios  $\{v_1, v_2, \dots, v_p\}$  organizados de manera que  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$ . Por lo tanto, el nuevo sistema de coordenadas definido por las componentes principales es dado por el producto matricial  $XV$

$$X = VZ$$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ x_{31} & x_{32} & \cdots & x_{3p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \psi_1 & \psi_2 & \cdots & \psi_p \\ | & | & & | \end{bmatrix}.$$

Aquí,  $Z$  contiene a las componentes principales organizadas en columnas, donde  $\psi_k$  representa la proyección de cada uno de los individuos del conjunto de datos  $X$ , sobre el eje definido por el vector propio  $v_k$ .

### 1.1.2. Relación con la descomposición en valores singulares (SVD)

Como ya vimos, el proceso de construcción de las componentes principales consiste sustancialmente en encontrar los valores y vectores propios de la matriz de correlación

$\Sigma = \frac{1}{n}X^\top X$ . Lo cual puede lograrse de manera eficiente y con un menor costo computacional mediante la descomposición en valores singulares (SVD) de  $X$ . La cual nos dice que, si  $X$  es una matriz de tamaño  $n \times p$ , esta puede ser escrita como

$$X = UDV^\top,$$

donde  $U \in \mathbb{R}^{n \times n}$  y  $V \in \mathbb{R}^{p \times p}$  son matrices ortogonales, y  $D \in \mathbb{R}^{n \times p}$  es una matriz diagonal cuyas entradas  $\sigma_i$  son los valores singulares de  $X$  ordenados de manera descendente [17]. Estos valores cumplen que

$$\sigma_i = \sqrt{\alpha_i},$$

donde  $\alpha_i$  es un valor propio de  $X^\top X$ . En cuanto a  $V$ , las columnas de esta matriz son los vectores propios de  $X^\top X$ , ordenados de acuerdo con  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p$ .

Por lo tanto, las columnas  $v_i$  de  $V$  cumplen que

$$\begin{aligned} X^\top X v_i &= \sigma_i^2 v_i, \\ \frac{1}{n} X^\top X v_i &= \frac{1}{n} \sigma_i^2 v_i. \end{aligned}$$

Por ende, si  $X$  ha sido previamente estandarizada (es decir, sus variables tienen media cero y desviación estándar igual a 1), entonces  $\frac{1}{n}X^\top X$  es precisamente la matriz de correlación.

Por lo tanto

$$\Sigma v_i = \frac{\sigma_i^2}{n} v_i.$$

Esto muestra que los  $v_i$  son exactamente los vectores propios de la matriz de correlación  $\Sigma$ , asociados a los valores propios  $\lambda_i = \frac{\sigma_i^2}{n}$ .

En consecuencia, basta con realizar la descomposición SVD de la matriz de datos  $X$  (estandarizada) y observar que

$$XV = UD = Z.$$

Lo que implica que la matriz  $Z$ , que contiene a las componentes principales como columnas, está dada por el producto  $UD$ . Lo cual resalta la relación directa entre el PCA y la SVD.

Además, dado que  $X^\top X v_i = \sigma_i^2 v_i$ , se tiene que

$$\begin{aligned} v_i^\top X^\top X v_i &= \sigma_i^2, \\ (X v_i)^\top X v_i &= \sigma_i^2, \\ \|X v_i\|^2 &= \sigma_i^2, \\ \|X v_i\| &= \sigma_i. \end{aligned}$$

Así, los valores singulares ( $\sigma_i$ ), en el contexto del PCA, representan exactamente la magnitud de cada componente principal  $\psi_i = X v_i$ . La cual está directamente relacionada con la cantidad de información que cada componente captura, o como suele denominarse en la literatura, su varianza explicada. Siendo así las primeras componentes principales las que capturan la mayor parte de la información, y las últimas aquellas que son menos significativas.

En el caso de la componente principal  $\psi_k$ , su varianza explicada es

$$Var(\psi_k) = Var(X v_k) = \lambda_k.$$

Por lo tanto, la proporción de varianza explicada por las primeras  $k$  componentes respecto a la varianza total del conjunto de datos es dada por

$$\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i}.$$

Finalmente, el objetivo es seleccionar un subconjunto de componentes principales que sea suficiente para capturar la información más relevante. La dificultad está en que no existe una regla universal para determinar el número óptimo de componentes principales a conservar, ya que depende en gran medida del conjunto de datos que se está analizando, y del criterio del analista. No obstante, existen diversas técnicas que pueden facilitar esta cuestión, como un gráfico de codo o un gráfico del porcentaje de varianza explicada.

### 1.1.3. Ejemplo práctico

Para ilustrar mejor los conceptos vistos, consideremos un pequeño ejemplo con un conjunto de datos en el que se evaluaron 7 características de 6 de zumos de naranja como se muestra en la [Tabla 1](#). Cada observación está descrita por 7 variables como la intensidad y el tipo de olor, la intensidad del sabor, y su carácter pulposo, ácido, amargo y azucarado.

	Intensidad olor	Tipo olor	Carácter pulposo	Intensidad sabor	Carácter ácido	Carácter amargo	Carácter azucarado
Pampryl amb.	2.82	2.53	1.66	3.46	3.15	2.97	2.60
Tropicana amb.	2.76	2.82	1.91	3.23	2.55	2.08	3.32
Fruvita fr.	2.83	2.88	4.00	3.45	2.42	1.76	3.38
Joker amb.	2.76	2.59	1.66	3.37	3.05	2.56	2.80
Tropicana fr.	3.20	3.02	3.69	3.12	2.33	1.97	3.34
Pampryl fr.	3.07	2.73	3.34	3.54	3.31	2.63	2.90

**Tabla 1.** Datos de los zumos de naranja. Fuente: tomada de [13]

Luego, haciendo uso de la librería Pandas en Python, podemos guardar estos datos en un dataframe, y calcular la matriz de correlación de la [Tabla 2](#) de la siguiente manera

```
import pandas as pd
data_jugos #dataframe con los datos originales
data_jugos_correlation = data_jugos.corr() #matriz de correlación
```

	Intensidad olor	Tipo olor	Carácter pulposo	Intensidad sabor	Carácter ácido	Carácter amargo	Carácter azucarado
Intensidad olor	1	0.58	0.66	-0.27	-0.15	-0.15	0.23
Tipo olor	0.58	1	0.77	-0.62	-0.84	-0.88	0.92
Carácter pulposo	0.66	0.77	1	-0.02	-0.47	-0.64	0.63
Intensidad sabor	-0.27	-0.62	-0.02	1	0.73	0.51	-0.57
Carácter ácido	-0.15	-0.84	-0.47	0.73	1	0.91	-0.90
Carácter amargo	-0.15	-0.88	-0.64	0.51	0.91	1	-0.98
Carácter azucarado	0.23	0.92	0.63	-0.57	-0.90	-0.98	1

**Tabla 2.** Matriz de correlación de los zumos de naranja. Fuente: tomada de [13]

Podemos observar como algunas variables tienen una fuerte correlación positiva, como el carácter azucarado y tipo de olor, o el carácter ácido con el carácter amargo. También una fuerte correlación negativa del carácter azucarado con el carácter ácido y el carácter amargo.

Posteriormente, calculamos los valores y vectores propios de esta matriz de correlación y tomamos los 2 primeros.

```
import numpy as np
eigen_vals, eigen_vecs = np.linalg.eig(data_jugos_correlation)

lambda_1 = eigen_vals[0] #valor propio 1
v1 = eigen_vecs[:,0] #vector propio 1

lambda_2 = eigen_vals[1] #valor propio 2
v2 = eigen_vecs[:,1] #vector propio 2
```

```
Vector propio asociado a lambda_1 = 4.74
[ 0.21  0.45  0.33 -0.3 -0.42 -0.43  0.44]
```

```
-----
Vector propio asociado a lambda_2 = 1.33
[ 0.65  0.12  0.53  0.37  0.3   0.16 -0.14]
```

Por lo tanto, al estandarizar la matriz de datos originales y proyectarla sobre estos vectores propios, obtendremos las primeras componentes principales como columnas.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data_jugos_scaled = scaler.fit_transform(data_jugos) #estandarización

V = np.array([v1,v2]).T #vectores propios como columnas
Z = data_jugos_scaled.dot(V) #proyección de los datos

print(Z)
```

```
[[ -2.98 -0.08]
 [ 0.89 -1.72]
 [ 1.94  0.04]
 [-1.9   -0.83]
 [ 3.19  0.59]
 [-1.13  2.  ]]
```

También podemos realizar la descomposición en valores singulares de la matriz estandarizada y comprobar que las componentes principales se encuentran en  $UD$ .

```
U, D, VT = np.linalg.svd(data_jugos_scaled)

UD = U.dot(np.diag(D)) #conjunto completo de componentes principales

print(UD[:,0:2]) #mostramos las primeras 2
```

```
[[ 2.98 -0.08]
 [-0.89 -1.72]
 [-1.94  0.04]
 [ 1.9   -0.83]
 [-3.19  0.59]
 [ 1.13  2.  ]]
```

Como podemos notar, la única diferencia se encuentra en la primera componente principal, la cual presenta un cambio de signo. Esto se debe a la forma en que se calculan los vectores propios y no representa un problema ya que solo invierte el sentido del vector sin alterar su dirección ni su estructura.

Otra alternativa para calcular las componentes principales es mediante la clase PCA de la librería sklearn, cuyo algoritmo también emplea una descomposición en valores singulares.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)

pca_jugos = pca.fit_transform(data_jugos_scaled) #cálculo de las comp.
print(pca_jugos)
```

```
[ [-2.98 -0.08]
 [ 0.89 -1.72]
 [ 1.94  0.04]
 [-1.9   -0.83]
 [ 3.19  0.59]
 [-1.13  2.  ]]
```

Como vemos, estas tres formas de calcular las componentes principales son consistentes. Siendo las dos ultimas más eficientes y numéricamente más estables ya que la descomposición SVD evita el cálculo explícito de la matriz de correlación, lo que reduce los problemas de precisión numérica.

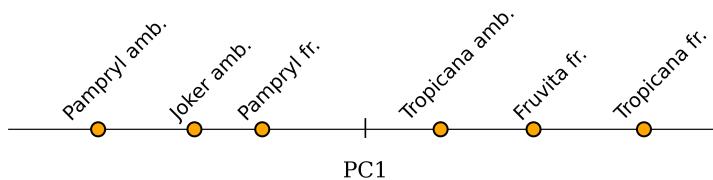
De este modo, hemos realizado la transformación del conjunto de variables originales, en un nuevo conjunto de variables ortogonales (no correlacionadas), ordenadas según la cantidad de varianza que explican.

Recordemos que la varianza explicada por cada componente principal está dada precisamente por el valor propio asociado a esa dirección, y por lo tanto, podemos calcular la proporción de varianza explicada por estas componentes respecto a la varianza total de la siguiente forma

```
var_exp_pc1 = lambda_1/sum(eigen_vals)*100
var_exp_pc2 = lambda_2/sum(eigen_vals)*100
```

```
varianza explicada por PC1: 67.77%
varianza explicada por PC2: 19.05%
```

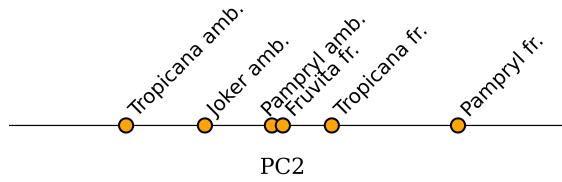
Lo que significa que el conjunto de datos que inicialmente se encuentra en 7 dimensiones, podríamos reducirlo a una sola dimensión, representandolo únicamente con la primera componente principal ([Figura 6](#)), y aun así capturar el 67.77 % de la información.



**Figura 6.** Datos de los zumos: Primera componente principal.

Este nuevo eje pareciera estar clasificando el conjunto de datos en dos grupos. Siendo los mas opuestos Pampryl ambiente y Tropicana fresco, cuya principal diferencia se encuentra en que Pampryl ambiente es el zumo con el tipo de olor más bajo, más amargo y menos dulce, mientras que Tropicana fresco tiene el tipo de olor más alto, es de los menos amargos y de los más dulces.

Por otro lado, note que si representaramos los datos no con la primera, sino con la segunda componente principal ([Figura 7](#)), estos quedarán un poco mas amontonados y, por ende, estaremos capturando mucha menos información (19.05 %). Además, el orden de las proyecciones claramente es distinto. En este caso los zumos de naranja que se oponen son Tropicana ambiente y Pampryl fresco, donde la única relación entre estos a simple vista es que Tropicana ambiente presenta el olor menos intenso, mientras que Pampryl fresco se encuentra entre los más intensos.



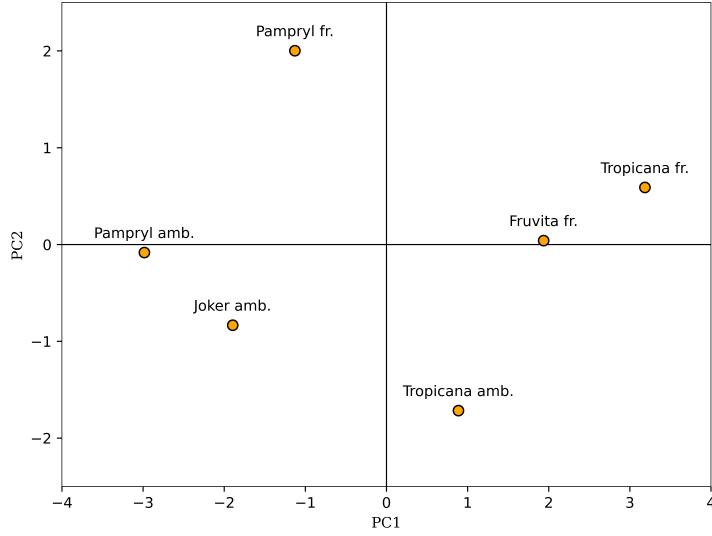
**Figura 7.** Datos de los zumos: Segunda componente principal.

Por ultimo, juntar estas 2 componentes principales nos dará la mejor representación planar de los datos ([Figura 8](#)). Lo que equivale a reducir 5 dimensiones, y capturar de este modo el 86.82 % de la información.

Ahora, es importante que entendamos lo que nos quieren representar estas componentes principales en función de las características que ya conocemos. En otras palabras, visualizar qué tanto contribuyen las variables originales en las direcciones de las componentes principales. Esto lo conseguimos calculando el coeficiente de correlación de cada variable con cada una de los componentes principales, dado por

$$\text{Corr}(X_i, \psi_k) = \frac{\text{Cov}(X_i, \psi_k)}{\sqrt{\text{Var}(X_i)} \sqrt{\text{Var}(\psi_k)}}.$$

Pero como cada  $X_i$  ha sido estandarizada, su varianza es igual a 1, y la covarianza entre



**Figura 8.** Datos de los zumos: Primeras 2 componentes principales. Fuente: [13]

esta y la componente principal  $\psi_k$  se puede expresar como

$$Cov(X_i, \psi_k) = \frac{1}{n} X_i \cdot \psi_k.$$

Sin embargo, recordemos que la matriz de datos  $X$  puede ser escrita de la siguiente forma

$$X = UDV^\top,$$

donde  $UD$  es la matriz de componentes principales y  $V^\top$  tiene como filas a los vectores propios de la matriz de correlación.

Esto significa que cada variable  $X_i$  puede ser escrita como combinación lineal de las componentes principales de la siguiente manera

$$X_i = \psi_1 v_{1i} + \psi_2 v_{2i} + \cdots + \psi_p v_{pi}.$$

Pero como las componentes principales son ortogonales entre ellas, se cumple que

$$\begin{aligned}
X_i \cdot \psi_k &= (\psi_1 v_{1i} + \psi_2 v_{2i} + \cdots + \psi_p v_{pi}) \cdot \psi_k, \\
&= v_{ki} (\psi_k \cdot \psi_k), \\
&= v_{ki} \sigma_k^2, \\
&= v_{ki} n \lambda_k,
\end{aligned}$$

donde  $v_{ki}$  representa la  $i$ -ésima entrada del  $k$ -ésimo vector propio,  $\sigma_k$  es la magnitud (valor singular correspondiente) de la componente principal  $\psi_k$ , y  $\lambda_k$  es el valor propio asociado a  $v_k$ .

Así, la covarianza entre la variable  $X_i$  y la componente principal  $\psi_k$  se simplifica a

$$\begin{aligned}
Cov(X_i, \psi_k) &= \frac{1}{n} (X_i \cdot \psi_k), \\
&= \frac{1}{n} (v_{ki} n \lambda_k), \\
&= v_{ki} \lambda_k.
\end{aligned}$$

De este modo, se reducen enormemente los cálculos, permitiendo obtener el coeficiente de correlación entre  $X_i$  y  $\psi_k$  simplemente con:

$$\begin{aligned}
Corr(X_i, \psi_k) &= \frac{Cov(X_i, \psi_k)}{\sqrt{Var(X_i)} \sqrt{Var(\psi_k)}}, \\
&= \frac{v_{ki} \lambda_k}{\sqrt{1} \sqrt{\lambda_k}}, \\
&= v_{ki} \sqrt{\lambda_k}.
\end{aligned}$$

Calculamos los coeficientes de correlación entre las variables originales y la primera componente principal con el siguiente código.

```

for i in range(data_jugos.shape[1]):
    corr = v1[i]*np.sqrt(lambda_1) #coeficiente de correlación (X_i,PC1)
    print(corr.round(2),end='\\t')

```

0.46 0.99 0.72 -0.65 -0.91 -0.93 0.95

Organizando estos coeficientes ([Tabla 3](#)), podemos observar como la primera componente principal ( $\psi_1$ ) está fuertemente correlacionada con 4 variables, positivamente con las variables tipo olor y carácter azucarado, y negativamente con carácter ácido y carácter amargo. Esto indica que los zumos con valores altos en la primera componente tienden a tener un olor más típico, ser más dulces, y al mismo tiempo menos ácidos y menos amargos.

	$\psi_1$	$\psi_2$
Intensidad olor	0.46	0.75
Tipo olor	0.99	0.13
Carácter pulposo	0.72	0.62
Intensidad sabor	-0.65	0.43
Carácter ácido	-0.91	0.35
Carácter amargo	-0.93	0.19
Carácter azucarado	0.95	-0.16

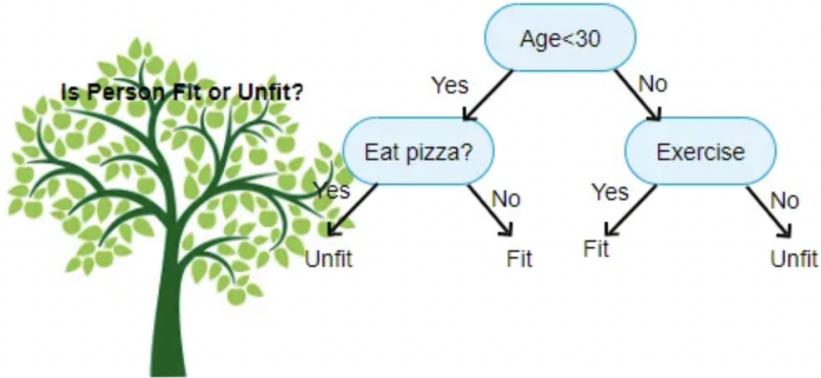
**Tabla 3.** Coeficientes de correlación de las variables originales con las dos primeras componentes principales. Fuente: tomada de [\[13\]](#).

En resumen, el análisis de componentes principales reexpresa el conjunto de datos en una base ortogonal que captura relaciones lineales entre las variables y concentra la información más relevante de los datos en unas pocas dimensiones. Es especialmente útil cuando existen muchas variables correlacionadas y, por ello, es ampliamente utilizado como método de pre procesamiento antes del entrenamiento de modelos supervisados o de clustering.

## 1.2. Árboles de decisión

Dentro de los métodos de machine learning que más se destacan por su popularidad y su capacidad para resolver problemas de clasificación, se encuentran los árboles de decisión. Estos hacen parte específicamente de los algoritmos de aprendizaje supervisado, una subcategoría del machine learning que toma un conjunto de datos etiquetados (cada muestra ha sido previamente clasificada) y entrena un algoritmo para que sea capaz de predecir resultados a partir de introducir nuevos datos.

Aunque este método es utilizado tanto para tareas de clasificación como de regresión, en esta sección nos interesamos principalmente en los árboles de decisión de clasificación,



**Figura 9.** Ejemplo de un árbol de decisión.<sup>2</sup>

donde la variable objetivo representa una categoría, a diferencia de la regresión donde lo que se busca es predecir un valor continuo.

Sin embargo, es importante tener en cuenta que estos algoritmos funcionan mejor cuando las categorías son binarias (“1” o “0”, “sí” o “no”) o nominales (sin un orden específico, como “perro”, “gato” o “conejo”). En contraste, pueden ser menos efectivos con categorías ordinales (como el estrato socioeconómico), ya que los árboles no consideran el orden implícito entre las categorías [24].

### 1.2.1. Algoritmo de Hunt

Los arboles de decisión tienen como base el algoritmo de Hunt [12], desarrollado en la decada de 1960 en el campo de la psicología con el objetivo de modelar el aprendizaje humano. Lo que se buscaba era entender como el ser humano toma decisiones y aprende a clasificar objetos basándose en reglas inductivas.

El algoritmo de Hunt consiste esencialmente en dividir el conjunto de datos de forma recursiva hasta obtener subconjuntos que contengan solo muestras de la misma clase. Esta idea se puede expresar de la siguiente forma:

---

<sup>2</sup>Imagen tomada de: [medium.com](https://medium.com)

1. Si todas las muestras en el conjunto pertenecen a una misma clase se etiqueta el conjunto con dicha clase.
2. Si el conjunto contiene más de una clase se divide en subconjuntos más pequeños en base a una característica y un criterio de partición.

Luego, se verifican estas 2 reglas para cada uno de los nuevos subconjuntos [24].

Esta forma de construcción, comúnmente denominada top-down recursive divide-and-conquer manner [10], no solo permite segmentar los datos de una manera sencilla, sino que también facilita la interpretación de los resultados, ya que genera una estructura de diagrama de flujo que representa el conjunto de reglas a seguir para clasificar una muestra. Por lo tanto, veamos la estructura de un árbol de decisión y las partes que la conforman.

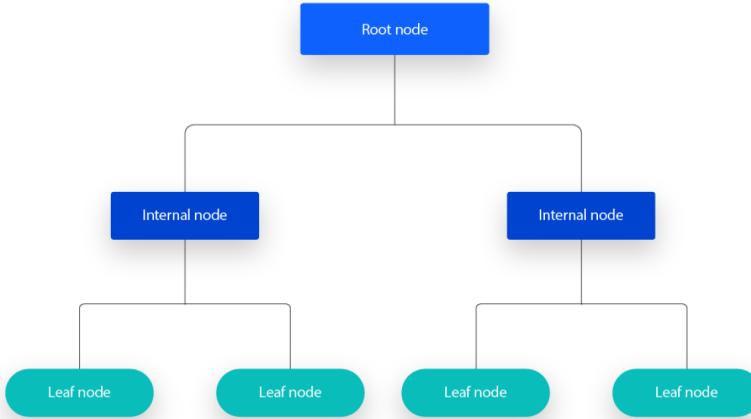
### 1.2.2. Estructura de un árbol de decisión

Un árbol de decisión está compuesto por un conjunto de nodos como se aprecia en la [Figura 10](#). El nodo raíz junto con los nodos internos representan atributos claves para hacer las divisiones, mientras que los nodos hojas representan las predicciones [24].

- **Nodo raíz:** No tiene aristas de entrada. Realiza la primera separación del conjunto de datos.
- **Nodos internos:** También llamados nodos de decisión. Tienen exactamente una arista de entrada y dos o más aristas de salida.
- **Nodos hoja:** Tienen exactamente una arista de entrada y ninguna arista de salida. Indican la clase a la que pertenece una muestra.

Se denomina profundidad del árbol a la mayor distancia entre el nodo raíz y un nodo hoja. En general, existe una preferencia por los árboles de decisión menos profundos, esto debido a razones como su interpretabilidad y la facilidad de análisis. Una menor profundidad implica una menor cantidad de toma de decisiones para clasificar una muestra correctamente. Para lograr esto, es fundamental seleccionar adecuadamente los atributos que se emplearán para realizar las particiones. ([18])

A modo de ejemplo, veamos el conjunto de datos de la [Tabla 4](#), en la cual contamos con 8 muestras caracterizadas por tres atributos: tamaño del borde (crust size), forma (shape)



**Figura 10.** Estructura básica de un árbol de decisión.<sup>3</sup>

y tamaño del relleno (filling size). Además, cada muestra está clasificada como positiva o negativa.

	crust size	shape	filling size	Class
e1	big	circle	small	+
e2	small	circle	small	+
e3	big	square	small	-
e4	big	triangle	small	-
e5	big	square	big	+
e6	small	square	small	-
e7	small	square	big	+
e8	big	circle	big	+

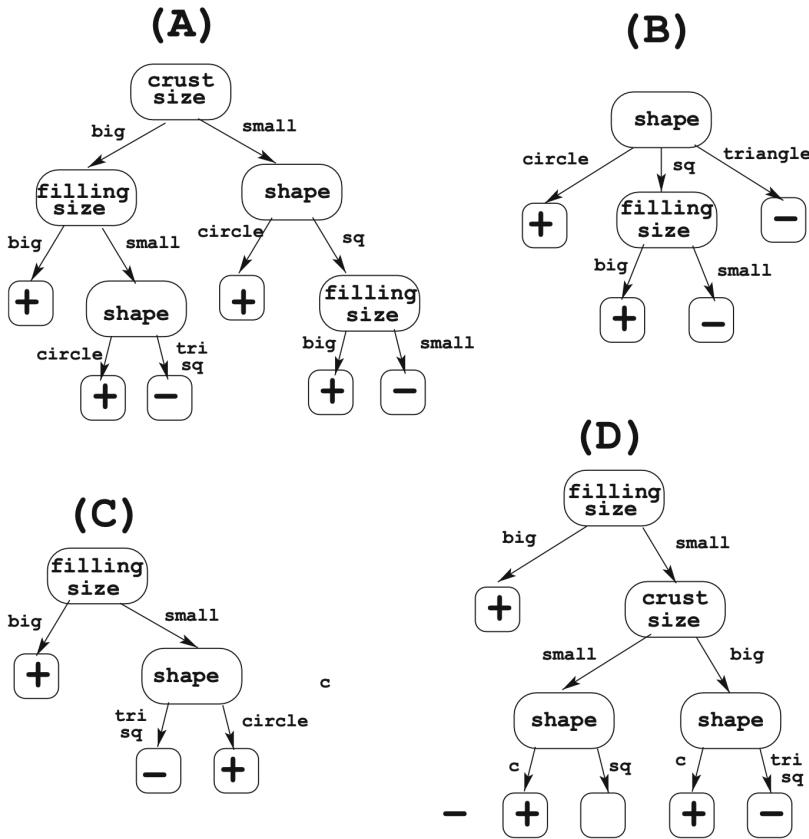
**Tabla 4.** Ejemplo de 8 muestras clasificadas como positivas o negativas según atributos de forma y tamaño. Fuente: tomada de [18]

A partir de estas muestras, utilizadas como conjunto de entrenamiento, se construye el conjunto de árboles de la [Figura 11](#). En el gráfico A, el primer atributo seleccionado para particionar el conjunto de datos es el tamaño del borde (crust size), mientras que en el gráfico B se utiliza la forma (shape). Por otro lado, en los gráficos C y D, el criterio inicial de división es el tamaño del relleno (filling size).

Cada árbol representa el conjunto de reglas a seguir para clasificar una muestra como positiva o negativa. En el gráfico B, por ejemplo, se observa que el atributo “shape” captura

---

<sup>3</sup>Imagen tomada de: [ibm.com](http://ibm.com)



**Figura 11.** 4 diferentes árboles de decisión creados a partir de la Tabla 4. Fuente: [18]

muy bien la relación que tiene la forma del objeto con la clase a la que pertenece. Así, podemos inferir fácilmente que si el objeto es un círculo, será clasificado como positivo, mientras que si es un triángulo, será negativo, sin necesidad de considerar los demás atributos. En el caso de los objetos con forma cuadrada, debemos verificar también su tamaño de relleno. No obstante, el árbol logra clasificar correctamente los datos de entrenamiento sin siquiera tener en cuenta el atributo “crust size”.

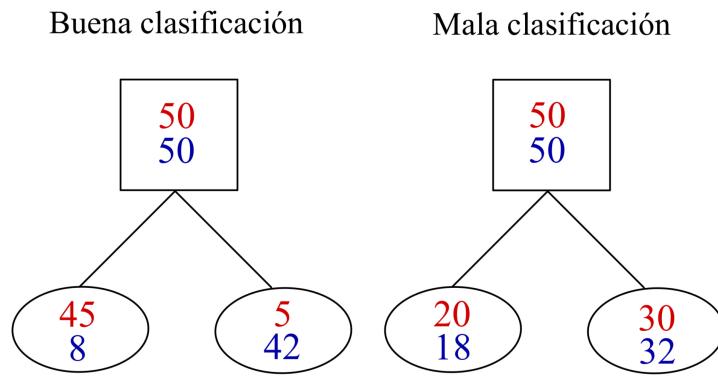
Lo interesante de estos algoritmos es que, además de predecir la clase de un nuevo dato, permiten comprender la razón detrás de su clasificación, a diferencia de las redes neuronales, que suelen funcionar como una caja negra. Por lo tanto, más que utilizarse únicamente como métodos predictivos, los árboles de decisión también pueden emplearse como herramientas descriptivas, ya que revelan relaciones entre los datos que resultarían difíciles de identificar con una simple observación de la tabla.

Sin embargo, hemos visto que a partir del mismo conjunto de datos pueden construirse árboles de decisión muy distintos. Esta variabilidad se debe principalmente a la selección de atributos utilizados para realizar las particiones, lo que da lugar a la pregunta: ¿qué atributos debemos elegir para lograr la mejor división de los datos? Para responder a esto, se definen criterios específicos de partición.

### 1.2.3. Criterios de partición

Lo más importante al momento de construir un árbol de decisión es elegir el nodo raíz, es decir, el atributo que realizará la primera partición del conjunto de datos. En un caso ideal, podríamos contar con un solo atributo que nos permitiera clasificar correctamente todas las muestras. Sin embargo, en la práctica, esto rara vez ocurre.

Por lo tanto, desde la primera división, el objetivo es que los subconjuntos resultantes sean lo más homogéneos posible, o más puros (como se suele denominar en la literatura). Es decir, se busca que en un nodo haya la mayor cantidad posible de muestras de una misma clase, y en el otro nodo, la mayor cantidad de muestras de la otra clase. Esto se ejemplifica en el gráfico de la izquierda de la [Figura 12](#), donde se genera una partición más efectiva en comparación con el árbol de la derecha, que deja una mezcla más equitativa de clases en cada rama.



**Figura 12.** Ejemplo de partición de 100 muestras: 50 de la clase roja y 50 de la clase azul. La primera división genera subconjuntos más homogéneos, mientras que la segunda deja una mayor mezcla de clases.

Existen diversas formas de medir esta homogeneidad o pureza en un nodo, siendo las métricas más populares la entropía basada en la teoría de la información y la impureza de Gini, las cuales definimos a continuación.

### Índice de Gini

Esta subsección es un compendio de ideas tomadas de [1, 11, 15, 23, 24].

Desarrollado en 1912 por el estadístico Corrado Gini, inicialmente con el propósito de medir la desigualdad en la distribución de riqueza en una población. Sin embargo, en el contexto de los árboles de decisión, es utilizado para cuantificar qué tan mezcladas están las clases en un nodo, y definir de este modo el grado de impureza de este.

Suponga que queremos medir la probabilidad de que dos muestras tomadas aleatoriamente (con reemplazo) de un nodo pertenezcan a clases distintas. La probabilidad de que la primera muestra pertenezca a la clase  $i$  es  $p_i$ , y la probabilidad de que la siguiente muestra no pertenezca a dicha clase es  $1 - p_i$ . Por lo tanto, como son eventos independientes, la probabilidad conjunta es igual a

$$p_i(1 - p_i).$$

Considerando el conjunto de las  $K$  posibles clases, se tiene que

$$\begin{aligned} \sum_{i=1}^K p_i(1 - p_i) &= \sum_{i=1}^K p_i - p_i^2 \\ &= \sum_{i=1}^K p_i - \sum_{i=1}^K p_i^2. \end{aligned}$$

lo que da lugar al índice de Gini, definido usualmente como

$$G = 1 - \sum_{i=1}^K p_i^2.$$

De este modo, si la probabilidad de que dos muestras pertenezcan a clases distintas es nula, es porque todas las muestras pertenecen a la misma clase  $i$ , lo que implica que  $G = 0$  y el conjunto sería un nodo puro.

Por otro lado, también se suele interpretar el índice de Gini como la varianza total entre las clases. Esto puesto que si tomamos una muestra al azar en un nodo, podemos modelar si pertenece o no a la clase  $i$  definiendo una variable  $X_i$  de la siguiente forma

$$X_i = \begin{cases} 1, & \text{si la muestra pertenece a la clase } i, \\ 0, & \text{si no pertenece a la clase } i. \end{cases}$$

Esto significa que  $X_i$  distribuye  $Bernoulli(p_i)$ , donde  $p_i$  es la probabilidad de pertenecer a la clase  $i$ , por ende, su varianza es

$$Var(X_i) = p_i(1 - p_i).$$

Considerando las  $K$  clases, la suma de las varianzas de cada  $X_i$  da como resultado nuevamente el índice de Gini

$$\begin{aligned} G &= \sum_{i=1}^K Var(X_i) \\ &= \sum_{i=1}^K p_i(1 - p_i) = 1 - \sum_{i=1}^K p_i^2. \end{aligned}$$

Ahora, sabemos que  $G$  alcanza su valor mínimo ( $G = 0$ ) cuando un nodo contiene muestras de una sola clase, esto es,  $p_i = 1$  para algún  $i = 1, 2, \dots, K$  y  $p_j = 0$  para todo  $j \neq i$ . Sin embargo, para encontrar los valores  $p_1, p_2, \dots, p_K$  que maximizan  $G$  bajo la condición  $\sum_{i=1}^K p_i = 1$ , podemos usar los multiplicadores de Lagrange. Esto da como resultado la siguiente función

$$\mathcal{L}(p_1, p_2, \dots, p_K, \lambda) = 1 - \sum_{i=1}^K p_i^2 + \lambda \left( \sum_{i=1}^K p_i - 1 \right).$$

Derivando con respecto a  $p_i$  e igualando a cero tenemos que

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial p_i} &= -2p_i + \lambda = 0, \\ p_i &= \frac{\lambda}{2}, \quad \forall i. \end{aligned}$$

Luego, usando la restricción tenemos que

$$\begin{aligned}\sum_{i=1}^K p_i &= 1, \\ \sum_{i=1}^K \frac{\lambda}{2} &= 1, \\ K \frac{\lambda}{2} &= 1, \\ \lambda &= \frac{2}{K}.\end{aligned}$$

Por lo tanto, reemplazando el valor de  $\lambda$  llegamos a que

$$p_i = \frac{\lambda}{2} = \frac{2}{K} \cdot \frac{1}{2} = \frac{1}{K}, \quad \forall i.$$

Esto significa que el índice de Gini  $G$  alcanza su valor máximo cuando las muestras están distribuidas de manera uniforme entre las clases, es decir, cuando la probabilidad de que una muestra sea de la clase  $i$  ( $i = 1, 2, \dots, K$ ) es la misma ( $p_i = 1/K$ ) para cada  $i$ . Esto corresponde a un estado de máximo incertidumbre, ya que no hay una clase predominante. En este caso, el valor máximo es dado por

$$\begin{aligned}G &= 1 - \sum_{i=1}^K p_i^2, \\ &= 1 - \sum_{i=1}^K \left(\frac{1}{K}\right)^2 \\ &= 1 - K \left(\frac{1}{K}\right)^2 \\ &= 1 - \frac{1}{K}.\end{aligned}$$

De este modo, el valor máximo que puede tomar  $G$  depende del número de clases, alcanzando valores de 0.5, 0.667 y 0.75 cuando se tienen dos, tres y cuatro clases respectivamente. En general, el valor máximo de impureza se acerca a  $G = 1$  a medida que el número de clases aumenta.

## Entropía

“What is it? I claim it’s the amount of information we don’t know about a situation.”

*John C. Baez, What is Entropy? [3]*

Se entiende comúnmente como entropía a la medida que cuantifica el desorden molecular en un sistema. Esto dentro del campo de la termodinámica. Sin embargo, en la teoría de la información introducida por Shanon, la entropía cuantifica la cantidad promedio de información contenida en una fuente de datos, o de manera equivalente, la incertidumbre esperada en la clasificación de una muestra. En este contexto, la formula de entropía se define como [9, 5]

$$E = - \sum_{i=1}^K p_i \log_2 p_i,$$

donde  $K$  es el numero de clases diferentes en el conjunto de datos y  $p_i$  es la frecuencia relativa de muestras que pertenecen a la clase  $i$ , con  $p_i \in (0, 1]$ . Como  $\log_2 p_i \leq 0$  para todo  $p_i \in (0, 1]$ , la entropía es siempre un valor no negativo, es decir,  $E \geq 0$ .

Note que la entropía es mínima cuando todas las muestras pertenecen a la misma clase. Esto significa que solo existe una clase no vacía  $j$  en la cual  $p_j = 1$ . Lo que da como resultado que  $E = -p_j \log_2 p_j = -\log_2 1 = 0$ . Por otro lado, análogamente a como se mostró para el índice Gini,  $E$  alcanza su valor máximo cuando la proporción de clases es la misma ( $p_i = 1/K$  para todo  $i$ ), y por lo tanto,

$$\begin{aligned} E &= - \sum_{i=1}^K \frac{1}{K} \log_2 \frac{1}{K} \\ &= -K \frac{1}{K} \log_2 \frac{1}{K} \\ &= -\log_2 \frac{1}{K} \\ &= \log_2 K. \end{aligned}$$

De este modo, el valor máximo que puede tomar la entropía depende (al igual que el índice de Gini) del numero de clases en el conjunto de datos.

En cuanto a interpretación, la entropía se mide en bits de información ya que representa la cantidad promedio de bits necesarios para codificar de manera óptima un conjunto de valores. De manera equivalente, corresponde al número promedio de preguntas binarias necesarias para identificar la clase de una muestra tomada al azar.

A modo de ejemplo, suponga que tenemos un conjunto de datos con dos clases (A y B) igualmente probables, es decir,  $p_A = 0.5 = p_B$ . La entropía en este caso es

$$\begin{aligned} E &= -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) \\ &= 1 \text{ bit.} \end{aligned}$$

Esto significa que en promedio necesitamos una pregunta binaria para determinar la clase de una muestra en este conjunto.

■ **¿Es de la clase A?**

Sí	→	es A
No	→	es B

Si el conjunto tuviera cuatro clases en lugar de dos, digamos A, B, C y D, con probabilidades de ocurrencia  $p_A = p_B = p_C = p_D = 0.25$ , entonces su entropía sería

$$\begin{aligned} E &= -4 (0.25 \log_2 0.25) \\ &= 2 \text{ bits.} \end{aligned}$$

Esto sugiere que se necesitan en promedio dos preguntas binarias para identificar la clase

■ **¿Es de la clase A o B?**

Sí → ¿Es A?

Sí	→	es A
No	→	es B

No → ¿Es C?

Sí	→	es C
No	→	es D

En ambos ejemplos, los conjuntos de datos son lo menos homogéneos posible, lo que implica una entropía máxima  $E = \log_2 K$ . Sin embargo, esto no significa que podamos

clasificar una muestra desconocida con exactamente  $E$  preguntas binarias, ya que no conocemos su clase y, por lo tanto, no podemos formular preguntas directas sobre esta. En su lugar, debemos hacer preguntas en base a los atributos que colectivamente determinan la clase de una muestra.

En la práctica, la medida de impureza utilizada no suele representar una gran diferencia, conduciendo a árboles de decisión muy similares. Sin embargo, la impureza de Gini es ligeramente más rápida de calcular en comparación con la entropía debido a la presencia del logaritmo en esta última [9] [21].

Por lo tanto, teniendo presente estas dos formas de medir la impureza de un nodo, la función objetivo que buscaremos optimizar es la ganancia de información, definida de la siguiente manera [21]

$$IG(S_p, f) = I(S_p) - \sum_{j=1}^m \frac{N_j}{N_p} I(S_j),$$

donde:

- $f$  representa la característica utilizada para realizar la partición.
- $S_p$  es el conjunto de datos en el nodo padre.
- $S_j$  es el conjunto de datos en el nodo hijo  $j$ .
- $I$  es la medida de impureza.
- $N_p$  es el número total de muestras en el nodo padre.
- $N_j$  es el número de muestras en el nodo hijo  $j$ .

En la construcción de un árbol de decisión, se emplea lo que se conoce como una greedy search o búsqueda exhaustiva [22]. Es decir, en cada paso, se selecciona la partición que maximice la ganancia de información en ese instante, optando por una solución localmente óptima. Este proceso consiste en evaluar todas las posibles particiones y elegir la mejor en cada iteración, sin considerar el impacto global de esta en la estructura final del árbol.

Note que la impureza del nodo padre,  $I(S_p)$ , mide qué tan mezclado está el conjunto de datos antes de realizar alguna partición. Dado que este valor es constante en cada división, maximizar la ganancia de información  $IG$  equivale a minimizar la suma ponderada de las impurezas en los nodos hijos. A continuación un ejemplo que ilustra mejor este concepto.

#### 1.2.4. Ejemplo

Observemos el conjunto de datos en la [Tabla 5](#), el cual contiene las características que determinan si el clima es apto para jugar un partido de tenis. Este ejemplo, tomado de [\[22\]](#), contiene las variables: Condición del tiempo, Temperatura, Humedad, Viento y la decisión de si se debe jugar el partido de tenis o no.

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

**Tabla 5.** Datos de entrenamiento para determinar si se debe jugar un partido de tenis.

Dado que el conjunto de datos contiene solo dos clases, cinco muestras clasificadas como “No” y nueve clasificadas como “Yes”, calculamos la entropía del conjunto de datos en el nodo padre  $S_p = \{D_1, D_2, \dots, D_{14}\}$  como

$$E(S_p) = - \left( \frac{5}{14} \log_2 \left( \frac{5}{14} \right) + \frac{9}{14} \log_2 \left( \frac{9}{14} \right) \right) \\ \approx 0,940 \text{ bits.}$$

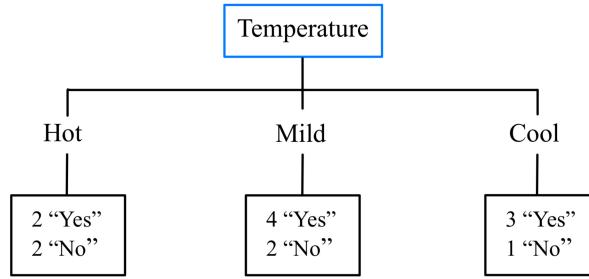
Luego, si seleccionamos la variable Temperature como nodo raíz para particionar los datos, tendremos el árbol de decisión parcial observado en la [Figura 13](#).

Calculando la entropía de cada nodo hijo tenemos

$$E_{\text{hot}} = - \left( \frac{2}{4} \log_2 \left( \frac{2}{4} \right) + \frac{2}{4} \log_2 \left( \frac{2}{4} \right) \right) = 1 \text{ bit.}$$

$$E_{\text{mild}} = - \left( \frac{4}{6} \log_2 \left( \frac{4}{6} \right) + \frac{2}{6} \log_2 \left( \frac{2}{6} \right) \right) \approx 0,918 \text{ bits.}$$

$$E_{\text{cool}} = - \left( \frac{3}{4} \log_2 \left( \frac{3}{4} \right) + \frac{1}{4} \log_2 \left( \frac{1}{4} \right) \right) \approx 0,811 \text{ bits.}$$

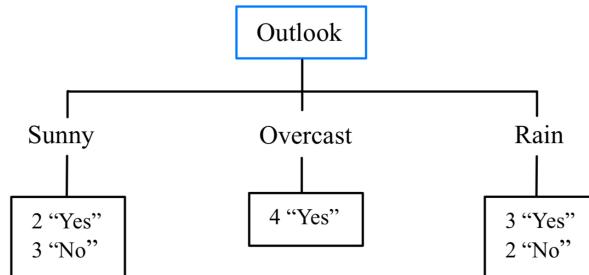


**Figura 13.** Árbol de decisión creado a partir de la variable Temperature.

De este modo, la ganancia de información con esta partición es

$$\begin{aligned} IG(S_p, \text{Temperature}) &\approx 0,940 - \left( \frac{4}{14} \cdot 1 + \frac{6}{14} \cdot 0,918 + \frac{4}{14} \cdot 0,811 \right) \\ &\approx 0,029 \text{ bits.} \end{aligned}$$

Por otro lado, si se realiza la división en base al atributo Outlook se obtiene el árbol de decisión de la [Figura 14](#) y los respectivos valores de entropía de los nodos hijos



**Figura 14.** Árbol de decisión creado a partir de la variable Outlook.

$$E_{\text{sunny}} = - \left( \frac{2}{5} \log_2 \left( \frac{2}{5} \right) + \frac{3}{5} \log_2 \left( \frac{3}{5} \right) \right) \approx 0,970 \text{ bits.}$$

$$E_{\text{overcast}} = - \frac{4}{4} \log_2 \left( \frac{4}{4} \right) = 0 \text{ bits.}$$

$$E_{\text{rain}} = - \left( \frac{3}{5} \log_2 \left( \frac{3}{5} \right) + \frac{2}{5} \log_2 \left( \frac{2}{5} \right) \right) \approx 0,970 \text{ bits.}$$

Así, la ganancia de información del atributo Outlook es

$$\begin{aligned} IG(S_p, \text{Outlook}) &\approx 0,940 - \left( \frac{5}{14} \cdot 0,970 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0,970 \right) \\ &\approx 0,247 \text{ bits.} \end{aligned}$$

Realizando el mismo procedimiento para los demás atributos obtenemos que

$$\begin{aligned} IG(S_p, \text{Temperature}) &\approx 0,029 \text{ bits,} \\ IG(S_p, \text{Wind}) &\approx 0,048 \text{ bits,} \\ IG(S_p, \text{Humidity}) &\approx 0,151 \text{ bits,} \\ IG(S_p, \text{Outlook}) &\approx 0,247 \text{ bits.} \end{aligned}$$

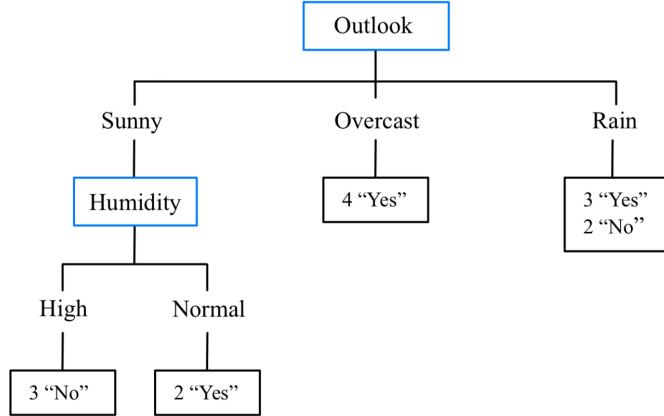
De este modo, el atributo que presenta una mayor ganancia de información es Outlook, pues es el que realiza una mejor partición de los datos, a diferencia de Temperature que nos deja una mayor mezcla de clases.

Note que si una muestra tiene “Overcast” como valor del atributo Outlook, esta inmediatamente es clasificada como “Yes”, sin siquiera tener en cuenta los demás atributos. Sin embargo, si el valor de este atributo es “Sunny” o “Rain”, tendremos que realizar más divisiones para identificar correctamente la clase.

Tomando entonces como nodo padre al conjunto de datos  $S_{\text{sunny}} = \{D_1, D_2, D_8, D_9, D_{11}\}$ , y particionandolo respecto a los demás atributos, la ganancia de información obtenida por estos es

$$\begin{aligned} IG(S_{\text{sunny}}, \text{Temperature}) &= 0,970 - \left( \frac{2}{5} \cdot 0 + \frac{2}{5} \cdot 1 + \frac{1}{5} \cdot 0 \right) = 0,570 \text{ bits,} \\ IG(S_{\text{sunny}}, \text{Wind}) &= 0,970 - \left( \frac{3}{5} \cdot 0,918 + \frac{2}{5} \cdot 1 \right) = 0,019 \text{ bits,} \\ IG(S_{\text{sunny}}, \text{Humidity}) &= 0,970 - \left( \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0 \right) = 0,970 \text{ bits.} \end{aligned}$$

Esto muestra que el atributo que mejor separa los datos dentro del conjunto  $S_{\text{sunny}}$  es Humidity y, por lo tanto, el árbol de decisión tomará forma de la [Figura 15](#).



**Figura 15.** Árbol de decisión creado a partir de los atributos Outlook y Humidity.

Dado que ambos subconjuntos resultantes separan completamente las clases, ya no se realizarán más particiones en esta rama, y dichos nodos serán etiquetados automáticamente con la única clase que contienen.

De la misma manera, al particionar el subconjunto de datos con  $\text{Outlook} = \text{Rain}$ , es decir,  $S_{\text{rain}} = \{\text{D}_4, \text{D}_5, \text{D}_6, \text{D}_{10}, \text{D}_{14}\}$ , se obtienen las siguientes ganancias de información para cada atributo

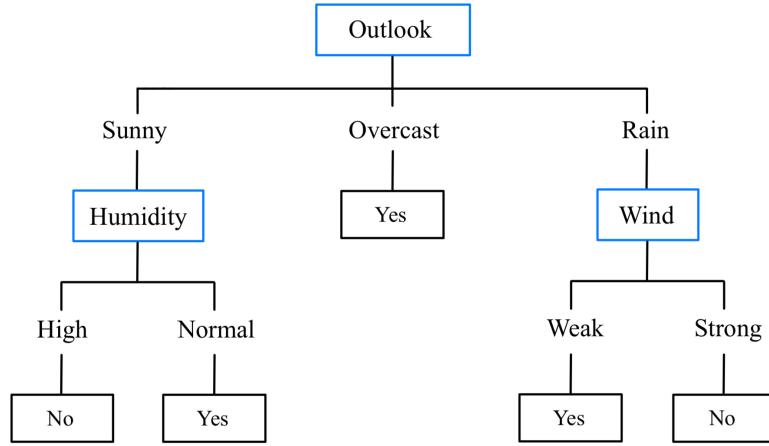
$$IG(S_{\text{rain}}, \text{Temperature}) = 0,019 \text{ bits},$$

$$IG(S_{\text{rain}}, \text{Humidity}) = 0,419 \text{ bits},$$

$$IG(S_{\text{rain}}, \text{Wind}) = 0,970 \text{ bits}.$$

Por lo tanto, en este caso, el atributo Wind es el que mejor separa las clases, dando como resultado el árbol de decisión mostrado en la [Figura 16](#). Además, es importante notar que el atributo Temperature no tuvo relevancia en la clasificación de las muestras del conjunto de entrenamiento. Sin embargo, esto no implica que sea así siempre, ya que si se agregaran más muestras al conjunto de datos, la estructura del árbol de decisión podría cambiar, y Temperature podría volverse un criterio relevante en alguna partición.

En resumen, los árboles de decisión son un método de aprendizaje supervisado ampliamente utilizado debido a que son intuitivos, simples de construir y generalmente fáciles de interpretar. A continuación veremos cómo, al combinarse de forma aditiva, estos árboles actúan como aprendices base del popular método de ensamble XGBoost.



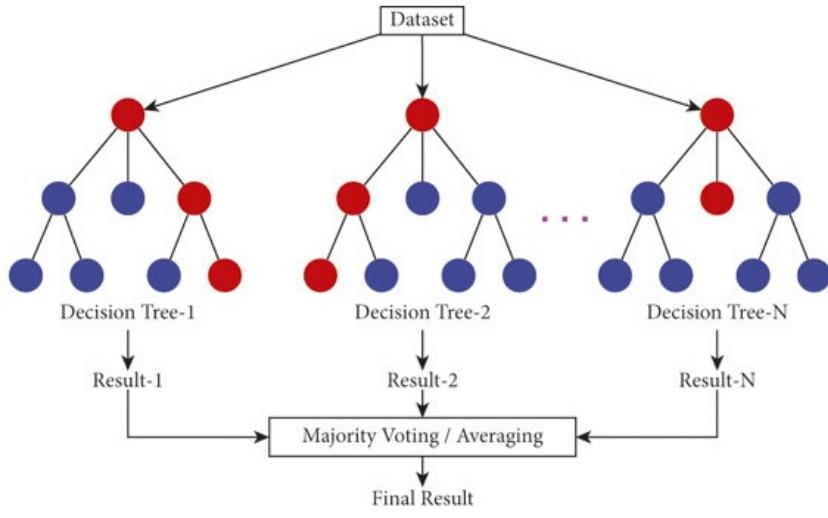
**Figura 16.** Árbol de decisión construido a partir del conjunto de datos de la [Tabla 5](#), particionado de acuerdo a los atributos que maximizan la ganancia de información.

### 1.3. eXtreme Gradient Boosting (XGBoost)

La teoría matemática expuesta en esta sección es tomada principalmente de [\[7\]](#).

En la sección anterior examinamos los árboles de decisión, un método clásico de aprendizaje supervisado ampliamente utilizado tanto en tareas de clasificación como de regresión. Sin embargo, en el campo del machine learning nos encontramos con un conjunto de métodos conocidos como métodos de ensamble, los cuales consisten en combinar las predicciones de múltiples modelos, con el propósito de mejorar la precisión y el rendimiento general, en lugar de confiar en un único predictor. La efectividad de estos métodos radica en aprovechar las fortalezas de cada modelo por individual al mismo tiempo que se mitigan sus debilidades o errores, dando lugar a un modelo más preciso y robusto [\[11\]](#).

Dentro de los métodos de ensamble más populares se encuentra el Random Forest, o Bosque Aleatorio, el cual está compuesto específicamente por un conjunto de árboles de decisión, donde todos los árboles son independientes y cada uno de ellos es construido a partir de un subconjunto aleatorio de características y de un muestreo con reemplazo de las observaciones en el conjunto de entrenamiento [\[11\]](#). De este modo, cada árbol contribuye con su propia predicción y después se toma una predicción final mediante un proceso de voto mayoritario o promedio (véase [Figura 17](#)), lo que reduce el riesgo de sobreajuste.



**Figura 17.** Estructura básica de un Random Forest.<sup>4</sup>

Por otro lado, el XGBoost es un método que también resulta de un ensamble de árboles de decisión, con la diferencia de que estos no son independientes uno del otro como en el Random Forest, sino que son construidos secuencialmente de modo que cada nuevo árbol intenta corregir los errores de su predecesor, aproximándose de forma iterativa a los valores reales del conjunto de muestras [9].

El XGBoost surge como una implementación de gradient boosting, el algoritmo propuesto por Friedman [8] que construye un modelo aditivo de árboles de decisión y que recibe su nombre debido a que dicho algoritmo optimiza una función de pérdida mediante un descenso de gradiente, ajustando los árboles de decisión a los gradientes negativos de la pérdida de forma que la suma de las predicciones converja a la solución real.

Para comenzar, conviene tener claro que el XGBoost es un método de aprendizaje supervisado, esto quiere decir que el modelo se entrena sobre un conjunto de datos etiquetados, cuyos valores verdaderos deseamos predecir mediante una función paramétrica. Así, dado un conjunto de datos con  $n$  individuos y  $p$  variables, para cada  $y_i \in \mathbb{R}$  y  $x_i \in \mathbb{R}^p$ , la predicción viene dada por [7]

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F},$$

---

<sup>4</sup>Imagen tomada de: [researchgate.net/figure/Illustration-of-random-forest](https://researchgate.net/figure/Illustration-of-random-forest)

aquí  $\mathcal{F}$  denota el conjunto de árboles de decisión, cada uno es expresado como

$$f_k(x_i) = w_{q(x_i)}, \quad w \in \mathbb{R}^T, q : \mathbb{R}^p \longrightarrow \{1, 2, \dots, T\}.$$

En este caso,  $T$  es el número de nodos hoja en el árbol. La función  $q$  nos indica a qué hoja del árbol pertenece cada muestra, y  $w = (w_1, \dots, w_T)$  es el vector de pesos que contiene los valores que retorna el árbol en cada hoja. De este modo, dado un árbol  $f_k$ , cualquier muestra pertenecerá a alguna de sus  $T$  hojas, por lo que podemos ver a  $f_k$  como una función por partes de la siguiente forma

$$f_k(x_i) = \begin{cases} w_1, & \text{si } x_i \in \text{hoja 1}, \\ w_2, & \text{si } x_i \in \text{hoja 2}, \\ \vdots \\ w_T, & \text{si } x_i \in \text{hoja } T. \end{cases}$$

Ahora bien, en este contexto entendemos el entrenamiento de un modelo como la búsqueda de los parámetros que mejor se ajustan a los datos de entrenamiento. Del mismo modo que buscamos los  $\beta_i$  en una regresión lineal  $y = \sum \beta_i X_i$ , debemos apoyarnos en una función objetivo, la cual buscaremos optimizar. En XGBoost, dicha función objetivo está compuesta de dos partes, una función de pérdida y un término de regularización.

$$\mathcal{L}(\phi) = \underbrace{\sum_{i=1}^n l(y_i, \hat{y}_i)}_{\text{función de pérdida}} + \underbrace{\sum_{k=1}^K \Omega(f_k)}_{\text{regularización}},$$

con

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2.$$

Por un lado, la función de pérdida nos mide qué tan buenas son las predicciones del modelo, y puede tomar distintas formas según el tipo de problema que estemos resolviendo. Por ejemplo, para una tarea de regresión,  $l$  puede ser de la siguiente forma

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2.$$

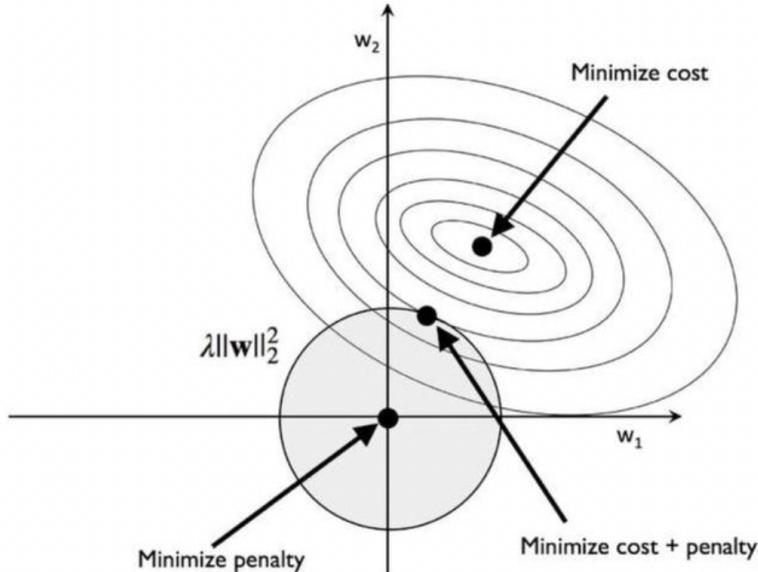
Mientras que para una clasificación binaria donde  $\hat{y}_i$  es la probabilidad de que una muestra pertenezca a alguna clase (veremos esto en detalle más adelante),  $l$  puede ser la función de perdida log-loss (o entropía cruzada)

$$l(y_i, \hat{y}_i) = -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)].$$

Sin embargo,  $l$  no está restringida a estas únicas dos funciones y puede tomar muchas otras formas siempre y cuando sea convexa y diferenciable.

Por otro lado, el término de regularización  $\Omega(f_k)$  penaliza la complejidad de cada árbol mediante un término de regularización L2 [21] más un término adicional  $\gamma T$  que penaliza el número de hojas, evitando que el árbol crezca demasiado mientras controla los pesos de cada hoja.

Note por un momento que, omitiendo el término  $\gamma T$ , la regularización L2 no es más que la búsqueda de los pesos  $w_j$  que minimizan la función de pérdida, sujeta a la condición de que dichos pesos pertenezcan a una bola centrada en el origen (véase [Figura 18](#)).



**Figura 18.** Regularización L2. Fuente: tomada de [21].

$$\operatorname{argmin}_w \sum_{i=1}^N l(y_i, \hat{y}_i) \quad \text{sujeta a} \quad \sum_{j=1}^T w_j^2 \leq C.$$

Por lo tanto, este es un problema que puede ser abordado haciendo uso de los multiplicadores de Lagrange. Es por esta razón que la función objetivo  $\mathcal{L}(\phi)$  que vimos hace un momento es en esencia el Lagrangiano de dicho problema [11].

Ahora bien, antes de buscar los pesos  $w_j$  que optimizan las hojas de cada árbol, primero enfoquémonos en el conjunto de funciones  $\{f_k\}$ . Como ya vimos, cada  $f_k$  representa un árbol de decisión, y por lo tanto, nuestro objetivo inmediato es encontrar el conjunto de árboles que minimicen la función objetivo

$$\begin{aligned}\mathcal{L}(\phi) &= \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k), \\ &= \sum_{i=1}^n l\left(y_i, \sum_{k=1}^K f_k(x_i)\right) + \sum_{k=1}^K \Omega(f_k).\end{aligned}$$

Como cada  $\hat{y}_i$  es la suma de un conjunto de  $K$  árboles, es imposible encontrar de forma analítica y en un solo paso el conjunto completo de árboles que minimizan el error. Por ello, la estrategia aquí es tomar

$$\begin{aligned}\hat{y}_i^{(t-1)} &= \sum_{k=1}^{t-1} f_k(x_i), \\ \hat{y}_i^{(t)} &= \hat{y}_i^{(t-1)} + f_t(x_i),\end{aligned}$$

y buscar el árbol  $f_t$  que corrija la predicción de los árboles anteriores minimizando así el error. De este modo, se emplea una construcción de forma aditiva, entrenando secuencialmente un árbol a la vez, y agregando aquel que optimiza nuestra función objetivo.

Por lo tanto, nuestra función objetivo a optimizar en el paso  $t$  es dada por

$$\begin{aligned}\mathcal{L}^{(t)} &= \sum_{i=1}^n l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^t \Omega(f_k), \\ &= \sum_{i=1}^n l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \sum_{k=1}^t \Omega(f_k),\end{aligned}$$

pero como el término  $\sum_{k=1}^{t-1} \Omega(f_k)$  no depende de  $f_t$ , podemos dejarlo simplemente como

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t).$$

Sin embargo, como veremos más adelante en el caso de clasificación, al momento de trabajar con una función de perdida como la log-loss puede ser muy difícil optimizar esta función objetivo y resolver para  $f_t$ . Por esto, se aproxima la función de perdida con un polinomio de Taylor de segundo orden.

La aproximación de Taylor de segundo orden de  $l(y_i, \hat{y}_i^{(t)})$  alrededor de la predicción actual  $\hat{y}_i^{(t-1)}$  es expresada como

$$l(y_i, \hat{y}_i^{(t)}) \approx l(y_i, \hat{y}_i^{(t-1)}) + \frac{\partial l(y_i, \hat{y}_i^{(t)})}{\partial \hat{y}_i^{(t)}} \Bigg|_{\hat{y}_i^{(t-1)}} (\hat{y}_i^{(t)} - \hat{y}_i^{(t-1)}) + \frac{1}{2} \frac{\partial^2 l(y_i, \hat{y}_i^{(t)})}{\partial (\hat{y}_i^{(t)})^2} \Bigg|_{\hat{y}_i^{(t-1)}} (\hat{y}_i^{(t)} - \hat{y}_i^{(t-1)})^2.$$

Por lo tanto, tomando

$$\frac{\partial l(y_i, \hat{y}_i^{(t)})}{\partial \hat{y}_i^{(t)}} \Bigg|_{\hat{y}_i^{(t-1)}} = g_i,$$

$$\frac{\partial^2 l(y_i, \hat{y}_i^{(t)})}{\partial (\hat{y}_i^{(t)})^2} \Bigg|_{\hat{y}_i^{(t-1)}} = h_i,$$

$$\hat{y}_i^{(t)} - \hat{y}_i^{(t-1)} = f_t(x_i),$$

la aproximación de Taylor de  $l(y_i, \hat{y}_i^{(t)})$  queda dada por

$$l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) \approx l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i).$$

Luego, reemplazamos esta en la función objetivo

$$\begin{aligned} \mathcal{L}^{(t)} &= \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t), \\ &\approx \sum_{i=1}^n [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t). \end{aligned}$$

Finalmente, eliminando los términos constantes independientes de  $f_t$ , nuestra función objetivo específica en la iteración  $t$  queda como

$$\begin{aligned}\tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t), \\ &= \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2.\end{aligned}$$

Note que  $\lambda$  es el multiplicador de Lagrange que controla el tamaño de los pesos de los nodos hoja del árbol  $f_t$ , acotando estos dentro de una bola centrada en el origen como se observó en la [Figura 18](#). Así, entre menor sea  $\lambda$ , mayor es el radio de la bola que contiene los pesos  $w_j$  y, por lo tanto, mayor libertad tienen estos de converger al mínimo de la función de pérdida. Por otro lado,  $\gamma$  juega un papel similar a  $\lambda$  penalizando el numero  $T$  de hojas.

Ahora, recordemos que  $f_t(x_i) = w_{q(x_i)}$  es una función por partes que toma diferentes valores dependiendo de la hoja del árbol en la que caiga la muestra  $x_i$ . Por lo tanto, reemplazando este valor en la función objetivo  $\tilde{\mathcal{L}}^{(t)}$  tenemos

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^n [g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2.$$

No obstante,  $w_{q(x_i)}$  es el mismo valor para todas las muestras que terminen en una misma hoja. Así que, en vez de sumar sobre  $i$  para cada muestra del conjunto de datos, sumemos sobre  $j$  para cada hoja del árbol  $f_t$ . Lo cual es equivalente pues cada muestra  $x_i$  pertenece exactamente a una hoja de este árbol. Dicho esto obtenemos

$$\begin{aligned}\tilde{\mathcal{L}}^{(t)} &= \sum_{j=1}^T \left[ \left( \sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left( \sum_{i \in I_j} h_i \right) w_j^2 \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2, \\ &= \sum_{j=1}^T \left[ \left( \sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left( \sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T,\end{aligned}$$

donde  $I_j = \{i : q(x_i) = j\}$  es el conjunto de muestras que pertenecen a la  $j$ -ésima hoja.

De esta manera, tomando

$$\sum_{i \in I_j} g_i = G_j,$$

$$\sum_{i \in I_j} h_i = H_j,$$

obtenemos lo siguiente

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^T [G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T.$$

Ahora bien, como los  $w_j$  son independientes entre ellos ya que  $w_j$  es el valor que toma la  $j$ -ésima hoja. Podemos buscar el  $w_j^*$  que minimiza la pérdida en cada hoja derivando e igualando a cero  $G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2$ . Es decir,

$$\frac{d}{dw_j} (G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2) = 0,$$

$$G_j + (H_j + \lambda)w_j = 0,$$

$$w_j^* = \frac{-G_j}{H_j + \lambda}.$$

Así, al momento de construir el árbol  $f_t$ , y al realizar una partición basada en alguna característica, ya conocemos el valor óptimo  $w_j^*$  que debe retornar cada hoja para minimizar la pérdida. Sin embargo, estos valores por sí solos no nos indican cuánto se reduce la pérdida con cada partición. Por lo tanto, para comparar distintas particiones reemplazamos  $w_j^*$  en la función objetivo  $\tilde{\mathcal{L}}^{(t)}$  de la siguiente manera

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &= \sum_{j=1}^T [G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T, \\ &= \sum_{j=1}^T \left[ G_j \left( \frac{-G_j}{H_j + \lambda} \right) + \frac{1}{2}(H_j + \lambda) \left( \frac{-G_j}{H_j + \lambda} \right)^2 \right] + \gamma T, \\ &= -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T. \end{aligned}$$

De esta forma cuantificamos la pérdida asociada a cada partición, cuanto más pequeño sea este valor, mejor será la partición, de forma análoga a como lo vimos en la sección de árboles de decisión cuando minimizabamos el índice de Gini o la entropía.

Luego, calculamos la ganancia de información (Gain) como la reducción neta de la función objetivo al dividir un nodo en dos. Es decir, calculamos la pérdida óptima del nodo padre  $I$  y le restamos la suma de las pérdidas óptimas de los nodos hijos  $I_L$  e  $I_R$ , donde  $I_L$  es el conjunto de muestras que caen en la hoja izquierda tras la partición binaria e  $I_R$  las que caen en la hoja derecha, de manera que  $I = I_L \cup I_R$  es el conjunto original de muestras antes de realizar la partición.

$$\begin{aligned} Gain &= -\frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} + \gamma T - \left[ -\frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} - \frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} + \gamma(T+1) \right] \\ &= \frac{1}{2} \left[ \sum_{j=1}^T \frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \sum_{j=1}^T \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \sum_{j=1}^T \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma, \end{aligned}$$

De este modo, cuanto menor sea la pérdida de los nodos hijos, mayor será la ganancia de información y por lo tanto, mejor será dicha partición.

### Objetivo clasificación.

Como acabamos de ver, la forma general de construir los árboles y encontrar los pesos en las hojas consiste principalmente en calcular los gradientes ( $g_i$ ) y los Hessianos ( $h_i$ ) de la función de pérdida, y en usarlos para ir ajustando gradualmente las predicciones. Este proceso es independiente de la tarea que estemos realizando. Por esta razón, para ilustrar su aplicación al objetivo central de este proyecto, nos centraremos en la clasificación binaria, donde la función de pérdida utilizada es la log-loss definida anteriormente para cada muestra como

$$l(y_i, \hat{y}_i) = -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)].$$

En este contexto,  $y_i \in \{0, 1\}$  (dos posibles clases), y  $\hat{y}_i$  es la probabilidad de pertenecer a la clase 1. Por lo tanto,  $l$  puede ser vista como

$$l(y_i, \hat{y}_i) = \begin{cases} -\log(\hat{y}_i), & \text{si } y_i = 1, \\ -\log(1 - \hat{y}_i), & \text{si } y_i = 0. \end{cases}$$

De este modo, si una muestra  $x_i$  pertenece a la clase 1, lo ideal es que la predicción  $\hat{y}_i$  se acerque a 1, y así  $-\log(\hat{y}_i)$  sea cercano a 0. De modo contrario, si la predicción es incorrecta y  $\hat{y}_i$  tiende a cero, el error crece indefinidamente.

Ahora, esta función de pérdida surge de asumir que el conjunto de muestras son independientes y que cada una sigue una distribución de Bernoulli, de modo que la función de probabilidad de cada muestra es dada por

$$f(y_i) = \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}.$$

Luego, como estas son independientes, la función de verosimilitud es simplemente

$$\begin{aligned} L &= \prod_{i=1}^n f(y_i), \\ &= \prod_{i=1}^n \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}. \end{aligned}$$

Así, maximizar esta verosimilitud equivale a estimar los  $\hat{y}_i$  que mejor se ajustan a los datos observados. Por lo tanto, en el caso más ideal el modelo predice correctamente los valores de  $y_i$  con una certeza total, y así el producto de probabilidades es  $L = 1$ . Sin embargo, para simplificar los cálculos se toma el logaritmo de  $L$

$$\begin{aligned} \log(L) &= \sum_{i=1}^n \log(\hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}), \\ &= \sum_{i=1}^n \log(\hat{y}_i^{y_i}) + \log((1 - \hat{y}_i)^{1-y_i}), \\ &= \sum_{i=1}^n y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i), \end{aligned}$$

y así, como el logaritmo es una función monótona creciente, maximizar la verosimilitud  $L$ , equivale a minimizar  $-\log(L)$ . De hecho, en el caso ideal, si  $L = 1$  entonces  $-\log(L) = 0$ .

Por esta razón se define la *Log loss* como medida de error para el conjunto completo de muestras como [4]

$$\text{Log loss} = -\log(L) = -\sum_{i=1}^n y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i).$$

Ahora bien, como ya vimos, la predicción  $\hat{y}_i$  no es directamente la clase asignada a la muestra  $x_i$ , sino la probabilidad de que pertenezca a la clase 1.

Por lo tanto, para no limitar al modelo a generar predicciones en el intervalo  $(0,1)$ , no se trabaja directamente con  $\hat{y}_i$ , sino con su logit

$$\text{logit}(\hat{y}_i) = \log\left(\frac{\hat{y}_i}{1 - \hat{y}_i}\right) = F(x_i),$$

de modo que ahora  $F(x_i) \in \mathbb{R}$ . Luego, para obtener nuevamente la probabilidad basta con aplicar la función inversa, es decir, la sigmoide

$$\sigma(F(x_i)) = \frac{1}{1 + e^{-F(x_i)}} = \hat{y}_i.$$

Así, reescribimos la log-loss de cada muestra

$$\begin{aligned} l(y_i, \hat{y}_i) &= -[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)], \\ &= -y_i \log(\hat{y}_i) - \log(1 - \hat{y}_i) + y_i \log(1 - \hat{y}_i), \\ &= -y_i [\log(\hat{y}_i) - \log(1 - \hat{y}_i)] - \log(1 - \hat{y}_i), \\ &= -y_i \log\left(\frac{\hat{y}_i}{1 - \hat{y}_i}\right) - \log\left(1 - \frac{1}{1 + e^{-F(x_i)}}\right), \\ &= -y_i F(x_i) - \log\left(\frac{1}{1 + e^{F(x_i)}}\right), \\ &= -y_i F(x_i) + \log(1 + e^{F(x_i)}), \end{aligned}$$

de modo que ahora queda expresada en términos de  $F(x_i)$ .

Luego, la predicción de los  $K$  árboles viene dada por

$$F(x_i) = \sum_{k=1}^K f_k(x_i),$$

y por lo tanto, la predicción en la iteración  $t$  es escrita como

$$F_t(x_i) = \sum_{k=1}^{t-1} f_k(x_i) + f_t(x_i) = F_{t-1}(x_i) + f_t(x_i).$$

Ahora, calculamos los  $g_i$  como

$$\begin{aligned} g_i &= \frac{\partial l(y_i, F_t(x_i))}{\partial F_t(x_i)} \Big|_{F_{t-1}(x_i)}, \\ &= -y_i + \frac{e^{F_{t-1}(x_i)}}{1 + e^{F_{t-1}(x_i)}}, \\ &= -y_i + \sigma(F_{t-1}(x_i)). \end{aligned}$$

Pero recordemos que  $\sigma(F(x_i)) = \hat{y}_i$ , por lo tanto

$$g_i = \hat{y}_i^{(t-1)} - y_i,$$

es precisamente la diferencia entre la probabilidad estimada en la última iteración y el valor real. De este modo,  $-g_i$  nos indica la dirección y la magnitud en la que debemos ajustar la predicción para minimizar la pérdida.

Por otro lado, calculamos los  $h_i$  como

$$\begin{aligned} h_i &= \frac{\partial^2 l(y_i, F_t(x_i))}{\partial (F_t(x_i))^2} \Big|_{F_{t-1}(x_i)}, \\ &= \frac{e^{-F_{t-1}(x_i)}}{(1 + e^{-F_{t-1}(x_i)})^2}, \\ &= \sigma(F_{t-1}(x_i))[1 - \sigma(F_{t-1}(x_i))], \end{aligned}$$

lo que significa que

$$h_i = \hat{y}_i^{(t-1)}(1 - \hat{y}_i^{(t-1)}),$$

es simplemente el producto de la probabilidad predicha y su complemento. Este valor positivo muestra que en efecto la función es localmente convexa, y además, al medir la curvatura, nos permite ajustar las predicciones con más precisión.

Sin embargo, como recién observamos, para obtener estos  $g_i$  y  $h_i$  necesitamos evaluar la predicción actual en dichas derivadas. Para ello necesitamos una predicción inicial, la cual se toma como un valor constante  $\gamma$  que minimiza la pérdida, calculado como

$$F_0 = \arg \min_{\gamma} \sum_{i=1}^n l(y_i, \gamma).$$

Por lo tanto, siendo consistentes con la función de pérdida que estamos utilizando, este valor es

$$\begin{aligned} F_0 &= \arg \min_{\gamma} \sum_{i=1}^n -y_i \gamma + \log(1 + e^{\gamma}), \\ &= \log \left( \frac{\bar{y}}{1 - \bar{y}} \right), \end{aligned}$$

donde  $\bar{y}$  es la proporción de muestras en la clase 1.

De este modo, dada una predicción inicial, el siguiente paso consiste en calcular los correspondientes  $g_i$  y  $h_i$  de cada muestra, y construir el árbol que maximice la ganancia de información.

A modo de ejemplo, note que en la [Figura 19](#), la partición que minimiza la pérdida es la que separa a los menores de 15 años al lado izquierdo, de los adultos al lado derecho. Sobre la rama izquierda, se realiza una partición adicional por género, dando lugar a un total de tres hojas en el árbol.

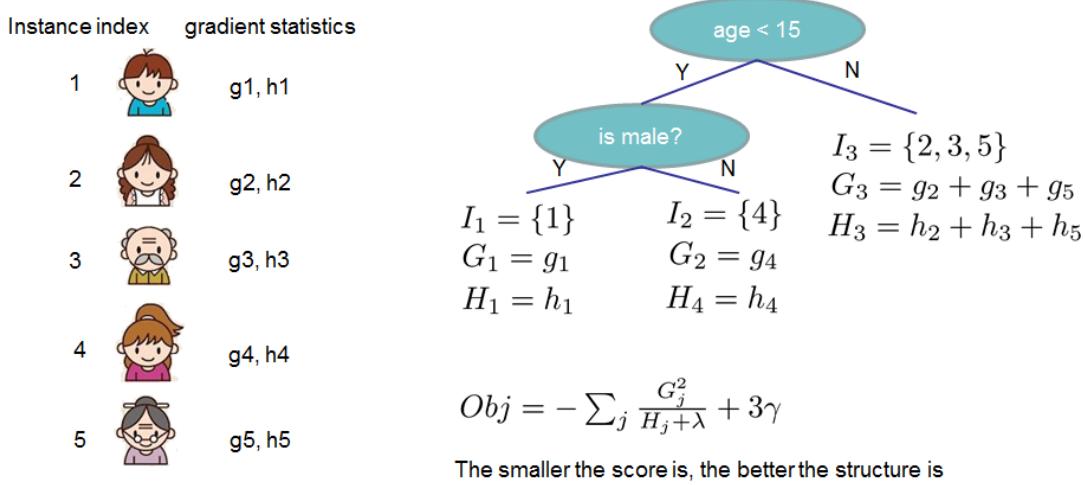
Luego, para cada hoja de dicho árbol se calcula su respectivo peso

$$w_j = -\frac{G_j}{H_j + \lambda},$$

de modo que ahora la nueva predicción es dada por

$$\begin{aligned} F_1(x_i) &= F_0 + \eta f_1(x_i), \\ &= F_0 + \eta w_{q(x_i)}, \end{aligned}$$

donde  $\eta$  es la tasa de aprendizaje que controla la contribución de cada árbol.



**Figura 19.** Ejemplo ilustrativo de un solo árbol en XGBoost. Fuente: tomada de [7]

Así, de forma general, tras  $m$  iteraciones

$$F_m(x_i) = F_{m-1}(x_i) + \eta f_m(x_i).$$

Cabe recordar que, dado que estamos prediciendo los logits, la salida final tras  $K$  árboles debe convertirse de nuevo en probabilidad mediante la sigmoide

$$\hat{y}_i = \sigma(F_K(x_i)) = \frac{1}{1 + e^{-F_K(x_i)}}.$$

En resumen, el XGBoost, mediante el entrenamiento iterativo de árboles de decisión, nos permite ajustar gradualmente las predicciones del conjunto de datos de entrenamiento de tal modo que sea capaz de predecir correctamente nuevos casos. A continuación, se presentan las métricas empleadas para evaluar el desempeño de un modelo en tareas de clasificación.

## 1.4. Métricas de desempeño

Las métricas de desempeño son utilizadas para medir o evaluar el rendimiento y la eficacia de modelos de clasificación. Es decir, nos proporcionan información sobre qué tan bueno es el modelo para realizar predicciones.

Para comprender mejor estas métricas, es de gran importancia definir una herramienta fundamental conocida como la Matriz de confusión ([Figura 20](#)).

		Valor real	
		+	-
Valor predicción	+	Verdadero positivo	Falso positivo
	-	Falso negativo	Verdadero negativo

**Figura 20.** Matriz de confusión.

La Matriz de confusión, en el caso de una clasificación binaria, organiza las predicciones del modelo en 4 categorías:

- Verdaderos positivos (VP): Instancias positivas que fueron correctamente clasificadas.
- Falsos positivos (FP): Instancias negativas que fueron incorrectamente clasificadas.
- Verdaderos negativos (VN): Instancias negativas que fueron correctamente clasificadas.
- Falsos negativos (FN): Instancias positivas que fueron incorrectamente clasificadas.

Teniendo esto en cuenta, se definen las siguientes métricas:

**Accuracy:** Es la proporción de predicciones correctas (tanto positivas como negativas) respecto al total de predicciones. Sin embargo, esta métrica es engañosa cuando se tiene un conjunto de datos desbalanceado,

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN}.$$

**Precision (Positive Predictive Value):** Es la proporción de instancias clasificadas como positivas que son realmente positivas,

$$\text{Precision} = \frac{VP}{VP + FP}.$$

**Negative predictive value:** Es la proporción de instancias clasificadas como negativas que son realmente negativas,

$$\text{Neg Pred Value} = \frac{VN}{VN + FN}.$$

**Recall (Sensitivity):** También conocida como tasa de verdaderos positivos. Es la proporción de instancias positivas que el modelo ha clasificado correctamente con respecto al total de instancias positivas,

$$\text{Recall} = \frac{VP}{\text{Real Positives}} = \frac{VP}{VP + FN}.$$

**Specificity:** También conocida como tasa de verdaderos negativos. Es la proporción de instancias negativas que el modelo ha clasificado correctamente con respecto al total de instancias negativas,

$$\text{Specificity} = \frac{VN}{\text{Real Negatives}} = \frac{VN}{VN + FP}.$$

**F1 Score:** Esta métrica proporciona un equilibrio entre Recall y Precision. Es decir, indica de manera conjunta la capacidad del modelo para predecir positivos correctamente y, a su vez, evitar los falsos negativos,

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

## 2. Metodología

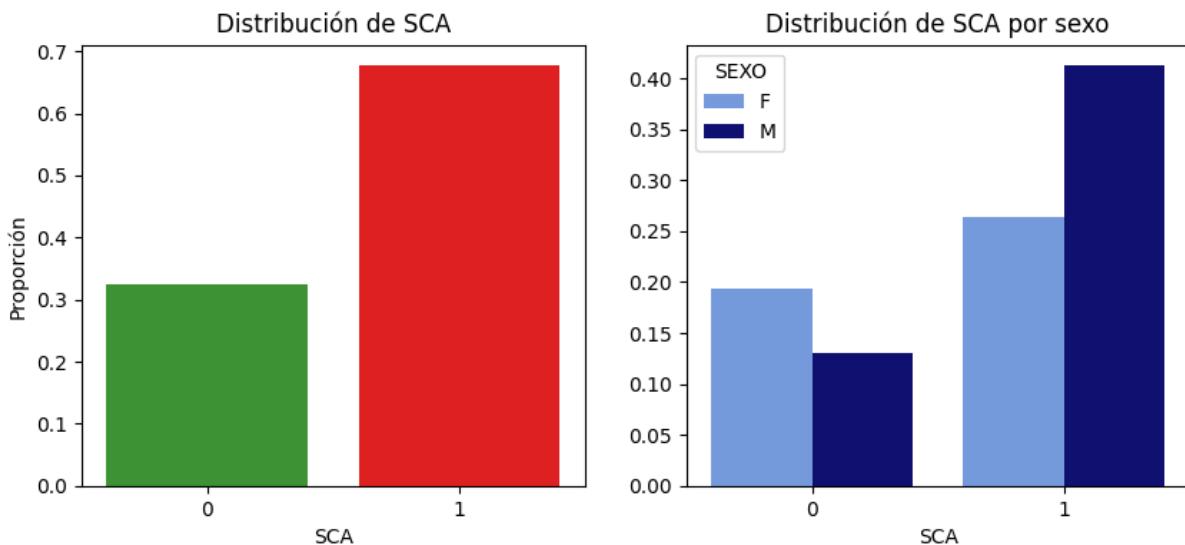
En esta sección se describen, de manera sintética, los pasos seguidos para el desarrollo del modelo. Toda la experimentación se desarrolló en *Python* (Google Colab), apoyándose en las librerías pandas, numpy, matplotlib, seaborn, sklearn, shap y xgboost. Para favorecer la reproducibilidad y el reporte de resultados se fijó una semilla aleatoria. A continuación se detallan las etapas del procedimiento.

- **Conjunto de Datos:** Se utilizó un conjunto de datos proporcionado por la Clínica de Occidente, con  $n = 859$  pacientes descritos por  $p = 24$  variables: 13 numéricas continuas, 10 categóricas binarias y la variable objetivo (presencia de SCA).
- **Análisis exploratorio de datos (EDA):** Se visualizaron las distribuciones de las diferentes variables y se calcularon algunos estadísticos descriptivos (promedios, desviaciones estandar, cuartiles) para obtener un panorama de los datos.
- **Partición de los datos:** Se realizó una partición estratificada de los datos en conjuntos de entrenamiento, validación y prueba (70/15/15).
- **Análisis de componentes principales (PCA):** Se ajustó el PCA únicamente sobre las 13 variables numéricas estandarizadas del conjunto de entrenamiento. Se utilizaron los gráficos de codo y de porcentaje de varianza explicada como criterio para elegir el número de componentes.
- **Modelos:** Se entrenó un XGBoost con las variables originales y otro con las componentes principales concatenadas a las variables binarias para comparar su desempeño. Adicionalmente, se entrenaron una regresión logística y un Random Forest como modelos de comparación. Para cada modelo se ajustaron los pesos de las clases con el fin de manejar el desbalance.
- **Selección de punto de corte:** Se utilizó el conjunto de validación para elegir el umbral de clasificación que maximizó el balanced accuracy de los diferentes modelos.
- **Evaluación:** Con el conjunto de datos reservado para prueba (Test-set) y el umbral seleccionado, se evaluó el rendimiento de los diferentes modelos (XGBoost, regresión logística y random forest), reportando valores de accuracy, precision, recall, specificity y F1-score.

- **Interpretabilidad:** Se calcularon los valores SHAP [20] para visualizar cómo contribuyen las diferentes variables a las predicciones del modelo.

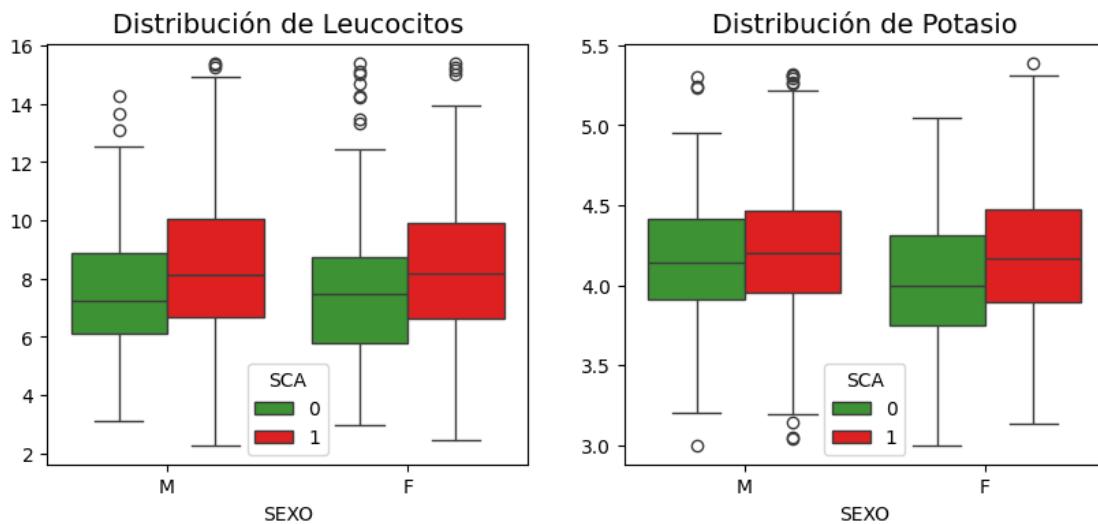
### 3. Resultados

Para comenzar conviene notar la diferencia en la proporción de clases de la variable objetivo, siendo en su mayoría los pacientes con síndrome coronario agudo (clase 1) como se observa en el gráfico izquierdo de la [Figura 21](#). De igual manera, en el gráfico de la derecha se observa cómo dentro de la clase negativa (clase 0) la mayoría son mujeres, mientras que en aquellos con síndrome coronario predominan los hombres.



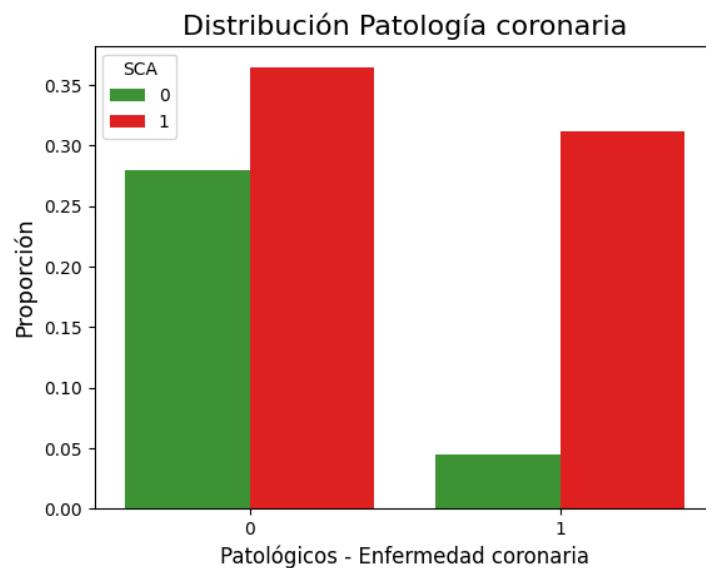
**Figura 21.** Distribución de la variable objetivo (SCA) en la muestra total (izquierda) y estratificada por sexo (derecha).

Seguidamente, al visualizar las distribuciones de las variables Leucocitos y Potasio (véase [Figura 22](#)), observamos cómo los pacientes con síndrome coronario tienden a presentar valores ligeramente más altos en estas variables. En la variable Leucocitos, los pacientes con síndrome coronario presentan un promedio de 8.5 frente a un 7.6 en aquellos de la clase negativa. De la misma manera, en la variable Potasio, la diferencia es más evidente en el sexo femenino, las cuales presentan un promedio de 4.2 frente a un 4.0 en las mujeres de la clase negativa.



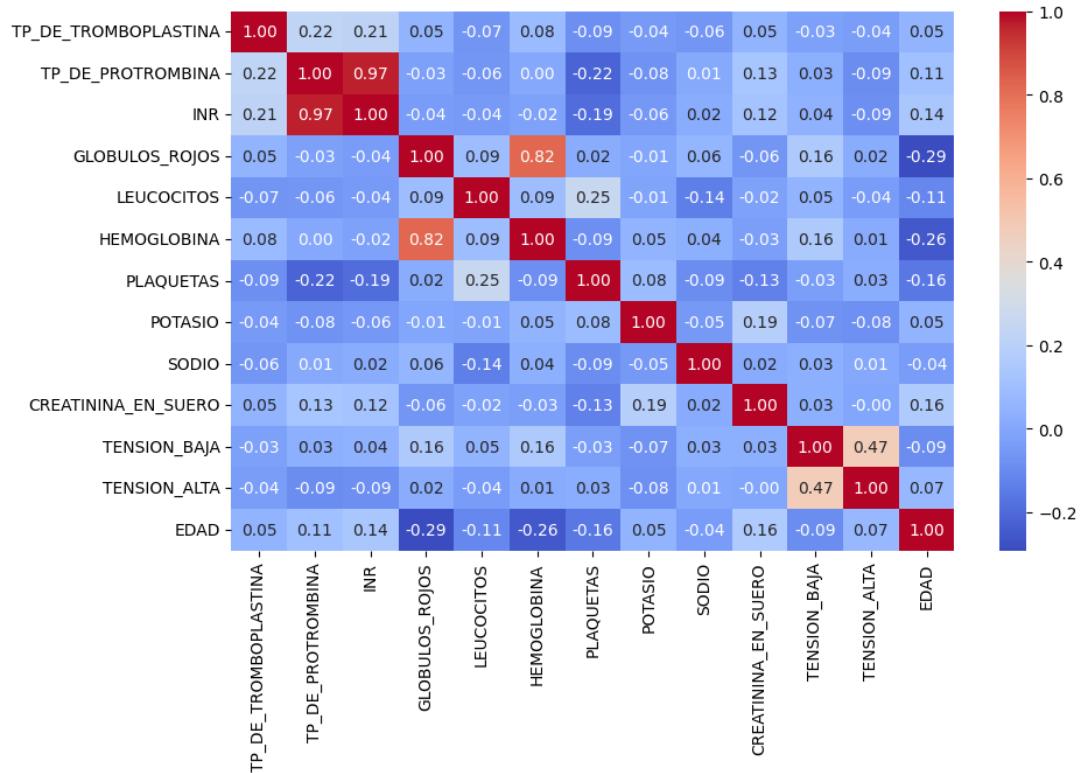
**Figura 22.** Distribuciones de leucocitos y potasio, estratificadas por sexo y SCA.

Dentro de las variables categóricas tenemos Patología coronaria, en la cual, como podemos observar, la diferencia en las proporciones es mucho mayor dentro de los pacientes con síndrome coronario (véase Figura 23), indicando que si un paciente tiene antecedentes de una enfermedad coronaria es bastante probable que sea clasificado como positivo.



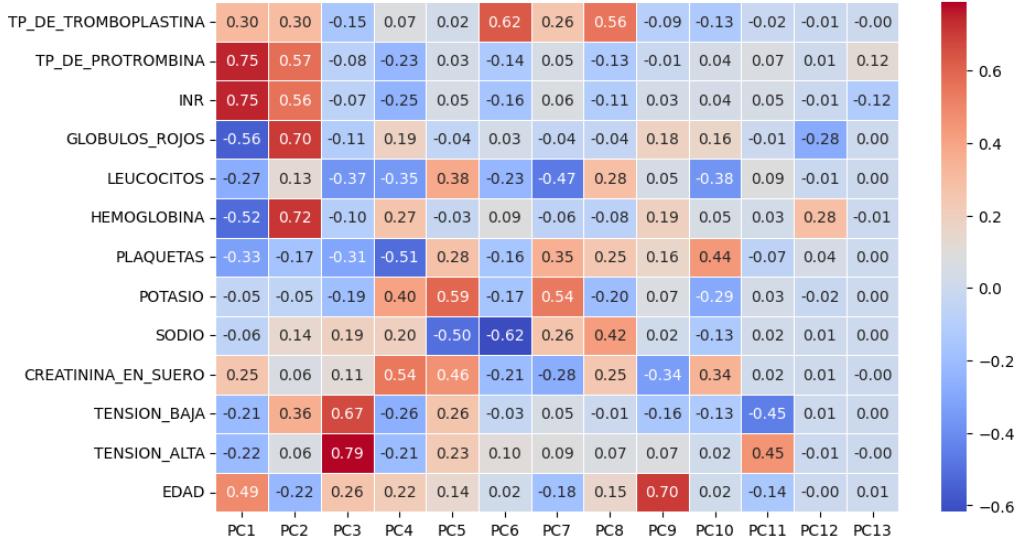
**Figura 23.** Distribución de Patología coronaria.

Ahora, la matriz de correlación en la [Figura 24](#) evidencia que existe una fuerte correlación entre las variables INR con TP. de protrombina, Hemoglobina con Glóbulos rojos, y una correlación un poco menos significativa entre Tensión baja con Tensión alta. Aunque las relaciones lineales entre las demás variables son débiles, se puede mitigar la redundancia en aquellas que presentan correlación y reducir un poco el ruido con el uso de PCA.



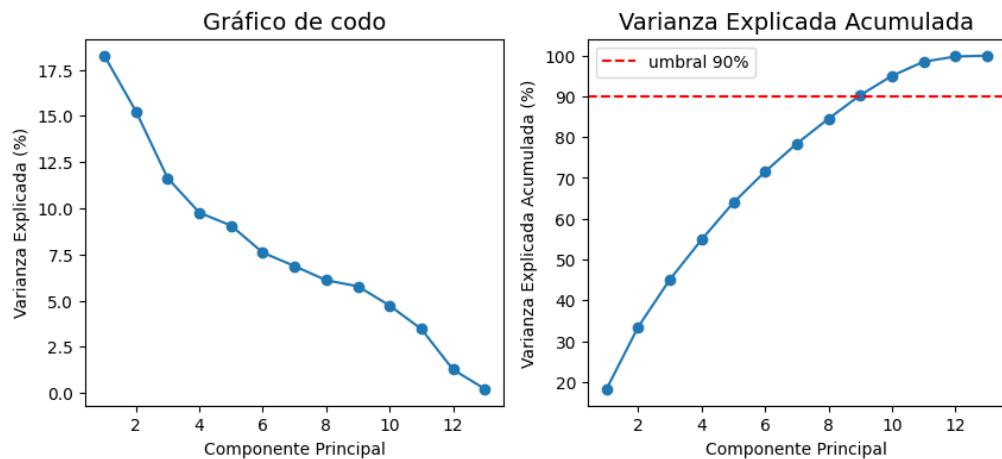
**Figura 24.** Matriz de coeficientes de correlación de Pearson.

Para realizar un correcto preprocesamiento y evitar la fuga de datos (data leakage) primero se realizó una partición estratificada en conjuntos de entrenamiento, validación y prueba (70/15/15). Luego, se realizó el análisis de componentes principales sobre el conjunto de entrenamiento y se calculó la correlación de las componentes con las diferentes variables. La [Figura 25](#) refleja cómo las tres primeras componentes concentran precisamente la información de las variables más colineales identificadas anteriormente en la [Figura 24](#), reduciendo así la redundancia.



**Figura 25.** Matriz de correlación entre variables originales y componentes principales.

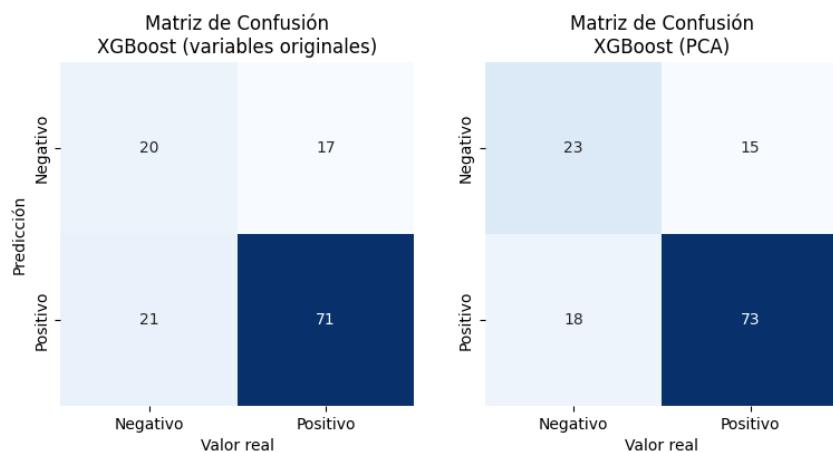
Sin embargo, tomar solo tres componentes no es suficiente para capturar la mayor parte de la variabilidad pues las demás componentes aún contienen información relevante, a excepción de las tres últimas las cuales muestran asociaciones débiles con las variables, lo que significa que su contribución es principalmente ruido. Esto se corrobora en el gráfico de varianza acumulada de la [Figura 26](#) donde el umbral de información relevante ( $\approx 90\%$ ) se alcanza alrededor de 9-10 componentes.



**Figura 26.** Gráfico de codo (varianza explicada por componente) y varianza explicada acumulada.

Con base en este análisis se tomó la decisión de trabajar con **10 componentes principales** y aprovechar este como especie de filtro para reducir la redundancia y el ruido.

Con el conjunto de entrenamiento, se entrenaron dos modelos: un XGBoost con las variables originales y un XGBoost con PCA (10 componentes) concatenado a las variables categóricas. La evaluación en el conjunto de validación mostró que el pipeline **PCA+XGBoost** superó en todas las métricas al XGBoost base (véase [Tabla 6](#)). En particular, podemos observar una ligera mejora tanto en la sensibilidad (recall) como en la especificidad, reduciendo de manera conjunta el número de falsos positivos y falsos negativos como se observa en la [Figura 27](#).



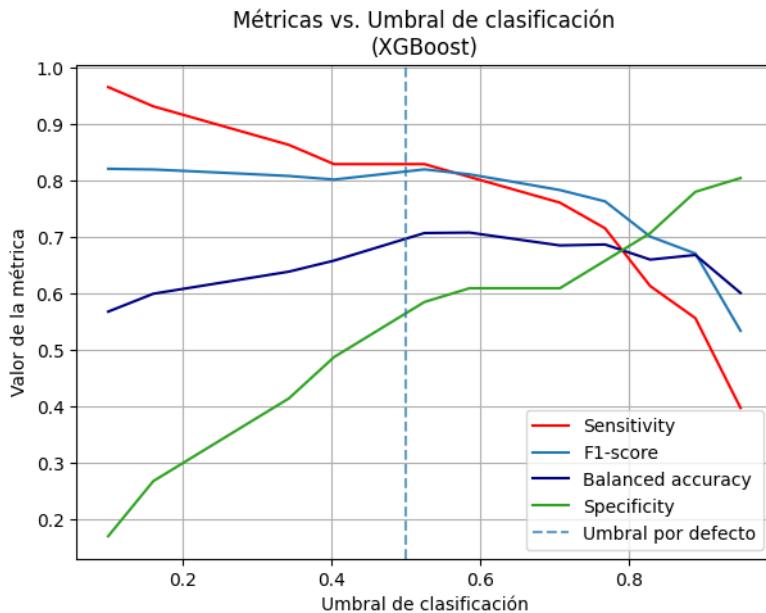
**Figura 27.** Matriz de confusión de XGBoost (originales) vs. PCA+XGBoost en el conjunto de validación.

	XGBoost (originales)	XGBoost (PCA)
Accuracy	0.71	0.74
Precision (clase 1)	0.77	0.80
Recall	0.81	0.83
F1-score	0.79	0.82
Precision (clase 0)	0.54	0.61
Specificity	0.49	0.56

**Tabla 6.** Resultados de los modelos XGBoost base y PCA+XGBoost (10 componentes) en el conjunto de validación.

Tras confirmar que el mejor desempeño fue obtenido mediante la configuración PCA +XGBoost, optamos por implementar esta estrategia y, por lo tanto, a partir de ahora nos referiremos a este solamente como XGBoost, teniendo presente que estaremos trabajando con las componentes principales.

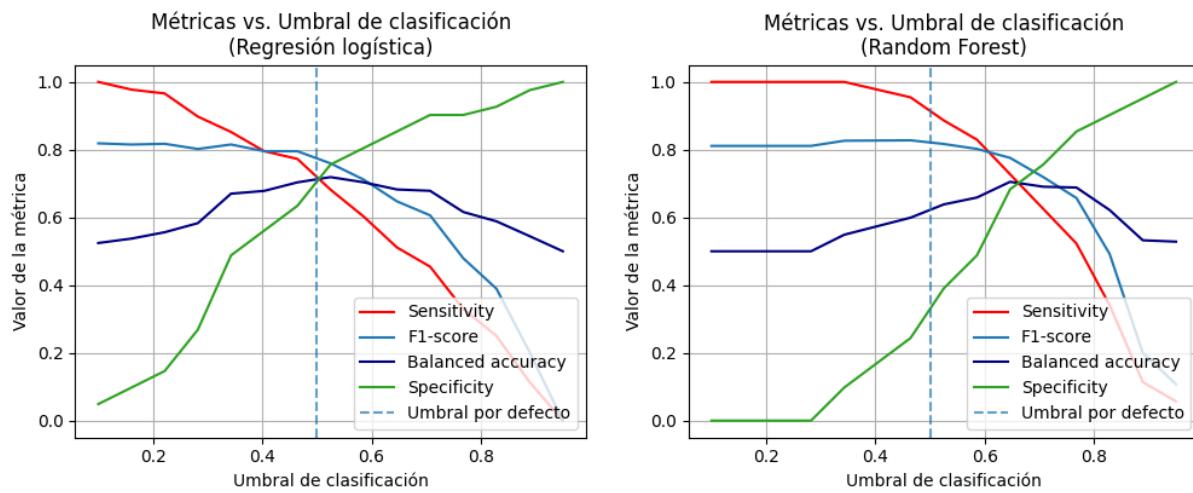
Ahora, los métodos de clasificación suelen producir probabilidades  $\hat{p} \in [0, 1]$  para etiquetar a cada muestra. En este contexto, el umbral por defecto que se utiliza es  $\tau = 0.5$ . Sin embargo, esto no es definitivo y dicho umbral puede ser ajustado según la métrica que deseemos priorizar. Para la selección de este umbral utilizamos el conjunto de validación (debidamente preprocesado) para evaluar diferentes valores y observar como cambian las métricas. En la [Figura 28](#) se evidencia cómo umbrales bajos aumentan la sensibilidad a costa de baja especificidad, mientras que umbrales altos hacen lo contrario.



**Figura 28.** Evolución de sensibilidad, F1, exactitud balanceada y especificidad en el conjunto de validación según el umbral de clasificación  $\tau$ .

Dado el desequilibrio entre clases y con el fin de evaluar el rendimiento global sin favorecer a la clase mayoritaria, seleccionamos el umbral  $\tau$  que maximiza la *exactitud balanceada* (balanced accuracy) en dicho conjunto de validación. Para este modelo el máximo balanced accuracy fue de 0.71 y se alcanzó con el umbral  $\tau = 0.58$ .

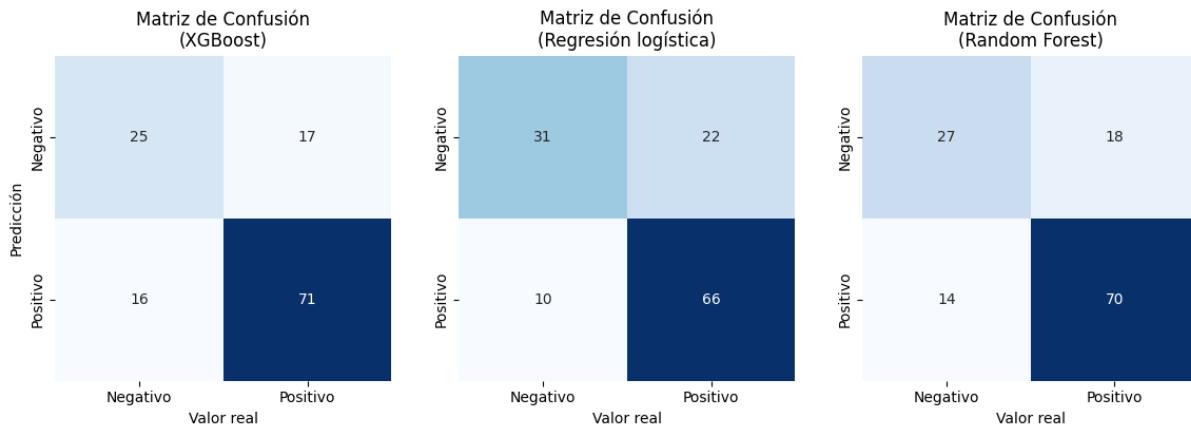
Seguidamente, con el conjunto de entrenamiento (con componentes principales), se entrenaron dos modelos: una regresión logística y un Random Forest. Luego, con el conjunto de validación, se realizó el mismo proceso para la selección de los mejores umbrales como se observa en la [Figura 29](#). Los resultados de estos fueron un balanced\_accuracy = 0.75 con umbral  $\tau = 0.48$  para el modelo logístico, y un balanced\_accuracy = 0.73 con umbral  $\tau = 0.64$  para el Random Forest.



**Figura 29.** Evolución de sensibilidad, F1, exactitud balanceada y especificidad en el conjunto de validación para la regresión logística (izquierda) y Random Forest (derecha).

El desempeño de cada modelo en el conjunto de validación, usando los umbrales seleccionados, se resumen en la [Tabla 7](#). Note que los modelos basados en árboles (XGBoost y Random Forest) arrojan resultados muy similares, a diferencia de la regresión logística, la cual presenta una mayor especificidad a costa de una menor sensibilidad (recall), capturando más verdaderos negativos pero disminuyendo los verdaderos positivos como se observa en la [Figura 30](#).

No obstante, esta comparación en el conjunto de validación puede ser optimista puesto que hemos usado este mismo conjunto de datos para la elección de los umbrales. En su lugar, debemos evaluar el desempeño mediante los datos del conjunto de prueba que reservamos en un comienzo y, de este modo, aproximarnos más a resultados reales.



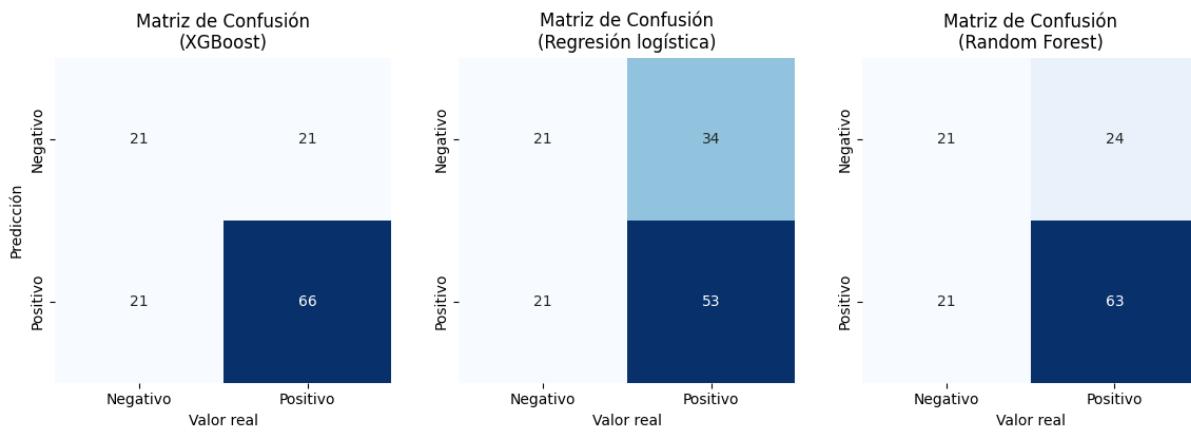
**Figura 30.** Matrices de confusión de XGBoost, regresión logística y Random Forest en el conjunto de validación (con umbral ajustado).

	XGBoost	Regresión logística	Random Forest
Accuracy	0.74	0.75	0.75
Precision (clase 1)	0.82	0.87	0.83
Recall	0.81	0.75	0.80
F1-score	0.81	0.80	0.81
Precision (clase 0)	0.60	0.58	0.60
Specificity	0.61	0.76	0.66

**Tabla 7.** Resultados de los modelos XGBoost, regresión logística y Random Forest en el conjunto de validación (Todos usando PCA).

Por lo tanto, para acercarnos al verdadero desempeño que tendrían los diferentes modelos con datos nunca vistos, el primer paso es realizar los debidos preprocesamientos, es decir, ajustar cada uno de los datos según los valores de estandarización obtenidos en el conjunto de entrenamiento y, seguidamente, proyectar estos datos estandarizados sobre los vectores propios obtenidos del PCA. Finalmente, se ajustaron las predicciones según el umbral obtenido anteriormente con el conjunto de validación, lo que dio como resultado las matrices de confusión de la [Figura 31](#) y las métricas de la [Tabla 8](#).

Podemos observar que de los tres modelos, el XGBoost presentó el mejor F1-score (0.76), es decir, el mejor balance entre sensibilidad y precisión puesto que obtuvo la mayor sensibilidad (0.76) y precisión positiva (0.76), detectando 66 de 87 casos de SCA, frente a



**Figura 31.** Matrices de confusión de XGBoost, regresión logística y Random Forest en el conjunto de prueba (datos nunca vistos).

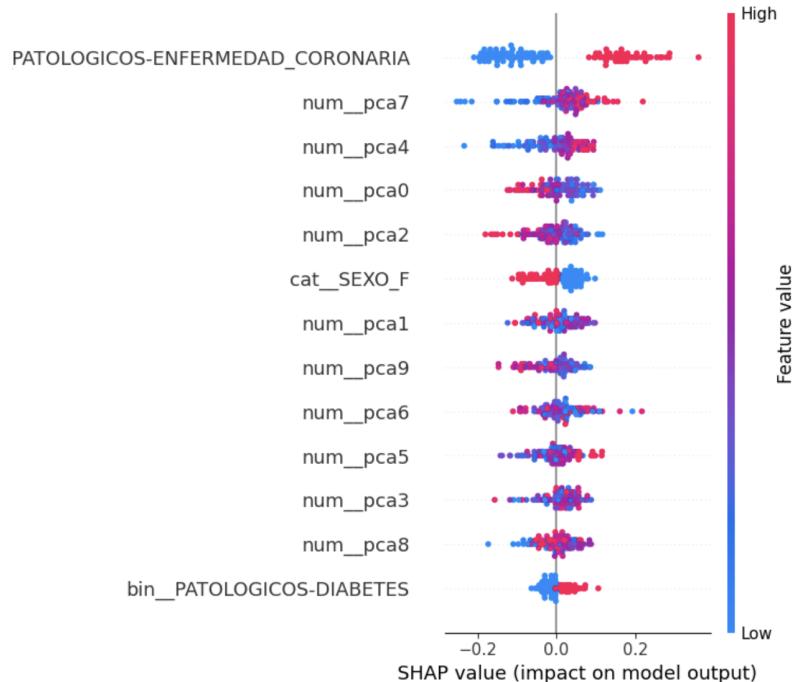
53 de la regresión logística y 63 del Random Forest. En términos absolutos, esto implica 13 y 3 falsos negativos menos, respectivamente. En cuanto a la especificidad, los tres modelos capturaron exactamente la mitad de los pacientes etiquetados como negativos (specificity = 0.50).

	XGBoost	Regresión logística	Random Forest
Accuracy	0.67	0.57	0.65
Precision (clase 1)	0.76	0.72	0.75
Recall	0.76	0.61	0.72
F1-score	0.76	0.66	0.74
Precision (clase 0)	0.50	0.38	0.47
Specificity	0.50	0.50	0.50

**Tabla 8.** Resultados de los modelos XGBoost, regresión logística y Random Forest en el conjunto de prueba (datos nunca vistos).

Dado que el mejor modelo (XGBoost) presentó una sensibilidad de 0.76 y una especificidad de 0.50, un resultado negativo no reduce lo suficiente la probabilidad de SCA, por lo que el clasificador no es adecuado para descartar casos. Sin embargo, un resultado positivo eleva la probabilidad de SCA, por lo que sirve como señal de alerta para priorizar una evaluación adicional, aunque no como confirmación definitiva.

Por último, para darle un poco de interpretabilidad al modelo, se calcularon los valores SHAP (SHapley Additive exPlanations) [20] los cuales surgen de la teoría de juegos y nos proporcionan información sobre como contribuyen las diferentes variables al momento de hacer predicciones. Dado que el clasificador es un ensamble de árboles, se utilizó específicamente el algoritmo TreeSHAP [19] para generar el gráfico de la [Figura 32](#).



**Figura 32.** SHAP values de las 13 variables más importantes. Cada punto representa un paciente; el color indica el valor de la característica (alto en rojo, bajo en azul). Valores SHAP positivos empujan la predicción hacia una mayor probabilidad de SCA y valores negativos la disminuyen.

El gráfico nos muestra que **PATLÓGICOS-ENFERMEDAD\_Coronaria** es la variable que tiene la mayor contribución. Esto significa que la presencia de antecedentes de enfermedad coronaria empuja hacia arriba la probabilidad de tener SCA, lo cual es acorde a lo observado en la distribución de esta variable ([Figura 23](#)). Seguidamente, **pca7** (PC8 realmente) le sigue en orden de importancia; dado que en esta componente las variables con mayor aporte, según la [Figura 25](#), son **TP\_DE\_TROMBOPLASTINA** y **SODIO**, esto nos dice que valores bajos en estas variables reducen la probabilidad de SCA. En contraste, valores altos en las componentes **pca0** y **pca2** (PC1 y PC3) disminuyen la probabilidad de SCA.

## 4. Conclusiones

- Este trabajo demuestra que la inteligencia artificial, el aprendizaje automático y la ciencia de datos están transformando de manera profunda el campo de la salud. La base de la IA contemporánea es, en gran medida, matemática y de alta complejidad; por ello, una comprensión profunda de estas herramientas exige ahondar más en sus componentes matemáticas. En este marco, los matemáticos con formación avanzada, usualmente de nivel doctoral, contribuyen de forma esencial a la investigación en inteligencia artificial. En particular, gran parte de la formación que he recibido como matemático ha sido aplicada aquí, especialmente el álgebra lineal, el cálculo, el análisis numérico, la probabilidad y la estadística.
- Además, este trabajo permitió constatar que la estrategia empleada fue eficaz para la detección de pacientes con SCA, identificando 66 de 87 casos positivos (sensibilidad = 0.76). Al menos para este conjunto de datos, el análisis de componentes principales mejoró los resultados al reducir la colinealidad entre variables y atenuar el ruido implícito. Del mismo modo, el método de ensamble XGBoost mostró el mejor desempeño en comparación con la regresión logística y el Random Forest, no por ser intrínsecamente superior a estos métodos, sino por resultar más adecuado para las características de este problema.
- Hasta el momento, este trabajo de grado es el primero del Programa de Matemáticas de la Universidad del Valle que aborda inteligencia artificial aplicada a un problema real (en este caso, clínico). Por ello, se constituye como un trabajo pionero que busca abrir nuevos enfoques y líneas de investigación en inteligencia artificial y ciencia de datos aplicada, promover colaboraciones interdisciplinarias e inspirar a futuras generaciones de matemáticos a interesarse en este campo.

## 5. Limitaciones

La inteligencia artificial representa una oportunidad notable de transformación y crecimiento en múltiples sectores. Tanto el sector salud como el social y el financiero están siendo revolucionados por las nuevas tecnologías y la disponibilidad de datos. Sin em-

bargo, su desempeño depende de forma crítica de la calidad de los datos, ya que estos representan la materia prima fundamental de los diferentes sistemas de IA.

En el Documento CONPES 4144 (Política Nacional de Inteligencia Artificial)<sup>5</sup>, se formula una política muy favorable para que nuestro país avance en la investigación y el desarrollo de inteligencia artificial. Mediante diferentes estrategias, se busca generar capacidades y condiciones que impulsen la transformación social y económica de Colombia. En cuanto al desarrollo de modelos, el documento menciona que «para asegurar la óptima creación de algoritmos de aprendizaje automático, es fundamental que los datos utilizados cumplan con criterios de calidad que incluyan la existencia de suficientes datos para obtener conclusiones válidas, datos adecuados para diferentes análisis y aplicaciones, datos claros, consistentes y veraces con respecto a la realidad que están describiendo»<sup>6</sup>.

No obstante, entre sus diagnósticos el CONPES señala que «en materia de datos, se evidencia en el país una baja disponibilidad de datos de calidad para el desarrollo de inteligencia artificial»<sup>7</sup>, lo que provoca limitaciones en la construcción y entrenamiento de modelos como se observó en el presente trabajo. Esta situación, sin embargo, representa para el país una oportunidad de mejora en cuanto a la implementación de estrategias de control de calidad de los datos y, de esta manera, crear una cultura de datos confiables que permita a las empresas de los diferentes sectores desplegar con éxito sistemas de inteligencia artificial, impactando directamente en su competitividad y contribuyendo al progreso como sociedad.

## 6. Anexos

Respositorio en github del código: [github.com/thesis-sca](https://github.com/thesis-sca). Notebook en colab: [Colab](#).

---

<sup>5</sup>Departamento Nacional de Planeación (2025), CONPES 4144. Disponible en [dnp.gov.co](http://dnp.gov.co)

<sup>6</sup>Departamento Nacional de Planeación (2025), CONPES 4144, sec. 3, p. 36.

<sup>7</sup>Departamento Nacional de Planeación (2025), CONPES 4144, sec. 4.2.3: “Baja disponibilidad de datos de calidad con representatividad para el desarrollo de la IA.”

## Referencias

- [1] C. C. Aggarwal. *Data Mining: The Textbook*. Springer, 2015.
- [2] T. W. Anderson. *An introduction to multivariate statistical analysis*, volume 2. Wiley New York, 1958.
- [3] J. C. Baez. What is entropy? *arXiv*, 2024.
- [4] C. M. Bishop. *Pattern Recognition and Machine Learning by Christopher M. Bishop*. Springer Science+ Business Media, LLC, 2006.
- [5] M. Bramer. *Principles of Data Mining*. Springer London, 2016.
- [6] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.
- [7] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [8] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
- [9] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc., 2022.
- [10] J. Han, J. Pei, and H. Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.
- [11] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.
- [12] E. B. Hunt, J. Marin, and P. J. Stone. *Experiments in induction*. Academic press, 1966.
- [13] F. Husson, S. Lê, and J. Pagès. *Análisis de datos con R*. Colección de matemáticas. Escuela Colombiana de Ingeniería Julio Garavito Editorial, Bogotá, Colombia, 1 edition, 2013.

- [14] L. Igual and S. Seguí. Introduction to data science. In *Introduction to Data Science: A Python Approach to Concepts, Techniques and Applications*. Springer, 2024.
- [15] G. James, D. Witten, T. Hastie, R. Tibshirani, et al. *An introduction to statistical learning*, volume 112. Springer, 2013.
- [16] I. Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.
- [17] D. R. Kincaid and E. W. Cheney. *Numerical analysis: mathematics of scientific computing*, volume 2. American Mathematical Soc., 2009.
- [18] M. Kubat. *An introduction to machine learning*, volume 2. Springer, 2017.
- [19] S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.
- [20] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.
- [21] V. Mirjalili and S. Raschka. *Python machine learning*. Marcombo, 2020.
- [22] T. M. Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.
- [23] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- [24] T. Pn, M. Steinbach, and V. Kumar. *Introduction to data mining*. Addison-Wesley, 2005.