LDA datos de clientes de aseguradora

RUBEN PIZARRO GURROLA

2025-09-23

# Inicializar datos

datos <- data.frame(  
 id = 1:10,  
 edad = c(25,28,32,29,35,50,55,60,48,52),  
 ingresos = c(15000,18000,20000,16000,21000,28000,30000,35000,27000,29000),  
 hijos = c(0,1,2,1,2,3,2,4,3,2),  
 gastos\_medicos = c(2000,2500,3000,2200,3100,6000,7000,8000,5500,6200),  
 ejercicio = c(6,5,3,4,2,1,0,1,2,0),  
 imc = c(22,24,27,23,28,30,31,29,28,32),  
 clase = factor(c("Joven","Joven","Joven","Joven","Joven","Mayor","Mayor","Mayor","Mayor","Mayor"))  
)  
  
  
datos

## id edad ingresos hijos gastos\_medicos ejercicio imc clase  
## 1 1 25 15000 0 2000 6 22 Joven  
## 2 2 28 18000 1 2500 5 24 Joven  
## 3 3 32 20000 2 3000 3 27 Joven  
## 4 4 29 16000 1 2200 4 23 Joven  
## 5 5 35 21000 2 3100 2 28 Joven  
## 6 6 50 28000 3 6000 1 30 Mayor  
## 7 7 55 30000 2 7000 0 31 Mayor  
## 8 8 60 35000 4 8000 1 29 Mayor  
## 9 9 48 27000 3 5500 2 28 Mayor  
## 10 10 52 29000 2 6200 0 32 Mayor

# Datos numéricos y factor Clase

# Matriz de variables numéricas (en el orden indicado)  
X <- as.matrix(datos[, c("edad","ingresos","hijos","gastos\_medicos","ejercicio","imc")])  
clase <- datos$clase  
niveles <- levels(clase) # c("Joven", "Mayor")  
d <- ncol(X) # número de variables (6)

# Medias aritméticas por clase

# Lista de medias por clase (cada elemento es un vector de longitud d)  
mu\_por\_clase <- lapply(niveles, function(k) colMeans(X[clase == k, , drop = FALSE]))  
names(mu\_por\_clase) <- niveles  
  
mu\_por\_clase

## $Joven  
## edad ingresos hijos gastos\_medicos ejercicio   
## 29.8 18000.0 1.2 2560.0 4.0   
## imc   
## 24.8   
##   
## $Mayor  
## edad ingresos hijos gastos\_medicos ejercicio   
## 53.0 29800.0 2.8 6540.0 0.8   
## imc   
## 30.0

# $Joven: c(29.8, 18000, 1.2, 2560, 4.0, 24.8)  
# $Mayor: c(53.0, 29800, 2.8, 6540, 0.8, 30.0)

# Matriz Sw por clase

A través del ciclo se obtiene la matriz de desviaciones .

* Centrar *Mk*: se resta las medias de la clase y queda las desviaciones internas.
* Producto \*t(Mk) % SIMBOLO \*\*% Mk \*: convierte esas desviaciones en una matriz de sumas de cuadrados y productos cruzados
* Acumular *(SW)*: se van sumando los resultados de todas las clases para construir.

La salida en modo consola de la matriz :

SW <- matrix(0, nrow = d, ncol = d)  
  
for (k in niveles) {  
 Xk <- X[clase == k, , drop = FALSE] # submatriz de la clase k (n\_k x d)  
 mu\_k <- mu\_por\_clase[[k]] # vector de medias de la clase k (long d)  
 Mk <- sweep(Xk, 2, mu\_k, FUN = "-") # centrar: Xk - mu\_k (n\_k x d)  
 SW <- SW + t(Mk) %\*% Mk # scatter de clase y suma  
}  
  
# Mostrar con redondeo  
round(SW, 2)

## edad ingresos hijos gastos\_medicos ejercicio imc  
## edad 146.8 93000 18.2 25160 -30.0 40.8  
## ingresos 93000.0 64800000 13800.0 16740000 -16200.0 25000.0  
## hijos 18.2 13800 5.6 2880 -3.2 4.2  
## gastos\_medicos 25160.0 16740000 2880.0 4764000 -3860.0 5360.0  
## ejercicio -30.0 -16200 -3.2 -3860 12.8 -20.0  
## imc 40.8 25000 4.2 5360 -20.0 36.8

# Calcular SB

cols <- c("edad","ingresos","hijos","gastos\_medicos","ejercicio","imc") # orden deseado  
X <- as.matrix(datos[, cols]) # matriz numérica N x d  
clase <- datos$clase # factor de clases  
niveles <- levels(clase) # niveles de clase (p.ej., "Joven", "Mayor")  
d <- ncol(X) # número de variables (d)  
  
# --- 3) Media global ---  
mu\_global <- colMeans(X) # vector (long d) con medias globales  
  
# --- 4) Construcción de S\_B según: S\_B = sum\_k n\_k \* (mu\_k - mu)(mu\_k - mu)^T ---  
SB <- matrix(0, nrow = d, ncol = d) # inicializa matriz d x d en ceros  
  
for (k in niveles) { # recorre cada clase k  
 Xk <- X[clase == k, , drop = FALSE] # submatriz de la clase k (n\_k x d)  
 mu\_k <- colMeans(Xk) # medias por clase (vector long d)  
 nk <- nrow(Xk) # tamaño de la clase k  
 diff <- matrix(mu\_k - mu\_global, ncol = 1) # vector columna (d x 1) con (mu\_k - mu)  
 SB <- SB + nk \* (diff %\*% t(diff)) # acumula n\_k \* producto exterior (d x d)  
}  
  
SB

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 1345.6 684400 92.8 230840 -185.6 301.6  
## [2,] 684400.0 348100000 47200.0 117410000 -94400.0 153400.0  
## [3,] 92.8 47200 6.4 15920 -12.8 20.8  
## [4,] 230840.0 117410000 15920.0 39601000 -31840.0 51740.0  
## [5,] -185.6 -94400 -12.8 -31840 25.6 -41.6  
## [6,] 301.6 153400 20.8 51740 -41.6 67.6

# Medias por clase

# Medias por clase  
muJ <- colMeans(X[clase=="Joven", , drop=FALSE])  
muM <- colMeans(X[clase=="Mayor", , drop=FALSE])  
d <- muM - muJ # vector diferencia de media  
d

## edad ingresos hijos gastos\_medicos ejercicio   
## 23.2 11800.0 1.6 3980.0 -3.2   
## imc   
## 5.2

# Solución del sistemas de ecuaciones para

# Resolver el sistema  
w <- solve(SW, d) # w = SW^{-1} d  
cat("\nSolución w (sin normalizar):\n")

##   
## Solución w (sin normalizar):

print(round(w, 6))

## edad ingresos hijos gastos\_medicos ejercicio   
## 14.671992 -0.017529 9.289174 -0.009192 35.465488   
## imc   
## 15.336134

# Normalizar w  
w\_norm <- w / sqrt(sum(w^2))  
cat("\nVector w normalizado (||w||=1):\n")

##   
## Vector w normalizado (||w||=1):

print(round(w\_norm, 4))

## edad ingresos hijos gastos\_medicos ejercicio   
## 0.3463 -0.0004 0.2193 -0.0002 0.8372   
## imc   
## 0.3620

# Razón de Fisher

# Numerador: w^T SB w  
numerador <- as.numeric(t(w) %\*% SB %\*% w)  
  
# Denominador: w^T SW w  
denominador <- as.numeric(t(w) %\*% SW %\*% w)  
  
# Razón de Fisher  
J <- numerador / denominador  
  
# Mostrar resultados  
paste0("Numerador (w^T SB w) :", numerador)

## [1] "Numerador (w^T SB w) :15243.3278178707"

paste0("Denominador (w^T SW w) :", denominador)

## [1] "Denominador (w^T SW w) :78.0854091821839"

paste0("Razón de Fisher J(w) :", J)

## [1] "Razón de Fisher J(w) :195.213522955434"

# Proyectar

## Primer dimensión D1

w\_norm

## edad ingresos hijos gastos\_medicos ejercicio   
## 0.3463464886 -0.0004137847 0.2192798765 -0.0002169953 0.8371969436   
## imc   
## 0.3620241772

X

## edad ingresos hijos gastos\_medicos ejercicio imc  
## [1,] 25 15000 0 2000 6 22  
## [2,] 28 18000 1 2500 5 24  
## [3,] 32 20000 2 3000 3 27  
## [4,] 29 16000 1 2200 4 23  
## [5,] 35 21000 2 3100 2 28  
## [6,] 50 28000 3 6000 1 30  
## [7,] 55 30000 2 7000 0 31  
## [8,] 60 35000 4 8000 1 29  
## [9,] 48 27000 3 5500 2 28  
## [10,] 52 29000 2 6200 0 32

D1 = as.numeric(X %\*% w\_norm)  
D1

## [1] 15.00561 14.80093 14.88121 14.84073 15.00959 16.78514 16.77786 16.77538  
## [9] 16.72788 16.68822

## Segunda dimensión D2

### Datos centrados

La función genera una matriz basada en la diferencia de los datos medio su media global.

La función *deep()* aplica alguna operación matricial de una matriz o *data.frame* sobre un vector por fila o por columna, la operación es reiterativa, sin necesidad de un ciclo explícito, similar a funciones *apply()*.

Al final los resultados Xc1 y Xc2 deben ser lo mismo solo que cengtrados de diferente manera.

# Datos centrados  
Xc1 <- scale(X, center = TRUE, scale = FALSE) # centrar por media global  
mu\_global <- colMeans(X)  
Xc2 <- sweep(X, 2, mu\_global, FUN = "-") # Aplica alguna operación matricial de una matriz/data.frame sobre un vector por fila o por columna, la operación es reitierativa, sin necesidad de un ciclo explícito, similar a funciones apply  
Xc1

## edad ingresos hijos gastos\_medicos ejercicio imc  
## [1,] -16.4 -8900 -2 -2550 3.6 -5.4  
## [2,] -13.4 -5900 -1 -2050 2.6 -3.4  
## [3,] -9.4 -3900 0 -1550 0.6 -0.4  
## [4,] -12.4 -7900 -1 -2350 1.6 -4.4  
## [5,] -6.4 -2900 0 -1450 -0.4 0.6  
## [6,] 8.6 4100 1 1450 -1.4 2.6  
## [7,] 13.6 6100 0 2450 -2.4 3.6  
## [8,] 18.6 11100 2 3450 -1.4 1.6  
## [9,] 6.6 3100 1 950 -0.4 0.6  
## [10,] 10.6 5100 0 1650 -2.4 4.6  
## attr(,"scaled:center")  
## edad ingresos hijos gastos\_medicos ejercicio   
## 41.4 23900.0 2.0 4550.0 2.4   
## imc   
## 27.4

Xc2

## edad ingresos hijos gastos\_medicos ejercicio imc  
## [1,] -16.4 -8900 -2 -2550 3.6 -5.4  
## [2,] -13.4 -5900 -1 -2050 2.6 -3.4  
## [3,] -9.4 -3900 0 -1550 0.6 -0.4  
## [4,] -12.4 -7900 -1 -2350 1.6 -4.4  
## [5,] -6.4 -2900 0 -1450 -0.4 0.6  
## [6,] 8.6 4100 1 1450 -1.4 2.6  
## [7,] 13.6 6100 0 2450 -2.4 3.6  
## [8,] 18.6 11100 2 3450 -1.4 1.6  
## [9,] 6.6 3100 1 950 -0.4 0.6  
## [10,] 10.6 5100 0 1650 -2.4 4.6

Xc <- Xc2 # o bien puede ser Xc <-Xc1  
Xc

## edad ingresos hijos gastos\_medicos ejercicio imc  
## [1,] -16.4 -8900 -2 -2550 3.6 -5.4  
## [2,] -13.4 -5900 -1 -2050 2.6 -3.4  
## [3,] -9.4 -3900 0 -1550 0.6 -0.4  
## [4,] -12.4 -7900 -1 -2350 1.6 -4.4  
## [5,] -6.4 -2900 0 -1450 -0.4 0.6  
## [6,] 8.6 4100 1 1450 -1.4 2.6  
## [7,] 13.6 6100 0 2450 -2.4 3.6  
## [8,] 18.6 11100 2 3450 -1.4 1.6  
## [9,] 6.6 3100 1 950 -0.4 0.6  
## [10,] 10.6 5100 0 1650 -2.4 4.6

## Generar los autovalores

Con la funcion *prcomp()* del modelo *PCA* se contruyen los componentes principales o los autovectores de la matriz de covarianza *S* con los datos centrados.

Se toma en cuenta solo el *pc1* porque en sus variables existe la gran mayoría de la explicación de la varianza, en este caso en la variable ingresos explica en el primer componente alrededor del *95.09%* y la variable gastos médicos explica alrededor del *30.95%*.

En términos del modelo *LDA* como ya se tiene la primera dimension *D1*, es necesario una segunda dimensión *D2*, se toma del primer componente del *PCA* dado que este explica la mayor proporción de la varianza de los datos, además de que el *pc1* es el componente que resulta ser más informativo con respecto a los demás.

# Componentes principales de los datos centrados  
pca <- prcomp(Xc, center = FALSE, scale. = FALSE)  
print("Los autovalores o matriz de covarianzas")

## [1] "Los autovalores o matriz de covarianzas"

PC <- as.data.frame(pca$rotation)  
print(PC)

## PC1 PC2 PC3 PC4  
## edad 0.0017926682 0.0040339732 -0.4409336192 -0.8249315650  
## ingresos 0.9509056801 -0.3094787446 0.0009496648 0.0001535115  
## hijos 0.0001397920 -0.0014188557 0.0501685894 -0.2506893779  
## gastos\_medicos 0.3094752354 0.9508957803 0.0010110500 0.0038231000  
## ejercicio -0.0002545527 0.0003977701 0.4507302189 0.0803779145  
## imc 0.0004102699 -0.0012951128 -0.7745327635 0.5001674719  
## PC5 PC6  
## edad -0.2468945510 -0.2531581532  
## ingresos -0.0005715875 0.0002053090  
## hijos 0.9470695910 -0.1941522743  
## gastos\_medicos 0.0024817444 0.0007044747  
## ejercicio -0.1810357095 -0.8704065197  
## imc 0.0965497629 -0.3749770605

print ("pc1")

## [1] "pc1"

pc1 <- PC[,1]  
print(round(pc1, 4))

## [1] 0.0018 0.9509 0.0001 0.3095 -0.0003 0.0004

## El método Gram–Schmidt

De acuerdo con ChatGPt el problema es que no hay garantía de que pc sea ortogonal a w. Si se prpyectos datos sobre w y pc1 directamente, esos ejes podrían estar correlacionados, la representación 2D sería engañosa.

Por lo anterior, se toma a pc1 y se le quita la “parte” que está alineada con.

## Procedimiento para construir vectores ortogonales con R Gram–Schmidt

# --- 1. Definir los vectores de 5 dimensiones ---  
u <- c(2, -1, 3, 0, 4)  
v <- c(1, 2, 1, 3, 1)  
  
# --- 2. Implementar la fórmula de Gram-Schmidt ---  
  
# 2a. Calcular el producto escalar (numerador: u · v)  
# El producto escalar es la suma del producto elemento a elemento  
u\_dot\_v <- sum(u \* v)  
# Resultado: 7  
  
# 2b. Calcular el cuadrado de la norma (denominador: ||v||^2)  
v\_norm\_sq <- sum(v \* v)  
# Resultado: 16  
  
# 2c. Calcular el escalar (coeficiente de la proyección)  
escalar <- u\_dot\_v / v\_norm\_sq  
# Resultado: 0.4375  
  
# 2d. Calcular la Proyección Vectorial (Proy\_v u)  
proyeccion <- escalar \* v  
  
# 2e. Calcular el vector Ortogonal (u\_perpendicular = u - Proy\_v u)  
u\_ortogonal <- u - proyeccion  
  
cat("--- Vectores ---\n")

## --- Vectores ---

cat("Vector V: ", v, "\n")

## Vector V: 1 2 1 3 1

cat("Vector U: ", u, "\n")

## Vector U: 2 -1 3 0 4

cat("\nVector U ortogonal a V (Gram-Schmidt):\n")

##   
## Vector U ortogonal a V (Gram-Schmidt):

print(u\_ortogonal)

## [1] 1.5625 -1.8750 2.5625 -1.3125 3.5625

# --- 3. Comprobación de Ortogonalidad ---  
# El producto escalar del resultado con V debe ser 0.  
comprobacion <- sum(u\_ortogonal \* v)  
  
cat("\n--- Comprobación ---\n")

##   
## --- Comprobación ---

cat("Producto Escalar (U\_ortogonal · V): ", comprobacion, "\n")

## Producto Escalar (U\_ortogonal · V): 0

Continuamos …

# Ortogonalziar  
print("Ortogonalizar")

## [1] "Ortogonalizar"

orto <- pc1 - sum(pc1 \* w\_norm) \* w\_norm  
  
orto

## edad ingresos hijos gastos\_medicos ejercicio   
## 0.0017489143 0.9509057323 0.0001120904 0.3094752628 -0.0003603157   
## imc   
## 0.0003645354

## Proyectar

# Proyección sobre el primer componente  
orto <- pc1 - sum(pc1 \* w) \* w  
  
# Si NO quisieras normalizar w, usarías:  
# v2\_star <- pc1 - (sum(pc1 \* w) / sum(w \* w)) \* w  
  
# Normaliza D2 (dirección unitaria)  
D2\_antes <- orto / sqrt(sum(orto^2))  
D2\_antes

## edad ingresos hijos gastos\_medicos ejercicio   
## -0.07482937 0.92748958 -0.04834677 0.30187260 -0.18535374   
## imc   
## -0.07964395

D2 <- as.numeric(X %\*% D2\_antes) # que hace?  
D2

## [1] 14511.35 17444.51 19450.21 15499.16 20407.77 27774.48 29931.11 34869.94  
## [9] 26696.18 28762.27

# Adjuntar al dataset original  
#res <- as.data.frame(D1, D2)  
#res

dimensiones <- data.frame(  
 id = datos$id,  
 clase = clase,  
 D1 = round(D1, 4),  
 D2 = round(D2, 4)  
)  
  
# Dimensiones   
print(dimensiones)

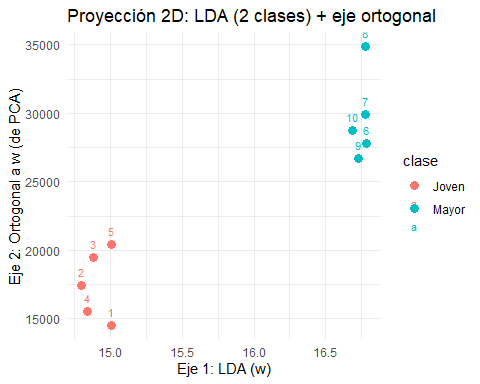
## id clase D1 D2  
## 1 1 Joven 15.0056 14511.35  
## 2 2 Joven 14.8009 17444.51  
## 3 3 Joven 14.8812 19450.21  
## 4 4 Joven 14.8407 15499.16  
## 5 5 Joven 15.0096 20407.77  
## 6 6 Mayor 16.7851 27774.48  
## 7 7 Mayor 16.7779 29931.11  
## 8 8 Mayor 16.7754 34869.94  
## 9 9 Mayor 16.7279 26696.18  
## 10 10 Mayor 16.6882 28762.27

# Visualizar dispersión de dimensiones

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.4.3

ggplot(dimensiones, aes(x = D1, y = D2, color = clase, label = id)) +  
 geom\_point(size = 3) +  
 geom\_text(vjust = -1, size = 3) +  
 labs(x = "Eje 1: LDA (w)", y = "Eje 2: Ortogonal a w (de PCA)",   
 title = "Proyección 2D: LDA (2 clases) + eje ortogonal") +  
 theme\_minimal()



# Comprobando

# Asumo que ya tienes:  
# datos (data.frame), cols <- c("edad","ingresos","hijos","gastos\_medicos","ejercicio","imc")  
X <- as.matrix(datos[, cols])  
# w = tu vector LDA (normalizado o lo normalizamos aquí)  
w <- c(0.346346, -0.000414, 0.219280, -0.000217, 0.837197, 0.362024)  
w <- w / sqrt(sum(w^2)) # asegurar norma 1  
  
# 1) Matriz de covarianzas global: S = cov(X centrado)  
Xc <- scale(X, center = TRUE, scale = FALSE)  
S <- cov(Xc)  
  
# 2) PCA: autovectores de S (cada columna es un PC)  
eig <- eigen(S)  
v1 <- eig$vectors[, 1] # PC1 (dirección de máxima varianza global)  
  
# 3) Ortogonalizar PC1 respecto a w (Gram–Schmidt)  
v2\_star <- v1 - sum(v1 \* w) \* w  
v2 <- v2\_star / sqrt(sum(v2\_star^2)) # normalizar  
  
# (opcional) Alinear el signo para que z2 salga positivo (estético):  
# si quieres que la mayoría de z2 sean positivos, puedes hacer:  
if (sum((X %\*% v2)) < 0) v2 <- -v2  
  
# 4) Proyección  
# a) Si quieres replicar los valores grandes que observaste (del orden de 1e4~3e4):  
z2 <- as.numeric(X %\*% v2) # usando X sin centrar  
# b) Si prefieres proyección centrada alrededor de 0:  
# z2 <- as.numeric(Xc %\*% v2) # usando X centrado  
  
# 5) (Opcional) z1 para tener ambos ejes  
z1 <- as.numeric(X %\*% w)  
  
# 6) Verificación de ortogonalidad (de direcciones, no de proyecciones):  
cat("w·v2 (debe ~0):", sum(w \* v2), "\n")

## w·v2 (debe ~0): 0

# 7) Tabla con las dos coordenadas  
proj2d <- data.frame(id = datos$id, clase = datos$clase,  
 z1\_LDA = round(z1, 4),  
 z2\_aux = round(z2, 2))  
print(proj2d)

## id clase z1\_LDA z2\_aux  
## 1 1 Joven 15.0024 14882.59  
## 2 2 Joven 14.7970 17890.05  
## 3 3 Joven 14.8769 19946.61  
## 4 4 Joven 14.8373 15895.40  
## 5 5 Joven 15.0050 20928.46  
## 6 6 Mayor 16.7791 28482.31  
## 7 7 Mayor 16.7713 30693.61  
## 8 8 Mayor 16.7678 35757.62  
## 9 9 Mayor 16.7220 27376.66  
## 10 10 Mayor 16.6819 29495.12