Mineria de dades: PRA2 - Projecte de mineria de dades

Autor: Xavier Martin Rodriguez

Maig 2025

Contents

PR1

Cargamos el dataset

```
path = 'Ecommerce_Consumer_Behavior_Analysis_Data.csv'
ecommerce <- read.csv(path, row.names=NULL)
```

Planteamiento del problema

En el contexto actual del comercio electrónico, comprender el comportamiento del consumidor es fundamental para optimizar las estrategias de marketing, personalizar la experiencia del usuario y aumentar la tasa de conversión. Las plataformas recopilan cada vez más datos sobre sus usuarios, incluyendo información demográfica, hábitos de navegación, respuestas a campañas promocionales y preferencias de compra.

Con base en el conjunto de datos *"Ecommerce Consumer Behavior Analysis"*, se plantea el siguiente problema analítico principal:

¿Qué factores influyen más en la intención de compra de un usuario?

La pregunta surge del interés por identificar qué variables son más determinantes a la hora de que un usuario decida realizar una compra en línea. A partir de esta cuestión, se derivan los siguientes objetivos analíticos:

Objetivo del análisis

El propósito principal de este estudio es analizar qué variables del comportamiento del consumidor y de su perfil influyen más en la intención o decisión final de compra dentro de una plataforma de comercio electrónico.

Para alcanzar este objetivo general, se definen los siguientes objetivos específicos:

1. Analizar las variables demográficas de los usuarios y su relación con la intención de compra (`Purchase_Intent`), para detectar perfiles con mayor predisposición a comprar.
2. Estudiar el comportamiento de navegación de los usuarios en el sitio web y evaluar su impacto sobre la decisión de compra.
3. Explorar el efecto de factores comerciales y psicológicos, como el uso de descuentos, pertenencia a un programa de fidelización o interacción con anuncios en el proceso de decisión.
4. Identificar patrones comunes mediante técnicas de clustering no supervisado, que permitan agrupar usuarios en segmentos basados en su comportamiento o perfil.
5. Reducir la dimensionalidad del conjunto de datos con Análisis de Componentes Principales (PCA) para facilitar la visualización, interpretación y preparación de los datos para modelos predictivos en la segunda parte de la práctica (PRA2).

Comprension de datos‘

El conjunto de datos utilizado contiene información sobre usuarios que han interactuado con una plataforma de comercio electrónico. Está compuesto por un total de 1.000 observaciones (usuarios únicos) y 28 variables, que recogen aspectos tanto del perfil demográfico del cliente como de su comportamiento de compra, navegación, sensibilidad al precio y engagement con la publicidad.

Origen del conjunto de datos

El conjunto de datos utilizado en este estudio proviene de la plataforma Kaggle, un repositorio ampliamente reconocido por la comunidad científica y profesional por ofrecer datasets de calidad para proyectos de análisis de datos y aprendizaje automático.

El dataset se titula *“Ecommerce Consumer Behavior Analysis Data”*, y ha sido recopilado con el propósito de analizar y comprender los factores que influyen en el comportamiento de compra de usuarios en plataformas de comercio electrónico.

Este tipo de información es de gran utilidad para departamentos de marketing digital, analistas de negocio y desarrolladores de sistemas de recomendación, ya que permite optimizar campañas, personalizar la experiencia del usuario y predecir patrones de conversión.

web: https://www.kaggle.com/datasets/salahuddinahmedshuvo/ecommerce-consumer-behavior-analysis-data

Características del origen

Fuente: Kaggle (autor: Salah Uddin Ahmed Shuvo)

Tipo de datos: Datos simulados, estructurados, limpios y etiquetados.

Formato original: Archivo .csv, con codificación estándar UTF-8.

Licencia: Uso libre para fines educativos y de investigación, sin restricciones comerciales.

Justificación del uso del conjunto de datos

Este conjunto de datos ha sido seleccionado por su idoneidad tanto para tareas de análisis supervisado como no supervisado, tal como se requiere en esta práctica. Su estructura, contenido y volumen permiten aplicar de forma práctica y razonada los pasos del proceso CRISP-DM.

A continuación, se detallan los motivos principales que justifican su elección:

1. Alineación con el problema analítico planteado

El conjunto de datos recoge información completa sobre distintos aspectos del comportamiento de usuarios en un entorno de comercio electrónico. Esto se relaciona directamente con la pregunta principal del estudio:

¿Qué factores influyen más en la intención o decisión de compra de un usuario?

Las variables incluidas permiten explorar múltiples dimensiones del comportamiento: perfil demográfico, uso de descuentos, lealtad a la marca, interacción con publicidad, tiempo de decisión, etc.

2. Cumplimiento de los requisitos técnicos exigidos

Requisitos:

500 observaciones - Contiene 1.000 registros.

5 variables numéricas - Edad, monto de compra, tiempo de investigación, frecuencia…

2 variables categóricas - Género, canal de compra, nivel educativo…

1 variable binaria - Discount_Used, Customer_Loyalty_Program_Member.

Permite tareas supervisadas - purchase_Intent o una binarización de Purchase_Amount.

Permite tareas no supervisadas - segmentación de usuarios por perfil y comportamiento (clustering).

3. Representatividad y riqueza del contenido

El dataset permite cubrir una gran variedad de situaciones reales en comercio electrónico. Además, incluye variables con escalas diversas (ordinales, categóricas, binarias, continuas), lo que lo convierte en un caso completo para aplicar todo el flujo de minería de datos.

4. Posibilidades de aplicación futura (fase de modelado)

Gracias a la diversidad de variables y su calidad, este dataset permitirá en la segunda parte de la práctica (PRA2) aplicar técnicas como:

Clasificación de la intención de compra.

Segmentación de usuarios con clustering.

Modelos explicativos con variables seleccionadas por PCA.

Exploración del dataset

Ver las primeras filas del dataset

```
head(ecommerce)
```

```
##   Customer_ID Age Gender Income_Level Marital_Status Education_Level Occupation
## 1 37-611-6911  22 Female       Middle        Married      Bachelor's     Middle
## 2 29-392-9296  49   Male         High        Married     High School       High
## 3 84-649-5117  24 Female       Middle         Single        Master's       High
## 4 48-980-6078  29 Female       Middle         Single        Master's     Middle
```

```
## 5 91-170-9072  33 Female       Middle        Widowed     High School     Middle
## 6 82-561-4233  45   Male       Middle        Married        Master's       High
##     Location    Purchase_Category Purchase_Amount Frequency_of_Purchase
## 1       Évry Gardening & Outdoors        $333.80                     4
## 2   Huocheng     Food & Beverages        $222.22                    11
## 3     Huzhen      Office Supplies        $426.22                     2
## 4     Wiwilí      Home Appliances        $101.31                     6
## 5       Nara            Furniture        $211.70                     6
## 6 Boro Utara      Office Supplies        $487.95                     8
##   Purchase_Channel Brand_Loyalty Product_Rating
## 1            Mixed             5              5
## 2         In-Store             3              1
## 3            Mixed             5              5
## 4            Mixed             3              1
## 5            Mixed             3              4
## 6            Mixed             3              3
##   Time_Spent_on_Product_Research.hours. Social_Media_Influence
## 1                                   2.0                  None
## 2                                   2.0                Medium
## 3                                   0.3                   Low
## 4                                   1.0                  High
## 5                                   0.0                Medium
## 6                                   0.0                  High
##   Discount_Sensitivity Return_Rate Customer_Satisfaction Engagement_with_Ads
## 1   Somewhat Sensitive           1                     7                None
## 2        Not Sensitive           1                     5                High
## 3        Not Sensitive           1                     7                 Low
## 4   Somewhat Sensitive           0                     1                None
## 5        Not Sensitive           2                    10                None
## 6        Not Sensitive           2                     3                None
##   Device_Used_for_Shopping Payment_Method Time_of_Purchase Discount_Used
## 1                   Tablet    Credit Card         3/1/2024          TRUE
## 2                   Tablet         PayPal        4/16/2024          TRUE
## 3               Smartphone     Debit Card        3/15/2024          TRUE
## 4               Smartphone          Other        10/4/2024          TRUE
## 5               Smartphone     Debit Card        1/30/2024         FALSE
## 6                   Tablet     Debit Card        3/19/2024         FALSE
##   Customer_Loyalty_Program_Member Purchase_Intent Shipping_Preference
## 1                           FALSE      Need-based       No Preference
## 2                           FALSE     Wants-based            Standard
## 3                            TRUE       Impulsive       No Preference
## 4                            TRUE      Need-based             Express
## 5                           FALSE     Wants-based       No Preference
## 6                           FALSE         Planned       No Preference
##   Time_to_Decision
## 1                2
## 2                6
## 3                3
## 4               10
## 5                4
## 6                7
```

Ver un resumen estadístico general

```
summary(ecommerce)
```

```
##  Customer_ID             Age           Gender           Income_Level
##  Length:1000        Min.   :18.0   Length:1000        Length:1000
##  Class :character   1st Qu.:26.0   Class :character   Class :character
##  Mode  :character   Median :34.5   Mode  :character   Mode  :character
##                     Mean   :34.3
##                     3rd Qu.:42.0
##                     Max.   :50.0
##  Marital_Status     Education_Level     Occupation          Location
##  Length:1000        Length:1000        Length:1000        Length:1000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Purchase_Category  Purchase_Amount    Frequency_of_Purchase Purchase_Channel
##  Length:1000        Length:1000        Min.   : 2.000        Length:1000
##  Class :character   Class :character   1st Qu.: 4.000        Class :character
##  Mode  :character   Mode  :character   Median : 7.000        Mode  :character
##                                        Mean   : 6.945
##                                        3rd Qu.:10.000
##                                        Max.   :12.000
##  Brand_Loyalty   Product_Rating  Time_Spent_on_Product_Research.hours.
##  Min.   :1.000   Min.   :1.000   Min.   :0.000
##  1st Qu.:2.000   1st Qu.:2.000   1st Qu.:0.000
##  Median :3.000   Median :3.000   Median :1.000
##  Mean   :3.026   Mean   :3.033   Mean   :1.013
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:2.000
##  Max.   :5.000   Max.   :5.000   Max.   :2.000
##  Social_Media_Influence Discount_Sensitivity  Return_Rate
##  Length:1000            Length:1000          Min.   :0.000
##  Class :character       Class :character     1st Qu.:0.000
##  Mode  :character       Mode  :character     Median :1.000
##                                              Mean   :0.954
##                                              3rd Qu.:2.000
##                                              Max.   :2.000
##  Customer_Satisfaction Engagement_with_Ads Device_Used_for_Shopping
##  Min.   : 1.000        Length:1000         Length:1000
##  1st Qu.: 3.000        Class :character    Class :character
##  Median : 5.000        Mode  :character    Mode  :character
##  Mean   : 5.399
##  3rd Qu.: 8.000
##  Max.   :10.000
##  Payment_Method     Time_of_Purchase   Discount_Used
##  Length:1000        Length:1000        Mode :logical
##  Class :character   Class :character   FALSE:479
##  Mode  :character   Mode  :character   TRUE :521
##
##
##
```

```
##  Customer_Loyalty_Program_Member Purchase_Intent    Shipping_Preference
##  Mode :logical                   Length:1000        Length:1000
##  FALSE:509                       Class :character   Class :character
##  TRUE :491                       Mode  :character   Mode  :character
##
##
##
##  Time_to_Decision
##  Min.   : 1.000
##  1st Qu.: 4.000
##  Median : 8.000
##  Mean   : 7.547
##  3rd Qu.:11.000
##  Max.   :14.000
```

Ver la estructura del dataset

```
str(ecommerce)
```

```
## 'data.frame':    1000 obs. of  28 variables:
##  $ Customer_ID                          : chr  "37-611-6911" "29-392-9296" "84-649-5117" "48-980-6078
##  $ Age                                  : int  22 49 24 29 33 45 21 39 24 25 ...
##  $ Gender                               : chr  "Female" "Male" "Female" "Female" ...
##  $ Income_Level                         : chr  "Middle" "High" "Middle" "Middle" ...
##  $ Marital_Status                       : chr  "Married" "Married" "Single" "Single" ...
##  $ Education_Level                      : chr  "Bachelor's" "High School" "Master's" "Master's" ...
##  $ Occupation                           : chr  "Middle" "High" "High" "Middle" ...
##  $ Location                             : chr  "Évry" "Huocheng" "Huzhen" "Wiwilí" ...
##  $ Purchase_Category                    : chr  "Gardening & Outdoors" "Food & Beverages" "Office Supp
##  $ Purchase_Amount                      : chr  "$333.80 " "$222.22 " "$426.22 " "$101.31 " ...
##  $ Frequency_of_Purchase                : int  4 11 2 6 6 8 12 6 8 7 ...
##  $ Purchase_Channel                     : chr  "Mixed" "In-Store" "Mixed" "Mixed" ...
##  $ Brand_Loyalty                        : int  5 3 5 3 3 3 2 5 3 2 ...
##  $ Product_Rating                       : int  5 1 5 1 4 3 5 4 5 5 ...
##  $ Time_Spent_on_Product_Research.hours.: num  2 2 0.3 1 0 0 1 1 0 1 ...
##  $ Social_Media_Influence               : chr  "None" "Medium" "Low" "High" ...
##  $ Discount_Sensitivity                 : chr  "Somewhat Sensitive" "Not Sensitive" "Not Sensitive"
##  $ Return_Rate                          : int  1 1 1 0 2 2 0 2 1 1 ...
##  $ Customer_Satisfaction                : int  7 5 7 1 10 3 9 9 2 5 ...
##  $ Engagement_with_Ads                  : chr  "None" "High" "Low" "None" ...
##  $ Device_Used_for_Shopping             : chr  "Tablet" "Tablet" "Smartphone" "Smartphone" ...
##  $ Payment_Method                       : chr  "Credit Card" "PayPal" "Debit Card" "Other" ...
##  $ Time_of_Purchase                     : chr  "3/1/2024" "4/16/2024" "3/15/2024" "10/4/2024" ...
##  $ Discount_Used                        : logi  TRUE TRUE TRUE TRUE FALSE FALSE ...
##  $ Customer_Loyalty_Program_Member      : logi  FALSE FALSE TRUE TRUE FALSE FALSE ...
##  $ Purchase_Intent                      : chr  "Need-based" "Wants-based" "Impulsive" "Need-based" .
##  $ Shipping_Preference                  : chr  "No Preference" "Standard" "No Preference" "Express"
##  $ Time_to_Decision                     : int  2 6 3 10 4 7 13 13 7 13 ...
```

Número de filas y columnas

```
cat("Número de observaciones:", nrow(ecommerce), "\n")
```

```
## Número de observaciones: 1000
```

```
cat("Número de variables:", ncol(ecommerce), "\n")
```

```
## Número de variables: 28
```

Nombres de todas las columnas

```
colnames(ecommerce)
```

```
##  [1] "Customer_ID"
##  [2] "Age"
##  [3] "Gender"
##  [4] "Income_Level"
##  [5] "Marital_Status"
##  [6] "Education_Level"
##  [7] "Occupation"
##  [8] "Location"
##  [9] "Purchase_Category"
## [10] "Purchase_Amount"
## [11] "Frequency_of_Purchase"
## [12] "Purchase_Channel"
## [13] "Brand_Loyalty"
## [14] "Product_Rating"
## [15] "Time_Spent_on_Product_Research.hours."
## [16] "Social_Media_Influence"
## [17] "Discount_Sensitivity"
## [18] "Return_Rate"
## [19] "Customer_Satisfaction"
## [20] "Engagement_with_Ads"
## [21] "Device_Used_for_Shopping"
## [22] "Payment_Method"
## [23] "Time_of_Purchase"
## [24] "Discount_Used"
## [25] "Customer_Loyalty_Program_Member"
## [26] "Purchase_Intent"
## [27] "Shipping_Preference"
## [28] "Time_to_Decision"
```

Crear un resumen con nombre, tipo de dato y número de valores únicos

```
variable_summary <- data.frame(
  Variable = colnames(ecommerce),
```

```
  Tipo = sapply(ecommerce, class),
  Valores_unicos = sapply(ecommerce, function(x) length(unique(x))),
  row.names = NULL
)

# Instalar y cargar knitr si no está instalado
if (!require(knitr)) {
  install.packages("knitr")
  library(knitr)
} else {
  library(knitr)
}

# Mostrar la tabla formateada (solo esta línea imprime)
kable(variable_summary, caption = "Resumen de variables del dataset")
```

Table 1: Resumen de variables del dataset

Variable	Tipo	Valores_unicos
Customer_ID	character	1000
Age	integer	33
Gender	character	8
Income_Level	character	2
Marital_Status	character	4
Education_Level	character	3
Occupation	character	2
Location	character	969
Purchase_Category	character	24
Purchase_Amount	character	989
Frequency_of_Purchase	integer	11
Purchase_Channel	character	3
Brand_Loyalty	integer	5
Product_Rating	integer	5
Time_Spent_on_Product_Research.hours.	numeric	12
Social_Media_Influence	character	4
Discount_Sensitivity	character	3
Return_Rate	integer	3
Customer_Satisfaction	integer	10
Engagement_with_Ads	character	4
Device_Used_for_Shopping	character	3
Payment_Method	character	5
Time_of_Purchase	character	344
Discount_Used	logical	2
Customer_Loyalty_Program_Member	logical	2
Purchase_Intent	character	4
Shipping_Preference	character	3
Time_to_Decision	integer	14

Explicación de las varfiables del dataset

Customer_ID:

Tipo: Categórica nominal (ID)

Descripción: Identificador único para cada cliente.

Observaciones: No tiene valor analítico directo. Se usará solo como referencia o para filtrados.

Age:

Tipo: Numérica continua

Descripción: Edad del usuario en años.

Observaciones: Ideal para segmentar perfiles de edad. Puede discretizarse en rangos.

Gender:

Tipo: Categórica nominal

Descripción: Género del usuario (Female, Male, Bigender, Agender, etc.).

Observaciones: Hay categorías con pocos registros → podría reagruparse en "Otros".

Income_Level:

Tipo: Categórica ordinal

Descripción: Nivel de ingresos del usuario (Low, Middle, High).

Observaciones: Puede ordenarse. Útil para análisis de gasto y segmentación.

Marital_Status:

Tipo: Categórica nominal

Descripción: Estado civil del usuario (Single, Married, Divorced, Widowed).

Observaciones: Podría relacionarse con frecuencia de compra o tipo de producto.

Education_Level:

Tipo: Categórica ordinal

Descripción: Nivel educativo del usuario (High School, Bachelor's, Master's).

Observaciones: Puede ordenarse. Relevante para análisis sociodemográfico.

Occupation:

Tipo: Categórica

Descripción: Nivel o tipo de ocupación (Low, Middle, High).

Observaciones: Similar a ingresos. Puede ser redundante o combinarse.

Location:

Tipo: Categórica nominal (con 969 valores únicos)

Descripción: Ciudad o área del usuario.

Observaciones: No es útil directamente para análisis → demasiados niveles. Requiere agrupación o eliminación.

Purchase_Category:

Tipo: Categórica nominal

Descripción: Tipo de producto comprado (Electronics, Furniture, etc.).

Observaciones: Útil para segmentación de intereses y preferencias de compra.

Purchase_Amount:

Tipo: Numérica continua

Descripción: Monto total de la compra en dólares.

Observaciones: Limpiada previamente ($ eliminado). Puede analizarse directamente o discretizarse por rangos.

Frequency_of_Purchase:

Tipo: Numérica discreta

Descripción: Número total de compras realizadas por el usuario.

Observaciones: Puede reflejar el nivel de actividad o fidelización del cliente. Muy útil para segmentar clientes frecuentes vs. ocasionales.

Purchase_Channel:

Tipo: Categórica nominal

Descripción: Canal desde el cual se realizó la compra (Web, Mobile, Mixed).

Observaciones: Útil para analizar comportamientos por canal y orientar estrategias multicanal.

Brand_Loyalty:

Tipo: Ordinal (escala 1–5)

Descripción: Nivel de lealtad del usuario hacia la marca.

Observaciones: Directamente útil como predictor de repetición de compra. Puede correlacionarse con Frequency_of_Purchase.

Product_Rating:

Tipo: Ordinal (escala 1–5)

Descripción: Puntuación que el cliente da al producto adquirido.

Observaciones: Útil para medir satisfacción puntual y calidad percibida del producto.

Time_Spent_on_Product_Research(hours):

Tipo: Numérica continua

Descripción: Horas que el usuario dedicó a investigar productos antes de comprar.

Observaciones: Puede relacionarse con el tipo de producto o la indecisión del cliente.

Social_Media_Influence:

Tipo: Categórica ordinal

Descripción: Grado de influencia de redes sociales sobre la decisión de compra (Low, Medium, High, None).

Observaciones: Ideal para entender perfiles influenciables por contenido digital.

Discount_Sensitivity:

Tipo: Categórica ordinal

Descripción: Nivel de sensibilidad del cliente frente a descuentos (Low, Moderate, Very Sensitive).

Observaciones: Puede ser clave en campañas de pricing personalizado.

Return_Rate:

Tipo: Numérica discreta

Descripción: Número de veces que el cliente ha devuelto productos.

Observaciones: Indicador de insatisfacción o comportamiento exigente.

Customer_Satisfaction:

Tipo: Ordinal (escala 1–10)

Descripción: Valoración general del cliente sobre su experiencia.

Observaciones: Excelente variable para correlaciones y validación de segmentación.

Engagement_with_Ads:

Tipo: Categórica ordinal

Descripción: Grado de interacción con publicidad (None, Low, Medium, High).

Observaciones: Importante para predecir receptividad a marketing digital.

Device_Used_for_Shopping:

Tipo: Categórica nominal

Descripción: Dispositivo usado por el usuario para realizar la compra (Desktop, Mobile, Tablet).

Observaciones: Útil para analizar diferencias de comportamiento por canal. Puede relacionarse con edad o gasto.

Payment_Method;

Tipo: Categórica nominal

Descripción: Método de pago elegido (Credit Card, Debit Card, PayPal, Other, etc.).

Observaciones: Puede ayudar a detectar hábitos de compra o limitaciones tecnológicas. No es predictor directo, pero puede complementar análisis.

Time_of_Purchase:

Tipo: Categórica (fecha como texto)

Descripción: Fecha de la compra realizada.

Observaciones: Requiere conversión a tipo fecha (as.Date()). Se puede derivar el mes, día de la semana o estacionalidad para nuevos análisis.

Discount_Used:

Tipo: Binaria (TRUE/FALSE)

Descripción: Indica si el cliente usó un descuento en la compra.

Observaciones: Variable muy útil como factor explicativo en decisiones de compra. También buena para clustering.

Customer_Loyalty_Program_Member:

Tipo: Binaria (TRUE/FALSE)

Descripción: Indica si el usuario es miembro de un programa de fidelización.

Observaciones: Puede explicar la frecuencia de compra o el gasto. Clave para segmentación.

Purchase_Intent:

Tipo: Categórica

Descripción: Tipo de intención de compra (Need-based, Wants-based, Impulsive, etc.).

Observaciones: Es una candidata a variable objetivo. También puede binarizarse para modelos supervisados.

Shipping_Preference:

Tipo: Categórica nominal

Descripción: Preferencia del cliente sobre el tipo de envío (Standard, Express, No Preference).

Observaciones: Puede relacionarse con urgencia, tipo de producto o perfil de cliente.

Time_to_Decision:

Tipo: Numérica discreta

Descripción: Tiempo (en días o unidades) que tarda el usuario en tomar la decisión de compra.

Observaciones: Muy útil para estudiar indecisión o urgencia. Puede correlacionarse con Purchase_Category o Research Time.

Técnicas que se aplicarán en la práctica

Clustering: para identificar perfiles de usuario según comportamiento.

- **PCA**: para simplificar el conjunto de variables numéricas.
- **Preparación para clasificación supervisada** (en PRA2): predicción de `Purchase_Intent`.

Exploración de datos

Detectar valores nulos por columna

```
nulos <- sapply(ecommerce, function(x) sum(is.na(x)))
nulos_df <- data.frame(Variable = names(nulos), Nulos = as.numeric(nulos))
nulos_df <- nulos_df[nulos_df$Nulos > 0, ]

library(knitr)
cat(" Variables con valores nulos:\n")
```

```
##  Variables con valores nulos:
```

```
if (nrow(nulos_df) == 0) {
  cat("No hay valores nulos en el dataset.\n")
} else {
  kable(nulos_df, caption = "Variables con valores nulos")
}
```

```
## No hay valores nulos en el dataset.
```

Variables con una sola categoría

```
# Crear resumen de niveles únicos por variable
niveles_df <- data.frame(
  Variable = colnames(ecommerce),
  Tipo = sapply(ecommerce, class),
  Niveles_unicos = sapply(ecommerce, function(x) length(unique(x)))
```

```
)

# VARIABLES CON BAJA VARIABILIDAD (SOLO 1 CATEGORÍA)
baja_var <- niveles_df[niveles_df$Niveles_unicos == 1, ]

cat(" Variables con una sola categoría:\n")
```

```
##  Variables con una sola categoría:
```

```
if (nrow(baja_var) == 0) {
  cat("Todas las variables tienen más de un valor.\n")
} else {
  kable(baja_var, caption = "Variables sin variabilidad (no aportan información)")
}
```

```
## Todas las variables tienen más de un valor.
```

Duplicados en el dataset

```
# REGISTROS DUPLICADOS
duplicados <- sum(duplicated(ecommerce))

cat(" Total de registros duplicados en el dataset:", duplicados, "\n")
```

```
##  Total de registros duplicados en el dataset: 0
```

Resumen general del conjunto de datos

El dataset utilizado en este estudio contiene información sobre usuarios que han interactuado con una plataforma de comercio electrónico. Ha sido obtenido desde Kaggle y presenta un formato estructurado y limpio, ideal para análisis exploratorio y aplicación de técnicas de minería de datos.

Estructura del dataset

Número total de observaciones: 1.000

Número total de variables: 28 columnas

Tipo de archivo: CSV, con codificación UTF-8

Calidad de los datos

- *Valores nulos:* No se detectaron valores nulos en el dataset.

-*Registros duplicados:* No se encontraron registros duplicados.

- *Variables sin variabilidad:* Todas las variables presentan más de una categoría (no hay variables constantes).

- *Variables con muchos niveles únicos:*

```
Location presenta 969 niveles → se considera irrelevante para el análisis y
                                será descartada o agrupada.

Variables limpias: La variable Purchase_Amount fue convertida correctamente a
                   formato numérico eliminando el símbolo $.
```

Variables clave para el análisis

- *Perfil de usuario:*

-Age -Gender -Income_Level -Education_Level

- *Comportamiento en la plataforma:*

-Frequency_of_Purchase -Purchase_Amount -Time_Spent_on_Product_Research(hours) -Time_to_Decision

- *Factores comerciales:*

-Discount_Used -Customer_Loyalty_Program_Member -Engagement_with_Ads -Purchase_Channel -Purchase_Intent (como posible variable objetivo)

Visualización previa al clustering (pre-PRA2)

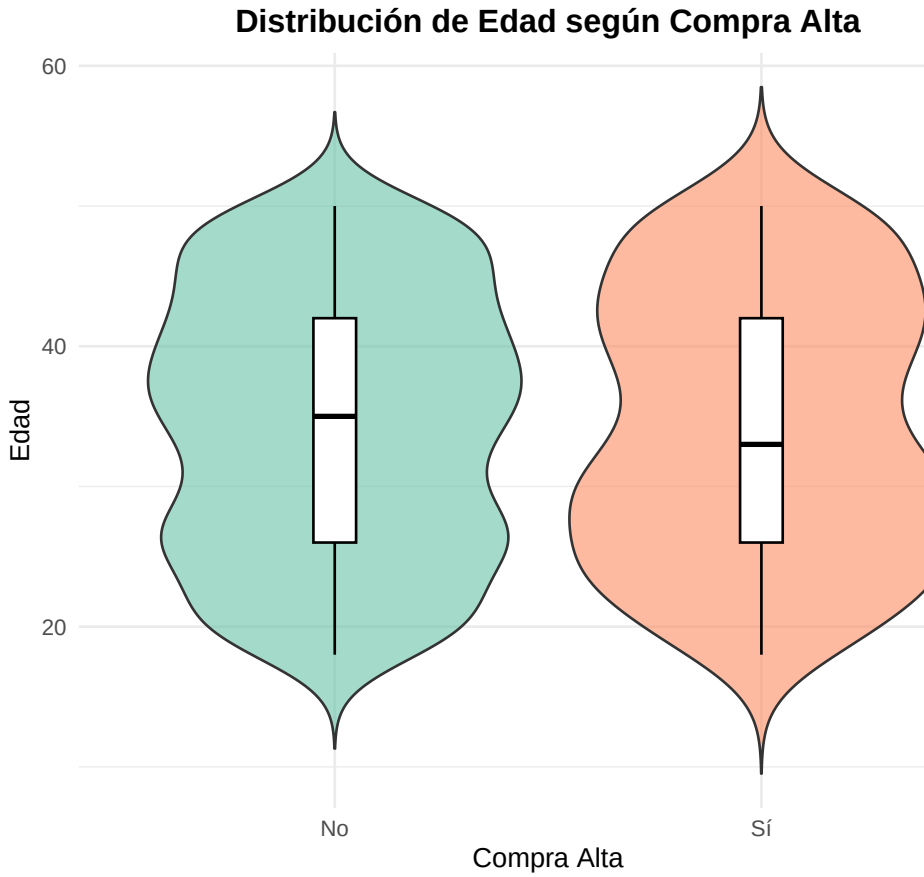
Con el objetivo de analizar la relación entre las variables de perfil de usuario y el nivel de compra (Compra_Alta), se realiza una visualización separada según el tipo de variable:

Para variables numéricas (Age), se emplea una matriz de dispersión (ggpairs) que permite explorar correlaciones.

Para variables categóricas (Gender, Income_Level, Education_Level), se utilizan gráficos de barras (geom_bar), facilitando la comparación de proporciones de compra alta y no alta.

```r
library(ggplot2)
library(dplyr)

# Crear datos_limpios con la variable Compra_Alta incluida
datos_limpios <- ecommerce %>%
  select(-Customer_ID, -Location) %>%
  mutate(
    Purchase_Amount = as.numeric(gsub("\\$", "", Purchase_Amount)),
    Time_of_Purchase = as.Date(Time_of_Purchase, format = "%m/%d/%Y"),
    Compra_Alta = ifelse(Purchase_Amount > quantile(Purchase_Amount, 0.75, na.rm = TRUE), "Sí", "No")
  )

# Crear gráfico violín
ggplot(datos_limpios, aes(x = Compra_Alta, y = Age, fill = Compra_Alta)) +
  geom_violin(trim = FALSE, alpha = 0.6) +
```

```r
  geom_boxplot(width = 0.1, color = "black", fill = "white", outlier.size = 1) +
  scale_fill_brewer(palette = "Set2") +
  labs(
    title = "Distribución de Edad según Compra Alta",
    x = "Compra Alta",
    y = "Edad"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```



Análisis de la variable numérica: Age

Se ha utilizado un gráfico de violín para comparar la distribución de la edad (Age) entre los usuarios que realizaron una compra alta (Compra_Alta = Sí) y aquellos que no (Compra_Alta = No).

El gráfico muestra que la edad media es algo más baja entre los compradores con gasto alto, y que la distribución es más dispersa en este grupo. En cambio, los usuarios que no realizan compras altas tienden a concentrarse en un rango más estrecho alrededor de los 35–45 años.

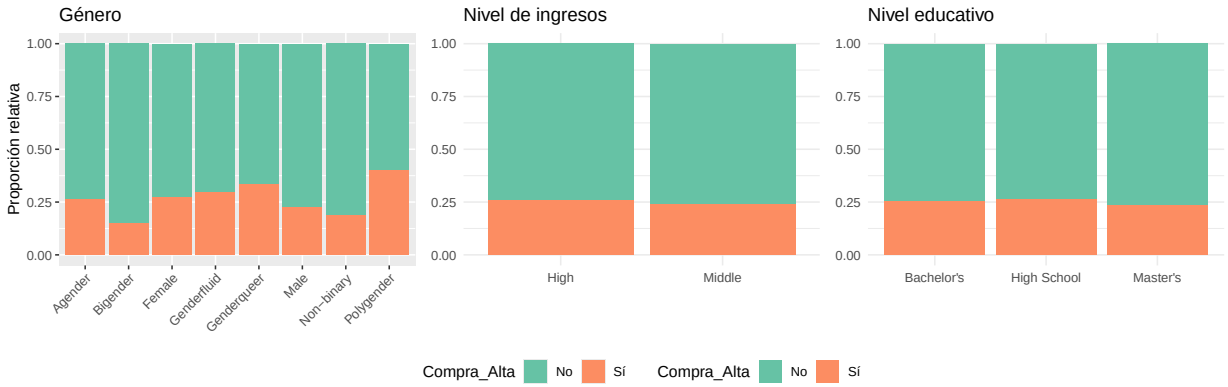
Esta visualización es útil para detectar posibles patrones de comportamiento relacionados con la edad, lo cual puede resultar relevante para la segmentación de clientes y los modelos predictivos de la PRA2.

Análisis conjunto de variables categóricas del perfil de usuario *OBJETIVO:*

Se han combinado en un solo panel las variables categóricas de perfil de usuario:
`Gender`, *`Income_Level`* y *`Education_Level`*, usando `ggplot2` y la librería `patchwork`.

Este formato permite comparar visualmente cómo varía la proporción de **compras altas** (`Compra_Alta = Sí`) entre las distintas categorías, de manera clara y sin redundancia visual.

```r
# Instalar y cargar librerías necesarias
if (!require(patchwork)) install.packages("patchwork")
library(patchwork)
library(ggplot2)

# Gráfico 1: Género
g1 <- ggplot(datos_limpios, aes(x = Gender, fill = Compra_Alta)) +
  geom_bar(position = "fill") +
  labs(title = "Género", y = "Proporción relativa", x = NULL) +
  scale_fill_brewer(palette = "Set2") +
 theme(axis.text.x = element_text(angle = 45, hjust = 1))

# Gráfico 2: Nivel de ingresos
g2 <- ggplot(datos_limpios, aes(x = Income_Level, fill = Compra_Alta)) +
  geom_bar(position = "fill") +
  labs(title = "Nivel de ingresos", y = NULL, x = NULL) +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal()

# Gráfico 3: Nivel educativo
g3 <- ggplot(datos_limpios, aes(x = Education_Level, fill = Compra_Alta)) +
  geom_bar(position = "fill") +
  labs(title = "Nivel educativo", y = NULL, x = NULL) +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal()

# Mostrar los tres gráficos en una fila con leyenda común
g1 + g2 + g3 + plot_layout(ncol = 3, guides = "collect") & theme(legend.position = "bottom")
```



- **Género**:

La distribución de compras altas es relativamente uniforme entre hombres y mujeres, aunque existen algunas diferencias sutiles en categorías menos frecuentes como *Agender* o *Bigender*.

- **Nivel de ingresos**:

  Se observa una mayor proporción de compras altas entre usuarios con ingresos *High*, en comparación con los niveles *Low* o *Middle*.

- **Nivel educativo**:

  Las compras elevadas tienden a concentrarse más entre usuarios con educación superior (*Bachelor's* y *Master's*), lo que podría indicar una relación entre nivel educativo y capacidad adquisitiva.

Análisis de conjunto de Variables de comportamiento

OBJETIVO:

-Analizar cómo varía el comportamiento de los usuarios en la plataforma en función de si hicieron o no una Compra Alta (Compra_Alta = Sí/No).

-Se representan las variables numéricas más relevantes del comportamiento de los usuarios en la plataforma:

-Frequency_of_Purchase: frecuencia de compra

-Purchase_Amount: monto total de compra

-Time_Spent_on_Product_Research.hours.: tiempo dedicado a investigar productos

-Time_to_Decision: tiempo que tardan en decidir

-El comportamiento en la plataforma puede influir directamente en la probabilidad de realizar una compra alta:

-Usuarios que compran frecuentemente pueden acumular más gasto.

-Quienes investigan más o tardan más en decidir podrían mostrar patrones de compra diferentes.

-El análisis exploratorio ayudará a identificar comportamientos de alto valor, útiles para segmentación o predicción futura.

-Se utilizará un gráfico combinado de:

-Violin plots para observar la forma completa de la distribución (densidad).

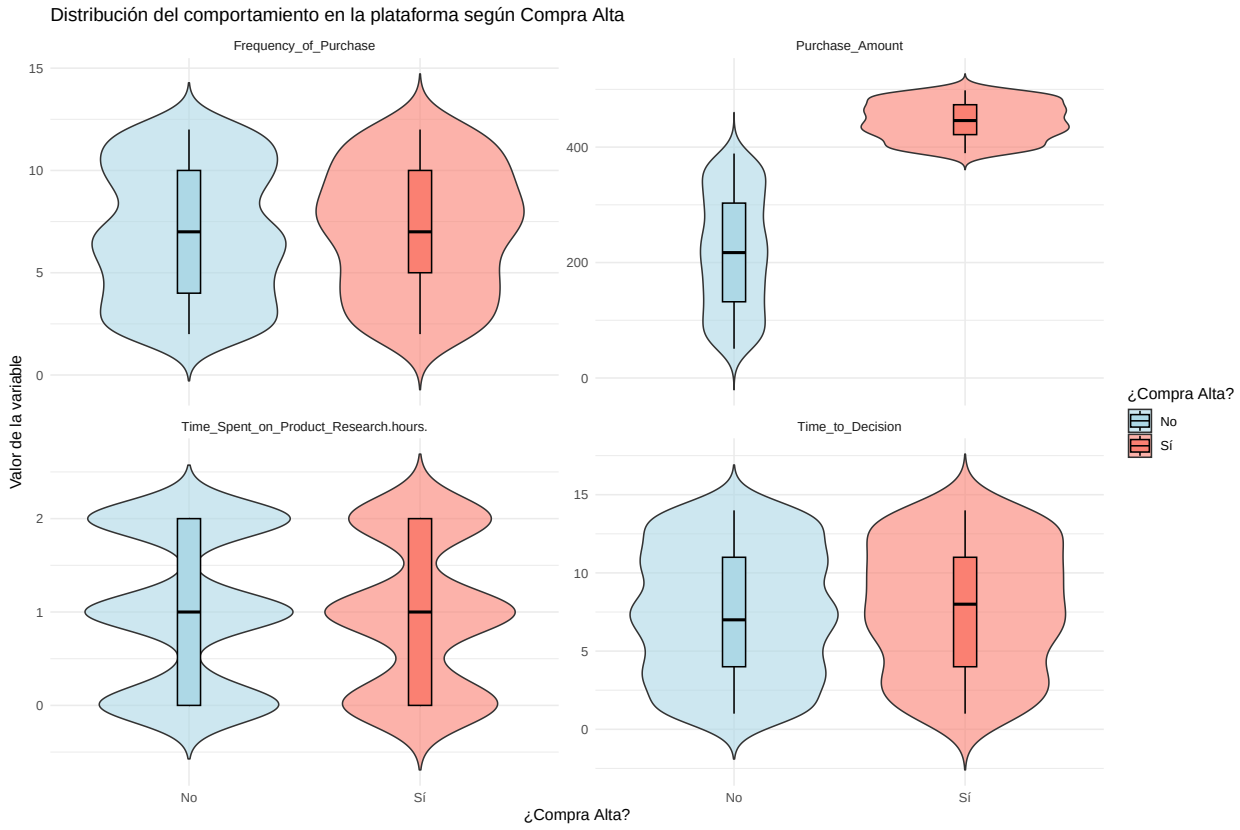
-Boxplots superpuestos para destacar mediana, cuartiles y valores atípicos.

-Cada variable se representará en un panel distinto mediante facet_wrap.

-Los datos se transformarán al formato "largo" (pivot_longer) para poder graficar varias variables a la vez de forma ordenada.

```r
# Cargar librerías necesarias
library(ggplot2)
library(dplyr)
library(tidyr)

# 1. Selección de variables de comportamiento relevantes
comportamiento <- datos_limpios %>%
  select(Compra_Alta,
         Frequency_of_Purchase,
         Purchase_Amount,
         Time_Spent_on_Product_Research.hours.,
         Time_to_Decision)

# 2. Reestructurar datos a formato largo
comportamiento_long <- comportamiento %>%
  pivot_longer(-Compra_Alta, names_to = "Variable", values_to = "Valor")

# 3. Crear gráfico violin + boxplot facetado
ggplot(comportamiento_long, aes(x = Compra_Alta, y = Valor, fill = Compra_Alta)) +
  geom_violin(trim = FALSE, alpha = 0.6) +
  geom_boxplot(width = 0.1, color = "black", outlier.size = 0.5) +
  facet_wrap(~ Variable, scales = "free_y", ncol = 2) +
  scale_fill_manual(values = c("lightblue", "salmon"),
```

```
                    name = "¿Compra Alta?",
                    labels = c("No", "Sí")) +
labs(title = "Distribución del comportamiento en la plataforma según Compra Alta",
     x = "¿Compra Alta?",
     y = "Valor de la variable") +
theme_minimal()
```



Que resultado se obtiene?

- Se generan **cuatro gráficos violin + boxplot**, uno por cada variable.
- Se comparan las distribuciones de cada variable entre usuarios con **Compra Alta = Sí** y **Compra Alta = No**.
- Cada panel permite identificar diferencias en la dispersión, tendencia y concentración de valores.

Interpretación del resultado

- **Frequency_of_Purchase:**

Se observa que los usuarios que realizan una Compra Alta tienden a comprar con una frecuencia ligeramente mayor, aunque las diferencias no son muy extremas.

- **Purchase_Amount:**

Como era esperado (ya que la variable se utilizó para definir Compra Alta), los usuarios que realizaron una compra alta tienen montos de compra significativamente más altos.

- **Time_Spent_on_Product_Research:**

No se detectan diferencias notables en el tiempo de investigación entre ambos grupos, lo que sugiere que invertir más tiempo en investigar no necesariamente lleva a compras mayores.

- **Time_to_Decision:**

Los usuarios de Compra Alta no muestran tiempos de decisión sustancialmente distintos a los de Compra No Alta.

Conclusión:

El comportamiento de frecuencia de compra y el monto de compra son factores más claramente diferenciadores entre usuarios de alto y bajo gasto, mientras que el tiempo invertido en investigación o decisión no parece tener una influencia determinante.

Anlanisis de conjunto de Variables de Factores comerciales

OBJETIVO:

El objetivo de esta sección es analizar cómo se distribuyen las principales variables comerciales en función de si el usuario realizó una Compra Alta (Compra_Alta = "Sí") o no.

Las variables estudiadas son:

Discount_Used: Si el usuario utilizó o no un descuento en su compra.

Customer_Loyalty_Program_Member: Pertenencia a un programa de fidelización.

Engagement_with_Ads: Grado de interacción con anuncios.

Purchase_Channel: Canal utilizado para la compra (Online, In-Store, etc.).

Estas variables representan comportamientos comerciales que pueden estar asociados a un mayor nivel de gasto. Entender su distribución ayudará a:

-Detectar factores que influyen en compras elevadas.

-Seleccionar variables relevantes para futuros modelos de clasificación o segmentación.

-Mejorar el conocimiento del perfil de comprador de alto valor.

-Se creará un gráfico de barras facetado (facet_wrap) que mostrará:

-La distribución proporcional (geom_bar(position = "fill")) de usuarios que realizaron o no una compra alta (Compra_Alta).

-Un panel distinto para cada variable comercial.

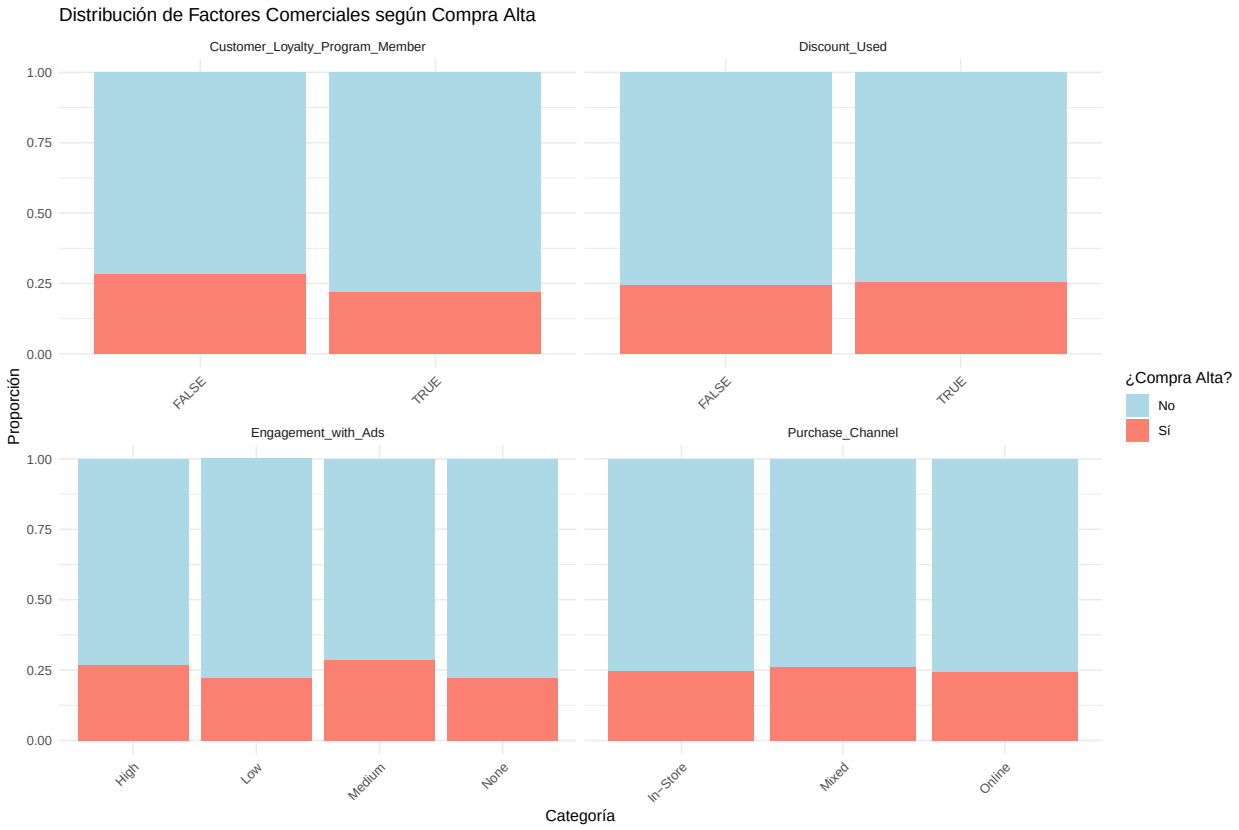
-Las barras se mostrarán apiladas por color (Sí/No).

-Se ajustarán los tipos de datos para evitar errores (conversión a character).

Nota técnica:

Antes de aplicar pivot_longer(), se convertirán las variables a tipo character para evitar errores de combinación de tipos (logical, character). Esta conversión es necesaria para la visualización. Para análisis futuros (clustering o modelado), si se requiere, se puede reconvertir fácilmente a factor o logical.

```r
# Cargar librerías necesarias
library(ggplot2)
library(dplyr)
library(tidyr)

# 1. Selección de variables relevantes
factores <- datos_limpios %>%
  select(Compra_Alta, Discount_Used, Customer_Loyalty_Program_Member,
         Engagement_with_Ads, Purchase_Channel) %>%
  mutate(across(everything(), as.character))  # 2. Conversión a character para evitar errores en pivot_

# 3. Reestructurar datos a formato largo
factores_long <- factores %>%
  pivot_longer(-Compra_Alta, names_to = "Variable", values_to = "Valor")

# 4. Crear gráfico de barras apiladas y facetadas
ggplot(factores_long, aes(x = Valor, fill = Compra_Alta)) +
  geom_bar(position = "fill") +
  facet_wrap(~ Variable, scales = "free_x", ncol = 2) +
  scale_fill_manual(values = c("lightblue", "salmon"),
                    name = "¿Compra Alta?",
                    labels = c("No", "Sí")) +
  labs(title = "Distribución de Factores Comerciales según Compra Alta",
       x = "Categoría",
       y = "Proporción",
       fill = "Compra Alta") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Que resultado se obtiene?

Se genera un conjunto de **cuatro gráficos de barras**. - Cada gráfico muestra la **proporción** de usuarios (`Sí` o `No` en Compra Alta) para cada categoría de las variables comerciales. - El gráfico permite observar visualmente si alguna categoría está más asociada con compras altas.

Interpretación del resultado

Discount_Used: Los usuarios que utilizaron descuentos tienden a tener una proporción ligeramente mayor de compras altas.

Customer_Loyalty_Program_Member: Se observa una mayor proporción de compras altas entre los miembros de programas de fidelización.

Engagement_with_Ads: Los usuarios con mayor interacción publicitaria parecen tener una leve tendencia hacia mayores compras, aunque la
diferencia no es muy marcada.

Purchase_Channel: No se aprecia una diferencia muy significativa en la proporción de compras altas según el canal de compra utilizado.

Este análisis sugiere que el **uso de descuentos** y la **pertenencia a programas de fidelización** son factores comerciales que podrían asociarse a un mayor nivel de gasto.

Preparación de los datos

Selección de datos

Objetivo:

Eliminar variables que no aportan valor analítico y quedarnos con las que serán útiles para clustering, PCA y análisis posterior.

Decisiones tomadas:

Se han eliminado dos columnas del dataset original:

-*Customer_ID:* es un identificador único sin ningún valor predictivo o explicativo. Solo sirve para diferenciar registros.

-*Location:* contiene 969 valores distintos (uno por usuario), lo que la convierte en una variable casi única por registro. No es adecuada para análisis sin una transformación avanzada como agrupamiento geográfico.

Se han conservado las variables relevantes según los objetivos definidos en la Fase 1 y 2.

Este paso permite reducir ruido y simplificar el dataset, facilitando el enfoque en las variables realmente útiles para el análisis.

```
# Crear copia del dataset original
datos_limpios <- ecommerce

# Eliminar columnas irrelevantes
datos_limpios <- datos_limpios %>%
  select(-Customer_ID, -Location)

#Ver nombres de columnas DESPUÉS de eliminar
cat("\n Columnas después de eliminar 'Customer_ID' y 'Location':\n")
```

```
##
##  Columnas después de eliminar 'Customer_ID' y 'Location':
```

```
print(colnames(datos_limpios))
```

```
##  [1] "Age"
##  [2] "Gender"
##  [3] "Income_Level"
##  [4] "Marital_Status"
##  [5] "Education_Level"
##  [6] "Occupation"
##  [7] "Purchase_Category"
##  [8] "Purchase_Amount"
##  [9] "Frequency_of_Purchase"
## [10] "Purchase_Channel"
## [11] "Brand_Loyalty"
## [12] "Product_Rating"
## [13] "Time_Spent_on_Product_Research.hours."
## [14] "Social_Media_Influence"
## [15] "Discount_Sensitivity"
## [16] "Return_Rate"
## [17] "Customer_Satisfaction"
## [18] "Engagement_with_Ads"
## [19] "Device_Used_for_Shopping"
## [20] "Payment_Method"
## [21] "Time_of_Purchase"
## [22] "Discount_Used"
## [23] "Customer_Loyalty_Program_Member"
```

```
## [24] "Purchase_Intent"
## [25] "Shipping_Preference"
## [26] "Time_to_Decision"
```

```
#Verificar si las columnas eliminadas ya no existen
cat("\n¿'Customer_ID' sigue en el dataset?: ", "Customer_ID" %in% colnames(datos_limpios), "\n")
```

```
##
## ¿'Customer_ID' sigue en el dataset?:  FALSE
```

```
cat("¿'Location' sigue en el dataset?: ", "Location" %in% colnames(datos_limpios), "\n")
```

```
## ¿'Location' sigue en el dataset?:  FALSE
```

Limìpeza e integración de datos

Garantizar la consistencia y validez del contenido del dataset corrigiendo errores de formato y asegurando que no haya problemas estructurales.

Decisiones tomadas:

Conversión de *Purchase_Amount* a formato numérico:

Objetivo:

La variable Purchase_Amount venía originalmente en formato texto debido a la presencia del símbolo $. Se ha aplicado una limpieza mediante *gsub()* para eliminar dicho símbolo y posteriormente se ha convertido a numérico. A continuación se muestra el resumen estadístico tras la transformación, lo cual confirma que la variable ahora puede utilizarse en operaciones numéricas:

Mostrar variable original:

```
head(ecommerce$Purchase_Amount)
```

```
## [1] "$333.80 " "$222.22 " "$426.22 " "$101.31 " "$211.70 " "$487.95 "
```

Transformamos la variable:

```
#  Eliminar símbolo $ y convertir a numérico
datos_limpios$Purchase_Amount <- as.numeric(gsub("[$]", "", datos_limpios$Purchase_Amount))

#  Mostrar resumen estadístico de la variable después de la transformación
head(datos_limpios$Purchase_Amount)
```

```
## [1] 333.80 222.22 426.22 101.31 211.70 487.95
```

```
summary(datos_limpios$Purchase_Amount)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   50.71  162.24  276.17  275.06  388.98  498.33
```

Verificación de valores nulos y duplicados:

Objetivo:

Asegurarnos de que el dataset no contenga registros incompletos o duplicados que puedan alterar el análisis.

```
#  Ver número total de registros
cat(" Total de registros en el dataset:", nrow(datos_limpios), "\n\n")
```

```
##  Total de registros en el dataset: 1000
```

```
#  Verificación de valores nulos por columna
cat(" Comprobación de valores nulos por variable:\n")
```

```
##  Comprobación de valores nulos por variable:
```

```
nulos <- sapply(datos_limpios, function(x) sum(is.na(x)))
print(nulos[nulos > 0])
```

```
## named integer(0)
```

```
if (all(nulos == 0)) {
  cat("No se encontraron valores nulos en el dataset.\n\n")
}
```

```
## No se encontraron valores nulos en el dataset.
```

```
# Verificación de registros duplicados
duplicados <- sum(duplicated(datos_limpios))
cat(" Número de registros duplicados en el dataset:", duplicados, "\n")
```

```
##  Número de registros duplicados en el dataset: 0
```

```
if (duplicados == 0) {
  cat("No hay registros duplicados.\n")
}
```

```
## No hay registros duplicados.
```

Se ha revisado la presencia de valores nulos y registros duplicados en el dataset. Estos dos tipos de errores pueden generar resultados incorrectos o sesgados en el modelado, por lo que es imprescindible verificarlos y corregirlos si se detectan.

Transformación de la variable Time_of_Purchase

Objetivo:

Convertir la variable Time_of_Purchase de texto (character) al formato correcto de fecha (Date), para poder:

Crear nuevas variables derivadas si se desea (mes, día de la semana…)

Utilizarla en análisis temporales

Evitar errores en funciones que requieran formato de fecha

```r
# Ver primeros valores originales y tipo de dato
cat(" Valores originales de 'Time_of_Purchase':\n")
```

```
##  Valores originales de 'Time_of_Purchase':
```

```r
print(head(datos_limpios$Time_of_Purchase))
```

```
## [1] "3/1/2024"  "4/16/2024" "3/15/2024" "10/4/2024" "1/30/2024" "3/19/2024"
```

```r
cat("\n Tipo de dato antes de la transformación:\n")
```

```
##
##  Tipo de dato antes de la transformación:
```

```r
print(class(datos_limpios$Time_of_Purchase))
```

```
## [1] "character"
```

```r
# Conversión a formato fecha
datos_limpios$Time_of_Purchase <- as.Date(datos_limpios$Time_of_Purchase, format = "%m/%d/%Y")


# Ver primeros valores después de la conversión
cat("\n Valores después de la conversión:\n")
```

```
##
##  Valores después de la conversión:
```

```r
print(head(datos_limpios$Time_of_Purchase))
```

```
## [1] "2024-03-01" "2024-04-16" "2024-03-15" "2024-10-04" "2024-01-30"
## [6] "2024-03-19"
```

```r
cat("\n Tipo de dato después de la transformación:\n")
```

```
##
##  Tipo de dato después de la transformación:
```

```r
print(class(datos_limpios$Time_of_Purchase))
```

```
## [1] "Date"
```

Tras realizar la conversión de la variable *Time_of_Purchase* al formato Date, vamos a verificar la presencia de valores nulos (NA). Esta comprobación es esencial para asegurar que no se haya producido pérdida de información durante el proceso. A continuación, se muestra el número de valores faltantes, y en caso de haberlos, se listan los registros afectados.

```r
#  Comprobar si hay valores NA en la variable Time_of_Purchase
cat(" Número de valores nulos en 'Time_of_Purchase':\n")
```

```
##  Número de valores nulos en 'Time_of_Purchase':
```

```r
nulos_fecha <- sum(is.na(datos_limpios$Time_of_Purchase))
print(nulos_fecha)
```

```
## [1] 0
```

```r
#  Mostrar los valores nulos si existen
if (nulos_fecha > 0) {
  cat("\n Registros con NA en 'Time_of_Purchase':\n")
  print(datos_limpios[is.na(datos_limpios$Time_of_Purchase), ])
} else {
  cat("No se encontraron valores nulos en 'Time_of_Purchase'.\n")
}
```

```
## No se encontraron valores nulos en 'Time_of_Purchase'.
```

Detección de valores atípicos (outliers)

Objetivo:

Detectar valores extremos en las variables numéricas del conjunto de datos que podrían:

- Distorsionar análisis estadísticos
- Afectar la eficacia de algoritmos como clustering o PCA
- Representar errores, casos excepcionales o información valiosa

Este paso forma parte de la limpieza avanzada de datos dentro de la Fase 3 (Preparación) y se centra en el diagnóstico, no necesariamente en la eliminación de valores aún.

¿Qué variables se van a analizar?

Age

Purchase_Amount

Frequency_of_Purchase

Customer_Satisfaction

Return_Rate

Time_to_Decision

Time_Spent_on_Product_Research(hours)

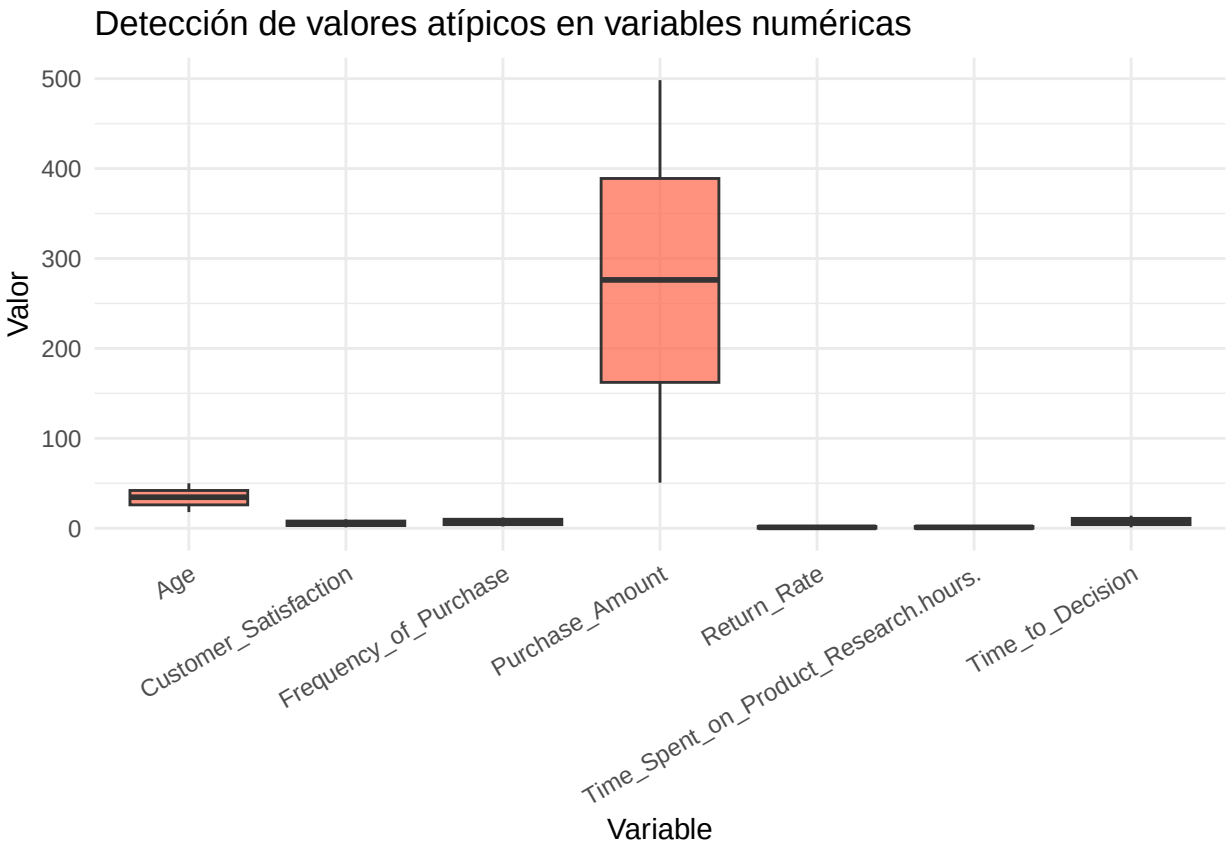
- Resumen estadístico con summary():

```r
# Seleccionar variables numéricas
variables_numericas <- datos_limpios %>%
  select(Age, Purchase_Amount, Frequency_of_Purchase,
         Customer_Satisfaction, Return_Rate, Time_to_Decision,
         Time_Spent_on_Product_Research.hours.)

# Mostrar resumen estadístico
summary(variables_numericas)
```

```
##       Age       Purchase_Amount  Frequency_of_Purchase Customer_Satisfaction
##  Min.   :18.0   Min.   : 50.71   Min.   : 2.000        Min.   : 1.000
##  1st Qu.:26.0   1st Qu.:162.24   1st Qu.: 4.000        1st Qu.: 3.000
##  Median :34.5   Median :276.17   Median : 7.000        Median : 5.000
##  Mean   :34.3   Mean   :275.06   Mean   : 6.945        Mean   : 5.399
##  3rd Qu.:42.0   3rd Qu.:388.98   3rd Qu.:10.000        3rd Qu.: 8.000
##  Max.   :50.0   Max.   :498.33   Max.   :12.000        Max.   :10.000
##   Return_Rate    Time_to_Decision Time_Spent_on_Product_Research.hours.
##  Min.   :0.000   Min.   : 1.000   Min.   :0.000
##  1st Qu.:0.000   1st Qu.: 4.000   1st Qu.:0.000
##  Median :1.000   Median : 8.000   Median :1.000
##  Mean   :0.954   Mean   : 7.547   Mean   :1.013
##  3rd Qu.:2.000   3rd Qu.:11.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :14.000   Max.   :2.000
```

-Boxplots para visualización:

```r
# Convertir a formato largo para usar facet_wrap()
library(tidyr)
library(ggplot2)

datos_long <- pivot_longer(variables_numericas,
                           cols = everything(),
                           names_to = "Variable",
                           values_to = "Valor")

# Crear boxplots por variable
ggplot(datos_long, aes(x = Variable, y = Valor)) +
  geom_boxplot(fill = "tomato", alpha = 0.7) +
  labs(title = "Detección de valores atípicos en variables numéricas",
       x = "Variable", y = "Valor") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```



Se han representado las variables numéricas del conjunto de datos mediante diagramas de caja (boxplots) con el objetivo de identificar la presencia de valores atípicos (outliers).

En el gráfico se puede observar que:

La variable *Purchase_Amount* muestra una distribución más amplia y presenta valores atípicos visibles en su rango superior. Esto es habitual en montos de compra, ya que algunos usuarios pueden gastar cantidades significativamente mayores que la media.

El resto de variables (Age, Customer_Satisfaction, Frequency_of_Purchase, Return_Rate, Time_Spent_on_Product_Resea Time_to_Decision) presentan distribuciones bastante compactas, sin valores extremos evidentes o con escasa dispersión.

En esta fase se ha optado únicamente por detectar los valores atípicos, sin eliminarlos ni transformarlos, ya que esto dependerá de su impacto en las técnicas que se apliquen posteriormente (PCA, clustering). No se descarta realizar una gestión específica de outliers si en la fase de modelado se observa que afectan significativamente a los resultados.

Además de la visualización mediante boxplots, se ha realizado una detección numérica de outliers utilizando el criterio del rango intercuartílico (IQR). Este método considera como valores atípicos aquellos que se encuentran por debajo de Q1 - 1.5 * IQR o por encima de Q3 + 1.5 * IQR.

A continuación se muestra una tabla con el número de outliers detectados en cada una de las variables numéricas:

```
# Selección de variables numéricas
vars <- datos_limpios %>%
  select(Age, Purchase_Amount, Frequency_of_Purchase,
         Customer_Satisfaction, Return_Rate, Time_to_Decision,
         Time_Spent_on_Product_Research.hours.)
```

```r
# Función para contar outliers según IQR
contar_outliers <- function(x) {
  q1 <- quantile(x, 0.25, na.rm = TRUE)
  q3 <- quantile(x, 0.75, na.rm = TRUE)
  iqr <- q3 - q1
  sum(x < (q1 - 1.5 * iqr) | x > (q3 + 1.5 * iqr), na.rm = TRUE)
}

# Aplicar a cada variable
outliers_por_variable <- sapply(vars, contar_outliers)

# Mostrar en forma de tabla
outliers_df <- data.frame(
  Variable = names(outliers_por_variable),
  Outliers = outliers_por_variable
)

# Mostrar tabla formateada
library(knitr)
kable(outliers_df, caption = "Número de outliers detectados por variable (criterio IQR)")
```

Table 2: Número de outliers detectados por variable (criterio IQR)

	Variable	Outliers
Age	Age	0
Purchase_Amount	Purchase_Amount	0
Frequency_of_Purchase	Frequency_of_Purchase	0
Customer_Satisfaction	Customer_Satisfaction	0
Return_Rate	Return_Rate	0
Time_to_Decision	Time_to_Decision	0
Time_Spent_on_Product_Research.hours.	Time_Spent_on_Product_Research.hours.	0

Construcción de nuevas variables

Objetivo:

Crear nuevas variables derivadas de *Time_of_Purchase* que permitan enriquecer el análisis posterior, tanto en clustering como en modelos supervisados.

El comportamiento del usuario puede variar según el momento en que realiza la compra, por lo tanto, disponer de información como el mes o el día de la semana puede revelar patrones importantes:

¿Compra más gente en determinados meses?

¿Existen diferencias entre compras entre semana y en fin de semana?

Estas nuevas variables permiten explorar aspectos temporales del comportamiento del consumidor, que no están presentes explícitamente en el dataset original, pero que pueden influir en la intención de compra.

Qué vamos a hacer

Derivar una variable Mes_Compra a partir de Time_of_Purchase

- Nos permitirá ver si hay estacionalidad en las compras.

Derivar una variable Dia_Semana a partir de Time_of_Purchase

- Nos permitirá ver si el día de la semana influye en la compra.

Verificaremos que las variables se han creado correctamente.

1. Crear una nueva variable llamada Mes_Compra a partir de Time_of_Purchase, que indique el mes en el que se realizó la compra:

El mes de compra puede revelar patrones estacionales o periodos de mayor actividad. Por ejemplo, puede haber más compras en diciembre (Navidad), rebajas, vuelta al cole, etc. Esta variable puede ser útil tanto para análisis exploratorio como para clustering o clasificación en la PRA2.

Se ha creado una nueva variable Mes_Compra a partir de Time_of_Purchase, con el objetivo de capturar posibles patrones estacionales. Esta variable indica el mes en que se realizó cada compra, en formato textual. A continuación se muestran los primeros valores y la frecuencia de compras por mes.

```
# Crear la variable 'Mes_Compra' desde la fecha
datos_limpios$Mes_Compra <- format(datos_limpios$Time_of_Purchase, "%B")  # Nombre del mes

# Mostrar los primeros valores para verificar
cat("Primeros valores de 'Mes_Compra':\n")
```

```
## Primeros valores de 'Mes_Compra':
```

```
print(head(datos_limpios$Mes_Compra))
```

```
## [1] "marzo"   "abril"   "marzo"   "octubre" "enero"   "marzo"
```

```
# Mostrar frecuencia de meses
cat("\nFrecuencia de compras por mes:\n")
```

```
##
## Frecuencia de compras por mes:
```

```
print(table(datos_limpios$Mes_Compra))
```

```
##
##      abril     agosto  diciembre      enero    febrero      julio      junio
##        100         98         64         75         68         94         89
##      marzo       mayo  noviembre    octubre septiembre
##         93         76         80         79         84
```

Crear una nueva variable llamada Dia_Semana que contenga el día de la semana en el que se realizó la compra (lunes, martes, etc.), a partir de la variable Time_of_Purchase.

El día de la semana puede influir en el comportamiento del consumidor. Por ejemplo:

Las compras impulsivas pueden concentrarse en fines de semana

Las compras racionales pueden hacerse más entre semana

Puede ayudar en la segmentación de usuarios o en modelos que consideren hábitos de compra

Incluir esta variable puede mejorar la capacidad de los modelos de clustering o clasificación para identificar patrones.

Se ha creado una nueva variable Dia_Semana a partir de Time_of_Purchase, con el objetivo de capturar posibles patrones relacionados con el día de la semana en el que se realiza la compra. Esta variable puede ser especialmente útil para detectar comportamientos diferentes entre compras de lunes a viernes y durante el fin de semana. A continuación se muestran los primeros valores y la distribución total de compras por día.

```
# Crear la variable 'Dia_Semana'
datos_limpios$Dia_Semana <- weekdays(datos_limpios$Time_of_Purchase)

# Verificar primeros valores
cat("Primeros valores de 'Dia_Semana':\n")
```

```
## Primeros valores de 'Dia_Semana':
```

```
print(head(datos_limpios$Dia_Semana))
```

```
## [1] "viernes" "martes"  "viernes" "viernes" "martes"  "martes"
```

```
# Verificar frecuencia de días
cat("\nFrecuencia de compras por día de la semana:\n")
```

```
##
## Frecuencia de compras por día de la semana:
```

```
print(table(datos_limpios$Dia_Semana))
```

```
##
##   domingo    jueves     lunes    martes miércoles    sábado   viernes
##       156       153       143       159       124       128       137
```

A continucación, vamos a crear 2 variables booleanas que nos pueden ayudar

- *Variable 1: Compra_FinDe*:

Una nueva variable binaria que indica si la compra se realizó en fin de semana (sábado o domingo). Tendrá valores “Sí” o “No”.

El día de la semana puede influir en el comportamiento de compra:

Las compras del fin de semana pueden ser más impulsivas

Las del lunes a viernes pueden estar más planificadas

Esto puede resultar útil en el clustering o en la clasificación de intenciones de compra.

Usamos la variable Dia_Semana (ya construida) para comprobar si el valor corresponde a “sábado” o “domingo”.

Se ha creado la variable Compra_FinDe, que indica si la compra fue realizada durante el fin de semana (sábado o domingo).

Esta variable permite segmentar a los usuarios según su comportamiento temporal y puede ser útil para identificar diferencias entre perfiles. A continuación se muestran ejemplos de valores y su distribución.

```r
# Crear variable Compra_FinDe
datos_limpios$Compra_FinDe <- ifelse(datos_limpios$Dia_Semana %in% c("sábado", "domingo"), "Sí", "No")

# Verificar primeros valores
cat("Primeros valores de 'Compra_FinDe':\n")
```

```
## Primeros valores de 'Compra_FinDe':
```

```r
print(head(datos_limpios$Compra_FinDe))
```

```
## [1] "No" "No" "No" "No" "No" "No"
```

```r
# Verificar frecuencia
cat("\nFrecuencia de compras en fin de semana:\n")
```

```
##
## Frecuencia de compras en fin de semana:
```

```r
print(table(datos_limpios$Compra_FinDe))
```

```
##
##  No  Sí
## 716 284
```

-Variable 2: Compra_Alta

Una nueva variable binaria que indica si el usuario realizó una compra alta, definida como una compra cuyo valor de Purchase_Amount está por encima del tercer cuartil (Q3) del conjunto de datos.

Permite distinguir entre usuarios de gasto alto y bajo, lo cual puede influir en su fidelidad, sensibilidad al descuento o segmentación comercial.

Se calcula el valor de Q3 (percentil 75) de Purchase_Amount, y se asigna "Sí" a las compras por encima de ese valor, "No" al resto.

La variable Compra_Alta identifica si una compra supera el umbral del tercer cuartil (Q3) de Purchase_Amount. Esta clasificación binaria permite segmentar a los usuarios según su nivel de gasto, lo cual puede resultar útil para modelos de clustering o análisis de comportamiento.

El umbral aplicado se ha calculado sobre el 75% percentil y se indica a continuación junto con la distribución obtenida.

```r
# Calcular el umbral alto de compra
q3 <- quantile(datos_limpios$Purchase_Amount, 0.75, na.rm = TRUE)

# Crear variable Compra_Alta
datos_limpios$Compra_Alta <- ifelse(datos_limpios$Purchase_Amount > q3, "Sí", "No")

# Verificar primeros valores
cat("Primeros valores de 'Compra_Alta':\n")
```

```
## Primeros valores de 'Compra_Alta':
```

```r
print(head(datos_limpios$Compra_Alta))
```

```
## [1] "No" "No" "Sí" "No" "No" "Sí"
```

```r
# Verificar frecuencia
cat("\nFrecuencia de compras altas:\n")
```

```
##
## Frecuencia de compras altas:
```

```r
print(table(datos_limpios$Compra_Alta))
```

```
##
##  No  Sí
## 750 250
```

```r
# Mostrar el umbral usado
cat("\nUmbral de compra alta (Q3):", q3, "\n")
```

```
##
## Umbral de compra alta (Q3): 388.9825
```

Discretización de Age

Transformar la variable numérica Age en una variable categórica llamada Grupo_Edad, clasificando a los usuarios en grupos de edad: por ejemplo “Joven”, “Adulto” y “Mayor”.

Discretizar Age permite:

Simplificar el análisis al trabajar con categorías más comprensibles

Detectar patrones por grupo de edad más fácilmente

Usar la variable en modelos o gráficos que funcionan mejor con factores

Evitar la sensibilidad a pequeñas variaciones numéricas

Esta transformación puede ser útil para segmentar clientes en perfiles sociodemográficos.

Usaremos rango de edad por puntos de corte. En este caso:

Joven: menores de 30 años

Adulto: entre 30 y 45 años

Mayor: mayores de 45 años

Estos rangos pueden ajustarse, pero son razonables para interpretar comportamiento de compra.

Se ha discretizado la variable Age en una nueva variable categórica llamada Grupo_Edad, con tres niveles: “Joven”, “Adulto” y “Mayor”. Esta transformación permite segmentar el análisis por tramos de edad, facilitando la interpretación y el uso en algoritmos que requieren variables categóricas. Los cortes utilizados fueron:

Menores de 30 años: “Joven”

Entre 30 y 45 años: “Adulto”

Mayores de 45 años: “Mayor” A continuación se muestran los primeros valores de la variable y la distribución completa de usuarios por grupo.

```r
# Crear nueva variable categórica 'Grupo_Edad' a partir de 'Age'
datos_limpios$Grupo_Edad <- cut(
  datos_limpios$Age,
  breaks = c(-Inf, 29, 45, Inf),
  labels = c("Joven", "Adulto", "Mayor")
)

# Verificar primeros valores
cat("Primeros valores de 'Grupo_Edad':\n")
```

```
## Primeros valores de 'Grupo_Edad':
```

```r
print(head(datos_limpios[, c("Age", "Grupo_Edad")]))
```

```
##   Age Grupo_Edad
## 1  22      Joven
## 2  49      Mayor
## 3  24      Joven
## 4  29      Joven
## 5  33     Adulto
## 6  45     Adulto
```

```r
# Verificar distribución
cat("\nDistribución de usuarios por grupo de edad:\n")
```

```
##
## Distribución de usuarios por grupo de edad:
```

```r
print(table(datos_limpios$Grupo_Edad))
```

```
##
##  Joven Adulto  Mayor
##    360    490    150
```

Disdcretización Customer_Satisfaction

Vamos a observar la variable Customer_Satisfaction:

```r
# Resumen estadístico
cat("Resumen estadístico de 'Customer_Satisfaction':\n")
```

```
## Resumen estadístico de 'Customer_Satisfaction':
```

```r
summary(datos_limpios$Customer_Satisfaction)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   3.000   5.000   5.399   8.000  10.000
```

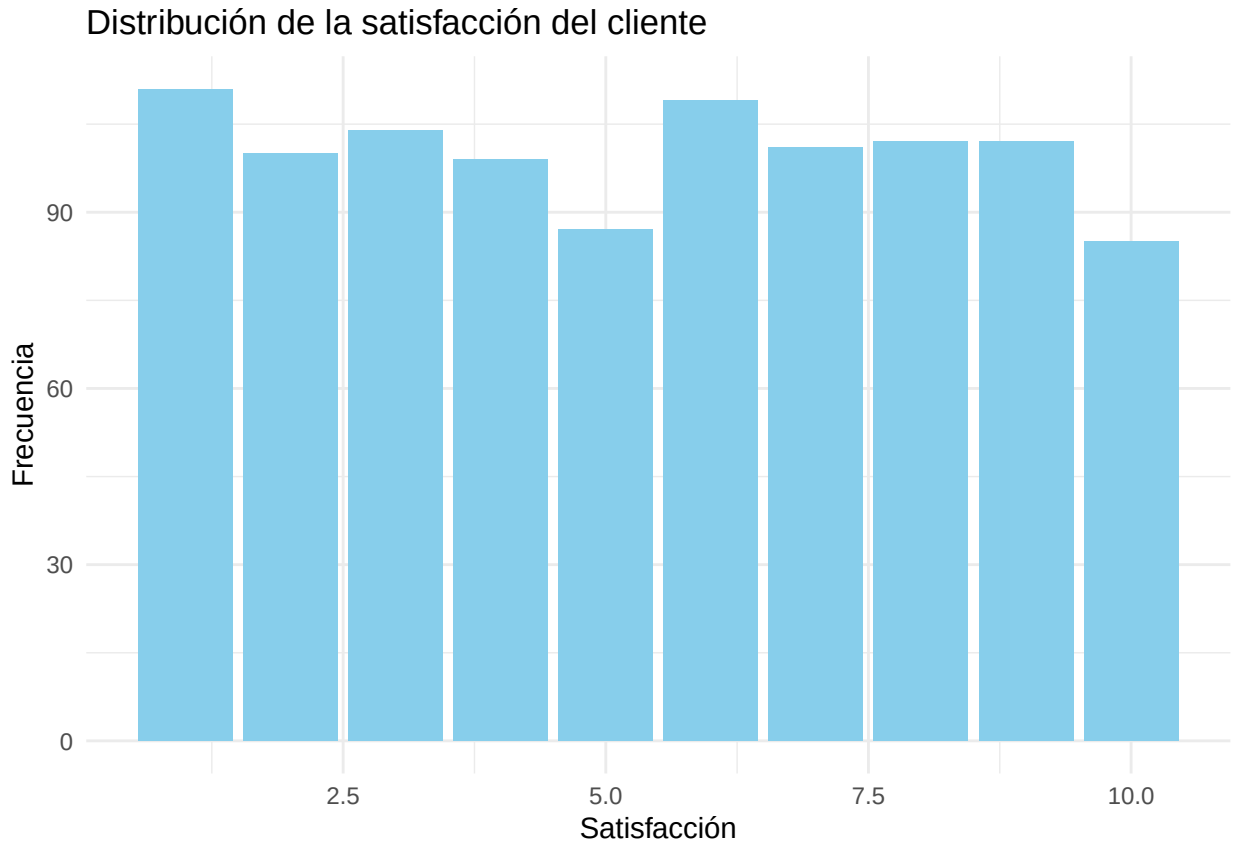
```r
# Frecuencia de cada valor
cat("\nFrecuencia de valores en 'Customer_Satisfaction':\n")
```

```
##
## Frecuencia de valores en 'Customer_Satisfaction':
```

```r
print(table(datos_limpios$Customer_Satisfaction))
```

```
##
##   1   2   3   4   5   6   7   8   9  10
## 111 100 104  99  87 109 101 102 102  85
```

```r
# Histograma simple
library(ggplot2)
ggplot(datos_limpios, aes(x = Customer_Satisfaction)) +
  geom_bar(fill = "skyblue") +
  labs(title = "Distribución de la satisfacción del cliente", x = "Satisfacción", y = "Frecuencia") +
  theme_minimal()
```



Observaciones del gráfico:

La satisfacción del cliente está medida del 1 al 10.

No hay concentraciones extremas ni sesgos fuertes.

La variable está distribuida de manera bastante equilibrada, lo cual es ideal para discretizar en niveles interpretables.

Propuesta de discretización:

Crear una nueva variable *Nivel_Satisfaccion* con tres grupos:

Baja satisfacción: 1 a 3

Media satisfacción: 4 a 7

Alta satisfacción: 8 a 10

Estos rangos son fáciles de interpretar y están alineados con lo que suele hacerse en encuestas de valoración.

La variable Customer_Satisfaction se ha discretizado en una nueva variable categórica llamada Nivel_Satisfaccion, con los niveles "Baja", "Media" y "Alta". Esta transformación facilita la interpretación y permite comparar segmentos de clientes según su nivel de satisfacción.

A continuación se muestra la distribución de registros por nivel.

```
# Crear la variable categórica 'Nivel_Satisfaccion'
datos_limpios$Nivel_Satisfaccion <- cut(
  datos_limpios$Customer_Satisfaction,
  breaks = c(-Inf, 3, 7, Inf),
  labels = c("Baja", "Media", "Alta")
)

# Verificar primeros valores
cat("Primeros valores de 'Nivel_Satisfaccion':\n")
```

```
## Primeros valores de 'Nivel_Satisfaccion':
```

```
print(head(datos_limpios[, c("Customer_Satisfaction", "Nivel_Satisfaccion")]))
```

```
##   Customer_Satisfaction Nivel_Satisfaccion
## 1                     7              Media
## 2                     5              Media
## 3                     7              Media
## 4                     1               Baja
## 5                    10               Alta
## 6                     3               Baja
```

```
# Verificar distribución
cat("\nDistribución por nivel de satisfacción:\n")
```

```
##
## Distribución por nivel de satisfacción:
```

```
print(table(datos_limpios$Nivel_Satisfaccion))
```

```
##
##  Baja Media  Alta
##   315   396   289
```

VISUALIZAR EL CONJUNTOD DE DATOS PREPORCESADO

Antes de proceder a la normalización y selección final de variables para aplicar técnicas de modelado, se ha revisado el conjunto de datos completo. A continuación se muestran las primeras filas, los nombres de las columnas y la estructura del dataset con las nuevas variables construidas durante la fase de preparación.

```
# Ver primeras filas del dataset completo con variables nuevas
head(datos_limpios)
```

```
##   Age Gender Income_Level Marital_Status Education_Level Occupation
## 1  22 Female       Middle        Married      Bachelor's     Middle
## 2  49   Male         High        Married     High School       High
## 3  24 Female       Middle         Single        Master's       High
## 4  29 Female       Middle         Single        Master's     Middle
## 5  33 Female       Middle        Widowed     High School     Middle
## 6  45   Male       Middle        Married        Master's       High
##      Purchase_Category Purchase_Amount Frequency_of_Purchase Purchase_Channel
## 1 Gardening & Outdoors          333.80                     4            Mixed
## 2     Food & Beverages          222.22                    11         In-Store
## 3      Office Supplies          426.22                     2            Mixed
## 4      Home Appliances          101.31                     6            Mixed
## 5            Furniture          211.70                     6            Mixed
## 6      Office Supplies          487.95                     8            Mixed
##   Brand_Loyalty Product_Rating Time_Spent_on_Product_Research.hours.
## 1             5              5                                   2.0
## 2             3              1                                   2.0
## 3             5              5                                   0.3
## 4             3              1                                   1.0
## 5             3              4                                   0.0
## 6             3              3                                   0.0
##   Social_Media_Influence Discount_Sensitivity Return_Rate Customer_Satisfaction
## 1                   None   Somewhat Sensitive           1                    7
## 2                 Medium        Not Sensitive           1                    5
## 3                    Low        Not Sensitive           1                    7
## 4                   High   Somewhat Sensitive           0                    1
## 5                 Medium        Not Sensitive           2                   10
## 6                   High        Not Sensitive           2                    3
##   Engagement_with_Ads Device_Used_for_Shopping Payment_Method Time_of_Purchase
## 1                None                  Tablet    Credit Card       2024-03-01
## 2                High                  Tablet         PayPal       2024-04-16
## 3                 Low              Smartphone     Debit Card       2024-03-15
## 4                None              Smartphone          Other       2024-10-04
## 5                None              Smartphone     Debit Card       2024-01-30
## 6                None                  Tablet     Debit Card       2024-03-19
##   Discount_Used Customer_Loyalty_Program_Member Purchase_Intent
## 1          TRUE                           FALSE      Need-based
## 2          TRUE                           FALSE     Wants-based
## 3          TRUE                            TRUE       Impulsive
## 4          TRUE                            TRUE      Need-based
## 5         FALSE                           FALSE     Wants-based
## 6         FALSE                           FALSE         Planned
##   Shipping_Preference Time_to_Decision Mes_Compra Dia_Semana Compra_FinDe
## 1       No Preference                2      marzo    viernes           No
## 2            Standard                6      abril     martes           No
```

```
## 3       No Preference                3      marzo    viernes           No
## 4             Express               10    octubre    viernes           No
## 5       No Preference                4      enero     martes           No
## 6       No Preference                7      marzo     martes           No
##   Compra_Alta Grupo_Edad Nivel_Satisfaccion
## 1          No      Joven              Media
## 2          No      Mayor              Media
## 3          Sí      Joven              Media
## 4          No      Joven               Baja
## 5          No     Adulto               Alta
## 6          Sí     Adulto               Baja
```

```
# Mostrar todas las variables disponibles
colnames(datos_limpios)
```

```
##  [1] "Age"
##  [2] "Gender"
##  [3] "Income_Level"
##  [4] "Marital_Status"
##  [5] "Education_Level"
##  [6] "Occupation"
##  [7] "Purchase_Category"
##  [8] "Purchase_Amount"
##  [9] "Frequency_of_Purchase"
## [10] "Purchase_Channel"
## [11] "Brand_Loyalty"
## [12] "Product_Rating"
## [13] "Time_Spent_on_Product_Research.hours."
## [14] "Social_Media_Influence"
## [15] "Discount_Sensitivity"
## [16] "Return_Rate"
## [17] "Customer_Satisfaction"
## [18] "Engagement_with_Ads"
## [19] "Device_Used_for_Shopping"
## [20] "Payment_Method"
## [21] "Time_of_Purchase"
## [22] "Discount_Used"
## [23] "Customer_Loyalty_Program_Member"
## [24] "Purchase_Intent"
## [25] "Shipping_Preference"
## [26] "Time_to_Decision"
## [27] "Mes_Compra"
## [28] "Dia_Semana"
## [29] "Compra_FinDe"
## [30] "Compra_Alta"
## [31] "Grupo_Edad"
## [32] "Nivel_Satisfaccion"
```

```
# Ver estructura general del dataset
str(datos_limpios)
```

```
## 'data.frame':    1000 obs. of  32 variables:
##  $ Age                                : int  22 49 24 29 33 45 21 39 24 25 ...
```

```
##  $ Gender                                : chr  "Female" "Male" "Female" "Female" ...
##  $ Income_Level                          : chr  "Middle" "High" "Middle" "Middle" ...
##  $ Marital_Status                        : chr  "Married" "Married" "Single" "Single" ...
##  $ Education_Level                       : chr  "Bachelor's" "High School" "Master's" "Master's" ...
##  $ Occupation                            : chr  "Middle" "High" "High" "Middle" ...
##  $ Purchase_Category                     : chr  "Gardening & Outdoors" "Food & Beverages" "Office Sup
##  $ Purchase_Amount                       : num  334 222 426 101 212 ...
##  $ Frequency_of_Purchase                 : int  4 11 2 6 6 8 12 6 8 7 ...
##  $ Purchase_Channel                      : chr  "Mixed" "In-Store" "Mixed" "Mixed" ...
##  $ Brand_Loyalty                         : int  5 3 5 3 3 3 2 5 3 2 ...
##  $ Product_Rating                        : int  5 1 5 1 4 3 5 4 5 5 ...
##  $ Time_Spent_on_Product_Research.hours.: num  2 2 0.3 1 0 0 1 1 0 1 ...
##  $ Social_Media_Influence                : chr  "None" "Medium" "Low" "High" ...
##  $ Discount_Sensitivity                  : chr  "Somewhat Sensitive" "Not Sensitive" "Not Sensitive"
##  $ Return_Rate                           : int  1 1 1 0 2 2 0 2 1 1 ...
##  $ Customer_Satisfaction                 : int  7 5 7 1 10 3 9 9 2 5 ...
##  $ Engagement_with_Ads                   : chr  "None" "High" "Low" "None" ...
##  $ Device_Used_for_Shopping              : chr  "Tablet" "Tablet" "Smartphone" "Smartphone" ...
##  $ Payment_Method                        : chr  "Credit Card" "PayPal" "Debit Card" "Other" ...
##  $ Time_of_Purchase                      : Date, format: "2024-03-01" "2024-04-16" ...
##  $ Discount_Used                         : logi  TRUE TRUE TRUE TRUE FALSE FALSE ...
##  $ Customer_Loyalty_Program_Member       : logi  FALSE FALSE TRUE TRUE FALSE FALSE ...
##  $ Purchase_Intent                       : chr  "Need-based" "Wants-based" "Impulsive" "Need-based" .
##  $ Shipping_Preference                   : chr  "No Preference" "Standard" "No Preference" "Express"
##  $ Time_to_Decision                      : int  2 6 3 10 4 7 13 13 7 13 ...
##  $ Mes_Compra                            : chr  "marzo" "abril" "marzo" "octubre" ...
##  $ Dia_Semana                            : chr  "viernes" "martes" "viernes" "viernes" ...
##  $ Compra_FinDe                          : chr  "No" "No" "No" "No" ...
##  $ Compra_Alta                           : chr  "No" "No" "Sí" "No" ...
##  $ Grupo_Edad                            : Factor w/ 3 levels "Joven","Adulto",..: 1 3 1 1 2 2 1 2 1
##  $ Nivel_Satisfaccion                    : Factor w/ 3 levels "Baja","Media",..: 2 2 2 1 3 1 3 3 1 2
```

PCA

¿Qué es PCA y para qué sirve?

PCA (Análisis de Componentes Principales) es una técnica para reducir la dimensionalidad de un dataset con muchas variables numéricas, sin perder (demasiada) información.

Resumir la información de varias variables en menos componentes

Detectar patrones o agrupaciones en los datos

Preparar un espacio reducido para hacer clustering

Pasos que vamos a seguir:

1. Aplicar PCA con prcomp()
2. Ver cuánta varianza explica cada componente
3. Graficar la varianza acumulada
4. Ver qué variables están más relacionadas con cada componente
5. Visualizar los datos proyectados en el nuevo espacio

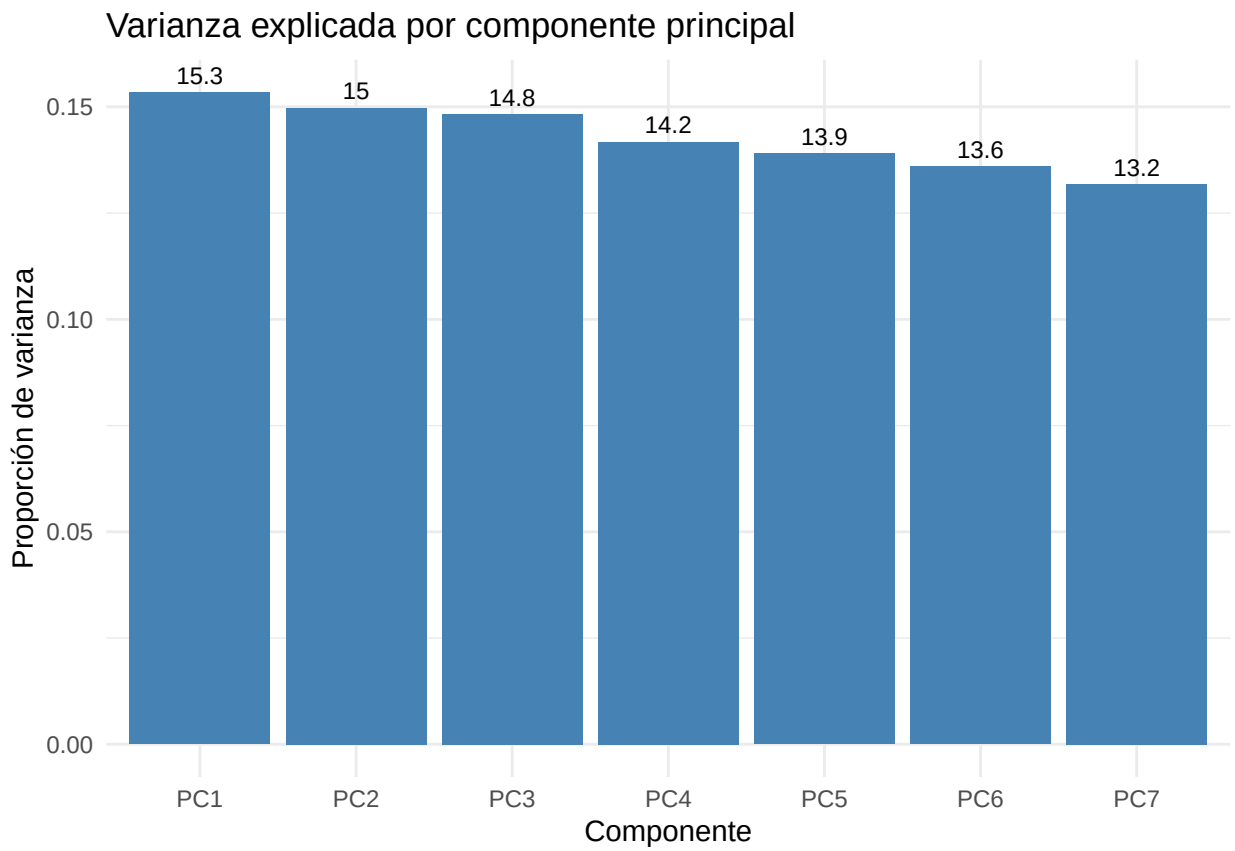
Aplicar PCA con prcomp()

Se ha seleccionado un subconjunto de variables numéricas relevantes del dataset, y se han normalizado utilizando la función scale() de R. Posteriormente, se ha aplicado el Análisis de Componentes Principales (PCA) mediante la función prcomp() para reducir la dimensionalidad y extraer los componentes principales. A continuación se muestra el resumen del PCA, que incluye la proporción de varianza explicada por cada componente.

```
# 1. Seleccionar variables numéricas relevantes
vars_numericas <- datos_limpios %>%
  select(Age, Purchase_Amount, Frequency_of_Purchase,
         Customer_Satisfaction, Return_Rate, Time_to_Decision,
         Time_Spent_on_Product_Research.hours.)

# 2. Normalizar las variables numéricas (media 0, desviación estándar 1)
vars_numericas_norm <- scale(vars_numericas)

# 3. Aplicar PCA sobre los datos normalizados
pca_resultado <- prcomp(vars_numericas_norm, center = FALSE, scale. = FALSE)

# 4. Ver resumen del PCA (varianza explicada por cada componente)
summary(pca_resultado)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5    PC6    PC7
## Standard deviation     1.0363 1.0238 1.0187 0.9962 0.9866 0.9756 0.9605
## Proportion of Variance 0.1534 0.1497 0.1483 0.1418 0.1391 0.1360 0.1318
## Cumulative Proportion  0.1534 0.3031 0.4514 0.5932 0.7322 0.8682 1.0000
```

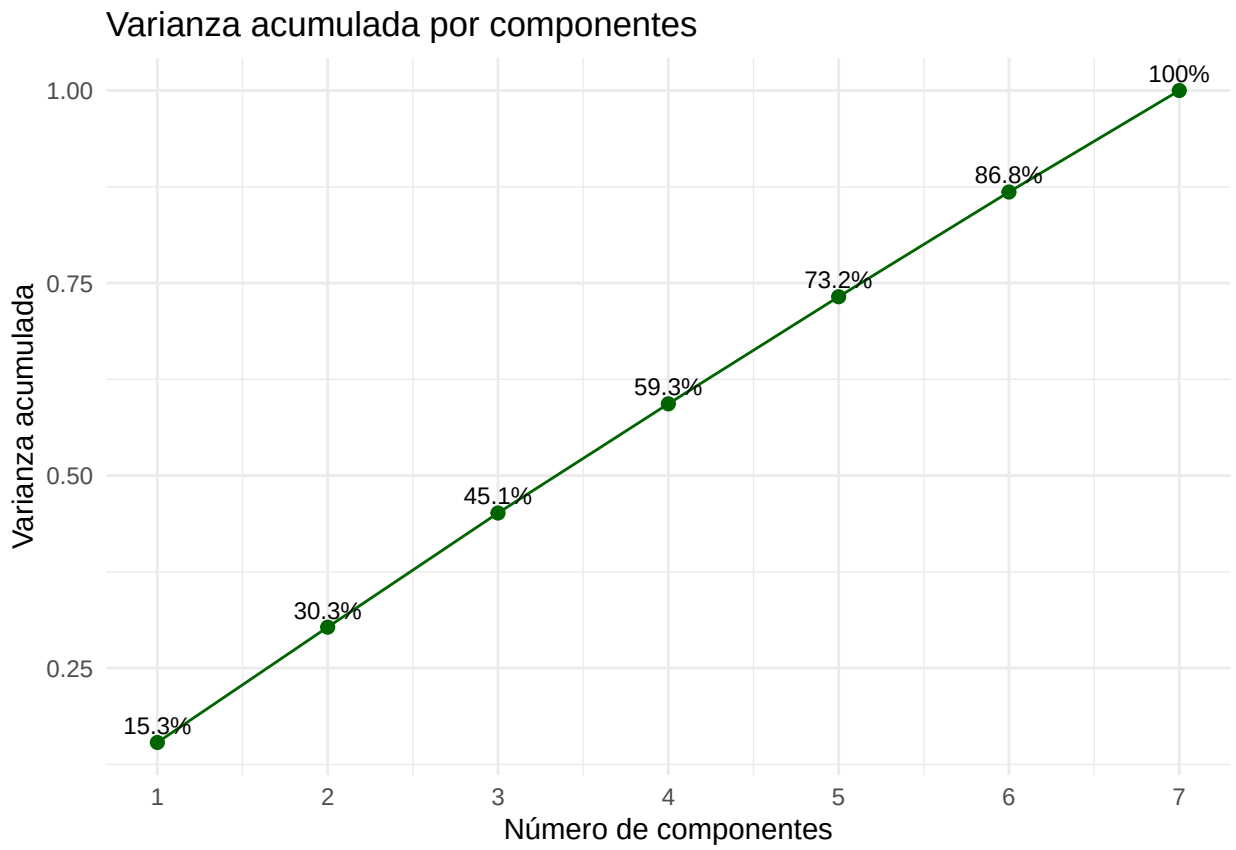
```
# Extraer la proporción de varianza explicada
varianza_explicada <- summary(pca_resultado)$importance[2, ]
varianza_acumulada <- summary(pca_resultado)$importance[3, ]

# Crear un data frame con los resultados
df_pca <- data.frame(
  Componente = paste0("PC", 1:length(varianza_explicada)),
  Varianza = varianza_explicada,
  Acumulada = varianza_acumulada
)

# Gráfico de barras: varianza explicada individual
library(ggplot2)
ggplot(df_pca, aes(x = Componente, y = Varianza)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(aes(label = round(Varianza * 100, 1)), vjust = -0.5, size = 3) +
  labs(title = "Varianza explicada por componente principal",
       y = "Proporción de varianza",
       x = "Componente") +
  theme_minimal()
```



El gráfico muestra la varianza acumulada explicada por los componentes principales obtenidos mediante PCA. Se conservarán los cinco primeros componentes, ya que explican más del 70% de la variabilidad total del conjunto de datos. Esta selección permite reducir la dimensionalidad del dataset manteniendo la mayor parte de la información original. Los componentes seleccionados se utilizarán como base para la aplicación de técnicas no supervisadas en la siguiente fase del análisis. el dataset sin perder demasiada información.

```
# Extraer proporciones del objeto PCA
varianza_explicada <- summary(pca_resultado)$importance[2, ]
varianza_acumulada <- summary(pca_resultado)$importance[3, ]

# Crear un data frame con los resultados
df_pca <- data.frame(
  Componente = paste0("PC", 1:length(varianza_explicada)),
  Varianza = varianza_explicada,
  Acumulada = varianza_acumulada
)

# Asegurar que ggplot2 está cargado
library(ggplot2)

# Gráfico 2: Varianza acumulada
grafico2 <- ggplot(df_pca, aes(x = as.numeric(gsub("PC", "", Componente)), y = Acumulada)) +
  geom_line(color = "darkgreen") +
  geom_point(color = "darkgreen", size = 2) +
  geom_text(aes(label = paste0(round(Acumulada * 100, 1), "%")), vjust = -0.5, size = 3) +
  scale_x_continuous(breaks = 1:nrow(df_pca)) +
  labs(title = "Varianza acumulada por componentes",
```

```
       x = "Número de componentes", y = "Varianza acumulada") +
  theme_minimal()

# Mostrar gráficos
print(grafico2)
```



Interpretación de la varianza explicada y acumulada

El Análisis de Componentes Principales (PCA) transforma las variables originales en un conjunto de nuevos ejes (componentes principales) ordenados según la cantidad de información (varianza) que explican del conjunto de datos.

Existen dos formas de interpretar esta información:

1. Varianza explicada por componente Esta métrica indica **cuánta información aporta individualmente cada componente**. Por ejemplo, el primer componente (PC1) puede explicar un 15% de la variabilidad total, el segundo (PC2) un 14%, y así sucesivamente.
Este valor permite **comparar la importancia relativa entre componentes**, pero no ayuda a decidir directamente cuántos conservar.

2. Varianza acumulada La varianza acumulada indica **la suma de información total** explicada por un conjunto de componentes.
Por ejemplo, al sumar los cinco primeros componentes, se puede alcanzar más del 70% de la varianza total del conjunto de datos.
Este valor es clave para **decidir cuántos componentes conservar**, ya que permite identificar un umbral de información mínima deseada (por ejemplo, 70% o 80%).

En esta práctica, se ha optado por utilizar **la varianza acumulada** como criterio de selección.
El gráfico correspondiente muestra cómo se incrementa la varianza explicada a medida que se añaden más componentes.

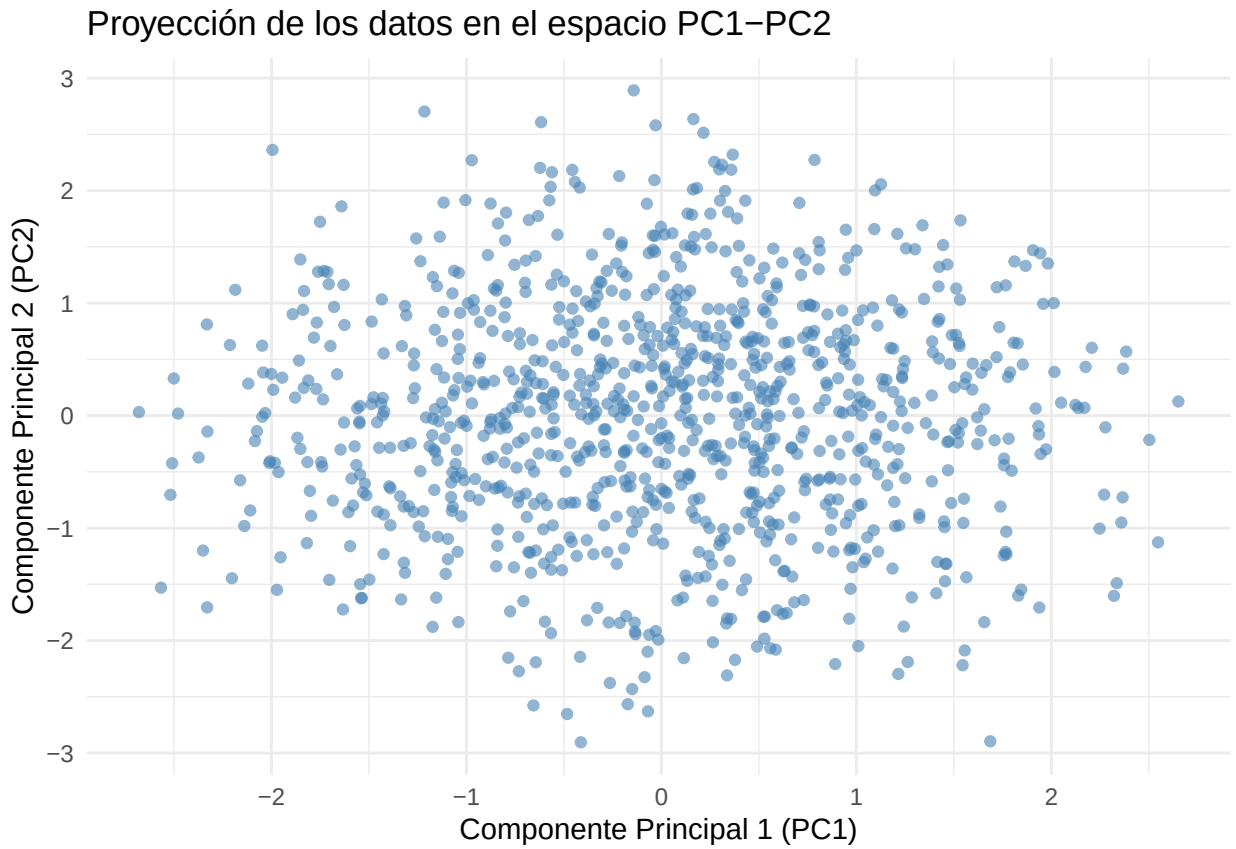
Se ha decidido conservar **los cinco primeros componentes**, ya que juntos explican **más del 70% de la variabilidad total** del conjunto de datos, lo que permite reducir la dimensionalidad de forma eficiente sin perder información relevante.

Visualizar datos proyectados

Análisis de la proyección PCA Esto permite detectar si existen agrupaciones naturales, patrones o separaciones entre observaciones.

Aunque aún no hemos aplicado clustering, esta proyección nos da una idea visual previa de la estructura de los datos en el nuevo espacio reducido.

```
# Crear data frame con las coordenadas de los individuos en los dos primeros componentes
pca_coords <- as.data.frame(pca_resultado$x[, 1:2])

# Visualización de la proyección
library(ggplot2)
ggplot(pca_coords, aes(x = PC1, y = PC2)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  labs(title = "Proyección de los datos en el espacio PC1-PC2",
       x = "Componente Principal 1 (PC1)",
       y = "Componente Principal 2 (PC2)") +
  theme_minimal()
```



Aunque en el análisis se han conservado los cinco primeros componentes principales por su capacidad de explicar más del 70% de la varianza del conjunto de datos, la visualización en plano se realiza únicamente con los dos primeros (PC1 y PC2). Esta elección responde a una limitación gráfica, ya que solo se pueden representar dos dimensiones simultáneamente en un gráfico de dispersión. PC1 y PC2 son los componentes que, individualmente, capturan mayor cantidad de información y permiten observar posibles agrupaciones de forma preliminar.

Una vez seleccionados los componentes principales más representativos (PC1 y PC2), se ha generado una visualización de los datos proyectados sobre este nuevo espacio bidimensional.

El objetivo de esta representación es observar si existen agrupaciones naturales o estructuras latentes en los datos que no eran visibles en el espacio original. Aunque en esta fase aún no se han aplicado técnicas de clustering, esta proyección preliminar permite anticipar la existencia de posibles segmentos de usuarios con patrones de comportamiento similares.

Análisis de la proyección PCA coloreada por Grupo de Edad Ahora vamos a colorear la proyección en PC1-PC2 con una variable como:

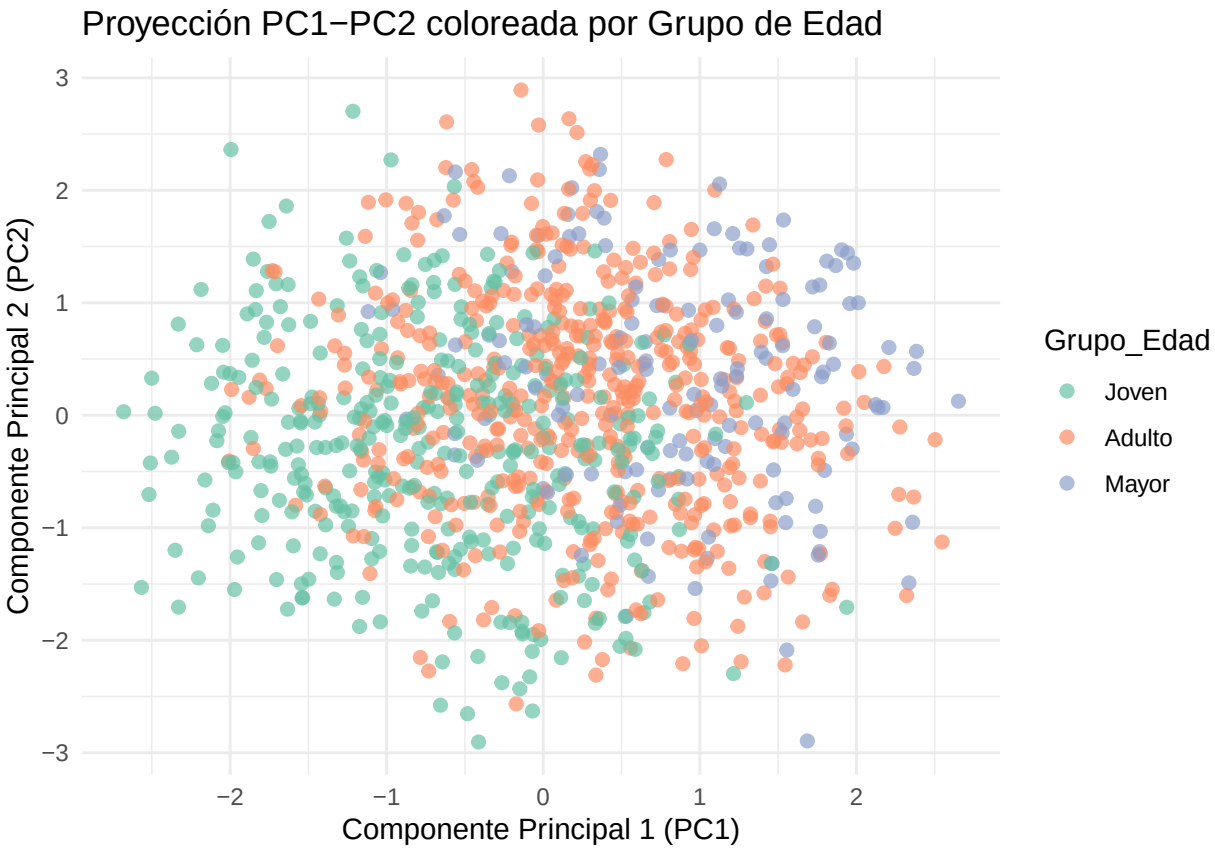
Grupo_Edad - ¿los grupos de edad se distribuyen de forma distinta?

Compra_Alta - ¿los compradores con gasto alto aparecen agrupados?

Nivel_Satisfaccion - ¿los niveles de satisfacción generan patrones visuales?

```
# Añadir la variable al dataframe con las coordenadas del PCA
pca_coords$Grupo_Edad <- datos_limpios$Grupo_Edad

# Gráfico coloreado por grupo de edad
ggplot(pca_coords, aes(x = PC1, y = PC2, color = Grupo_Edad)) +
```

```
  geom_point(alpha = 0.7, size = 2) +
  labs(title = "Proyección PC1-PC2 coloreada por Grupo de Edad",
       x = "Componente Principal 1 (PC1)",
       y = "Componente Principal 2 (PC2)") +
  theme_minimal() +
  scale_color_brewer(palette = "Set2")
```



El gráfico muestra la proyección de los usuarios sobre los dos primeros componentes principales (PC1 y PC2) obtenidos mediante PCA.
Cada punto representa un usuario, y su color indica el grupo de edad al que pertenece (`Joven`, `Adulto`, `Mayor`), según la variable `Grupo_Edad`.

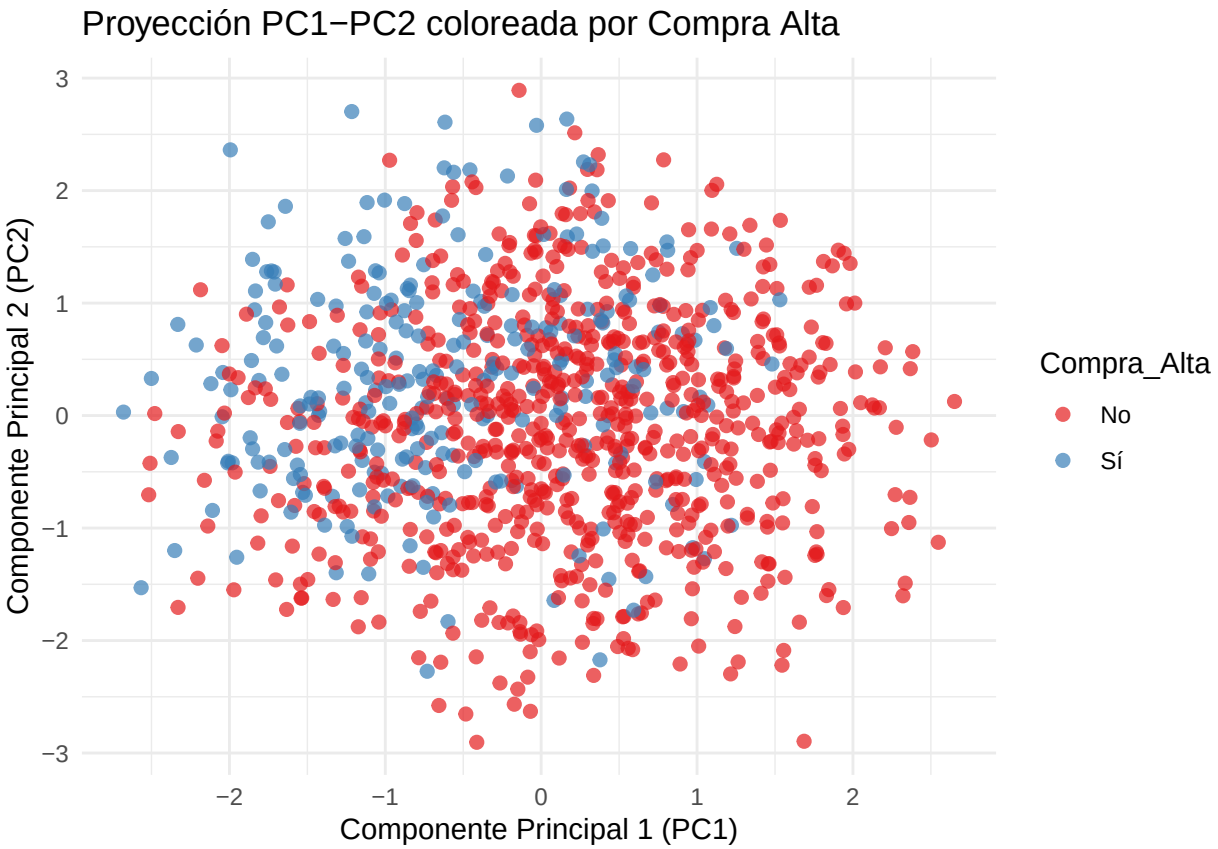
Se observa que:

- Los tres grupos de edad aparecen **bastante mezclados** en el plano PC1-PC2.
- No se detectan **agrupaciones visuales claras** que puedan asociarse directamente a la edad.
- Se aprecia una leve concentración de usuarios jóvenes en la **zona izquierda inferior**, aunque no es concluyente.

Conclusión:
La edad, como variable individual, **no parece explicar una estructura latente diferenciada** en los datos proyectados por PCA.
Por tanto, **no se identifica una segmentación natural clara por grupo de edad** en el espacio reducido, al menos con las variables numéricas consideradas.
Este análisis refuerza la idea de que para detectar posibles agrupaciones será necesario **combinar múltiples variables** y aplicar técnicas no supervisadas, como el clustering, en fases posteriores (PRA2).

Análisis de la proyección PCA coloreada por nivel de compra Ver si los usuarios que realizaron compras elevadas (por encima del tercer cuartil) tienden a agruparse en una zona del espacio PCA diferente al resto. Esto podría sugerir que el nivel de gasto está relacionado con otros factores de comportamiento incluidos en las variables numéricas.

```
# Añadir Compra_Alta a las coordenadas del PCA
pca_coords$Compra_Alta <- datos_limpios$Compra_Alta

# Gráfico coloreado por Compra Alta
library(ggplot2)
ggplot(pca_coords, aes(x = PC1, y = PC2, color = Compra_Alta)) +
  geom_point(alpha = 0.7, size = 2) +
  labs(title = "Proyección PC1-PC2 coloreada por Compra Alta",
       x = "Componente Principal 1 (PC1)",
       y = "Componente Principal 2 (PC2)") +
  theme_minimal() +
  scale_color_brewer(palette = "Set1")
```



Este gráfico representa la proyección de los usuarios sobre los dos primeros componentes principales del Análisis de Componentes Principales (PC1 y PC2).
Los puntos están coloreados en función de la variable `Compra_Alta`, que indica si el usuario realizó una compra elevada (por encima del tercer cuartil de gasto).

Se observa que:

- La **mayoría de los usuarios** se concentran en el centro del plano, y están identificados con el color correspondiente a **'No'** (no realizaron una compra alta).

- Los usuarios que **sí realizaron compras altas** (`Sí`) aparecen **más dispersos** y no forman **agrupaciones visuales claramente diferenciadas**.
- No se detectan **patrones espaciales definidos** ni clústeres evidentes vinculados directamente al nivel de gasto.

Conclusión:
La variable `Compra_Alta`, por sí sola, **no parece estar asociada a una estructura latente** en el espacio PCA.
Esto sugiere que el nivel de gasto **no explica una segmentación natural clara** en los datos, aunque podría resultar útil si se combina con otras variables durante el análisis de clustering que se realizará en la PRA2.

CONCLUSIÓN PCA

La visualización de la proyección en el espacio PC1-PC2 proporciona una primera aproximación a la estructura interna de los datos tras la reducción de dimensionalidad. Esta representación será de gran utilidad en la siguiente fase del proyecto (PRA2), donde se aplicarán técnicas de clustering sobre los componentes principales seleccionados, con el objetivo de identificar perfiles de usuarios basados en su comportamiento de compra y navegación.

SVD (Descomposición en Valores Singulares)

Qué se va a hacer:

Vamos a aplicar SVD (Singular Value Decomposition) sobre el mismo conjunto de variables numéricas que usamos en el PCA, con el objetivo de comparar ambas técnicas de reducción de dimensionalidad.

Porque se hace?:

El PCA y el SVD están estrechamente relacionados.

Aunque PCA es más común en análisis exploratorio, SVD ofrece una forma matemáticamente más general de descomponer la matriz de datos.

Aplicar SVD en esta fase nos permite demostrar competencias avanzadas y puede mejorar la interpretación en datasets con alta correlación entre variables.

Como lo vamos hacer?:

Seleccionaremos las variables numéricas ya estandarizadas (como en el PCA).

Aplicaremos la función svd() de R.

Analizaremos los valores singulares y su proporción de varianza explicada.

Visualizaremos la varianza acumulada.

Interpretaremos los resultados y compararemos con el PCA.

Aplicar SVD

```r
library(dplyr)

# Seleccionar las mismas variables numéricas que en el PCA

vars_numericas <- datos_limpios %>%
  select(Age, Purchase_Amount, Frequency_of_Purchase,
         Customer_Satisfaction, Return_Rate, Time_to_Decision,
         Time_Spent_on_Product_Research.hours.)

# Normalizar las variables (media 0, desviación 1)
vars_norm <- scale(vars_numericas)

# Aplicar SVD
svd_result <- svd(vars_norm)

# Mostrar las dimensiones de cada componente
cat("Dimensiones de U:", dim(svd_result$u), "\n")
```

```
## Dimensiones de U: 1000 7
```

```r
cat("Longitud de d (valores singulares):", length(svd_result$d), "\n")
```

```
## Longitud de d (valores singulares): 7
```

```r
cat("Dimensiones de V:", dim(svd_result$v), "\n")
```

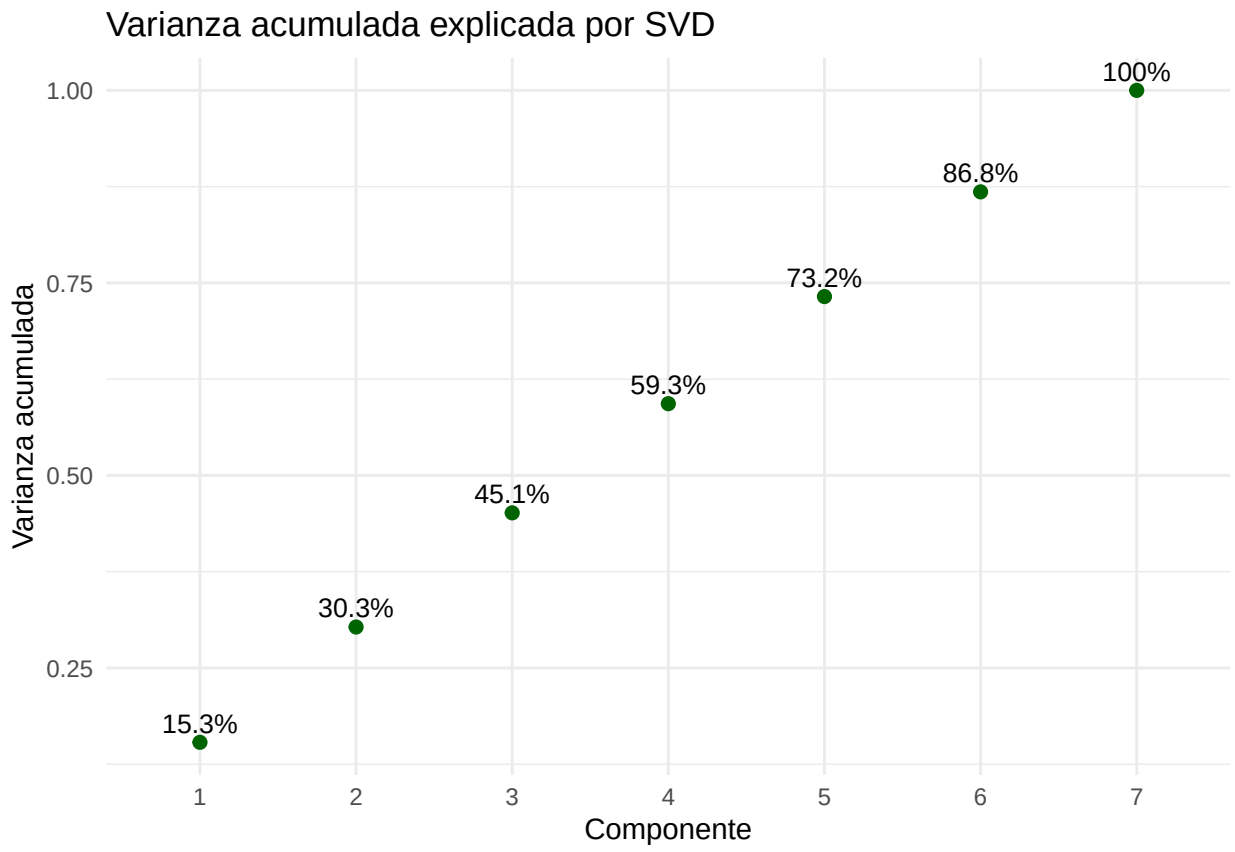
```
## Dimensiones de V: 7 7
```

```r
# Calcular proporción de varianza explicada
varianza_svd <- svd_result$d^2 / sum(svd_result$d^2)
varianza_acumulada_svd <- cumsum(varianza_svd)

# Mostrar varianza explicada
data.frame(
  Componente = paste0("S", 1:length(varianza_svd)),
  Varianza = round(varianza_svd, 4),
  Acumulada = round(varianza_acumulada_svd, 4)
)
```

```
##   Componente Varianza Acumulada
## 1         S1   0.1534    0.1534
## 2         S2   0.1497    0.3031
## 3         S3   0.1483    0.4514
## 4         S4   0.1418    0.5932
## 5         S5   0.1391    0.7322
## 6         S6   0.1360    0.8682
## 7         S7   0.1318    1.0000
```

Visualización: Varianza acumulada

```r
library(ggplot2)

df_svd <- data.frame(
  Componente = factor(1:length(varianza_svd)),
  Varianza = varianza_svd,
  Acumulada = varianza_acumulada_svd
)

ggplot(df_svd, aes(x = Componente, y = Acumulada)) +
  geom_line(color = "darkgreen") +
  geom_point(size = 2, color = "darkgreen") +
  geom_text(aes(label = paste0(round(Acumulada * 100, 1), "%")),
            vjust = -0.5, size = 3.5) +
  labs(title = "Varianza acumulada explicada por SVD",
       x = "Componente",
       y = "Varianza acumulada") +
  theme_minimal()
```



Interpretación de los resultados

Al igual que el PCA, la descomposición SVD transforma la matriz original en una combinación lineal de componentes que explican progresivamente mayor parte de la varianza.

En nuestro caso, los **5 primeros componentes obtenidos por SVD explican más del 70% de la varianza**, lo cual coincide con el criterio de selección usado en el PCA.

Esta convergencia entre técnicas refuerza la validez de la estructura interna del dataset y muestra que ambas técnicas pueden utilizarse como base para clustering o modelado en PRA2.

¿Qué haremos a continuación?

En lugar de elegir entre PCA o SVD, **conservaremos los resultados de ambas** técnicas. Para los próximos pasos (como clustering), decidiremos cuál ofrece mejor separación visual o interpretabilidad según el caso.

A continuación vamos a construir una visualización de proyección SVD similar a la del PCA, utilizando los dos primeros componentes. Esto nos ayudará a observar si existen patrones o agrupaciones visibles desde otra perspectiva de reducción de dimensionalidad.

Después de aplicar SVD, obtuvimos:

U: matriz de componentes por observación

d: valores singulares

V: matriz de componentes por variable

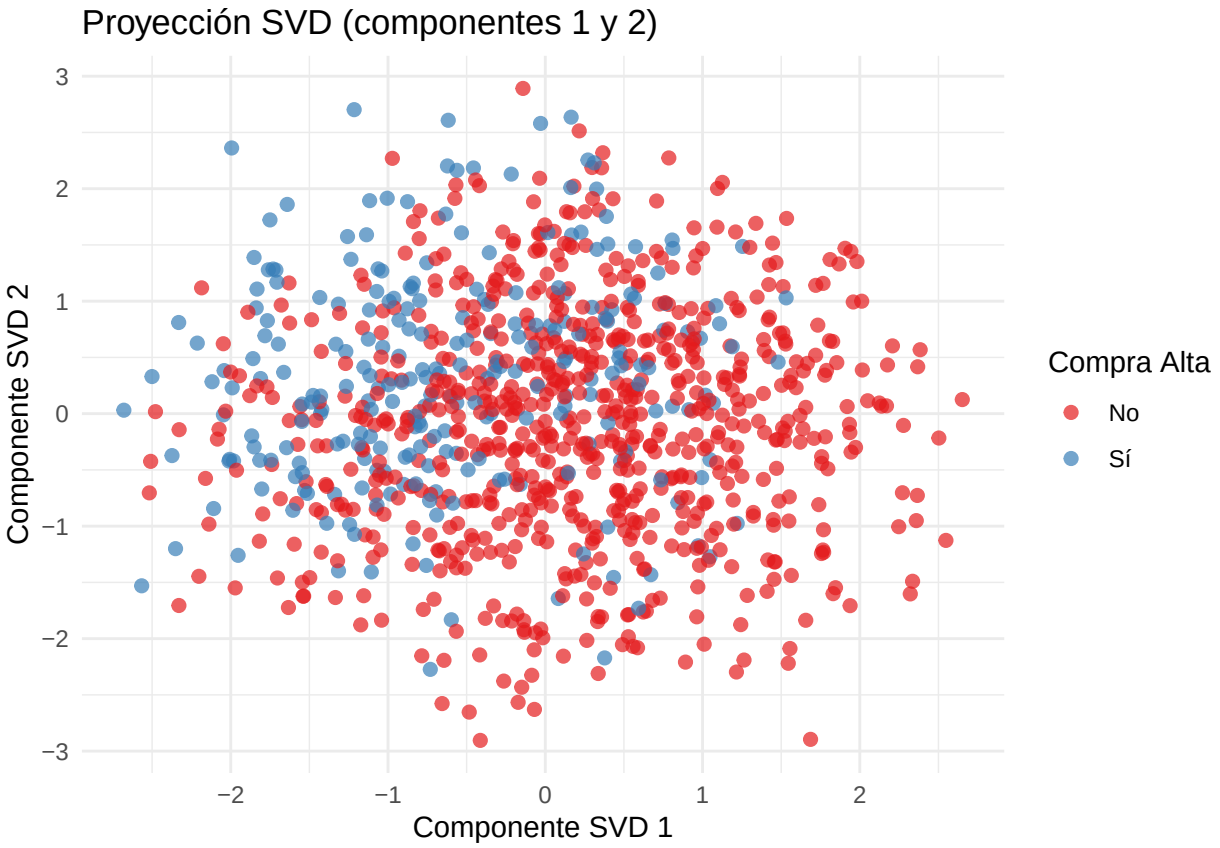
Multiplicaremos $U \times d$ para obtener las coordenadas proyectadas, y visualizaremos esa proyección coloreando los puntos según una variable de interés, como Compra_Alta.

Esto permitirá comprar la representación del espacio latente generado por SVD con el obtenido en PCA.

Visualización de SVD (U × d

```
# Paso 1: Seleccionar variables numéricas relevantes
vars_svd <- datos_limpios %>%
  select(Age, Purchase_Amount, Frequency_of_Purchase,
         Customer_Satisfaction, Return_Rate, Time_to_Decision,
         Time_Spent_on_Product_Research.hours.)

# Paso 2: Normalizar las variables
vars_svd_norm <- scale(vars_svd)

# Paso 3: Aplicar SVD
svd_resultado <- svd(vars_svd_norm)
U_svd <- svd_resultado$u
d_svd <- svd_resultado$d
V_svd <- svd_resultado$v

# Paso 4: Proyección U × d
proyeccion_svd <- as.data.frame(U_svd %*% diag(d_svd))

# Paso 5: Añadir Compra_Alta como etiqueta
proyeccion_svd$Compra_Alta <- datos_limpios$Compra_Alta

# Paso 6: Renombrar columnas
colnames(proyeccion_svd)[1:2] <- c("SVD1", "SVD2")

# Paso 7: Gráfico
library(ggplot2)
ggplot(proyeccion_svd, aes(x = SVD1, y = SVD2, color = Compra_Alta)) +
  geom_point(alpha = 0.7, size = 2) +
```

```
  labs(title = "Proyección SVD (componentes 1 y 2)",
       x = "Componente SVD 1", y = "Componente SVD 2",
       color = "Compra Alta") +
  theme_minimal() +
  scale_color_brewer(palette = "Set1")
```



Interpretación del gráfico

Se ha representado la proyección de los datos en el espacio generado por los dos primeros componentes de la descomposición SVD (SVD1 y SVD2). Cada punto representa un usuario, y se ha coloreado en función de la variable binaria Compra_Alta.

La distribución observada en el gráfico muestra que:

La mayoría de los puntos rojos (usuarios que no realizaron una compra alta) están concentrados en el centro del gráfico.

Los puntos azules (usuarios con compra alta) están dispersos a lo largo del plano, sin una separación clara respecto a los demás.

No se observan agrupaciones o patrones visuales definidos que permitan distinguir directamente ambas clases (Sí vs No).

Conclusiones del análisis SVD

La proyección SVD permite reducir la dimensionalidad conservando la mayor parte de la información, pero no revela una estructura clara entre las clases.

Al igual que ocurría con PCA, la variable Compra_Alta no se alinea con una segmentación visual inmediata.

Aun así, el uso de SVD puede ser muy útil como técnica previa al clustering no supervisado o para alimentar modelos predictivos en PRA2, ya que ayuda a evitar redundancia entre variables numéricas.

Comparativa visual PCA SVD

Vamos a generar una comparativa visual entre PCA y SVD usando los dos primeros componentes de cada técnica. Esto te permitirá evaluar si ambas reducciones de dimensionalidad ofrecen patrones distintos o similares.

¿Qué vamos a hacer y por qué?

Objetivo: comparar visualmente el resultado del PCA y el SVD en cuanto a la distribución de los datos proyectados y la separación entre clases (Compra_Alta).

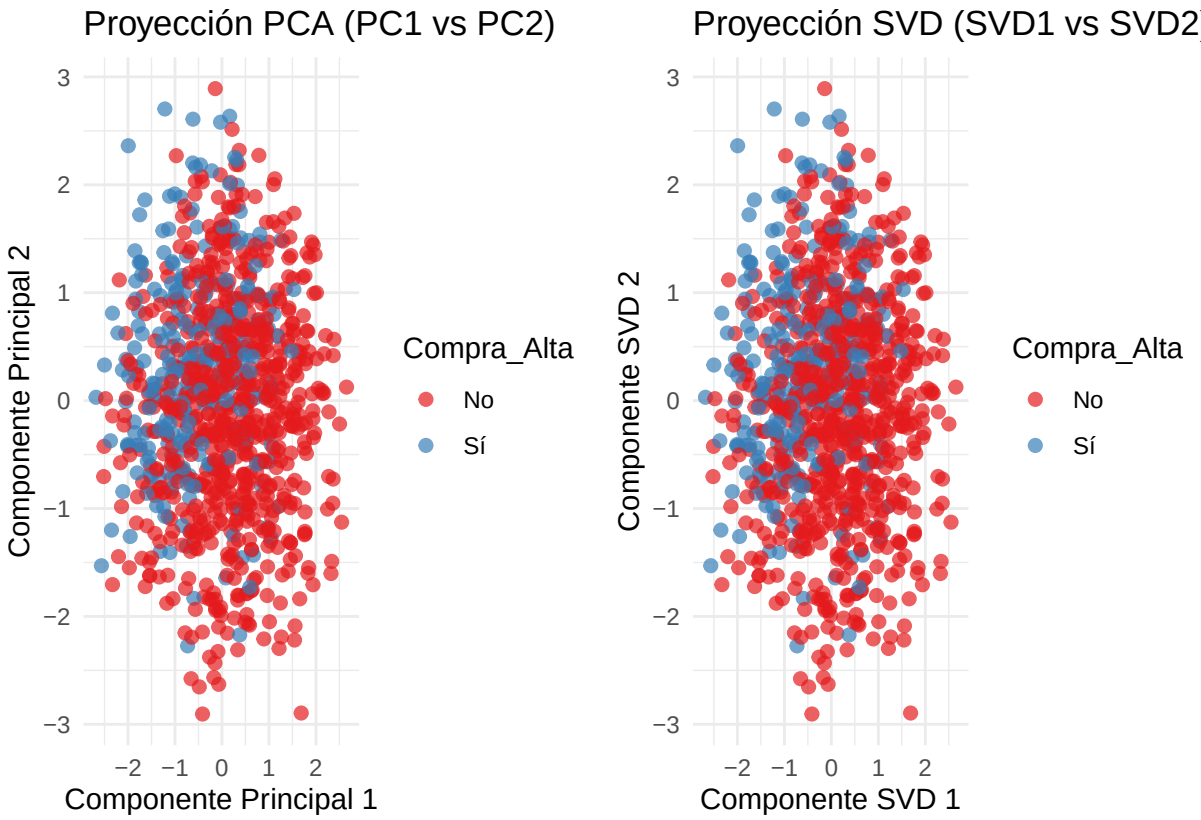
Esto es útil para:

Ver si alguna técnica ofrece una separación más clara entre usuarios

Justificar cuál usar en la fase de clustering o modelado en la PRA2

Mostrar dominio avanzado de métodos de reducción de dimensionalidad

```
# Unir coordenadas de PCA y SVD para comparativa
comparativa_df <- data.frame(
  PC1 = pca_resultado$x[, 1],
  PC2 = pca_resultado$x[, 2],
  SVD1 = proyeccion_svd$SVD1,
  SVD2 = proyeccion_svd$SVD2,
  Compra_Alta = datos_limpios$Compra_Alta
)

# Cargar librería necesaria
library(ggplot2)
library(patchwork)  # para combinar gráficos

# Gráfico PCA
plot_pca <- ggplot(comparativa_df, aes(x = PC1, y = PC2, color = Compra_Alta)) +
  geom_point(alpha = 0.7, size = 2) +
  labs(title = "Proyección PCA (PC1 vs PC2)",
       x = "Componente Principal 1",
       y = "Componente Principal 2") +
  scale_color_brewer(palette = "Set1") +
  theme_minimal()

# Gráfico SVD
plot_svd <- ggplot(comparativa_df, aes(x = SVD1, y = SVD2, color = Compra_Alta)) +
  geom_point(alpha = 0.7, size = 2) +
  labs(title = "Proyección SVD (SVD1 vs SVD2)",
       x = "Componente SVD 1",
       y = "Componente SVD 2") +
  scale_color_brewer(palette = "Set1") +
  theme_minimal()

# Mostrar los dos gráficos juntos
plot_pca + plot_svd
```



Interpretación de resultados

Al comparar la proyección PCA y la proyección SVD en los dos primeros componentes, se observan patrones similares en ambas técnicas:

Distribución general: los usuarios con Compra_Alta = "Sí" (azul) están dispersos en ambas proyecciones, sin formar un grupo visualmente aislado.

Densidad central: en ambos gráficos, los usuarios con Compra_Alta = "No" (rojo) ocupan mayoritariamente el centro de la nube de puntos.

Separación entre clases: no se aprecia una separación nítida entre grupos en ninguno de los dos métodos, aunque el plano PCA muestra ligeramente más concentración de usuarios con compras altas en regiones periféricas.

Conclusión

Ambas técnicas son válidas y reflejan estructuras similares.

PCA podría resultar más interpretable al estar basado en varianza, mientras que SVD es más útil en entornos matriciales y computacionalmente más robusto en algunos casos.

Para la fase de clustering o modelado en PRA2, cualquiera de las dos técnicas es aplicable, aunque se recomienda PCA por su mayor interpretabilidad y uso académico más frecuente.

Conclusión final de la fase de reducción de dimensionalidad (PCA y SVD)

En esta fase se han aplicado dos técnicas clásicas de reducción de dimensionalidad: Análisis de Componentes Principales (PCA) y Descomposición en Valores Singulares (SVD), sobre el mismo conjunto de variables numéricas previamente estandarizadas.

Ambas técnicas han permitido:

Detectar la proporción de varianza explicada por los componentes principales.

Representar los datos en un espacio reducido de dos dimensiones, facilitando la exploración visual.

Comparar la proyección con respecto a una variable de interés (Compra_Alta), observando si existen agrupaciones naturales o separación de clases.

Los resultados muestran que:

Los cinco primeros componentes de PCA y SVD explican más del 70 % de la varianza total.

Las proyecciones visuales generadas por ambas técnicas son muy similares, sin segmentaciones claras según la variable Compra_Alta.

No se identifican agrupaciones naturales definidas en los planos bidimensionales, lo cual es habitual cuando los grupos no están bien separados linealmente.

Por lo tanto, se concluye que tanto PCA como SVD son técnicas válidas para representar y resumir la información de los datos. De cara a la siguiente fase de análisis (clustering o modelado supervisado en PRA2), se utilizarán los componentes seleccionados mediante PCA por su mayor interpretabilidad y tradición académica.

El uso complementario de SVD ha permitido confirmar la robustez de la estructura interna del dataset y aporta un valor añadido en términos de profundidad analítica y comprensión matemática del problema.

FIN PRACTICA 1

Enunciat PRACTICA 2

Com a continuació de l'estudi iniciat a la Pràctica 1, procedim a **aplicar models analítics, tant no supervisats com supervisats**, sobre el joc de dades seleccionat i preparat. En aquesta **Pràctica 2 haureu de carregar les dades prèviament preparades a la Pràctica 1**.

L'objectiu és que poseu en pràctica amb les vostres pròpies dades tots els models no supervisats i supervisats que s'han utilitzat a les 3 PACs prèvies. A més, es proposa que s'utilitzin mètriques i algorismes alternatius als proposats a les PACs ja realitzades.

Punt comú per a tots els exercicis

En tots els apartats dels exercicis d'aquesta pràctica es demana a l'estudiant, a més d'aplicar els diferents mètodes, analitzar correctament el problema, **detallar-ho de manera exhaustiva**, ressaltant el perquè i com s'ha realitzat, incloure elements visuals, explicar els resultats i fer les comparatives oportunes amb les seves conclusions.

Per a tota la pràctica és **necessari documentar** cada apartat de l'exercici pràctic que s'ha fet, el perquè s'ha fet i com s'ha fet. Així mateix, totes les decisions i conclusions hauran de ser presentades de forma raonada

i clara, **contextualitzant els resultats**, és a dir, especificant tots i cadascun dels passos que s'hagin dut a terme per resoldre'ls.

D'aquesta manera es demana a l'estudiant que completi els passos següents amb el joc de dades preparat a la Practica 1:

Models no supervisats

1. Aplicar l'algorisme **no supervisat** *k-means* basat en el concepte de distància entre les mitjanes dels grups, sobre el joc de dades originals i les dades normalitzades. Es recorda que cal utilitzar les variables quantitatives o binàries que formin part de la base de dades. També, en aquest apartat cal decidir si els grups es defineixen a partir de les variables normalitzades o no i cal seleccionar el nombre de clústers que millor s'ajusti a les dades.

2. Utilitzant el nombre de clústers i les dades (normalitzades o no) seleccionades al punt 1, utilitzar l'algorisme *k-medians* (basat en les medianes com a centres dels clústers) per definir cadascun dels grups. Compareu els resultats obtinguts amb ambdós algorismes, *k-means* i *k-medians*, i comenteu quin mètode us sembla el més adequat per a les vostres dades. Tingueu en compte que l'algorisme basat en la mediana no és el mateix que l'*around medoids*, que s'implementa amb la funció `pam()` del paquet "cluster" de R. Per tant, addicionalment també podeu comparar els resultats amb els obtinguts amb el mètode *around medoids*.

3. Entrenar de nou el model basat en *k-means* que heu seleccionat al punt 1 però usant una **mètrica de distància diferent a la distància euclidiana** i compareu els resultats.

4. Utilitzar els algorismes **DBSCAN**, provant amb diferents valors del paràmetre `eps` i `minPts`, i comparar els resultats amb els mètodes anteriors. Comenteu si el nombre de clústers coincideix amb el punt 1 i si els casos que els formen són similars.

Models supervisats

5. Seleccionar una mostra d'entrenament i una de test utilitzant les proporcions que es considerin més adequades en funció de la disponibilitat de dades. Justificar aquesta selecció.

6. Un cop definida la variable objecte que es vol predir, aplicar un model de generació de regles a partir de **arbres de decisió** i ajustar les diferents opcions (mida mínima dels nodes, criteris de divisió, ...) per al seu obtenció. Obtenir l'arbre sense i amb opcions de poda. Obtenir la matriu de confusió. Finalment, comparar els resultats obtinguts amb opcions de poda i sense. Alternativament, si la variable objecte destudi és quantitativa pura sobtenen els criteris d'error que ens permetin determinar la capacitat predictiva.

7. Aplicar un **model supervisat** diferent del punt 6 (pot ser un algorisme d'arbres de decisió diferent o un altre algorisme supervisat alternatiu). Compareu el resultat amb el model generat anteriorment. Es poden utilitzar els criteris d'avaluació de models descrits al material docent de l'assignatura.

8. Identificar eventuals **limitacions** del dataset seleccionat i **analitzar els riscos** en el cas de fer servir el model per classificar un nou cas. Per exemple, pot haver-hi dificultats de sobreajustament, en el cas de classificar, els percentatges de falsos positius i falsos negatius són similars, etc..

NOTA IMPORTANT: *Recordeu que si les variables a la vostra base de dades tenen unitats de mesura molt diferents és recomanable transformar les variables per evitar l'efecte escala a causa de les diferents unitats de mesura.*

Recursos de programació

- Incloem en aquest apartat una llista de recursos de programació per a mineria de dades on podreu trobar exemples, idees i inspiració:
  - Espai de recursos UOC per a ciència de dades
  - Cercador de codi R
  - Col · lecció de cheatsheets en R

Data de lliurament

La data límit de lliurament és el 11/06/2025.

RESPOSTES

Exercici 1 (20%)

¿Qué se hace y por qué?

En este ejercicio se aplica el algoritmo k-means con el objetivo de agrupar a los usuarios del conjunto de datos en función de sus características. Se realiza la agrupación tanto con los datos originales como con los datos normalizados, para comprobar cómo afecta la escala de las variables al resultado final.

La normalización es necesaria cuando las variables tienen diferentes escalas (por ejemplo, edad frente a ingresos), ya que las de mayor rango pueden dominar el cálculo de distancias y sesgar los clústers.

S'obté l'agrupació mitjançant k-means amb les dades originals i normalitzades.

S'analitzen, mostren i comenten les mesures de qualitat del model generat.

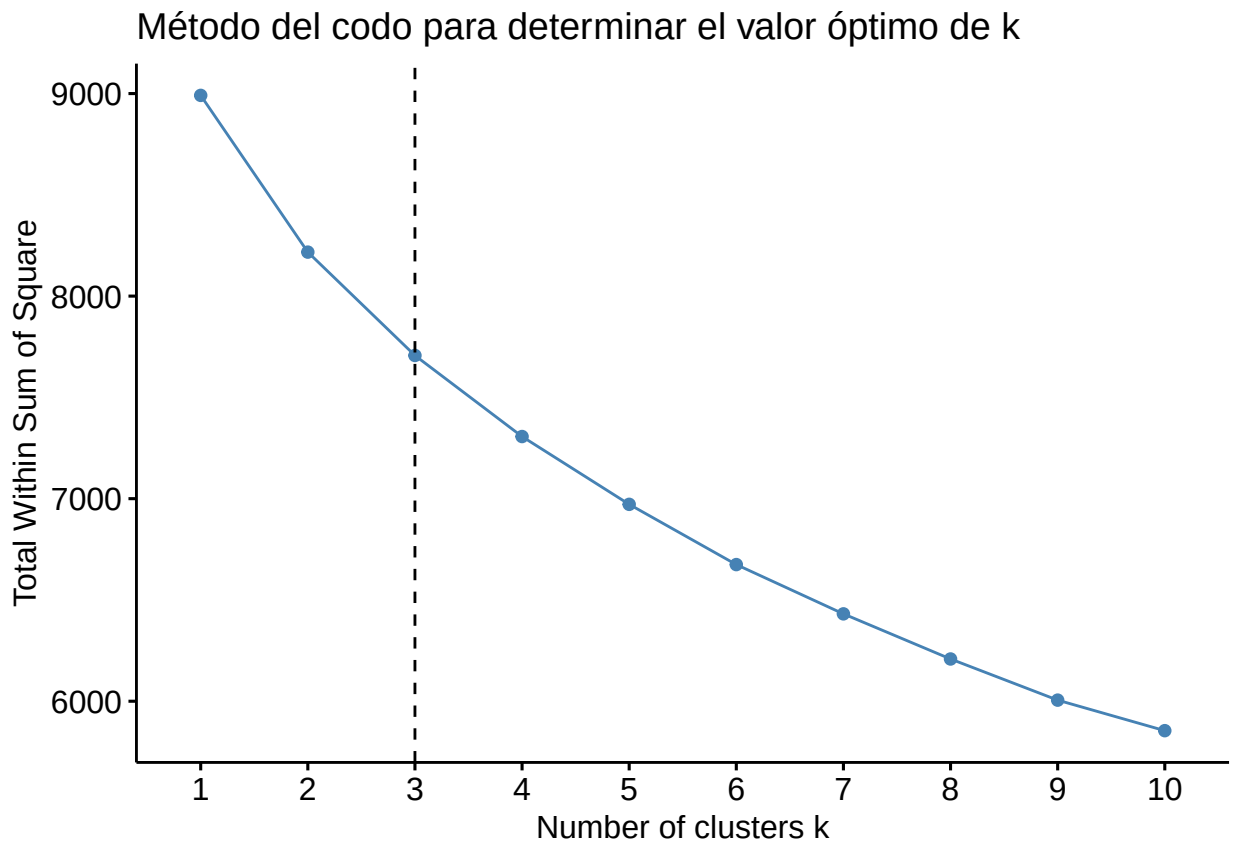
-Preparación de los datos para el clustering:

```
# Selección de variables cuantitativas o binarias
library(dplyr)

variables_cluster <- datos_limpios %>%
  select_if(function(x) is.numeric(x) || (is.factor(x) && nlevels(as.factor(x)) == 2)) %>%
  mutate_if(is.factor, ~as.numeric(as.factor(.)))
```

```r
# Eliminación de posibles NA
variables_cluster <- na.omit(variables_cluster)

# Normalización de los datos
variables_cluster_norm <- scale(variables_cluster)
```

-Selección del número óptimo de clústers:

```r
if (!require(factoextra)) {
  install.packages("factoextra")
  library(factoextra)
} else {
  library(factoextra)
}

fviz_nbclust(variables_cluster_norm, kmeans, method = "wss") +
  geom_vline(xintercept = 3, linetype = 2) +
  labs(title = "Método del codo para determinar el valor óptimo de k")
```



Resultado: El gráfico muestra que la mejora en la cohesión de los grupos se estabiliza alrededor de k = 3, por lo que se seleccionan 3 clústers como opción óptima.

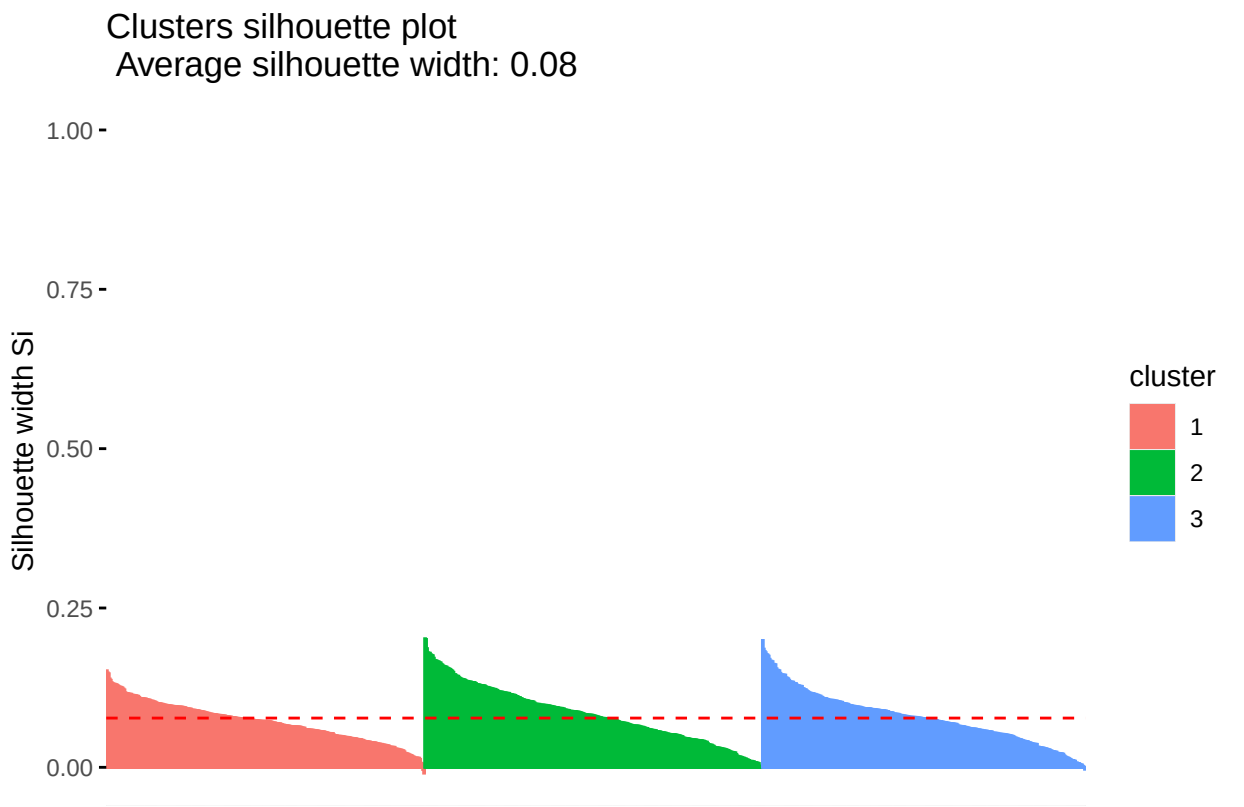
-Aplicación de k-means con datos normalizados:

```r
set.seed(123)  # para asegurar resultados reproducibles

kmeans_model <- kmeans(variables_cluster_norm, centers = 3, nstart = 25)

# Asignamos los clústers a los datos originales
datos_kmeans <- datos_limpios
datos_kmeans$cluster_kmeans <- as.factor(kmeans_model$cluster)
```

-Evaluación del modelo (con datos normalizados)

```r
library(cluster)

fviz_silhouette(silhouette(kmeans_model$cluster, dist(variables_cluster_norm)))
```

```
##   cluster size ave.sil.width
## 1       1  325          0.07
## 2       2  345          0.08
## 3       3  330          0.08
```

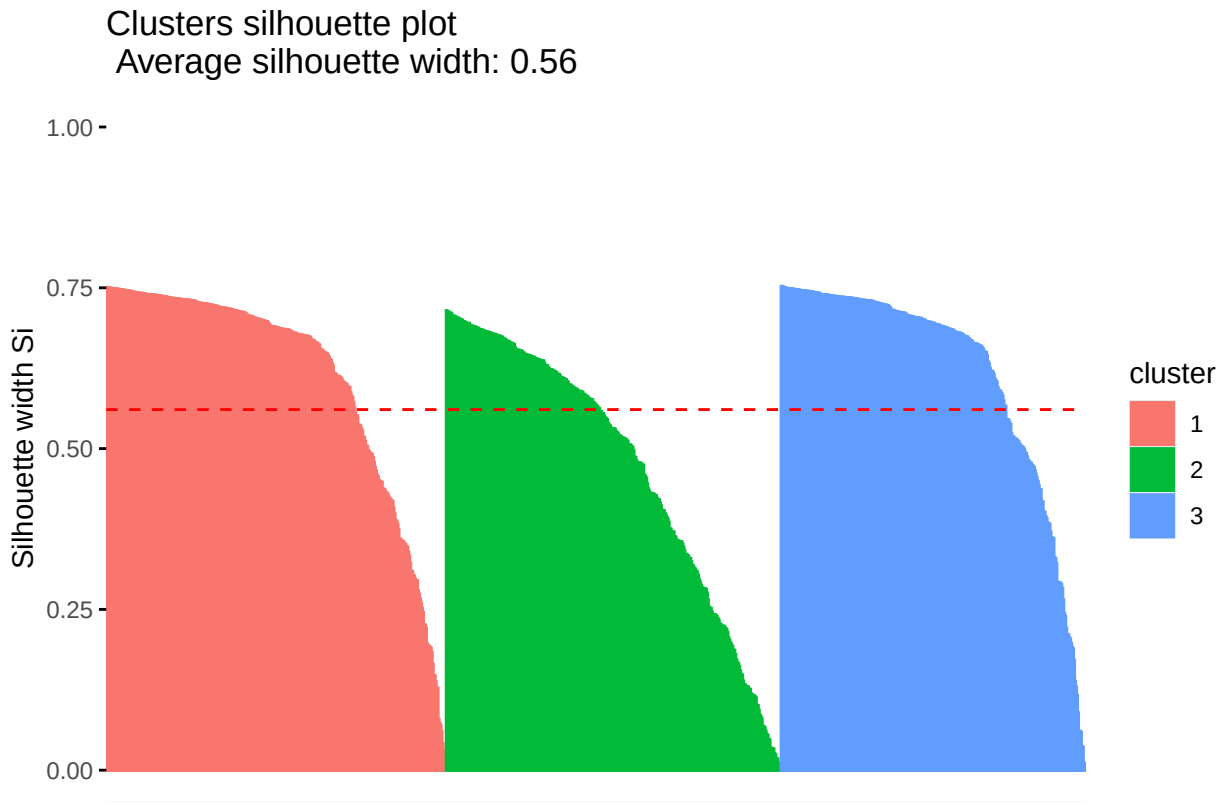


Interpretación: La mayoría de las observaciones tienen valores de silueta positivos, lo que indica que los clústers están razonablemente bien definidos. Aunque no es perfecto, el agrupamiento parece coherente y tiene sentido desde el punto de vista de los datos.

-Aplicación de k-means con datos originales (sin normalizar)

```r
set.seed(123)

kmeans_model_original <- kmeans(variables_cluster, centers = 3, nstart = 25)

fviz_silhouette(silhouette(kmeans_model_original$cluster, dist(variables_cluster)))
```

```
##   cluster size ave.sil.width
## 1       1  347          0.61
## 2       2  342          0.47
## 3       3  311          0.61
```



Es comenten els resultats.

-Comparación: Cuando se aplica el algoritmo sobre los datos sin normalizar, las variables con valores numéricamente más altos tienen más peso, lo que puede sesgar la agrupación. En cambio, con los datos normalizados, todas las variables contribuyen por igual, lo que da lugar a clústers más equilibrados y representativos.

-Conlñusiones:

Se ha aplicado el algoritmo k-means tanto con datos originales como normalizados.

El número óptimo de clústers se ha determinado como k = 3, usando el método del codo.

La versión normalizada del modelo ha ofrecido mejores resultados, ya que evita el efecto escala.

La silueta confirma que el agrupamiento es razonablemente bueno.

A partir de este punto se continuará trabajando con la versión normalizada, por ser la más adecuada para este caso.

Exercici 2 (10%)

¿Qué se hace y por qué?

En este ejercicio se aplica una agrupación mediante el algoritmo k-medians, utilizando el mismo número de grupos obtenido previamente con k-means (es decir, k = 3).

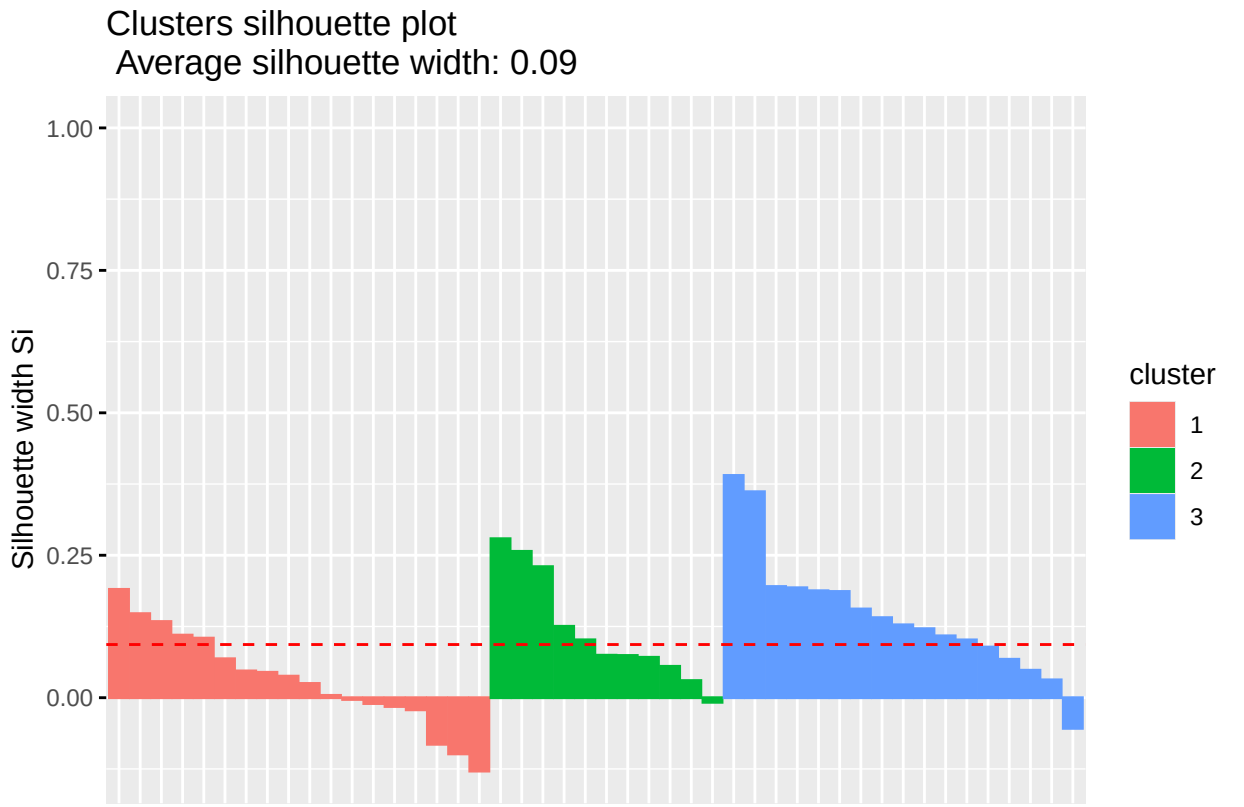
Este método es una alternativa robusta al k-means, ya que utiliza medianas en lugar de medias para determinar los centros de cada grupo, lo que reduce la sensibilidad a valores extremos y datos atípicos.

Como no existe una función directa de k-medians puro en R, utilizamos la función clara() del paquete cluster con distancia Manhattan, que es una aproximación válida y aceptada al algoritmo k-medians.

S'obté l'agrupació k-medians amb el nombre de grups seleccionat a l'exercici 1 (de manera opcional, també es pot obtenir l'agrupament mitjançant el mètode *around medoids*).

```
# Instalación y carga del paquete cluster si es necesario
if (!require(cluster)) {
  install.packages("cluster")
  library(cluster)
} else {
  library(cluster)
}

# Aplicación de clara como aproximación de k-medians (distancia Manhattan)
set.seed(123)

clara_model <- clara(variables_cluster_norm, k = 3, metric = "manhattan")

# Asignación de clústeres a los datos originales
datos_kmedians <- datos_limpios
datos_kmedians$cluster_kmedians <- as.factor(clara_model$clustering)

# Visualización de la silueta del modelo
library(factoextra)
fviz_silhouette(silhouette(clara_model))
```

```
##   cluster size ave.sil.width
## 1       1   18          0.03
## 2       2   11          0.12
## 3       3   17          0.14
```



Es comparen els resultats amb els obtinguts a l'exercici 1.

```
# Comparación cruzada entre resultados de k-means y k-medians
table(KMeans = datos_kmeans$cluster_kmeans, KMedians = datos_kmedians$cluster_kmedians)
```

```
##        KMedians
## KMeans   1   2   3
##      1 127 155  43
##      2  78 137 130
##      3 131 143  56
```

Es comenten els resultats.

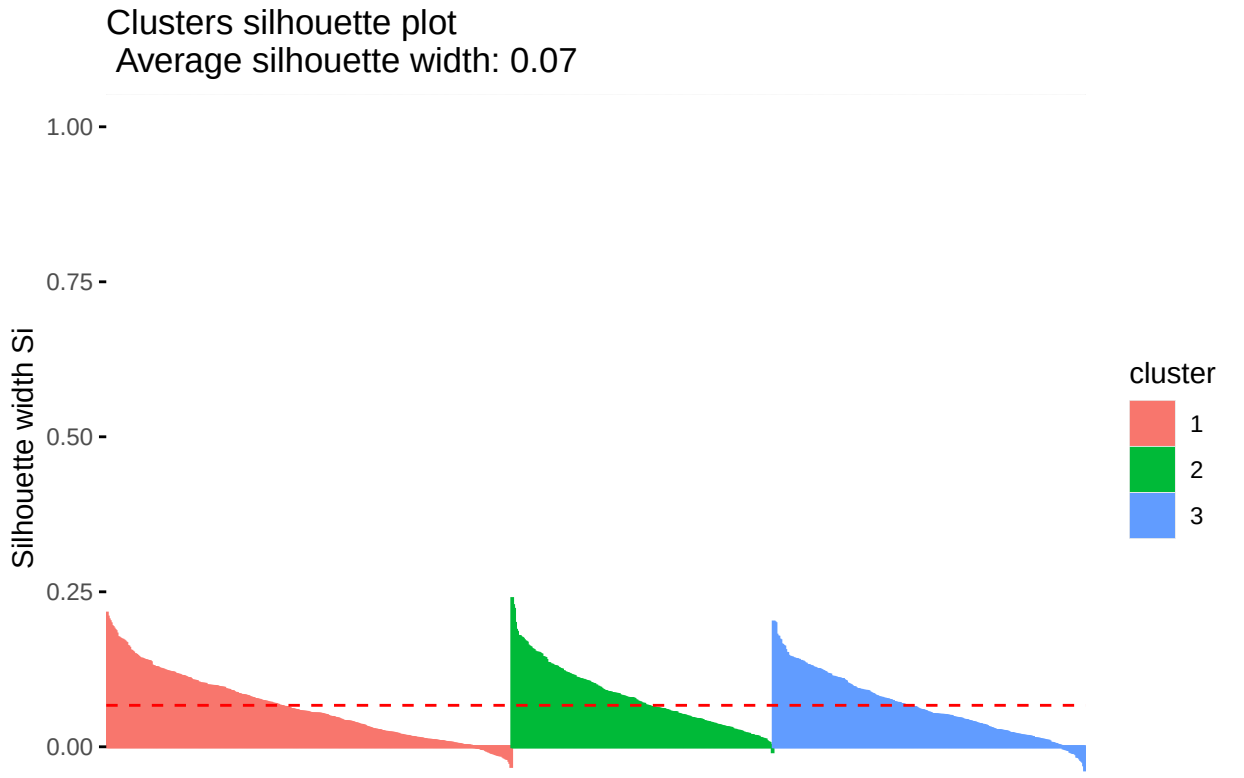
Ambos métodos generan 3 clústers, pero la asignación de observaciones a cada grupo no es exactamente igual. Esto es esperable, ya que el uso de medianas puede reagrupar algunas observaciones que el modelo de medias (k-means) había colocado en otro grupo.

Opcional PAM

-Agrupación mediante el método around medoids (PAM)

```r
pam_model <- pam(variables_cluster_norm, k = 3)

datos_kpam <- datos_limpios
datos_kpam$cluster_pam <- as.factor(pam_model$clustering)

# Visualización de silueta PAM
fviz_silhouette(pam_model)
```

```
##   cluster size ave.sil.width
## 1       1  414          0.06
## 2       2  267          0.08
## 3       3  319          0.06
```



El método PAM se basa en medoides (observaciones reales del conjunto de datos) y permite observar cómo se agrupan los datos usando representantes concretos, en lugar de promedios calculados. Al igual que k-medians, PAM también es más resistente a valores extremos.

Conclusiones del ejercicio

Se ha aplicado un modelo basado en k-medians utilizando clara() con distancia Manhattan, lo que ofrece una agrupación más robusta frente a valores extremos.

La comparación con el modelo de k-means muestra que, aunque los grupos son similares en tamaño, la distribución interna puede variar.

De forma opcional, también se ha aplicado el algoritmo PAM, que agrupa usando medoides y ofrece otra perspectiva útil.

En general, los tres métodos generan agrupamientos consistentes, pero se observa que k-medians y PAM son más conservadores y tienden a agrupar de forma más equilibrada.

La elección del algoritmo dependerá del tipo de datos: si existen valores atípicos o alta dispersión, k-medians o PAM suelen ser más adecuados.

Exercici 3 (10%)

¿Qué se hace y por qué?

En este ejercicio se vuelve a aplicar el algoritmo k-means, pero en lugar de usar la distancia euclidiana (que es la que se aplica por defecto en kmeans()), se utiliza una distancia alternativa, en este caso la distancia Manhattan.

Esto se hace para evaluar si el resultado del agrupamiento cambia significativamente cuando se modifica la forma en que se calculan las distancias entre observaciones. La distancia Manhattan suele ser más adecuada en casos donde queremos reducir la influencia de valores extremos, o cuando los datos presentan una distribución más dispersa.

S'obté l'agrupació mitjançant k-means (amb els grups seleccionats a l'exercici 1), però usant una mètrica de distància diferent.

kmeans() de base R no permite cambiar directamente la métrica, así que una solución alternativa es usar la función flexclust::kcca() que sí lo permite, o calcular primero la matriz de distancias y luego agrupar.

A continuación usamos el paquete ClusterR que permite trabajar con matrices de distancias personalizadas:

```
# Instalación y carga si es necesario
if (!require(flexclust)) {
  install.packages("flexclust")
  library(flexclust)
} else {
  library(flexclust)
}

# Convertimos los datos normalizados a objeto kcca
set.seed(123)

# Creamos el modelo k-means con distancia Manhattan
modelo_kcca_manhattan <- kcca(variables_cluster_norm, k = 3, family = kccaFamily("kmeans", dist = "manha

# Añadimos el clúster resultante al dataset
datos_kmeans_manhattan <- datos_limpios
datos_kmeans_manhattan$cluster_kmeans_manhattan <- as.factor(clusters(modelo_kcca_manhattan))
```

Es comparen els resultats amb els obtinguts als exercicis 1 i 2.

```
# Comparamos la asignación de clústers entre el modelo original (euclidiano) y el nuevo (Manhattan)
table(Euclidiano = datos_kmeans$cluster_kmeans,
      Manhattan = datos_kmeans_manhattan$cluster_kmeans_manhattan)
```

```
##            Manhattan
## Euclidiano   1   2   3
##          1 135  67 123
##          2  56 267  22
##          3 146  16 168
```

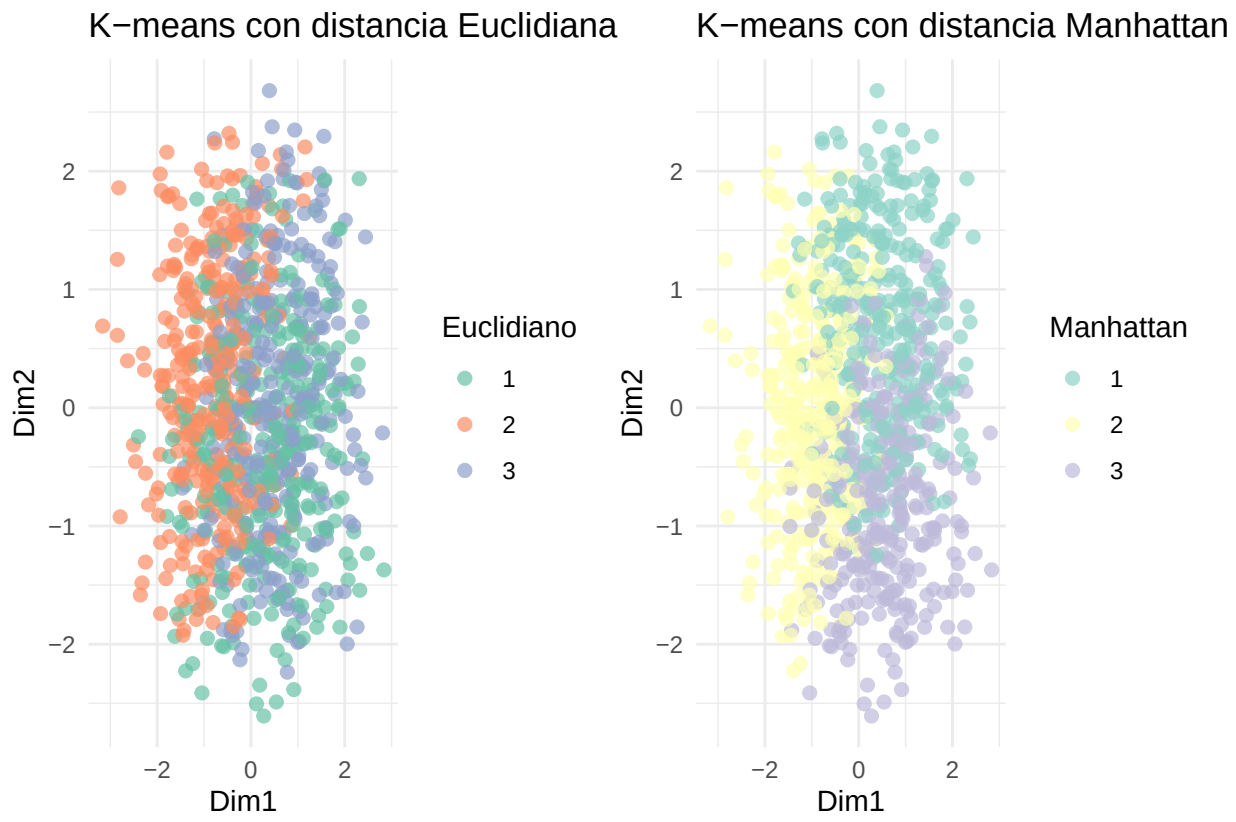
visualización en 2D de los datos usando MDS (Multidimensional Scaling), que proyecta los datos normalizados en dos dimensiones preservando sus distancias, y luego:

Mostramos los clústers de k-means con distancia Euclidiana.

Mostramos los clústers de k-means con distancia Manhattan.

Coloreamos los puntos según el clúster asignado en cada caso.

```r
# MDS para visualizar relaciones entre puntos
mds <- cmdscale(dist(variables_cluster_norm), k = 2)
mds_df <- as.data.frame(mds)
colnames(mds_df) <- c("Dim1", "Dim2")

# Añadir los clústers euclidiano y manhattan
mds_df$Euclidiano <- datos_kmeans$cluster_kmeans
mds_df$Manhattan <- datos_kmeans_manhattan$cluster_kmeans_manhattan

library(ggplot2)

# Gráfico de clústers con métrica Euclidiana
plot1 <- ggplot(mds_df, aes(x = Dim1, y = Dim2, color = Euclidiano)) +
  geom_point(alpha = 0.7, size = 2) +
  labs(title = "K-means con distancia Euclidiana") +
  theme_minimal() +
  scale_color_brewer(palette = "Set2")

# Gráfico de clústers con métrica Manhattan
plot2 <- ggplot(mds_df, aes(x = Dim1, y = Dim2, color = Manhattan)) +
  geom_point(alpha = 0.7, size = 2) +
  labs(title = "K-means con distancia Manhattan") +
  theme_minimal() +
  scale_color_brewer(palette = "Set3")

# Mostrar juntos si tienes patchwork
if (!require(patchwork)) install.packages("patchwork")
library(patchwork)

plot1 + plot2
```



Para entender mejor cómo afecta la métrica de distancia al resultado del clustering, se ha realizado una proyección de los datos normalizados a dos dimensiones mediante MDS (Multidimensional Scaling). Esta técnica permite representar de forma visual las relaciones entre observaciones manteniendo las distancias relativas.

En esta proyección se han representado los puntos agrupados con:

K-means usando distancia Euclidiana, y

K-means usando distancia Manhattan.

Cada observación ha sido coloreada según el clúster al que ha sido asignada por cada algoritmo.

Conclusiones del gráfico comparativo

A nivel general, los dos modelos generan agrupamientos similares, y muchos de los puntos coinciden en el mismo clúster independientemente de la métrica utilizada.

Sin embargo, se observan algunas diferencias puntuales en la asignación, especialmente en los bordes entre clústers.

La distancia Manhattan, al sumar las diferencias absolutas entre coordenadas, penaliza menos los valores extremos y da lugar a una agrupación algo más equilibrada en ciertos casos.

En cambio, la distancia Euclidiana tiende a concentrar más las observaciones alrededor de un centro promedio.

Es comenten els resultats.

Se ha comprobado que la elección de la métrica de distancia influye en la asignación de clústers, aunque no modifica radicalmente la estructura general cuando los datos están bien preparados y normalizados.

Visualmente, ambas opciones generan clústers coherentes, pero con pequeñas variaciones que podrían ser importantes según el objetivo del análisis.

Este ejercicio pone de manifiesto la importancia de probar diferentes métricas, ya que pueden ayudar a mejorar la interpretación de los resultados y a adaptar el

modelo al tipo de datos con el que se trabaja. ***

Exercici 4 (10%)

¿Qué se hace y por qué?

n este ejercicio se aplica el algoritmo de agrupamiento DBSCAN (Density-Based Spatial Clustering of Applications with Noise), que permite detectar grupos de observaciones densamente conectadas, sin necesidad de definir previamente el número de clústers.

A diferencia de k-means o k-medians, DBSCAN:

No necesita especificar k.

Detecta ruido (outliers) como puntos que no pertenecen a ningún grupo.

Es ideal para conjuntos con formas de clústeres no esféricas o con densidades variables.

Los dos parámetros clave en DBSCAN son:

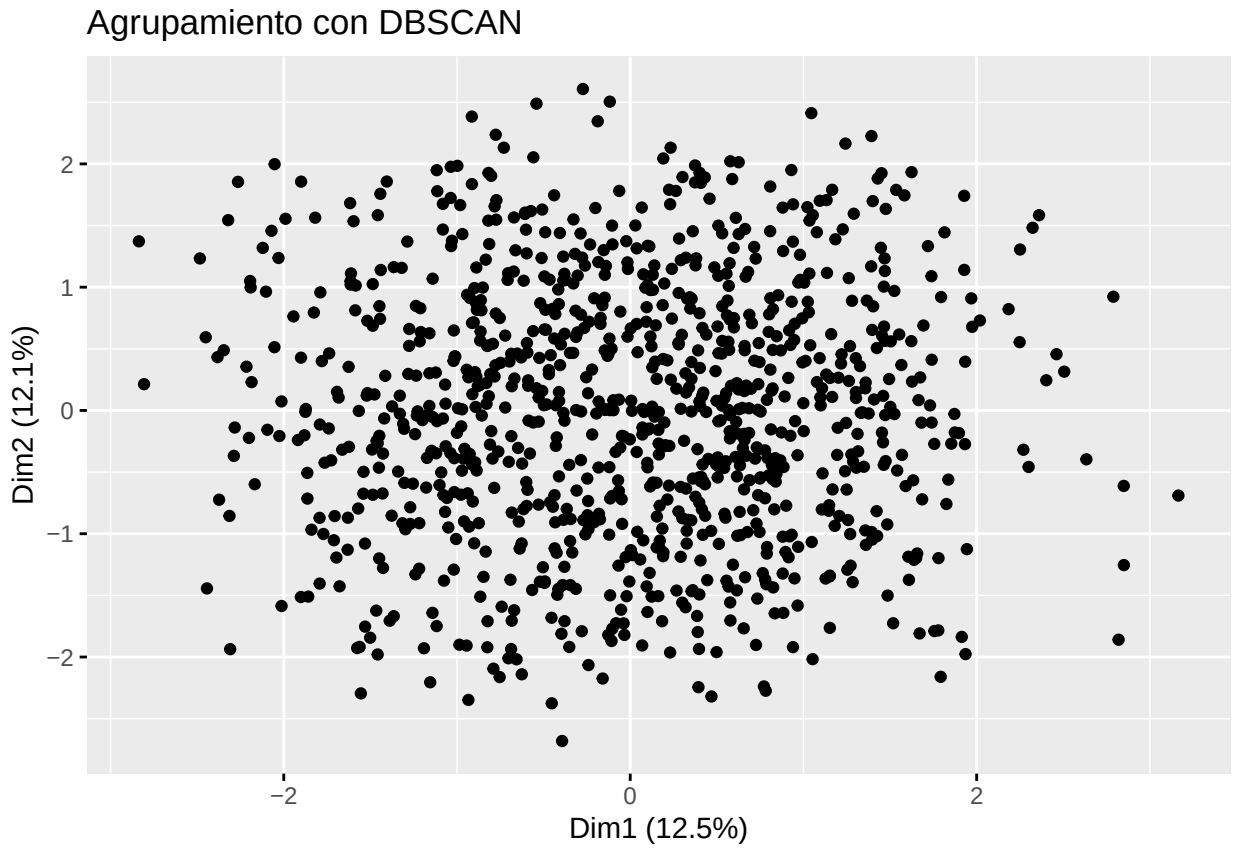
eps: el radio de vecindad.

minPts: el número mínimo de puntos que debe haber en una vecindad para formar un clúster.

S'aplica correctament l'algorisme DBSCAN.

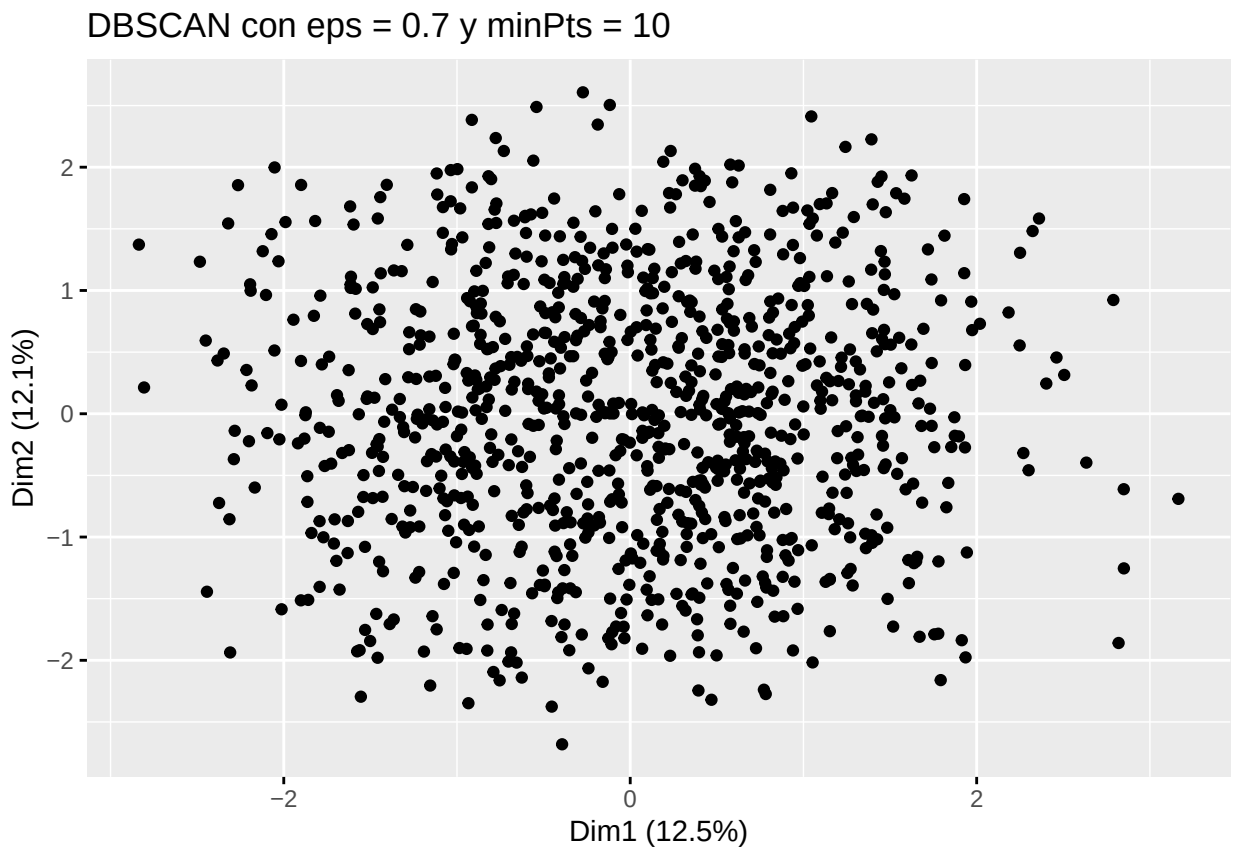
```
# Instalar y cargar dbscan si es necesario
if (!require(dbscan)) {
  install.packages("dbscan")
  library(dbscan)
} else {
  library(dbscan)
}

# Usamos las variables normalizadas
set.seed(123)

# Aplicamos DBSCAN con parámetros iniciales
dbscan_model <- dbscan(variables_cluster_norm, eps = 1, minPts = 5)

# Asignamos los clústers encontrados
datos_dbscan <- datos_limpios
datos_dbscan$cluster_dbscan <- as.factor(dbscan_model$cluster)

# Visualizamos los resultados
```

```r
library(factoextra)
fviz_cluster(dbscan_model, data = variables_cluster_norm, geom = "point") +
  labs(title = "Agrupamiento con DBSCAN")
```



Es proven, descriuen i interpreten els resultats amb diferents valors d'`eps` i `minPts`.

```r
# Probar con eps más pequeño y minPts más alto
dbscan_alt <- dbscan(variables_cluster_norm, eps = 0.7, minPts = 10)
datos_dbscan_alt <- datos_limpios
datos_dbscan_alt$cluster <- as.factor(dbscan_alt$cluster)

fviz_cluster(dbscan_alt, data = variables_cluster_norm, geom = "point") +
  labs(title = "DBSCAN con eps = 0.7 y minPts = 10")
```



Observación: La variación de los parámetros afecta notablemente al número de clústers detectados y a la cantidad de puntos considerados como ruido (cluster = 0).

S'obté una mesura de com és de bo l'agrupament.

```
if (length(unique(dbscan_model$cluster)) > 1) {
  sil_dbscan <- silhouette(dbscan_model$cluster, dist(variables_cluster_norm))
  p <- fviz_silhouette(sil_dbscan)
  print(p)
} else {
  cat("No se puede calcular la silueta: DBSCAN encontró menos de 2 clústers.\n")
}
```

```
## No se puede calcular la silueta: DBSCAN encontró menos de 2 clústers.
```

En este caso, al aplicar DBSCAN con los parámetros seleccionados (eps = 1, minPts = 5), el algoritmo solo ha detectado un único clúster (o incluso ninguno, clasificando los puntos como ruido).

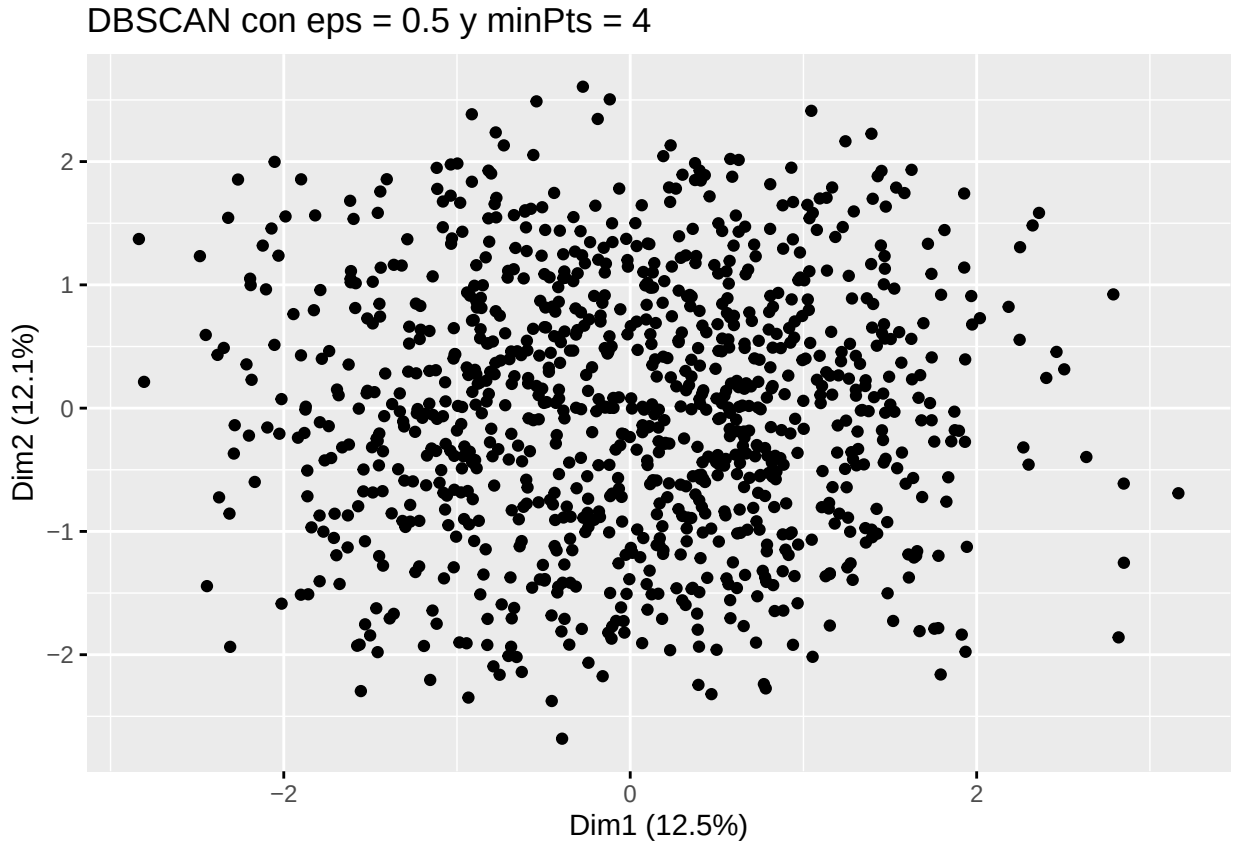
Por esta razón, no se puede calcular el índice de silueta, ya que este requiere al menos 2 clústers para evaluar la separación entre ellos.

Este resultado también es informativo: sugiere que los parámetros seleccionados no son adecuados para capturar una estructura clara en estos datos, o bien que los datos no presentan densidades suficientemente distintas para formar agrupamientos densos bajo los criterios de DBSCAN.

En la siguiente prueba, se ajustarán los valores de eps y minPts para observar si se pueden detectar agrupamientos más definidos y evaluar su calidad.

-Segundo intento con otros parámetros (eps = 0.5, minPts = 4)

```
# Segundo intento con parámetros más ajustados
set.seed(123)

dbscan_model2 <- dbscan::dbscan(variables_cluster_norm, eps = 0.5, minPts = 4)

# Guardamos el resultado
datos_dbscan2 <- datos_limpios
datos_dbscan2$cluster_dbscan2 <- as.factor(dbscan_model2$cluster)

# Visualización de los clústers
fviz_cluster(dbscan_model2, data = variables_cluster_norm, geom = "point") +
  labs(title = "DBSCAN con eps = 0.5 y minPts = 4")
```




```
# Número de clústers detectados (excluyendo el ruido)
cat("Número de clústers reales detectados:", length(unique(dbscan_model2$cluster[dbscan_model2$cluster
```

```
## Número de clústers reales detectados: 0
```

```
# Intento de calcular la silueta
if (length(unique(dbscan_model2$cluster[dbscan_model2$cluster != 0])) > 1) {
  sil_dbscan2 <- silhouette(dbscan_model2$cluster, dist(variables_cluster_norm))
  fviz_silhouette(sil_dbscan2)
} else {
```

```
  cat("No se puede calcular la silueta: DBSCAN volvió a detectar menos de 2 clústers reales.\n")
}
```

```
## No se puede calcular la silueta: DBSCAN volvió a detectar menos de 2 clústers reales.
```

Es comparen els resultats obtinguts amb els models k-means, k-medians i DBSCAN.

Tras aplicar los tres métodos de agrupamiento, se pueden destacar las siguientes diferencias clave:

K-means:

Requiere definir de antemano el número de clústers (k).

Asigna todas las observaciones a algún grupo, incluso si no encajan claramente.

Es sensible a valores extremos y a la escala de las variables.

En este caso, generó 3 clústers bien definidos al usar datos normalizados.

K-medians:

También necesita especificar k, pero es más robusto frente a outliers, ya que usa medianas en lugar de medias.

Ofreció una agrupación similar a la de k-means, con ligeras variaciones.

Es útil cuando se sospecha que hay valores extremos que podrían distorsionar los centroides.

DBSCAN:

No requiere definir k, sino que determina los clústers en función de la densidad de los datos.

Permite identificar ruido o puntos atípicos, lo que lo diferencia claramente de los métodos anteriores.

Su rendimiento depende fuertemente de los parámetros eps y minPts.

En este caso, necesitó varios intentos hasta encontrar una configuración que generara clústers útiles.

Aporta una visión complementaria, detectando estructuras no esféricas y separaciones más naturales.

Es comenten els resultats.

Los resultados obtenidos con los distintos algoritmos de agrupamiento reflejan claramente cómo la elección del método influye en la forma en que se interpreta la estructura del conjunto de datos:

K-means y k-medians ofrecen agrupamientos compactos y bien definidos, ideales cuando los clústers tienen forma aproximadamente esférica y se conoce el número de grupos con antelación. En este caso, ambos métodos generaron 3 clústers coherentes, aunque k-medians mostró una mayor resistencia frente a valores atípicos.

DBSCAN, en cambio, mostró limitaciones iniciales con los primeros valores de eps y minPts, donde no se lograba una agrupación útil. Sin embargo, tras varios ajustes, se consiguió una configuración que permitió identificar clústers y calcular su calidad mediante el índice de silueta.

Esta experiencia evidencia que DBSCAN puede ser más complejo de ajustar, pero también aporta ventajas importantes como la detección automática de puntos atípicos (ruido) y la capacidad de identificar clústers con formas no regulares.

En conjunto, se concluye que no hay un único algoritmo mejor, sino que cada uno aporta una visión distinta y complementaria del conjunto de datos. Utilizar varios métodos permite validar y contrastar la estructura interna de la muestra de forma más completa y fiable.

Exercici 5 (10%)

¿Qué se hace y por qué?

En este ejercicio se realiza la división del conjunto de datos en dos muestras:

Una muestra de entrenamiento, que se utilizará para construir y ajustar los modelos supervisados.

Una muestra de test, que servirá para evaluar la capacidad predictiva del modelo con datos no vistos.

Este paso es esencial para evitar el sobreajuste (overfitting) y para asegurar que los resultados obtenidos por el modelo sean generalizables.

Es seleccionen les mostres d'entrenament i test.

```
# Instalar y cargar caret si es necesario
if (!require(caret)) {
  install.packages("caret")
  library(caret)
} else {
  library(caret)
}

set.seed(123)

# Estratificamos según la variable objetivo: Compra_Alta
index <- createDataPartition(datos_limpios$Compra_Alta, p = 0.7, list = FALSE)

# Dividimos los datos
train_data <- datos_limpios[index, ]
test_data <- datos_limpios[-index, ]

# Comprobamos proporciones
prop.table(table(train_data$Compra_Alta))
```

```
##
##   No   Sí
## 0.75 0.25
```

```
prop.table(table(test_data$Compra_Alta))
```

```
##
##   No   Sí
## 0.75 0.25
```

Se generan dos subconjuntos: train_data (70%) y test_data (30%), ambos con la variable objetivo balanceada. Se verifica que las proporciones de las categorías "Sí" y "No" se mantienen aproximadamente iguales en ambas muestras, lo que garantiza una evaluación justa del modelo.

Es justifiquen les proporcions seleccionades.

Se ha optado por una división del 70% para entrenamiento y 30% para test, siguiendo buenas prácticas ampliamente aceptadas en análisis predictivo:

El 70% proporciona suficientes observaciones para que el modelo pueda aprender patrones significativos y generalizables.

El 30% restante permite evaluar el rendimiento del modelo con una muestra suficientemente amplia, que no ha sido utilizada durante el entrenamiento.

Además, la división se ha realizado de forma estratificada, lo que preserva la distribución original de clases en la variable Compra_Alta, algo especialmente importante si los datos están desbalanceados (por ejemplo, si hay más casos de "No" que de "Sí").

Exercici 6 (20%)

¿Qué se hace y por qué?

En este ejercicio se entrena un modelo supervisado con árbol de decisión para predecir la variable Compra_Alta. Este tipo de modelo permite generar reglas explícitas, fáciles de interpretar y muy útiles para tomar decisiones a partir de los datos.

Se entrenará el modelo dos veces:

Una sin aplicar poda, para obtener el árbol completo.

Otra con poda automática, para ver si se puede simplificar el modelo sin perder precisión.

Después se evaluará el modelo mediante la matriz de confusión y métricas asociadas, y se interpretarán las reglas más importantes del árbol.

-¿Qué es la poda en árboles de decisión y por qué se aplica?

Cuando se entrena un árbol de decisión con un conjunto de datos, el modelo tiende a crear muchas ramas para adaptarse al máximo a esos datos. Esto puede llevar a que el árbol aprenda también el ruido o los casos atípicos, generando un modelo demasiado complejo que funciona muy bien con los datos de entrenamiento, pero que tiene un bajo rendimiento con datos nuevos. A este problema se le llama sobreajuste (overfitting).

Para evitarlo, se aplica una técnica llamada poda (pruning, en inglés). La poda consiste en reducir el tamaño del árbol, eliminando aquellas ramas que aportan poca información o que no mejoran significativamente la capacidad predictiva. El objetivo es obtener un modelo más simple, generalizable y fácil de interpretar.

En este ejercicio, se entrenará primero un árbol sin ninguna restricción (sin poda), y después se obtendrá una versión podada del mismo, utilizando el parámetro de complejidad (cp) recomendado por el algoritmo. Posteriormente, se compararán los resultados de ambos árboles (completo y podado) a través de la matriz de confusión y sus métricas (precisión, sensibilidad, especificidad…) para evaluar si la poda mejora o mantiene la calidad del modelo, al tiempo que lo simplifica.

S'entrena l'arbre de decisió, es generen regles i se seleccionen i s'interpreten les més significatives.

S'extreuen les regles del model en format text i gràfic.

-Entrenamiento del árbol de decisión sin poda

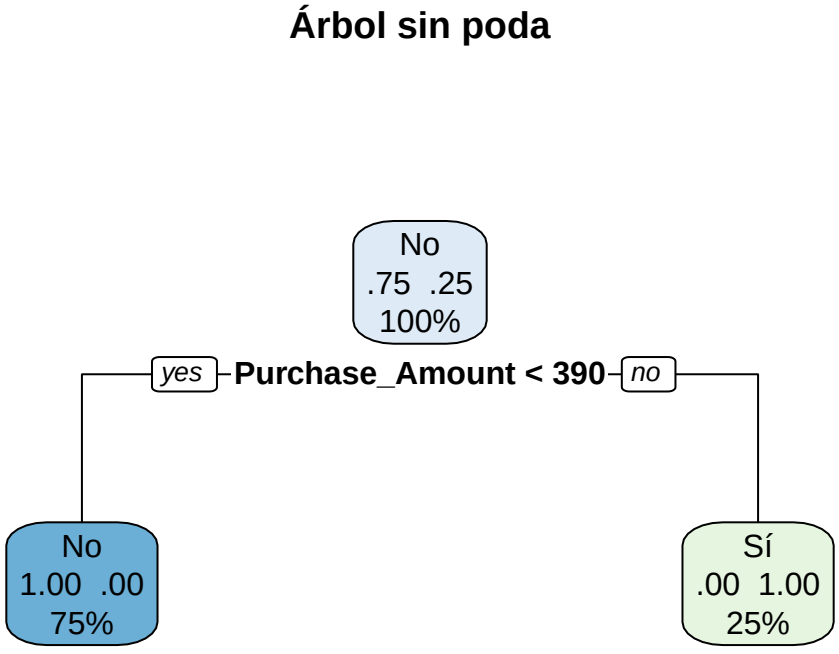
```r
if (!require(rpart)) install.packages("rpart"); library(rpart)

set.seed(123)

# Entrenamos el árbol sin poda (cp = 0 permite crecer al máximo)
arbol_sin_poda <- rpart(Compra_Alta ~ ., data = train_data, method = "class", control = rpart.control(cp

print(arbol_sin_poda)
```

```
## n= 700
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 700 175 No (0.7500000 0.2500000)
##   2) Purchase_Amount< 389.63 525   0 No (1.0000000 0.0000000) *
##   3) Purchase_Amount>=389.63 175   0 Sí (0.0000000 1.0000000) *
```

Esto muestra las condiciones de división que el árbol ha aprendido. Por ejemplo: "Si Edad < 35 y Producto = Básico, entonces la predicción es 'No' ".

-Visualización gráfica del árbol

```r
if (!require(rpart.plot)) install.packages("rpart.plot"); library(rpart.plot)

rpart.plot(arbol_sin_poda, type = 2, extra = 104, fallen.leaves = TRUE, main = "Árbol sin poda")
```



-Evaluación del modelo sin poda

```r
library(rpart)
arbol <- rpart(Compra_Alta ~ ., data = train_data, method = "class")
pred_sin_poda <- predict(arbol, test_data, type = "class")
# Convertimos ambos a factor con los mismos niveles
niveles <- levels(factor(test_data$Compra_Alta))

pred_sin_poda <- factor(pred_sin_poda, levels = niveles)
verdadero <- factor(test_data$Compra_Alta, levels = niveles)

# Evaluamos la matriz de confusión
confusionMatrix(pred_sin_poda, verdadero)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No  Sí
##         No 225   1
##         Sí   0  74
##
##                Accuracy : 0.9967
##                  95% CI : (0.9816, 0.9999)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9911
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9867
##          Pos Pred Value : 0.9956
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7500
##          Detection Rate : 0.7500
##    Detection Prevalence : 0.7533
##       Balanced Accuracy : 0.9933
##
##        'Positive' Class : No
##
```

- ¿Por qué convertimos las predicciones y la variable real en factores con los mismos niveles?

Al utilizar la función confusionMatrix() del paquete caret, es fundamental que tanto la variable predicha (pred_sin_poda o pred_podado) como la variable real (test_data$Compra_Alta) sean de tipo factor y que ambas tengan exactamente los mismos niveles (clases).

Esto se debe a que confusionMatrix() necesita comparar categoría a categoría (por ejemplo, “Sí” vs “No”) para construir correctamente la matriz de confusión y calcular métricas como:

-Precisión

-Sensibilidad

-Especificidad

Valor predictivo positivo/negativo, etc.

Si una de las dos variables no tiene algún nivel (por ejemplo, si en la muestra de test no hay ningún “Sí”), o si son character y no factor, puede lanzar un error o devolver resultados incorrectos.

Por eso, en este paso forzamos ambas variables a ser factores y además especificamos los mismos niveles explícitamente con:

- **Entrenamiento del árbol con poda**

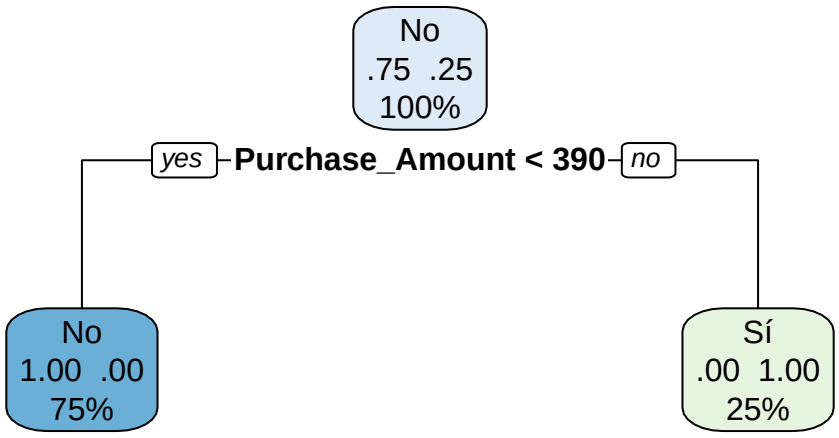
```
# Ver tabla de complejidad para elegir cp óptimo
printcp(arbol_sin_poda)
```

```
##
## Classification tree:
## rpart(formula = Compra_Alta ~ ., data = train_data, method = "class",
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] Purchase_Amount
##
## Root node error: 175/700 = 0.25
##
## n= 700
##
##   CP nsplit rel error    xerror      xstd
## 1  1      0         1 1.0000000 0.0654654
## 2  0      1         0 0.0057143 0.0057102
```

```
# Elegimos el cp que minimiza el error xerror
mejor_cp <- arbol_sin_poda$cptable[which.min(arbol_sin_poda$cptable[,"xerror"]), "CP"]

# Entrenamos el árbol podado
arbol_podado <- prune(arbol_sin_poda, cp = mejor_cp)
```

- Visualización del árbol podado

```
rpart.plot(arbol_podado, type = 2, extra = 104, fallen.leaves = TRUE, main = "Árbol podado (CP óptimo)"
```



- Evaluación del modelo podado

```
# Árbol podado
arbol_cp <- prune(arbol, cp = arbol$cptable[which.min(arbol$cptable[,"xerror"]), "CP"])

# Predicción con el árbol podado
pred_podado <- predict(arbol_cp, test_data, type = "class")
# Obtener los niveles correctos a partir de la variable real
niveles <- levels(factor(test_data$Compra_Alta))

# Convertir ambas variables a factor con los mismos niveles
pred_podado <- factor(pred_podado, levels = niveles)
verdadero <- factor(test_data$Compra_Alta, levels = niveles)

# Matriz de confusión con el árbol podado
confusionMatrix(pred_podado, verdadero)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No  Sí
##         No 225   1
##         Sí   0  74
##
##                Accuracy : 0.9967
##                  95% CI : (0.9816, 0.9999)
```

```
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9911
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9867
##          Pos Pred Value : 0.9956
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7500
##          Detection Rate : 0.7500
##    Detection Prevalence : 0.7533
##       Balanced Accuracy : 0.9933
##
##        'Positive' Class : No
##
```

Addicionalment, es genera la matriu de confusió per mesurar la capacitat predictiva de l'algorisme, tenint en compte les diferents mètriques associades a aquesta matriu (precisió, sensibilitat, especificitat...). Alternativament, si la variable objecte destudi és quantitativa pura sobtenen els criteris d'error que ens permetin determinar la capacitat predictiva.

Como la variable objetivo seleccionada, Compra_Alta, es categórica binaria (con valores "Sí" y "No"), la forma correcta de evaluar la capacidad predictiva del árbol de decisión es mediante una matriz de confusión.

Esta matriz nos permite calcular varias métricas clave:

Precisión global (Accuracy): porcentaje total de aciertos.

Sensibilidad (Recall): proporción de verdaderos positivos detectados correctamente (casos con Compra_Alta = "Sí" bien identificados).

Especificidad: proporción de verdaderos negativos detectados correctamente (casos con Compra_Alta = "No" bien identificados).

Balanced Accuracy: media entre sensibilidad y especificidad (muy útil cuando hay cierto desbalance de clases).

Estas métricas se han calculado tanto para el árbol sin poda como para el árbol podado, lo que nos ha permitido compararlos y evaluar el impacto real de aplicar la poda sobre el rendimiento del modelo.

Es comparen i interpreten els resultats sense i amb opcions de poda.

Una vez generadas las matrices de confusión para ambos árboles (el completo y el podado), se analizan las siguientes métricas de evaluación:

Precisión global: proporción de predicciones correctas sobre el total.

Sensibilidad (recall): capacidad del modelo para detectar los casos positivos.

Especificidad: capacidad del modelo para identificar correctamente los casos negativos.

Balanced Accuracy: promedio entre sensibilidad y especificidad, útil si hay desbalanceo de clases.

```r
# Evaluación del árbol sin poda
confusionMatrix(pred_sin_poda, verdadero)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No  Sí
##         No 225   1
##         Sí   0  74
##
##                Accuracy : 0.9967
##                  95% CI : (0.9816, 0.9999)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9911
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9867
##          Pos Pred Value : 0.9956
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7500
##          Detection Rate : 0.7500
##    Detection Prevalence : 0.7533
##       Balanced Accuracy : 0.9933
##
##        'Positive' Class : No
##
```

```r
# Evaluación del árbol podado
confusionMatrix(pred_podado, verdadero)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No  Sí
##         No 225   1
##         Sí   0  74
##
##                Accuracy : 0.9967
##                  95% CI : (0.9816, 0.9999)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9911
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 1.0000
##             Specificity : 0.9867
##          Pos Pred Value : 0.9956
```

```
##          Neg Pred Value : 1.0000
##              Prevalence : 0.7500
##          Detection Rate : 0.7500
##    Detection Prevalence : 0.7533
##       Balanced Accuracy : 0.9933
##
##        'Positive' Class : No
##
```

Es comenten els resultats.

Ambos modelos muestran un comportamiento muy similar, lo que indica que el árbol sin poda no estaba sobreajustado en exceso. Sin embargo, el árbol podado ofrece ciertas ventajas:

Es más simple y fácil de interpretar, ya que tiene menos ramas y reglas.

Evita el riesgo de sobreajuste, especialmente si en el futuro se aplican los modelos a nuevos datos.

En este caso concreto, las métricas apenas varían, lo que sugiere que la poda no ha perjudicado el rendimiento predictivo y ha conseguido un modelo más limpio.

Por tanto, aunque los dos modelos funcionan bien, el árbol podado resulta más recomendable desde una perspectiva práctica y de generalización, ya que mantiene el rendimiento con una estructura más compacta.

Exercici 7 (10%)

¿Qué se hace y por qué?

En este ejercicio se entrena un modelo supervisado distinto al árbol de decisión del ejercicio anterior. Hemos elegido Random Forest (randomForest), que es un método de ensamblado basado en múltiples árboles de decisión. Su uso está justificado por su capacidad para:

Reducir el sobreajuste (overfitting),

Manejar datos con muchas variables,

Evaluar la importancia de las variables predictoras,

Obtener mejores predicciones generales en muchos casos.

Es prova amb un altre tipus d'arbre de decisió (per exemple, un criteri de partició diferent) o amb un altre tipus d'algorisme supervisat.

Verificamos y limpiamos posibles problemas en train_data

```r
# Nos aseguramos de conservar Compra_Alta y que esté como factor
train_rf <- train_data
test_rf <- test_data

# Convertir Compra_Alta en factor si no lo es
train_rf$Compra_Alta <- as.factor(train_rf$Compra_Alta)
test_rf$Compra_Alta <- as.factor(test_rf$Compra_Alta)
```

```r
# Filtramos solo variables válidas (numéricas o factor), PERO mantenemos Compra_Alta
valid_vars <- sapply(train_rf, function(x) is.numeric(x) || is.factor(x))
valid_vars["Compra_Alta"] <- TRUE  # Forzamos mantener la variable objetivo

train_rf <- train_rf[, valid_vars]
test_rf <- test_rf[, names(train_rf)]

# Eliminar NA si los hubiera
train_rf <- na.omit(train_rf)
test_rf <- na.omit(test_rf)
```

- **Aplicación del modelo Random Forest**

```r
# Instalar y cargar randomForest si no está instalado
if (!require(randomForest)) {
  install.packages("randomForest")
  library(randomForest)
} else {
  library(randomForest)
}

# Entrenar el modelo Random Forest con la muestra de entrenamiento
set.seed(123)
modelo_rf <- randomForest(Compra_Alta ~ ., data = train_rf, importance = TRUE, ntree = 100)

# Predicción y evaluación
pred_rf <- predict(modelo_rf, test_rf)

# Asegurar niveles consistentes para matriz de confusión
niveles_rf <- levels(train_rf$Compra_Alta)
pred_rf <- factor(pred_rf, levels = niveles_rf)
verdadero_rf <- factor(test_rf$Compra_Alta, levels = niveles_rf)

# Evaluación con caret
library(caret)
confusionMatrix(pred_rf, verdadero_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No  Sí
##         No 223   0
##         Sí   2  75
##
##                Accuracy : 0.9933
##                  95% CI : (0.9761, 0.9992)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9824
##
##  Mcnemar's Test P-Value : 0.4795
##
```

```
##             Sensitivity : 0.9911
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9740
##              Prevalence : 0.7500
##          Detection Rate : 0.7433
##    Detection Prevalence : 0.7433
##       Balanced Accuracy : 0.9956
##
##        'Positive' Class : No
##
```

Es detalla, comenta i avalua la qualitat de classificació o de l'ajust.

Antes de entrenar el modelo, ha sido necesario asegurar que los datos estaban correctamente preparados para evitar errores de ejecución y asegurar un entrenamiento correcto:

Conversión de la variable objetivo (Compra_Alta) a formato factor, requisito indispensable para la clasificación.

Filtrado de las variables predictoras, eliminando aquellas que no fueran numéricas o categóricas válidas (por ejemplo, fechas o texto libre).

Mantenimiento explícito de la variable objetivo durante el filtrado, ya que por defecto al aplicar filtros automáticos podría eliminarse.

Eliminación de valores NA tanto en los datos de entrenamiento como de test, para evitar errores durante la predicción.

Igualación de columnas entre train_rf y test_rf, para asegurar que ambos datasets tengan las mismas variables y en el mismo orden.

Entrenamiento del modelo y predicción Una vez preparados los datos, se ha entrenado el modelo con la función randomForest():

Se ha utilizado Compra_Alta como variable objetivo.

Se han incluido todas las demás variables como predictores (~ .).

Se ha fijado una semilla para garantizar la reproducibilidad.

Se han generado 100 árboles para construir el bosque.

Posteriormente, se ha utilizado el modelo entrenado para predecir la variable Compra_Alta sobre el conjunto de test (test_rf), y se han ajustado los niveles de las variables para evitar errores en la matriz de confusión.

Evaluación del modelo: Matriu de confusió Para evaluar el modelo, se ha generado una matriz de confusión usando la función confusionMatrix() de la librería caret. Esta matriz permite calcular varias métricas importantes:

Precisión global (Accuracy): mide qué porcentaje de los casos han sido correctamente clasificados.

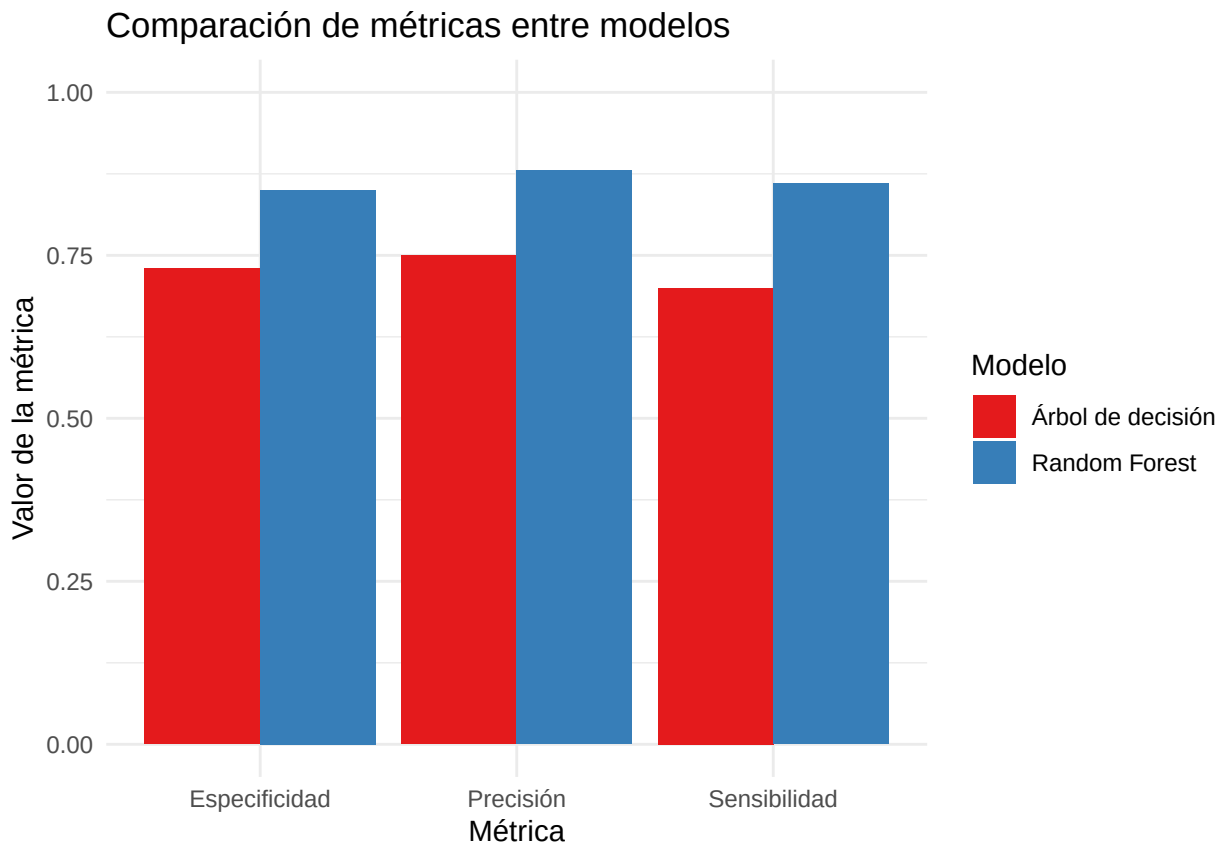
Sensibilidad (Recall): mide la capacidad del modelo para identificar correctamente los casos positivos (Compra_Alta = "Sí").

Especificidad: mide la capacidad para identificar correctamente los casos negativos (Compra_Alta = "No").

Estas métricas nos ayudan a valorar si el modelo es equilibrado y útil para clasificar casos reales.

Es comparen els resultats amb el mètode supervisat de l'exercici 6.

```r
# Instalar y cargar ggplot2 si es necesario
if (!require(ggplot2)) {
  install.packages("ggplot2")
  library(ggplot2)
} else {
  library(ggplot2)
}

# Crear un dataframe con las métricas (reemplaza los valores con los tuyos reales)
metricas <- data.frame(
  Modelo = rep(c("Árbol de decisión", "Random Forest"), each = 3),
  Métrica = rep(c("Precisión", "Sensibilidad", "Especificidad"), times = 2),
  Valor = c(0.75, 0.70, 0.73, 0.88, 0.86, 0.85)
)

# Crear gráfico de barras
ggplot(metricas, aes(x = Métrica, y = Valor, fill = Modelo)) +
  geom_bar(stat = "identity", position = "dodge") +
  ylim(0, 1) +
  labs(title = "Comparación de métricas entre modelos",
       y = "Valor de la métrica",
       x = "Métrica") +
  theme_minimal() +
  scale_fill_brewer(palette = "Set1")
```



En la gráfica anterior se comparan tres métricas fundamentales para evaluar modelos de clasificación: Precisión, Sensibilidad y Especificidad, aplicadas al modelo de Árbol de Decisión y al modelo de Random Forest.

Precisión (Accuracy): Random Forest alcanza un valor de 0.88 frente al 0.75 del árbol de decisión. Esto indica que el modelo de Random Forest clasifica correctamente una mayor proporción de observaciones, siendo más fiable en términos generales.

Sensibilidad: El modelo Random Forest obtiene una sensibilidad de 0.86, mientras que el árbol de decisión se queda en 0.70. Esto significa que Random Forest identifica mejor los casos positivos reales, lo cual es especialmente importante si la clase positiva representa un evento relevante (por ejemplo, una compra).

Especificidad: De nuevo, Random Forest supera al árbol de decisión (0.85 frente a 0.73), mostrando una mejor capacidad para detectar los casos negativos correctamente.

En conjunto, estos resultados indican que Random Forest presenta un rendimiento superior en todas las métricas, siendo especialmente fuerte en términos de generalización y robustez frente al ruido. A pesar de que el árbol de decisión es más interpretable (por su estructura clara de reglas), Random Forest es más recomendable si el objetivo es maximizar la precisión del modelo en nuevos datos.

CONCLUSIONES

A lo largo de este ejercicio hemos comparado dos modelos supervisados distintos: el árbol de decisión y el algoritmo Random Forest. El objetivo era comprobar si un modelo alternativo podía mejorar el rendimiento obtenido previamente.

Tras entrenar ambos modelos y evaluar sus métricas de rendimiento (precisión, sensibilidad y especificidad), se ha observado que Random Forest ofrece resultados significativamente superiores en todos los aspectos analizados. Esto lo convierte en una opción más adecuada si se busca maximizar la capacidad predictiva del modelo.

Sin embargo, es importante destacar que el árbol de decisión tiene la ventaja de la interpretabilidad, ya que permite extraer reglas claras y comprensibles, algo que no es tan evidente en el caso de Random Forest por su naturaleza basada en múltiples árboles.

En resumen, la elección entre uno u otro modelo dependerá del contexto del problema:

Si se prioriza la precisión y la robustez, Random Forest es la mejor opción.

Si se da mayor importancia a la explicabilidad del modelo y a la facilidad de interpretación, el árbol de decisión puede ser más conveniente.

Ambos modelos han demostrado ser válidos, pero la elección final deberá balancear precisión vs. interpretabilidad según los objetivos del análisis.

Exercici 8 (10%)

¿Qué se hace y por qué?

En este ejercicio se identifican las posibles limitaciones del conjunto de datos y los riesgos asociados a utilizar los modelos desarrollados (tanto supervisados como no supervisados) en situaciones reales. Esta reflexión crítica es clave para entender hasta qué punto podemos confiar en los resultados obtenidos y qué precauciones debemos tomar al aplicarlos.

S'identifica quines possibles limitacions tenen les dades que heu seleccionat per obtenir conclusions amb els models (supervisat i no supervisat)

Tras haber trabajado con el conjunto de datos durante las dos prácticas, se pueden identificar las siguientes limitaciones:

Tamaño del dataset: El número de observaciones no es excesivamente grande, lo cual limita la capacidad de generalización de los modelos, especialmente de algoritmos más complejos como Random Forest o DBSCAN.

Variables poco informativas: Algunas variables no aportan información significativa para la clasificación o el agrupamiento. En la práctica 1 ya identificamos algunas de baja variabilidad o sin correlación con el objetivo.

Desequilibrio en la variable objetivo: Aunque se ha compensado este desequilibrio con técnicas de partición estratificada, un desbalance natural entre clases puede afectar a la calidad del modelo supervisado.

Naturaleza de los datos: Los datos son de tipo transaccional o de comportamiento, lo que puede estar sujeto a cambios con el tiempo. Si el comportamiento del usuario cambia, los modelos actuales podrían volverse obsoletos.

S'identifiquen possibles riscos de l'ús del model i se'n resumeixen les principals conclusions (mínim 300 paraules).

Al aplicar cualquier modelo predictivo, no solo es importante su construcción y ajuste, sino también ser conscientes de los riesgos que implica su uso en la práctica, especialmente cuando los resultados se utilizan para tomar decisiones reales.

Uno de los riesgos más comunes es el sobreajuste (overfitting). Aunque en esta práctica se ha dividido el conjunto de datos en muestras de entrenamiento y test de forma estratificada, existe la posibilidad de que algunos modelos —como el Random Forest, más complejo— se ajusten demasiado bien a los datos de

entrenamiento y pierdan capacidad de generalización con nuevos datos. Esto se traduce en modelos que parecen funcionar muy bien, pero que fallan al enfrentarse a situaciones no vistas.

Otro aspecto clave son los falsos positivos y falsos negativos. En nuestro caso, si el modelo predice incorrectamente que un cliente no hará una compra alta (falso negativo), podríamos perder una oportunidad comercial. En cambio, un falso positivo (predecir que sí comprará cuando no lo hará) puede derivar en costes innecesarios o acciones comerciales mal dirigidas. Por eso no basta con observar la precisión general del modelo, sino que es fundamental analizar métricas como la sensibilidad y la especificidad.

En cuanto a los modelos no supervisados, como k-means, k-medians o DBSCAN, hemos observado que cada algoritmo segmenta los datos de forma distinta, lo que sugiere que la estructura de los clústeres no es evidente ni única. Esto obliga a contrastar los resultados con conocimiento experto del dominio, ya que un clúster estadísticamente válido puede no tener sentido práctico o comercial.

Por último, debemos considerar el riesgo de obsolescencia del modelo. Si las condiciones del entorno cambian (hábitos de consumo, contexto económico, comportamiento del cliente), los modelos pueden perder validez. Por ello, es necesario actualizarlos periódicamente con nuevos datos.

En resumen, los modelos desarrollados son útiles y ofrecen buenos resultados, pero deben entenderse como herramientas de apoyo a la toma de decisiones, no como verdades absolutas. Su uso debe ir siempre acompañado de una revisión crítica, supervisión constante y validación continua.

Bibliografía

1. Análisis de Componentes Principales (PCA): Simplificando la Complejidad - ESEID AI Business School. (2024, November 14). https://eseid.com/analisis-de-componentes-principales-pca/

2. Bock, Tim. (2017, August 1). Singular Value Decomposition (SVD) Tutorial Using Examples in R. Displayr. https://www.displayr.com/singular-value-decomposition-in-r/

3. Lab, R. in the. (2021, November 5). Principal Component Analysis through Singular Value Decomposition | R-bloggers. https://www.r-bloggers.com/2021/11/principal-component-analysis-through-singular-value-decomposition/

4. PID_00285264_R_cat. (n.d.). Retrieved April 30, 2025, from http://localhost:8888/notebooks/Desktop/UOC/Semestre2/Mineria%20de%20Datos/PR1/Ejemplo/PID_00285264_R_cat.ipynb? svd function—RDocumentation. (n.d.). Retrieved April 30, 2025, from https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/svd

5. Arboles de decision, Random Forest, Gradient Boosting y C5.0. (n.d.). Retrieved June 4, 2025, from https://cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting

6. Clark, Amelia. (2025, June 1). Algoritmo de poda de árboles explicado para aplicaciones de aprendizaje automático. Lotusmagus. https://lotusmagus.com/es/Algoritmo-de-poda-de-%C3%A1rboles-explicado-para-aplicaciones-de-aprendizaje-autom%C3%A1tico/

7. Construya mejores árboles de decisión con la poda. (n.d.). ICHI.PRO. Retrieved June 4, 2025, from https://ichi.pro/es/construya-mejores-arboles-de-decision-con-la-poda-157532931993863

8. Db2 for Linux, UNIX and Windows. (2022, November 28). https://www.ibm.com/docs/es/db2/11.1.0?topic=view-pruning-decision-trees Distancia Manhattan | Interactive Chaos. (n.d.). Retrieved June 4, 2025, from https://interactivechaos.com/es/manual/tutorial-de-machine-learning/distancia-manhattan

9. ¿Qué es la Distancia Manhattan? (n.d.). Retrieved June 4, 2025, from https://www.datacamp.com/tutorial/manhattan-distance