

Ventas de asientos de seguridad para niños en automóviles



MINERÍA DE DATOS

En este análisis, estamos utilizando un conjunto de datos que contiene información sobre las ventas de asientos de seguridad para niños en automóviles (sales).

El objetivo principal es identificar las relaciones entre las características de las tiendas y comunidades con las ventas, y utilizar estas relaciones para predecir las ventas en otras ubicaciones.

Por lo tanto, el problema que estamos abordando es un problema de regresión, ya que la variable de interés (sales) es continua y queremos predecir su valor en función de otras variables.

SE PRESENTAN EL CONJUNTO DE DATOS DE ASIENTOS DE SEGURIDAD PARA NIÑOS EN AUTOMOVIL (CHILD CAR SAFETY SEATS)

Variable dependiente:

- sales (ventas): Ventas unitarias en miles

Variable independiente:

- comp_price (precio_comp): Precio cobrado por el competidor en cada ubicación
- income (ingreso): Nivel de ingresos de la comunidad en miles de dólares
- adversiting (publicidad): Presupuesto publicitario local en cada ubicación en miles de dólares
- population (población): El pop regional en miles
- price (precio): Precio de las sillas de coche en cada sitio
- shelveloc (estanteLoc): Malo, Bueno o Medio indica la calidad de la ubicación de las estanterías.
- age (edad): Nivel de edad de la población
- education (educación): Nivel educativo en la ubicación
- urban (urbano): Los niveles de factor 'Sí' o 'No' se utilizan para indicar si una tienda está en una ubicación urbana o rural.
- us (a nosotros): Los niveles de factor 'Sí' o 'No' se utilizan para indicar si una tienda está en Estados Unidos o no.

ANÁLISIS EXPLORATORIO DE LOS DATOS

Se realiza un analisis de los datos para poder realizar los modelos de aprendizaje supervisado mas a delante.

SE REvisa LAS DIMENSIONES, ESTRUCTURA Y RESUMEN
ESTADÍSTICO DE LOS DATOS

DIM(DATA)

STR(DATA)

SUMMARY(DATA)

Aquí se puede apreciar que tenemos
datos en formato carácter por lo que
debemos convertir estos datos a
factorial, 400 datos con 11 variables
ya mencionadas anteriormente

```
> dim(data)
[1] 400 11
> str(data)
'data.frame': 400 obs. of 11 variables:
 $ sales      : num  9.5 11.22 10.06 7.4 4.15 ...
 $ comp_price : int  138 111 113 117 141 124 115 136 132 132 ...
 $ income     : int  73 48 35 100 64 113 105 81 110 113 ...
 $ advertising: int  11 16 10 4 3 13 0 15 0 0 ...
 $ population : int  276 260 269 466 340 501 45 425 108 131 ...
 $ price      : int  120 83 80 97 128 72 108 120 124 124 ...
 $ shelveLoc  : chr  "Bad" "Good" "Medium" "Medium" ...
 $ age        : int  42 65 59 55 38 78 71 67 76 76 ...
 $ education  : int  17 10 12 14 13 16 15 10 10 17 ...
 $ urban      : chr  "Yes" "Yes" "Yes" "Yes" ...
 $ us         : chr  "Yes" "Yes" "Yes" "Yes" ...
> summary(data)
      sales      comp_price      income      advertising      population
Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000   Min.   : 10.0
1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000   1st Qu.:139.0
Median : 7.490   Median :125   Median : 69.00   Median : 5.000   Median :272.0
Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635   Mean   :264.8
3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000   3rd Qu.:398.5
Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000   Max.   :509.0

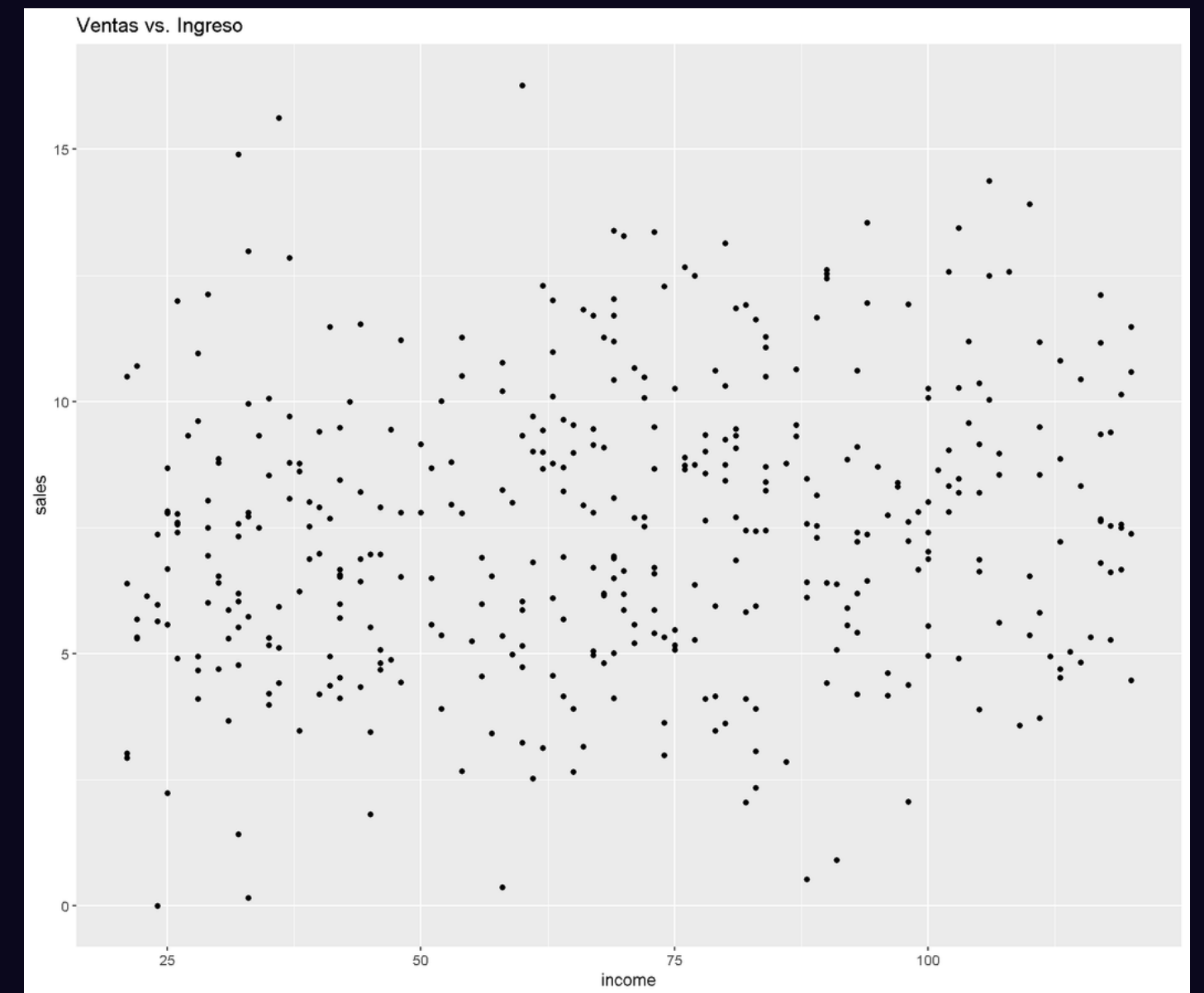
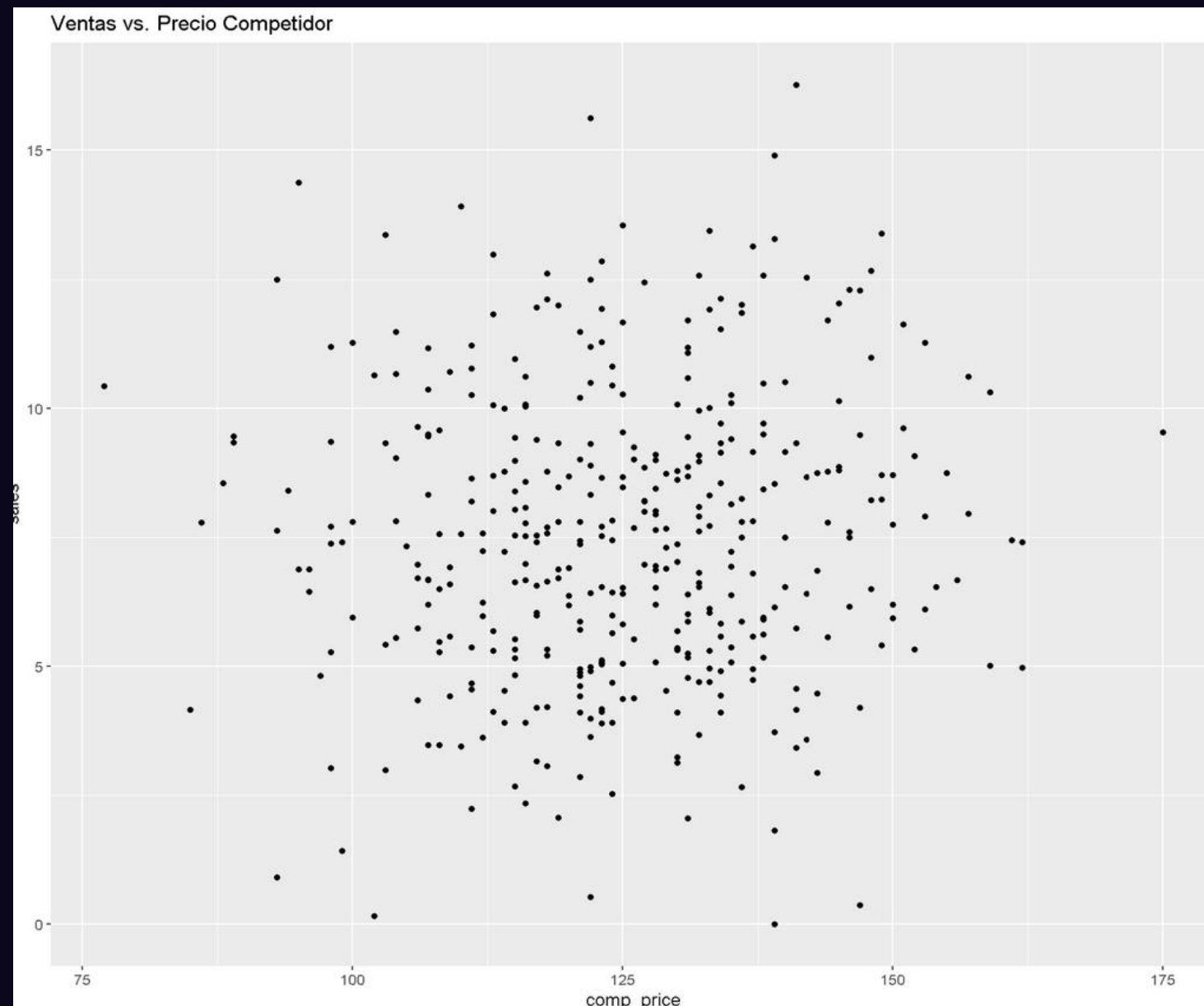
      price      shelveLoc      age      education      urban
Min.   : 24.0   Length:400   Min.   :25.00   Min.   :10.0   Length:400
1st Qu.:100.0   Class :character   1st Qu.:39.75   1st Qu.:12.0   Class :character
Median :117.0   Mode  :character   Median :54.50   Median :14.0   Mode  :character
Mean   :115.8                                Mean   :53.32   Mean   :13.9
3rd Qu.:131.0                                3rd Qu.:66.00   3rd Qu.:16.0
Max.   :191.0                                Max.   :80.00   Max.   :18.0

      us
Length:400
Class :character
Mode  :character
```

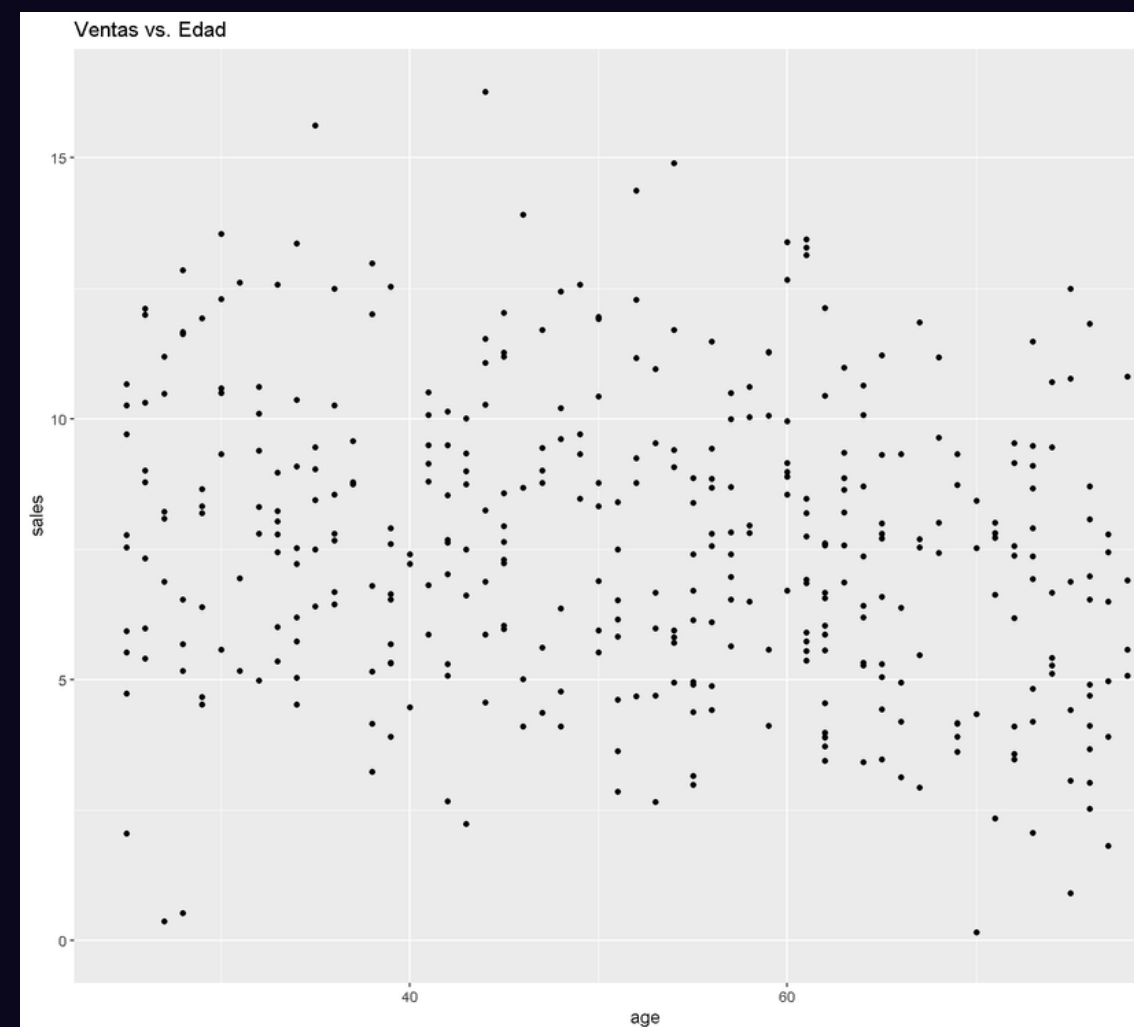
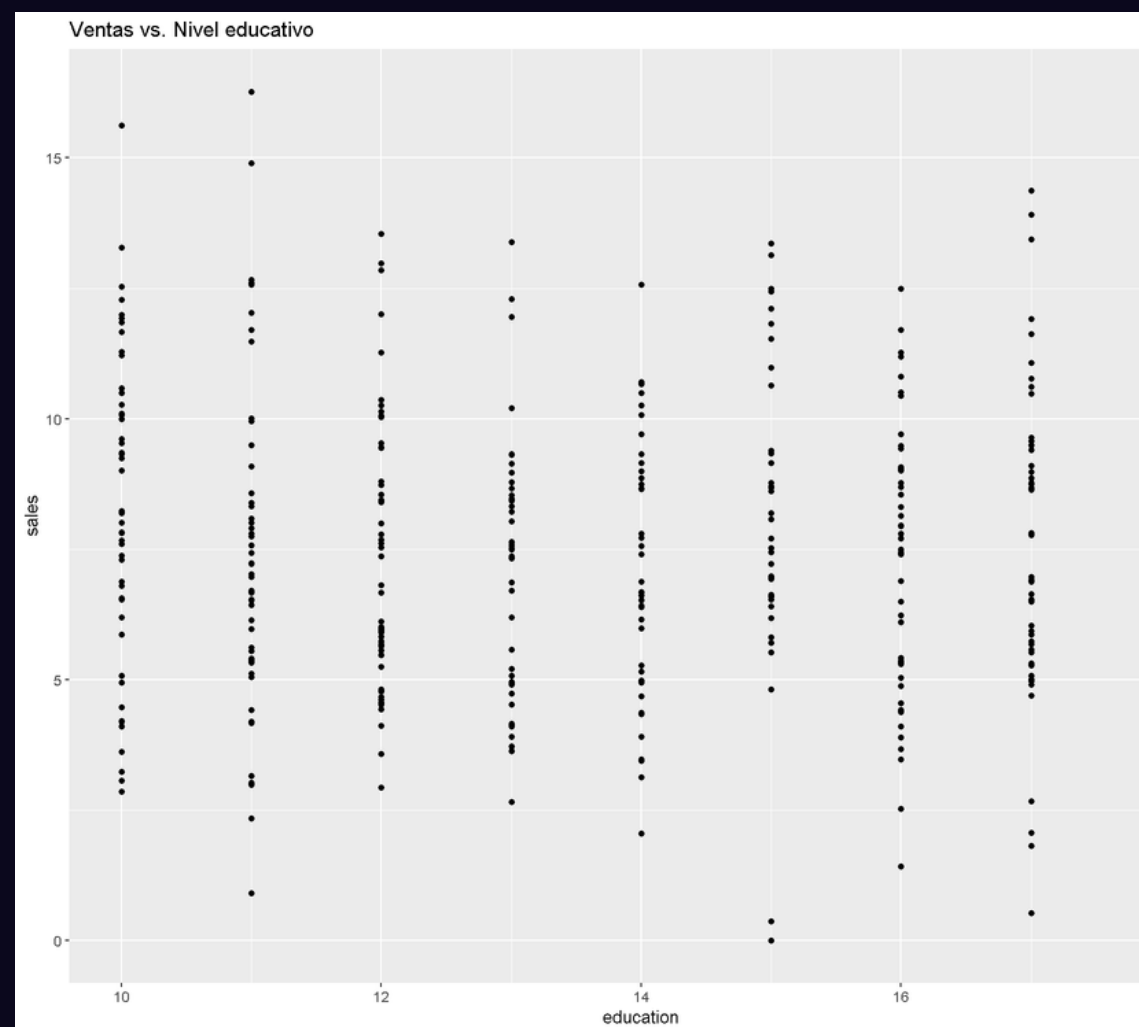
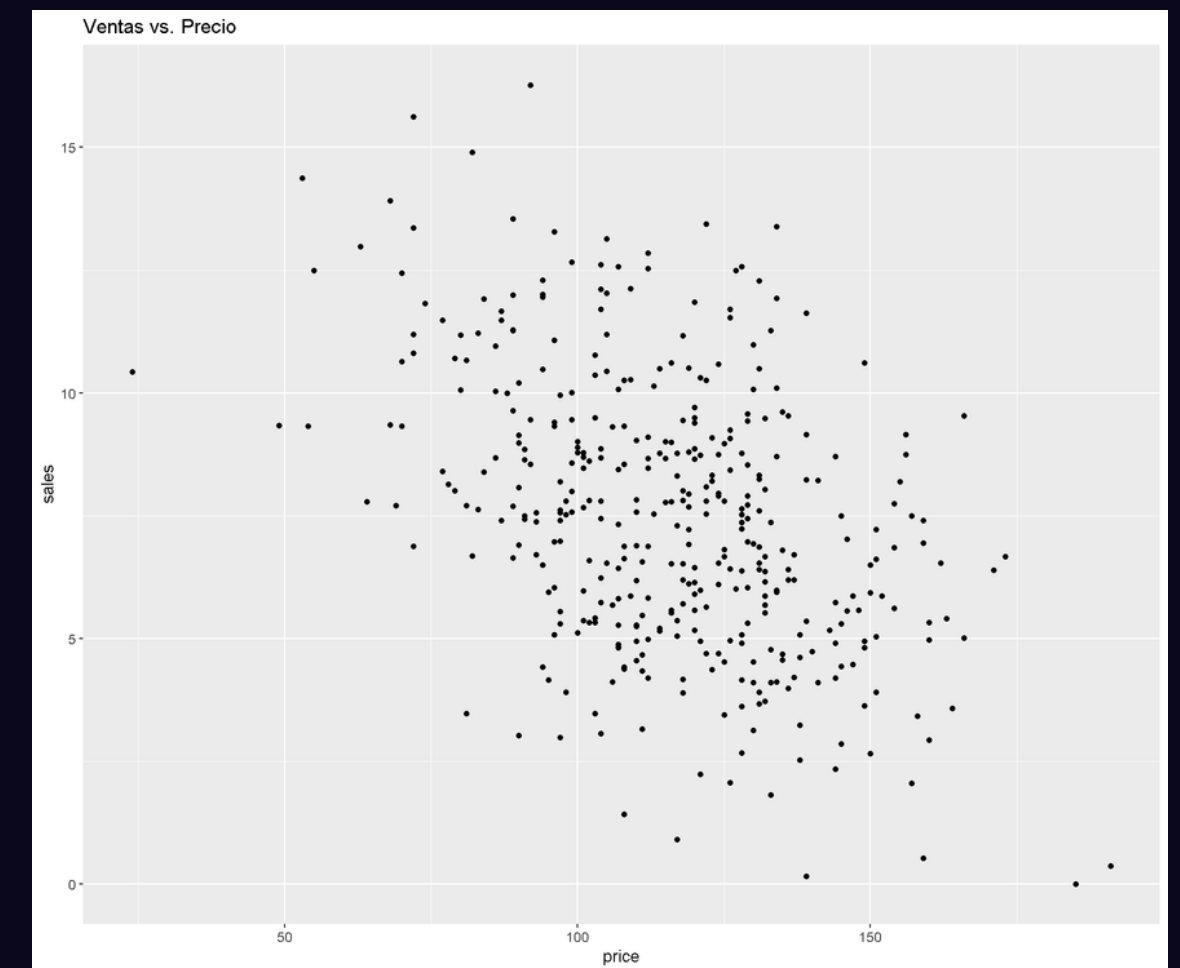
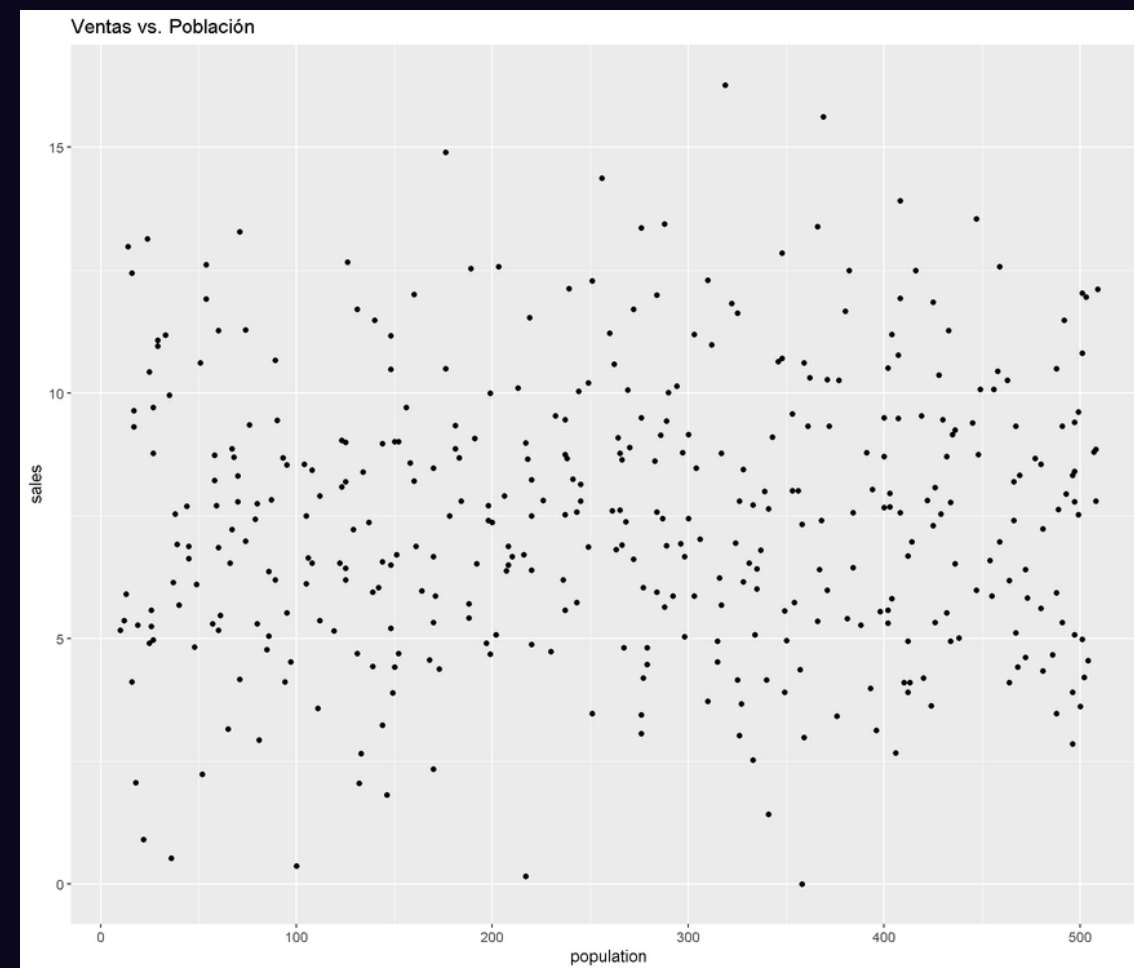
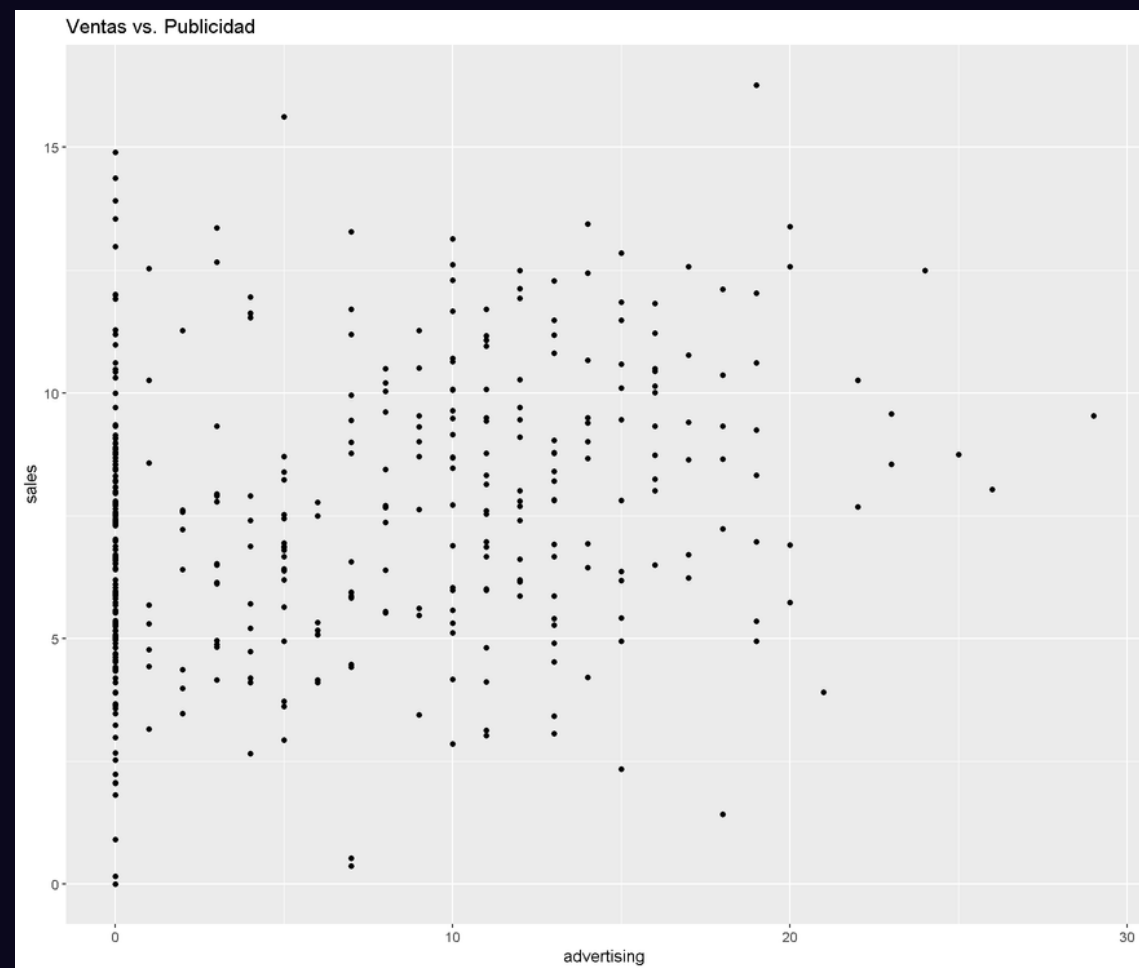

Luego se debe analizar si hay datos faltantes en este caso se hizo con el siguiente código `any(is.na(data))`

- Aquí se indica que no existen datos faltantes

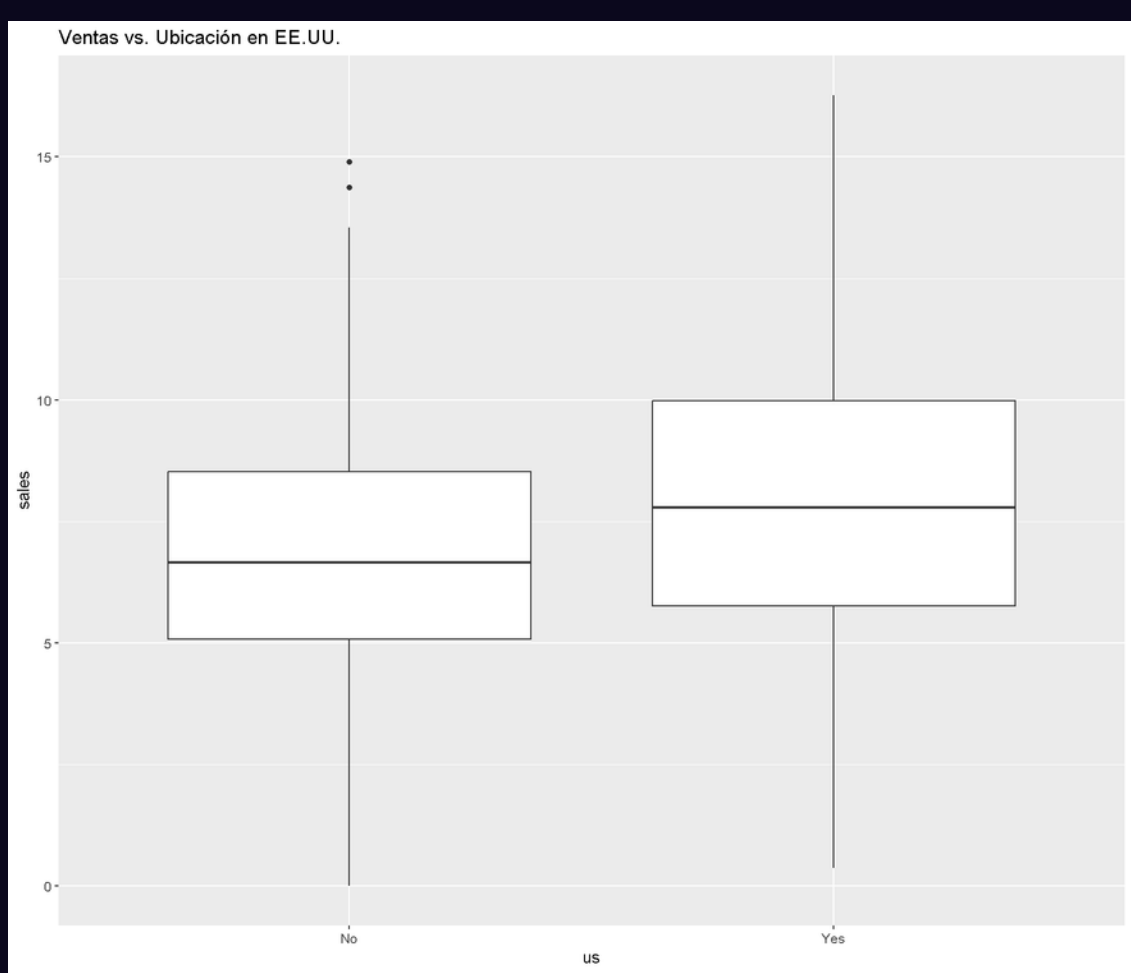
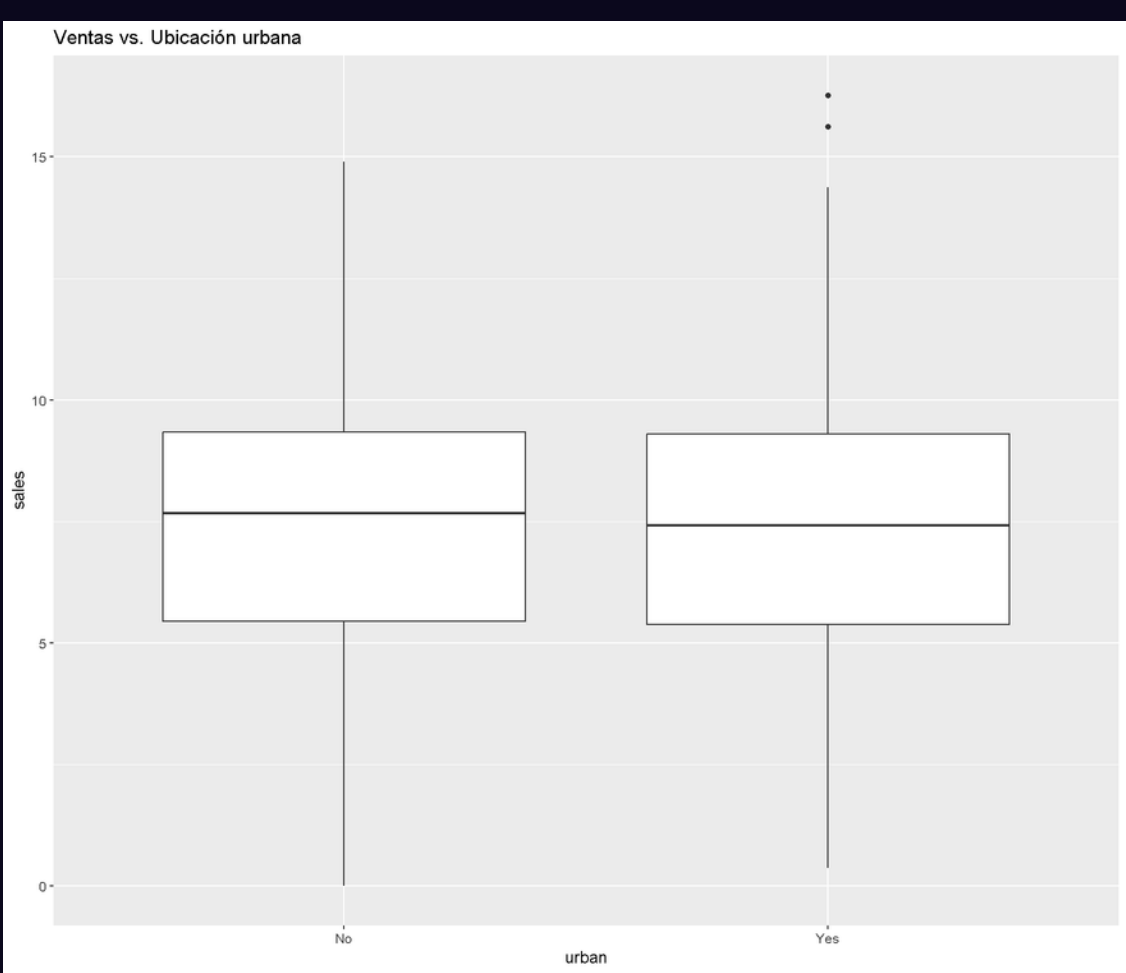
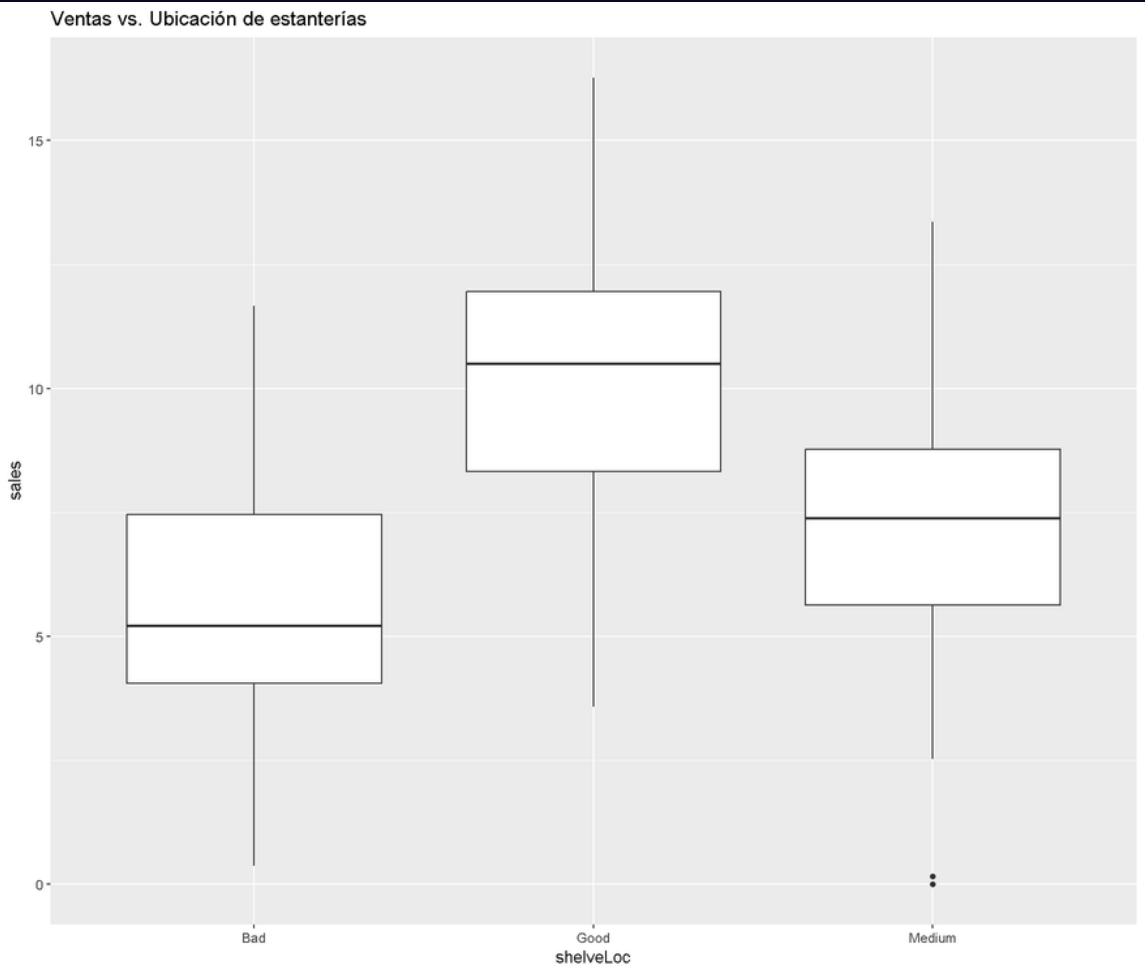
Ahora, se analiza el comportamiento de las variables predictoras con ventas utilizando la librería `ltidyverse`



Los puntos están bastante dispersos en ambos gráficos, lo que indica que no hay una relación clara entre el ingreso y las ventas y precio competidor y ventas.



A pesar que algunos gráficos muestran una pequeña homogeneidad, la gran mayoría muestra una dispersión entre las variables publicidad, población, precio, nivel educativo y edad con respecto a ventas



La ubicación de la estantería parece tener un impacto significativo en las ventas. Los productos clasificados como "Good" tienen, en promedio, las mayores ventas también hay variabilidad en las ventas dentro de cada categoría.

No hay una diferencia significativa entre zonas urbanas y no urbanas. La dispersión en los valores de ventas es similar en ambos grupos. La presencia de valores atípicos en el grupo de zonas urbanas sugiere que puede haber algunas ubicaciones urbanas con ventas excepcionalmente altas.

La caja para EE.UU. es más alta, lo que indica una mayor variabilidad en las ventas dentro de EE.UU. en comparación con fuera de EE.UU.

Las variables como publicidad, población, precio, nivel educativo y edad muestran alta dispersión, indicando que no existe una relación lineal clara con las ventas. Sin embargo, algunas variables como la ubicación de la estantería sí tienen un impacto notable. Esto sugiere que las ventas están influenciadas por una combinación de factores, por lo que es necesario utilizar todas las variables disponibles en el conjunto de datos para predecirlas de manera efectiva a través de modelos de machine learning.



DECISION TREE - REGRESIÓN

Exploración de datos:

- sales: Ventas en miles de unidades (variable dependiente).
- comp_price: Precio de la competencia.
- income: Ingreso promedio.
- advertising: Gasto en publicidad.
- population: Población del área.
- price: Precio del asiento de seguridad.
- shelveLoc: Ubicación en la estantería.
- age: Edad promedio de la población.
- education: Nivel de educación promedio.
- urban: Indicador de si el área es urbana.
- us: Indicador de si el área está en Estados Unidos.

División de los Datos:

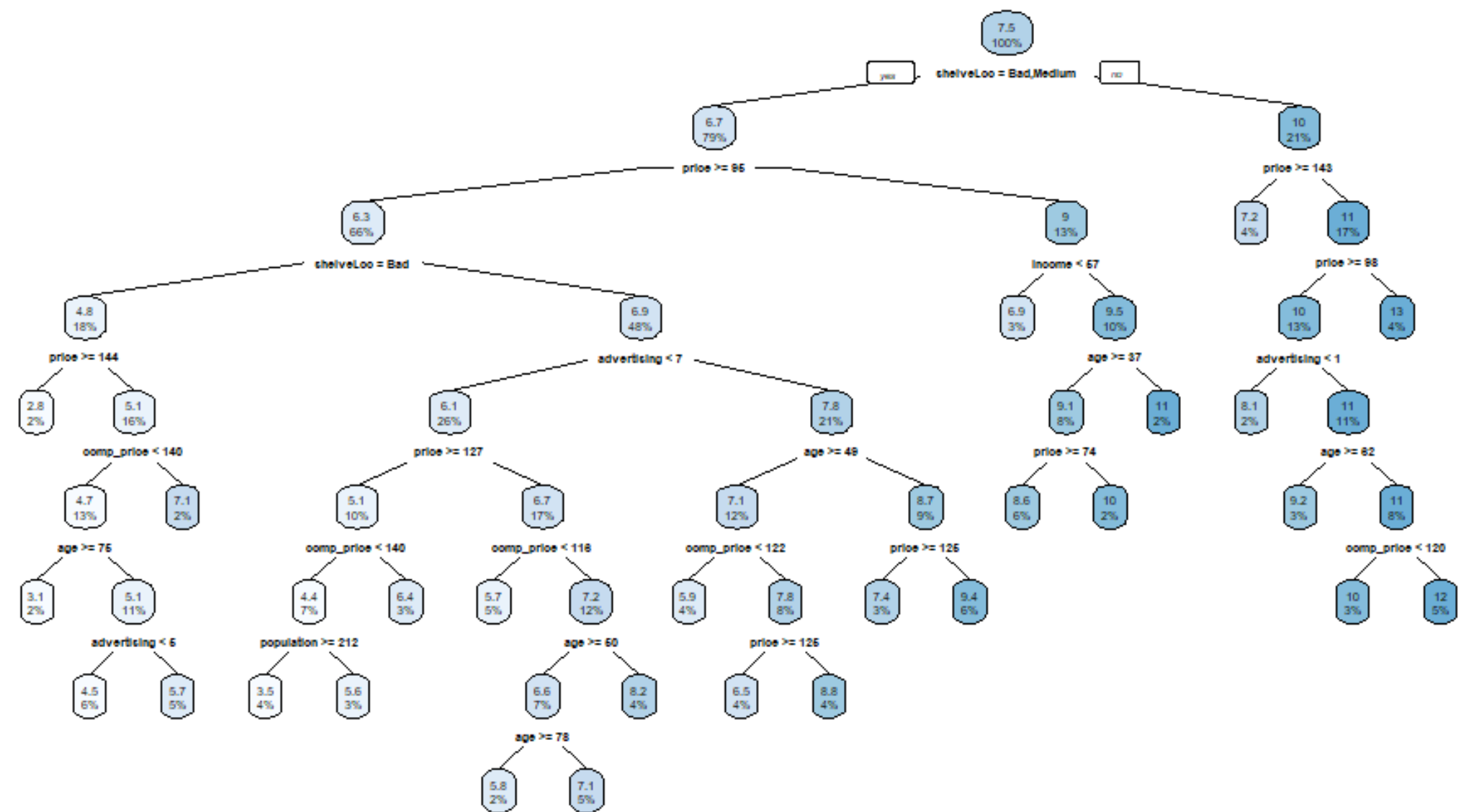
- Se dividieron los datos en un conjunto de entrenamiento y un conjunto de prueba. El 80% de los datos se utilizó para entrenar el modelo, mientras que el 20% restante se reservó para evaluar el rendimiento del modelo.

Creación del Modelo de Árbol de Decisión:

- Se generó un modelo de Árbol de Decisión utilizando la función `rpart`, donde se especificó que el objetivo es predecir la variable `sales` (ventas).
- El método utilizado fue "anova", apropiado para modelos de regresión.
- Se especificó un valor mínimo de "costo de complejidad" (`cp=0.001`) para evitar sobreajustes del modelo.

Visualización del Árbol:

- Finalmente, se dibujó el árbol de decisión utilizando `rpart.plot`, permitiendo visualizar cómo las diferentes variables influyen en las ventas.



Analizando el árbol, podríamos tener las siguientes conclusiones:

- `price` (precio): Es una de las variables más importantes, ya que aparece en la mayoría de las primeras divisiones. Esto sugiere que el precio del asiento de seguridad es un factor clave que influye en las ventas.
- `shelveLoc` (ubicación en la estantería): También es una variable predictiva importante. Mejores ubicaciones en la estantería ("Good") tienden a estar asociadas con mayores ventas.
- `advertising` (publicidad), `comp_price` (precio de la competencia), e `income` (ingreso promedio) también juegan un papel significativo en ciertas ramas del árbol.

Identificación del Mejor Valor de Costo de Complejidad (cp):

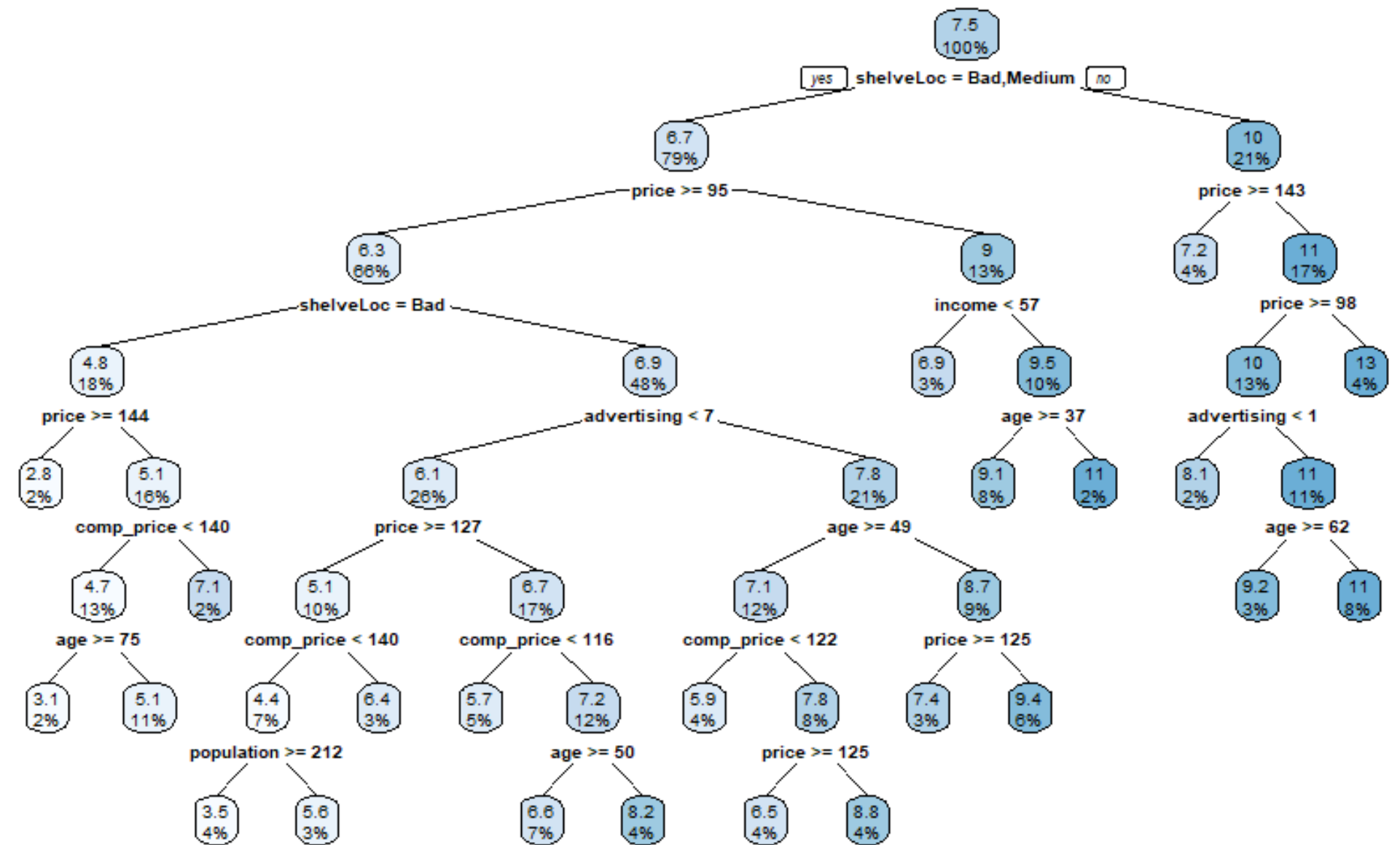
- Se analizó la tabla de complejidad (cptable) generada por el modelo inicial de árbol de decisión. Esta tabla contiene información sobre el error de validación cruzada (medido como xerror) para diferentes valores del parámetro de complejidad (cp).
- Se seleccionó el valor de cp que minimiza el xerror, lo que indica el punto donde el modelo logra el mejor equilibrio entre precisión y complejidad. Este valor óptimo de cp se almacenó en la variable mejor_cp.

Poda del Árbol:

- Utilizando el valor de cp identificado como óptimo, se procedió a podar el árbol de decisión original. La poda es un proceso que elimina ramas del árbol que no contribuyen significativamente a la predicción, reduciendo así el riesgo de sobreajuste.
- El árbol podado es más simple y generaliza mejor a nuevos datos, al enfocarse en las divisiones más importantes.

Visualización del Árbol Podado:

- Se generó una nueva visualización del árbol de decisión, esta vez mostrando el árbol podado. Al reducir el tamaño del árbol, se facilita la interpretación del modelo y se mejora su capacidad predictiva en el conjunto de prueba.



Análisis de la poda

Nodo Raíz:

- El nodo raíz muestra la media general de las ventas (sales) para todos los datos de entrenamiento, que es aproximadamente 7.5.

Primer Nivel de Divisiones:

- La primera división se realiza en función de shelveLoc (ubicación en la estantería). Esta variable parece ser una de las más importantes, ya que determina si se van a seguir diferentes caminos basados en si la ubicación es "Bad/Medium" o no.
- Si shelveLoc no es "Bad/Medium", el promedio de ventas sube a 10.

Subsecuentes Divisiones:

- Camino para shelveLoc = Bad/Medium:
 - El siguiente criterio de división es price (precio) con un umbral de 95. Si el precio es mayor o igual a 95, el promedio de ventas baja ligeramente a 6.7.
 - Luego, se consideran otros factores como advertising (publicidad), comp_price (precio de la competencia), y age (edad promedio).
- Camino para shelveLoc ≠ Bad/Medium:
 - Si el precio (price) es mayor o igual a 143, el promedio de ventas es de 7.2.
 - Otras divisiones importantes incluyen la edad (age) y el gasto en publicidad (advertising).

Las siguientes variables son las que influyen mas en el modelo:

- shelveLoc sigue siendo una variable clave en la determinación de las ventas.
- price también es crucial en varias ramas del árbol, lo que indica su fuerte influencia en las decisiones de compra.
- advertising y comp_price también son importantes pero menos determinantes en comparación con shelveLoc y price.

Validación de modelos y comparación

Predicciones con los Modelos:

- `val_pred1 = predict(arbol, datos_prueba)`: Se utiliza el modelo del árbol de decisión sin podar (`arbol`) para hacer predicciones sobre los datos de prueba (`datos_prueba`).
- `val_pred2 = predict(arbol_podado, datos_prueba)`: De manera similar, se utiliza el modelo del árbol de decisión podado (`arbol_podado`) para hacer predicciones sobre los datos de prueba.

Cálculo del Coeficiente de Determinación (R^2) para el Árbol sin Podar:

Se calcula la Suma Total de Cuadrados (TSS), la cual mide la variabilidad total en los datos de prueba en relación con su media. Cuantifica la variación total de las ventas observadas (sales) en los datos de prueba. Además de la Suma de los Cuadrados de los Residuos (RSS), la cual mide la variabilidad de las ventas que el modelo no pudo explicar, es decir, el error de predicción del modelo sin podar.

Finalmente se saca el R^2 (coeficiente de determinación) el cual se calcula restando la proporción de la variabilidad total no explicada por el modelo (`rss / tss`) de 1. Un R^2 más cercano a 1 indica un mejor ajuste del modelo.

Cálculo del Coeficiente de Determinación (R^2) para el Árbol Podado:

- El cálculo para el árbol podado sigue un procedimiento similar al del árbol sin podar:
 - Se vuelve a calcular TSS (que es el mismo para ambos modelos porque usa los mismos datos de prueba).
 - Se calcula RSS para el árbol podado usando `val_pred2`.
 - Se calcula R^2 para el árbol podado como `r_cuadrado2`.

Comparación de Modelos:

```
if (r_cuadrado1 > r_cuadrado2){  
  print("Modelo Árbol sin poda es mejor")  
}else{  
  print("Modelo Árbol con poda es mejor")  
}
```

Con este trozo de código se compara el valor de R^2 de ambos modelos. Si el R^2 del árbol sin podar ($r_cuadrado1$) es mayor que el R^2 del árbol podado ($r_cuadrado2$), se concluye que el árbol sin podar es mejor. De lo contrario, se considera que el árbol podado es mejor.

Resultados:

- El valor de R^2 para el árbol sin podar es 0.5562676, mientras que para el árbol podado es 0.5358276.
- Dado que el R^2 del árbol sin podar es mayor, el código concluye que el modelo de árbol sin poda es mejor.



RANDOM FOREST - REGRESIÓN




Exploración de datos:

- sales: Ventas en miles de unidades (variable dependiente).
- comp_price: Precio de la competencia.
- income: Ingreso promedio.
- advertising: Gasto en publicidad.
- population: Población del área.
- price: Precio del asiento de seguridad.
- shelveLoc: Ubicación en la estantería.
- age: Edad promedio de la población.
- education: Nivel de educación promedio.
- urban: Indicador de si el área es urbana.
- us: Indicador de si el área está en Estados Unidos.

División de los Datos:

- Se dividieron los datos en un conjunto de entrenamiento y un conjunto de prueba. como podemos notar en la imagen se notan que de los 400 observaciones totales se dividieron en 320 en datos de entrenamiento y 80 corresponden a datos de prueba.

Data

▶ datos	400 obs. of 11 va...	
▶ datos_entrenam...	320 obs. of 11 va...	
▶ datos_prueba	80 obs. of 11 var...	

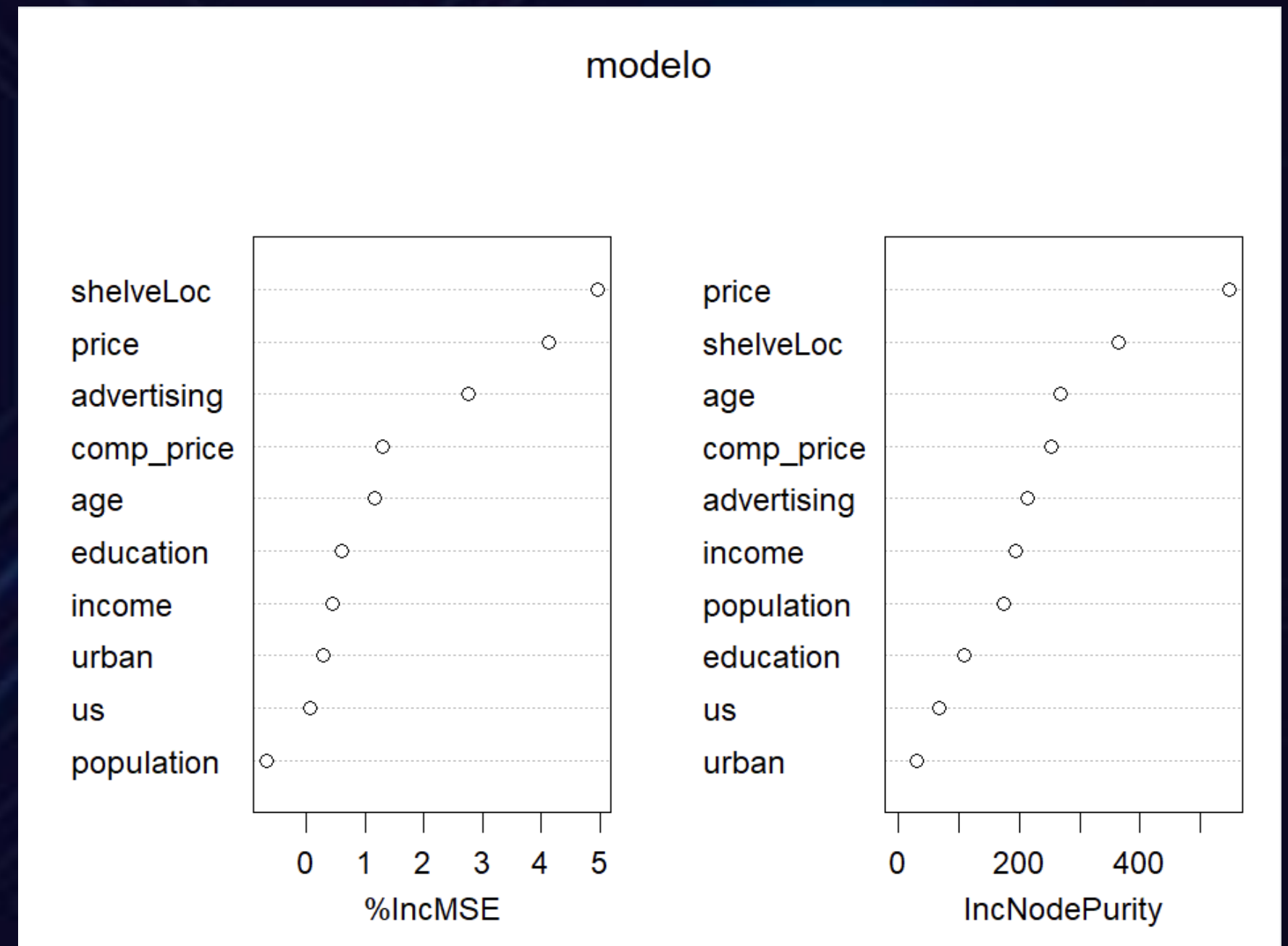
Values

Para comenzar le decimos a R que nos genere el modelo considerando lo siguiente:

- Numero de arboles: 100, especifica que queremos construir 100 árboles en el bosque.
- establecer cada división de nodo, en este caso el algoritmo considera 2 características aleatorias.

El siguiente gráfico que muestra la importancia de las variables:

- MeanDecreaseAccuracy: Muestra cuanto disminuye la precisión del modelo si se elimina una variable. Una mayor disminución, indica que la variable es más importante. en este caso las dos variables mas importantes son shelveLoc y Price
- MeanDecreaseGini: Muestra la disminución promedio de la pureza (o el índice de Gini) de los nodos de los árboles de decisión en el modelo. Una mayor disminución significa que la variable es más importante para la creación de los nodos de los árboles. En este caso son Price y shelveLoc.



Predicciones sobre los datos de prueba

- cada predicción será un valor numérico continuo que el modelo estima para cada uno de los 80 datos de prueba.
- Aquí, las 80 predicciones serán una lista de valores numéricos continuos.
- Para evaluar el rendimiento del modelo con estas predicciones, podrías realizar las siguientes acciones:
- Comparar las predicciones con los valores reales: Esto te permitirá calcular métricas como el error cuadrático medio (MSE), el error absoluto medio (MAE) o el coeficiente de determinación (R^2) si es un problema de regresión.

```
> predicciones
      8      15      17      21      23      26      29      48      50
8.970000 8.526154 8.526154 4.140000 4.140000 11.110000 3.909286 7.566250 7.132857
      52      65      67      68      69      70      77      83      87
5.887778 6.907143 8.953077 6.907143 11.152143 8.953077 9.365556 7.132857 7.177647
      99     101     104     117     119     123     129     131     132
11.152143 5.697895 5.887778 2.509000 6.958333 5.697895 4.140000 9.365556 6.958333
      137     140     160     167     168     169     172     174     177
10.036000 10.036000 11.091429 6.759286 6.958333 8.729231 9.365556 2.509000 6.759286
      189     190     192     203     216     217     222     231     234
9.365556 8.729231 7.132857 4.140000 6.759286 5.791667 8.953077 8.729231 8.729231
      241     256     272     273     275     278     280     281     286
10.036000 5.887778 5.697895 13.174000 10.036000 5.190769 5.190769 6.759286 5.190769
      289     293     295     300     302     305     306     309     310
6.958333 11.110000 11.110000 8.953077 9.365556 11.152143 5.190769 6.759286 7.566250
      320     330     334     335     336     339     343     344     347
5.190769 13.174000 5.791667 11.091429 5.697895 6.447333 6.469231 6.447333 5.791667
      349     350     358     359     373     379     382     389
13.174000 8.680000 11.091429 7.566250 6.958333 6.469231 6.759286 5.887778
```


Conclusiones del modelo random forest:

- El MSE del modelo nos da un 3.958141 lo cual nos da un valor bastante considerable.
- Y un R2 de 0,52 con 100 arboles y a medida que el numero de arboles aumenta, este valor mas se aproxima a 1.
- Utilizando un solo árbol y los mismos datos de entrenamiento nos da un R2 de 0,45, lo cual nos da una conclusión de que a mas arboles, el modelo resulta bastante mejor.

Values	
mse	3.95814132695
particiones	int [1:320] 49 321 153 74 228 146 ...
predicciones	Named num [1:80] 9.55 8.68 7.59 6....
r2	0.52957155227139
rss	316.651306156
tss	673.11258

mse	3.95814132695
particiones	int [1:320] 49 321 153 74 228 146 ...
predicciones	Named num [1:80] 8.97 8.53 8.53 4.
r2	0.456794648434618
rss	365.638355661981
tss	673.11258



NEURONAL NETWORK - REGRESIÓN

Exploración de datos:

- sales: Ventas en miles de unidades (variable dependiente).
- comp_price: Precio de la competencia.
- income: Ingreso promedio.
- advertising: Gasto en publicidad.
- population: Población del área.
- price: Precio del asiento de seguridad.
- shelveLoc: Ubicación en la estantería.
- age: Edad promedio de la población.
- education: Nivel de educación promedio.
- urban: Indicador de si el área es urbana.
- us: Indicador de si el área está en Estados Unidos.

División de los Datos:

- Se dividieron los datos en un conjunto de entrenamiento y un conjunto de prueba. El 80% de los datos se utilizó para entrenar el modelo, mientras que el 20% restante se reservó para evaluar el rendimiento del modelo.

Preprocesamiento de Datos

- **Normalización:** Se normalizaron las variables numéricas para mejorar el rendimiento de la red neuronal.
- **Codificación de Variables Categóricas:** Las redes neuronales requieren que todos los datos sean numéricos. Las variables categóricas se convierten a números mediante la función `factor()`, que convierte las categorías en niveles numéricos, y `as.numeric()` convierte esos niveles en números.

Data	
▶ datos	400 obs. of 11 variables
datos_categoricos	400 obs. of 0 variables
▶ datos_entrenamiento	321 obs. of 11 variables
▶ datos_n	400 obs. of 11 variables
▶ datos_numericos	400 obs. of 11 variables
▶ datos_numericos_n	400 obs. of 11 variables
▶ datos_prueba	79 obs. of 11 variables

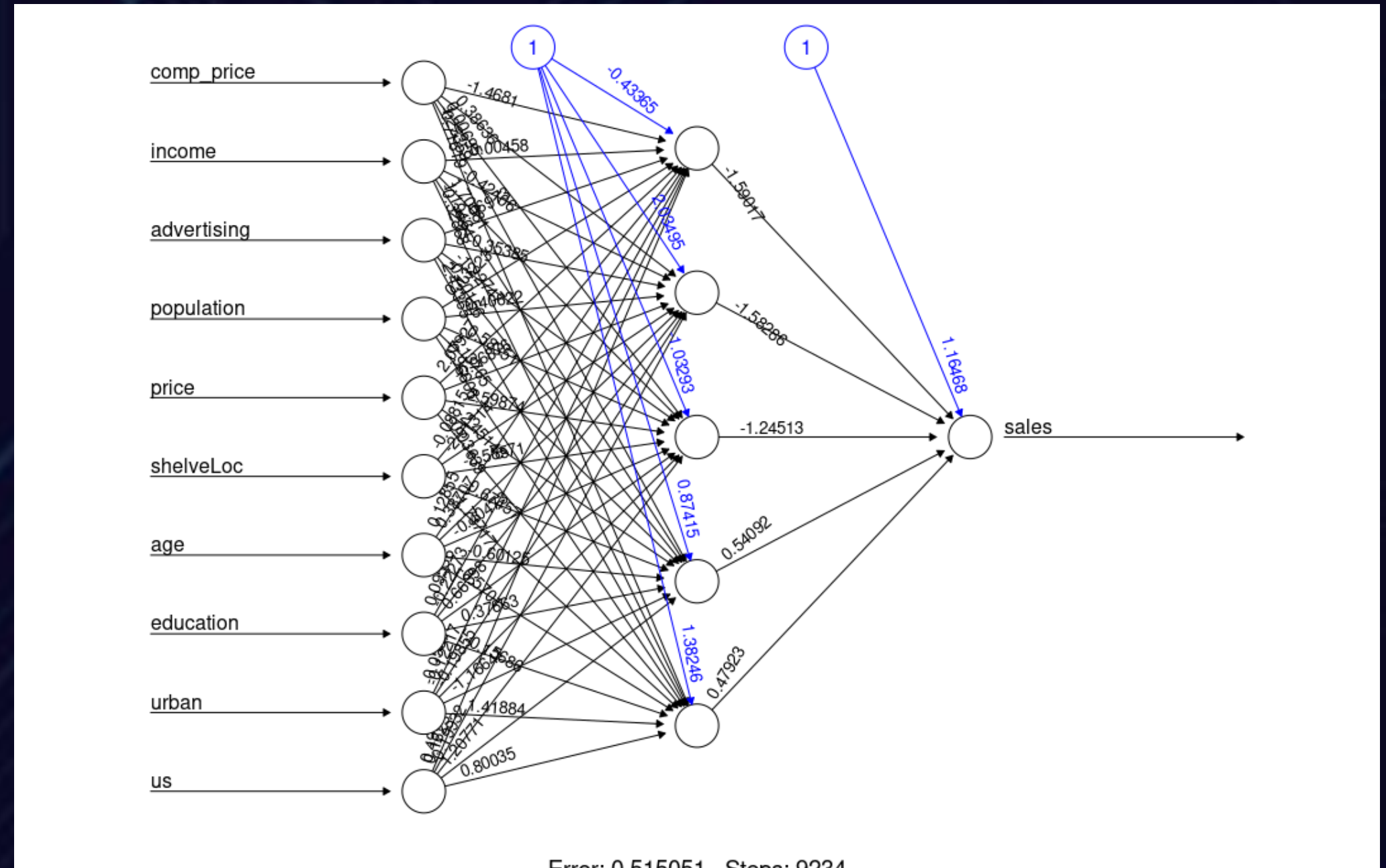
Modelo 1: Red Neuronal con 1 Capa Oculta y 5 Neuronas.

Arquitectura

- Número de Capas: 2 (1 capa oculta + 1 capa de salida).
- Número de Neuronas en la Capa Oculta: 5
- Capa de Entrada: Las variables de entrada del modelo.
- Capa Oculta: 5 neuronas.
- Capa de Salida: 1 neurona (para la predicción continua de sales).

Características

- **Simplicidad:** Este modelo tiene una sola capa oculta, lo que lo hace más simple en comparación con modelos más complejos.
- **Entrenamiento:** Generalmente más rápido debido a la menor cantidad de neuronas y conexiones.
- **Capacidad de Modelado:** Puede no capturar relaciones complejas en los datos, ya que solo tiene una capa oculta con un número limitado de neuronas.



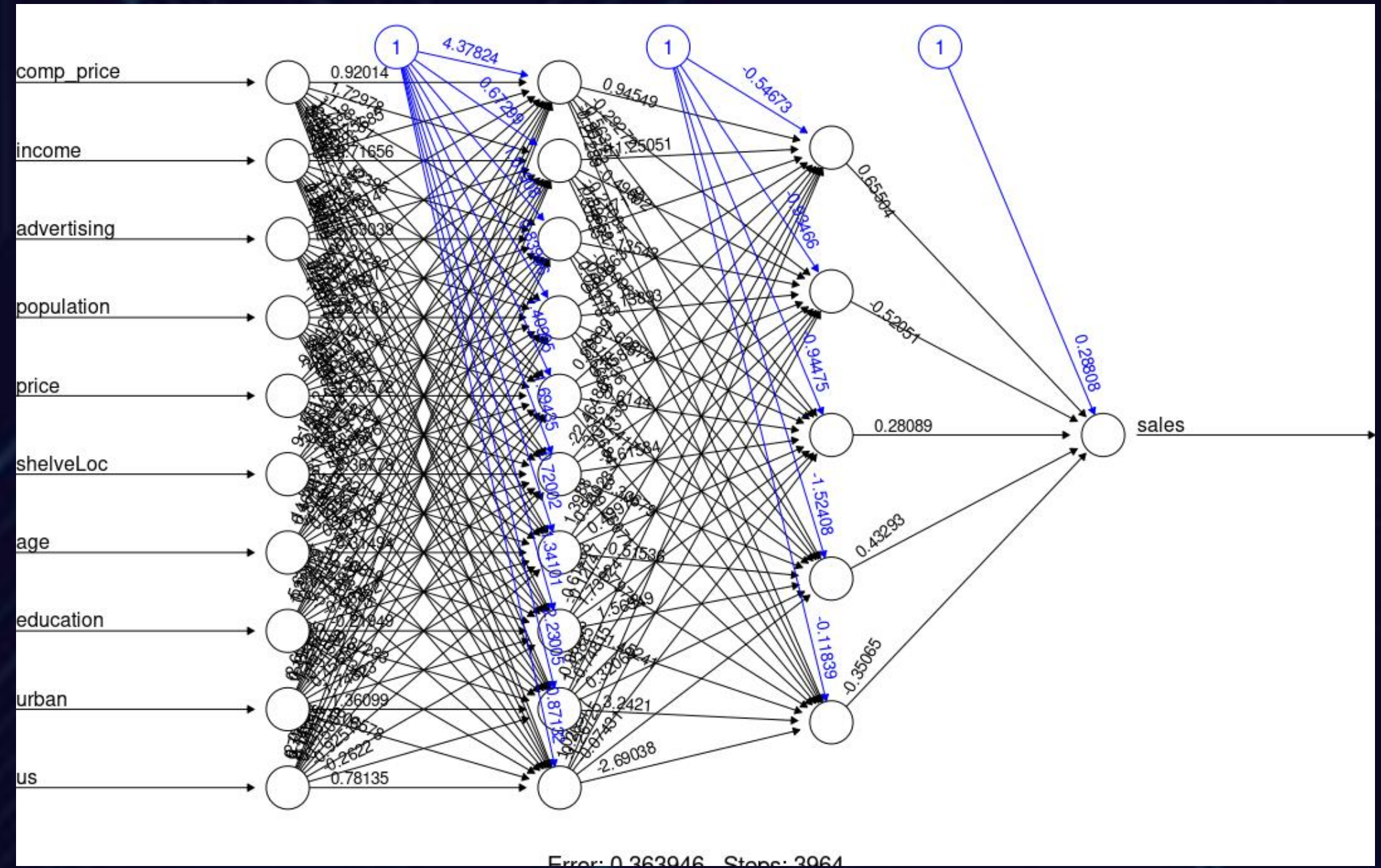
Modelo 2: Red Neuronal con 2 Capas Ocultas y 10 Neuronas en la Primera Capa, 5 en la Segunda.

Arquitectura

- Número de Capas: 3 (2 capas ocultas + 1 capa de salida)
- Número de Neuronas en la Primera Capa Oculta: 10
- Número de Neuronas en la Segunda Capa Oculta: 5
- Capa de Entrada: Las variables de entrada del modelo.
- Primera Capa Oculta: 10 neuronas.
- Segunda Capa Oculta: 5 neuronas.
- Capa de Salida: 1 neurona (para la predicción continua de sales).

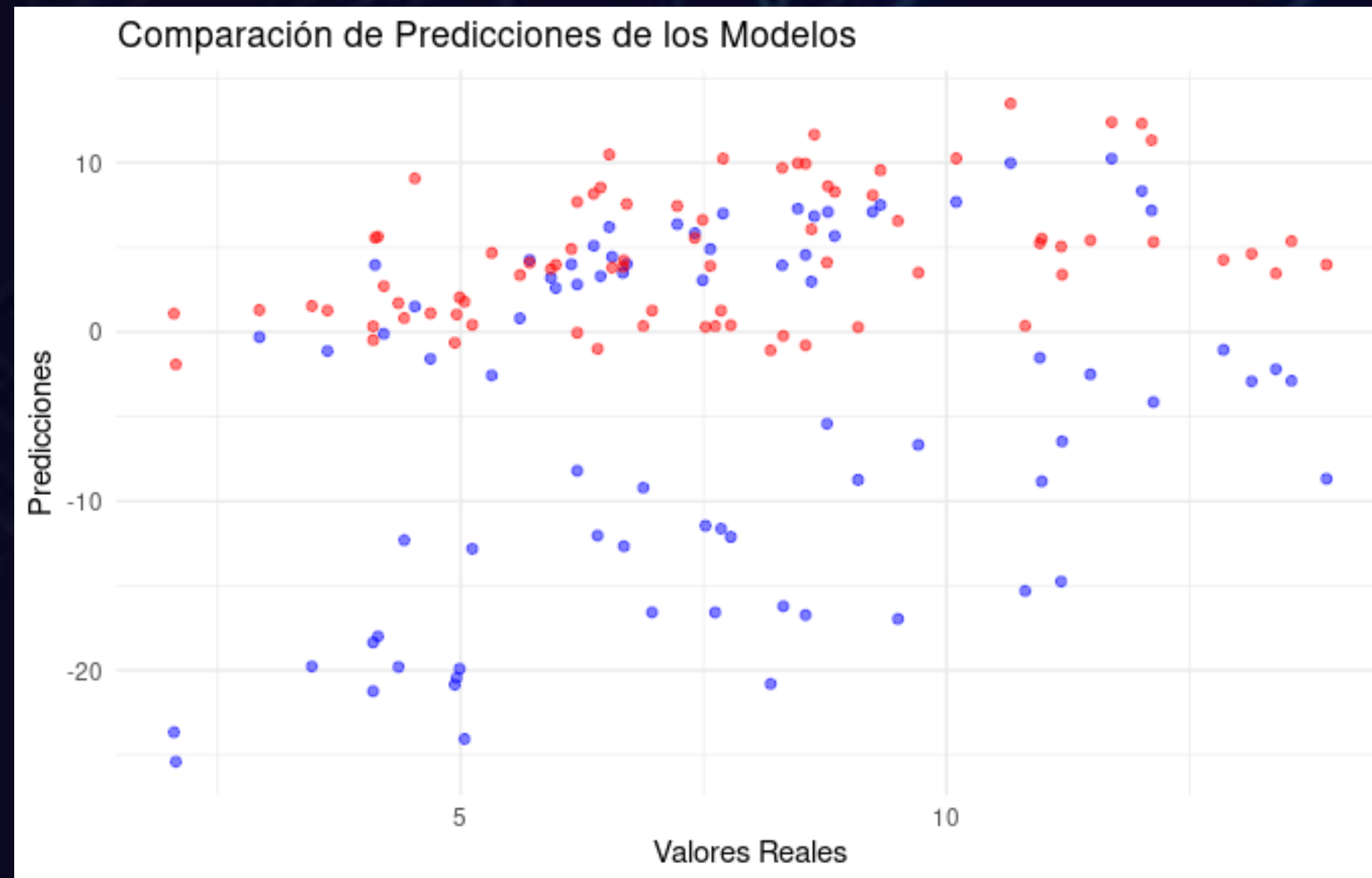
Características

- **Complejidad:** Este modelo es más complejo debido a la presencia de dos capas ocultas y una mayor cantidad de neuronas en la primera capa.
- **Capacidad de Modelado:** Puede capturar relaciones más complejas y no lineales en los datos debido a su mayor capacidad de modelado.
- **Entrenamiento:** Generalmente más lento debido a la mayor cantidad de neuronas y conexiones.



Predicciones sobre los datos de prueba

- Se compara las predicciones generadas por los Modelos 1 y 2 con los valores reales de las ventas (sales) para evaluar cuál modelo proporciona mejores resultados.
- **Modelo 1 (Azul):** Se observa cómo las predicciones se alinean con los valores reales. Una dispersión significativa indica que el modelo podría no estar capturando bien los patrones.
- **Modelo 2 (Rojo):** Se compara cómo las predicciones del Modelo 2 se alinean con los valores reales. Un ajuste más cercano a los valores reales sugiere un mejor rendimiento.
- **Comparación:** La superposición de los puntos de predicción y los valores reales muestra qué modelo se acerca más a los valores reales.



Conclusiones del modelo neuronal network:

- **Modelo con Mejor Rendimiento:** Modelo 2 (10 Neuronas, 2 Capas Ocultas).
- **R^2 Superior:** El Modelo 2 presenta un valor de R^2 más alto en comparación con el Modelo 1. Esto sugiere que captura de manera más efectiva la relación entre las variables independientes y la variable dependiente (sales).
- **Predicciones Más Precisos:** Los errores de predicción (MAE y MSE) son menores en el Modelo 2, lo que indica que sus predicciones están más cerca de los valores reales.
- El Modelo 2 presenta una mayor capacidad de modelado y flexibilidad, permitiéndole capturar relaciones complejas en los datos y proporcionar predicciones más precisas, como lo indica su superior valor de R^2 y menores errores (MAE y MSE). Sin embargo, esta mayor capacidad viene con desventajas: su mayor complejidad puede dificultar la interpretación del modelo y prolongar el tiempo de entrenamiento, requiriendo más recursos computacionales.

Values	
datos_prueba_real	num [1:79] 9.5 4.15 10.81 4.69 10.96 ...
formula	(unknown)
maxs	Named num [1:11] 16.3 175 120 29 509 ...
mins	Named num [1:11] 0 77 21 0 10 24 1 25 10 1 ...
R2_1	-28.8099188083424
R2_2	-1.90196435219585
SSE1	19143.5764413763
SSE2	1863.60710217242
SST1	642.188144303797
SST2	642.188144303797
valores_reales1	num [1:79] 9.5 4.15 10.81 4.69 10.96 ...
valores_reales2	num [1:79] 9.5 4.15 10.81 4.69 10.96 ...

SUPPORT VECTOR MACHINES (SVM)

Una vez hecho el análisis de los datos y que se hayan hecho la conversión de carácter a factorial, se dividen los datos y se crea el conjunto de entrenamiento con un 80% de los datos, para luego crear el conjunto de prueba.

```
# Dividir los datos en conjuntos de entrenamiento y prueba
set.seed(123)
#createDataPartition() es una función del paquete caret en R
#que se utiliza para dividir un conjunto de datos en subconjuntos,
#generalmente para separar los datos en conjuntos de entrenamiento y prueba
#en este caso se entrena el 80% de los datos.
training_samples <- createDataPartition(data$sales, p = 0.8, list = FALSE)
# Aquí se está creando el conjunto de entrenamiento.
#se usa los índices almacenados en training_samples,
#se selecciona las filas correspondientes del conjunto de datos original data
#para formar el conjunto de entrenamiento train_data.
train_data <- data[training_samples, ]
#Aquí se está creando el conjunto de prueba.
#se utiliza los índices negativos [-training_samples]
#para seleccionar las filas que no fueron seleccionadas para el conjunto de entrenamiento,
#y así crear el conjunto de prueba test_data.
test_data <- data[-training_samples, ]
```

Se convierte los datos factorial a tipo numérico ya que, SVM solo permite datos numéricos y se aplica esta conversión a los datos de entrenamiento y prueba para luego, normalizar los datos predictorios.

```
# Convertir factores a numéricos y normalizar las variables predictorias
convert_to_numeric <- function(df) {
  df_numeric <- df
  for (col in names(df)) {
    if (is.factor(df[[col]])) {
      df_numeric[[col]] <- as.numeric(df[[col]])
    }
  }
  return(df_numeric)
}

# Aplicar la conversión a los datos de entrenamiento y prueba
train_data_numeric <- convert_to_numeric(train_data)
test_data_numeric <- convert_to_numeric(test_data)

# Identificar columnas predictorias (todas excepto 'sales')
predictor_cols <- setdiff(names(train_data_numeric), "sales")

# Normalizar las variables predictorias
train_data_scaled <- train_data_numeric
train_data_scaled[, predictor_cols] <- scale(train_data_numeric[, predictor_cols])

test_data_scaled <- test_data_numeric
test_data_scaled[, predictor_cols] <- scale(test_data_numeric[, predictor_cols])
```


Ahora, se comienza a entrenar el modelo, se realizan predicciones y se calcula rmse y el rcuadrado

```
# Entrenar el modelo SVM lineal
modelo_SVM <- svm(sales ~ ., data = train_data, kernel = "linear", cost = 1, scale = TRUE)

# Realizar predicciones con el modelo
predicciones <- predict(modelo_SVM, test_data)

# Calcular MSE y RMSE
mse <- mean((predicciones - test_data$sales)^2)
rmse <- sqrt(mse)
print(paste("Raíz del error cuadrático medio (RMSE): ", rmse))

# Calcular R-cuadrado
suma_total_cuadrados <- sum((test_data$sales - mean(test_data$sales))^2)
suma_cuadrado_residual <- sum((predicciones - test_data$sales)^2)
r_cuadrado <- 1 - (suma_cuadrado_residual / suma_total_cuadrados)
print(paste("R-cuadrado: ", r_cuadrado))

#RMSE de 1.09: La desviación promedio de las predicciones
#con respecto a los valores reales es de 1.09 unidades.
#R-cuadrado de 0.852: El modelo explica aproximadamente el 85.2% de la variabilidad en las ventas,
#lo que indica una buena capacidad predictiva.
#Ambos resultados sugieren que el modelo está funcionando bastante bien,
#captura una buena parte de la variabilidad en las ventas
#y con errores de predicción relativamente bajos.
```

```
[1] "Raíz del error cuadrático medio (RMSE):  1.09516590701396"
```

```
[1] "R-cuadrado:  0.852454951734289"
```


Se define una matriz de combinaciones de hiperparámetros que se utilizarán para ajustar un modelo de machine learning SVM con un kernel radial.

- `expand.grid()` contiene todas las combinaciones posibles de los valores de los hiperparámetros que se especifiquen.
- `C` controla la penalización de errores de clasificación en el margen.
- `sigma = 2(-5:2)` es el hiperparámetro en el kernel radial (RBF) que define la escala de la función gaussiana usada por el kernel 2⁻⁵ a 2².

```
# Definir la matriz de combinaciones de hiperparámetros
matriz_combinaciones <- expand.grid(C = 2(-5:2), sigma = 2(-5:2))

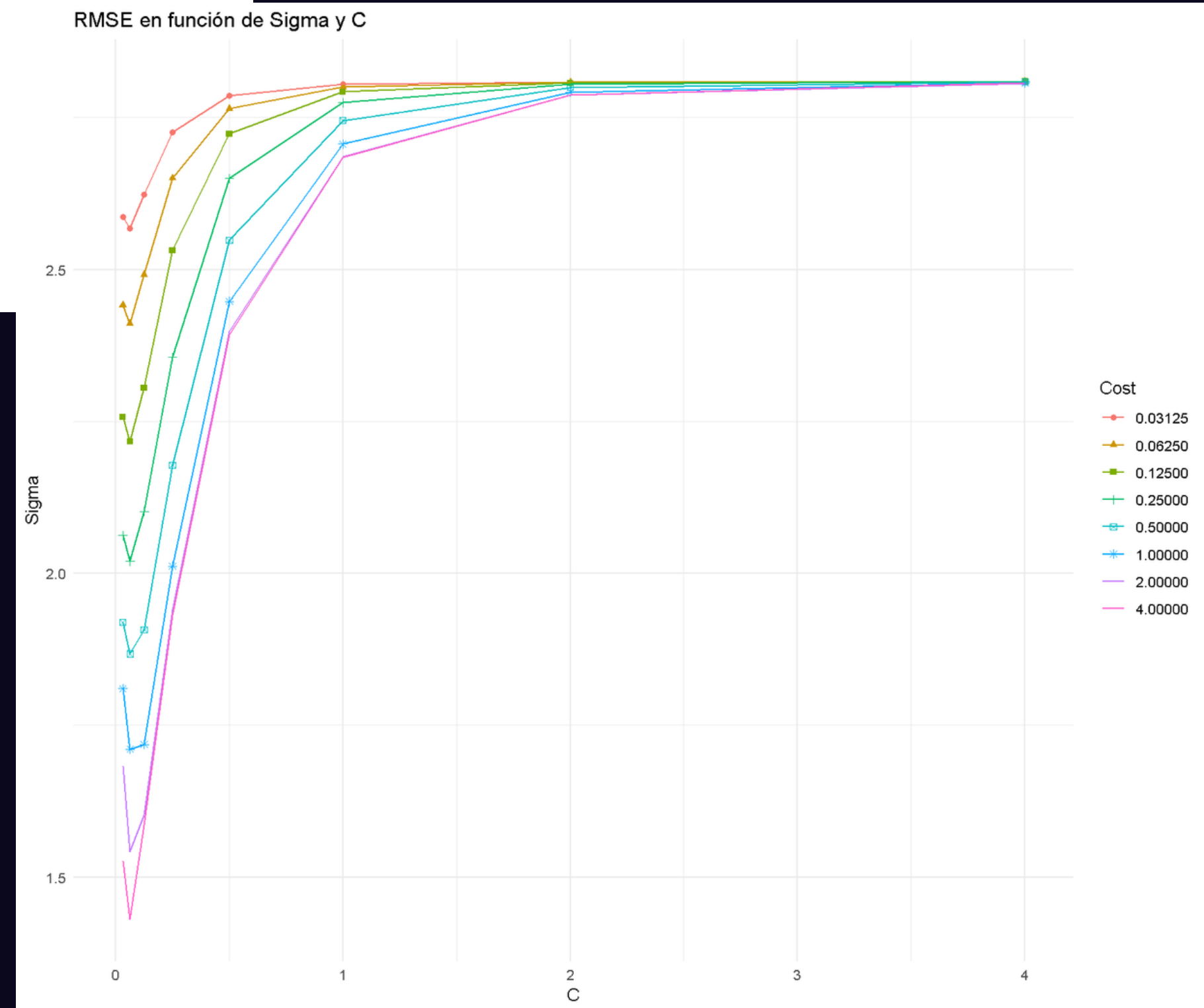
# Entrenar con validación cruzada usando svmRadial
modelo_ajustado <- train(sales ~ ., data = train_data_scaled, method = "svmRadial",
                        tuneGrid = matriz_combinaciones, trControl = trainControl(method = "cv"))

# Ver el mejor modelo
print(modelo_ajustado)
```

C	sigma	RMSE	Rsquared	MAE
0.03125	0.03125	2.586630	0.41999076	2.066989
0.03125	0.06250	2.567320	0.42077358	2.049394
0.03125	0.12500	2.623129	0.40225289	2.094183
0.03125	0.25000	2.725364	0.35780061	2.178846
0.03125	0.50000	2.786200	0.25455844	2.230634
0.03125	1.00000	2.804621	0.11395221	2.246607
0.03125	2.00000	2.808367	0.03814734	2.250016
0.03125	4.00000	2.808890	0.01862678	2.250525
0.06250	0.03125	2.441042	0.45241975	1.944438
0.06250	0.06250	2.410947	0.45632320	1.917570
0.06250	0.12500	2.491699	0.43150649	1.981033
0.06250	0.25000	2.650332	0.36812130	2.112821
0.06250	0.50000	2.764559	0.26171369	2.211570
0.06250	1.00000	2.800346	0.11679769	2.243084
0.06250	2.00000	2.807944	0.03875286	2.250105
0.06250	4.00000	2.809086	0.01868897	2.251199
0.12500	0.03125	2.257191	0.49772117	1.794723
0.12500	0.06250	2.216754	0.50458787	1.756311

```
ggplot(modelo_ajustado, aes(x = C, y = sigma, fill = RMSE)) +
  geom_tile() +
  scale_fill_viridis_c(option = "plasma") +
  labs(title = "RMSE en función de Sigma y C",
       x = "C",
       y = "Sigma",
       fill = "RMSE") +
  theme_minimal()
```

A medida que aumenta el valor de "C",
el RMSE tiende a disminuir hasta
alcanzar un punto mínimo y luego se
estabiliza y la relación entre Sigma y
RMSE depende del valor de "C". Por lo
que el valor más apropiado es cuando
 $c=4$



El **modelo SVM** ha demostrado un buen rendimiento en la predicción de ventas, con un **RMSE de 1.09**, indicando errores de predicción relativamente bajos, y un **R-cuadrado de 0.852**, que muestra que el modelo explica el **85.2%** de la variabilidad en las ventas.

Estos resultados sugieren que **el modelo es preciso y eficaz**.

Los hiperparámetros óptimos seleccionados fueron $\sigma = 0.0625$ y $C = 4$, los cuales optimizaron el rendimiento del modelo.

K-NEAREST NEIGHBORS (KNN)

Una vez que se convierten los datos de carácter a numérico se dividen el conjunto de entrenamiento y prueba.

Luego se definen los valores de k y el tipo de distancia (Manhattan y Euclidiana)

```
# Dividir los datos en conjuntos de entrenamiento y prueba
set.seed(123)
particiones <- sample(1:nrow(data), size = 0.8 * nrow(data))
datos_entrenamiento <- data[particiones, ]
datos_prueba <- data[-particiones, ]

# Definir los valores de k y tipos de distancia a experimentar
valores_k <- c(1,2,3,4, 5,6, 7,8, 9,10,11,12,13,15)
distancias <- c(1, 2) # 1: Manhattan, 2: Euclidiana
```

Se crea un dataframe para almacenar los resultados y se comienza a iterar según cada valor de k y su distancia para realizar la predicciones obteniendo lo siguiente:

```
# Crear un dataframe para almacenar los resultados
resultados <- data.frame(k = integer(), distancia = character(), MSE = numeric(), R2 = numeric())

# Iterar sobre los valores de k y distancias
for (k in valores_k) {
  for (dist in distancias) {

    # Aplicar el modelo KNN
    modelo_knn <- kknn(sales ~ .,
                      datos_entrenamiento,
                      datos_prueba[, -1], # Excluyendo la columna de ventas
                      k = k,
                      distance = dist)

    # Obtener las predicciones
    predicciones_knn <- fitted(modelo_knn)

    # Calcular el MSE
    mse_knn <- mean((datos_prueba$sales - predicciones_knn)^2)

    # Calcular el coeficiente de determinación (R^2)
    sum_total_cuadrado_knn <- sum((datos_prueba$sales - mean(datos_prueba$sales))^2)
    rss_knn <- sum((datos_prueba$sales - predicciones_knn)^2)
    r2_knn <- 1 - (rss_knn / sum_total_cuadrado_knn)

    # Almacenar los resultados
    dist_name <- ifelse(dist == 1, "Manhattan", "Euclidiana")
    resultados <- rbind(resultados, data.frame(k = k, distancia = dist_name, MSE = mse_knn, R2 = r2_knn))
  }
}

# Mostrar los resultados
print(resultados)
```

```
# Print(resultados)
  k  distancia      MSE      R2
1  Manhattan 6.290154 0.04625276
1  Euclidiana 5.108562 0.22541203
2  Manhattan 4.985948 0.24400345
2  Euclidiana 4.501453 0.31746526
3  Manhattan 4.288302 0.34978445
3  Euclidiana 4.153166 0.37027443
4  Manhattan 3.951838 0.40080081
4  Euclidiana 3.878741 0.41188424
5  Manhattan 3.769002 0.42852352
5  Euclidiana 3.650328 0.44651745
6  Manhattan 3.683435 0.44149756
6  Euclidiana 3.505873 0.46842046
7  Manhattan 3.608995 0.45278463
7  Euclidiana 3.424315 0.48078685
8  Manhattan 3.550111 0.46171291
8  Euclidiana 3.380053 0.48749806
9  Manhattan 3.522429 0.46591024
9  Euclidiana 3.380496 0.48743091
10 Manhattan 3.501335 0.46910852
10 Euclidiana 3.395593 0.48514174
11 Manhattan 3.482937 0.47189820
11 Euclidiana 3.403018 0.48401597
12 Manhattan 3.463598 0.47483041
12 Euclidiana 3.408619 0.48316668
13 Manhattan 3.444771 0.47768514
13 Euclidiana 3.416026 0.48204358
15 Manhattan 3.434909 0.47918040
```

```
# Identificar la configuración con el menor MSE
mejor_mse <- resultados[which.min(resultados$MSE), ]
print(paste("Mejor configuración basada en MSE:"))
print(mejor_mse)

# Identificar la configuración con el mayor R^2
mejor_r2 <- resultados[which.max(resultados$R2), ]
print(paste("Mejor configuración basada en R^2:"))
print(mejor_r2)
```

Se escoje el mejor mse y

k	distancia	MSE	R2
8	Euclidiana	3.380053	0.4874981

Conclusión:

Aunque el modelo con $k=8$ y la distancia Euclidiana ofrece el mejor resultado en comparación con otras combinaciones, el valor de $R^2=0.49$, se sugiere que el modelo no es particularmente robusto en su capacidad para predecir las ventas.

Esto indica que podría ser necesario mejorar el modelo, ya sea ajustando más parámetros, explorando diferentes métodos de modelado, o revisando las características del conjunto de datos.

CONCLUSIONES

Para evaluar el desempeño de los modelos de predicción de ventas, hemos utilizado dos métricas principales: el RMSE (raíz del error cuadrático medio) y el R^2 (coeficiente de determinación).

- El RMSE (raíz del error cuadrático medio) indica el error medio de las predicciones, mientras más cercano a cero menor es el error de predicción.
- El R cuadrado (coeficiente de determinación) mide la variabilidad de la variables dependiente, en este caso, ventas. Mientras más cercano a uno menor es la dispersión de los datos.

En este análisis, se compararon diferentes modelos de aprendizaje supervisado para predecir las ventas, incluyendo Árboles de Decisiones (con y sin poda), Random Forest, Redes Neuronales, Máquinas de Soporte Vectorial (SVM) y K-Nearest Neighbors (KNN). Los resultados obtenidos se muestran en la tabla adjunta:

Modelo	[valor aproximado]	RMSE [valor aproximado]
Árbol de decisiones sin poda	0,56	1,89
Árbol de decisiones con poda	0,54	1,94
Random forest	0,53	1,98
Neuronal network	-1,90	4,86
Support Vector Machines	0,85	1,09
K-Nearest Neighbors	0,49	1,84

- El Modelo de Máquinas de Soporte Vectorial (SVM) mostró el mejor desempeño general con el RMSE más bajo (1.09) y el R^2 más alto (0.85). Esto indica que el modelo SVM proporciona las predicciones más precisas y explica mejor la variabilidad en las ventas.
- Árboles de Decisiones (con y sin poda) y K-Nearest Neighbors (KNN) tienen un desempeño aceptable con valores intermedios de RMSE y R^2 , pero no alcanzan el rendimiento del SVM.
- Random Forest y Árboles de Decisiones con poda presentan resultados ligeramente peores, con RMSE superiores y R^2 menores.
- Red Neuronal resultó ser el peor modelo en términos de RMSE y R^2 , indicando que no se ajustó bien a los datos y sus predicciones son menos confiables.

- Tanto los aspectos teóricos y empíricos obtenidos sugieren que el modelo de máquinas de soporte vectorial (SVM) es la opción más efectiva, dada su capacidad para minimizar el error de predicción y explicar la variabilidad en los datos de manera más precisa.
- Los otros modelos, especialmente el de Red Neuronal, no ofrecen un rendimiento superior y podrían no ser adecuados para predecir las ventas de sillas de seguridad para niños en automóviles.



THANK YOU

