

Trabajo 2

Pablo Setrakian Bearzotti

2025-04-05

Contents

1. Introducción	2
2. Conjunto de Datos	2
2.1 Descripción general	3
2.2 Variables disponibles	3
2.3 Justificación de la elección	3
2.4 Limpieza del conjunto de datos	3
2.5 Visualización del conjunto de datos	5
3. Aplicación de la Técnica de Regresión	7
3.1 Regresión Lineal múltiple	7
3.1.1 Conjuntos de entrenamiento y validación	7
3.1.2 Aplicación del modelo de regresión lineal múltiple	8
3.1.3 Contraste de Hipótesis	9
3.1.4 Interpretación de los Coeficientes	11
3.1.5 Selección de variables predictoras	13
3.1.6 Diagnosis del modelo	13
3.1.7 Relación lineal entre la variable respuesta y las variables predictoras	13
3.1.8 Homocedasticidad de los residuos	17
3.1.8.1 Residuals vs Fitted	19
3.1.8.2 Normal Q-Q Plot	19
3.1.8.3 Scale-Location (Spread-Location)	19
3.1.8.4 Residuals vs Leverage	19
3.1.9 Independencia de los residuos	20
3.1.10 Normalidad de los residuos	20
3.1.11 Multicolinealidad	20
3.1.12 Estudio de valores atípicos	29
3.1.13 Estudio de Leverages	30
3.1.14 Reajuste del modelo	33
3.1.15 Medidas de precisión	38
3.1.15.1 Técnicas de Validación: K-Folds Cross-Validation	39
3.1.16 Predicción	42
3.1.17 Conclusión	42
3.2 Regresión Penalizada	42
3.2.1 Regresión Ridge	44
3.2.2 Regresión Lasso	48
3.2.3 Conclusión	52
3.3 K-Nearest Neighbors (KNN) para Regresión	52
3.3.1 Depuración del conjunto de datos	52
3.3.2 Búsqueda del valor de K	53

3.3.3 Conjuntos de entrenamiento y validación	54
3.3.4 Aplicación del modelo KNN	55
3.3.5 Predicción y Medidas de Precisión	55
3.3.6 Conclusión	56
3.4 Árboles Aleatorios para Regresión	56
3.4.1 Depuración del conjunto de datos	56
3.4.2 Búsqueda óptima para los parámetros: profundidad y complejidad	57
3.4.3 Conjuntos de entrenamiento y validación	59
3.4.4 Aplicación del modelo Árboles Aleatorios	60
3.4.5 Predicción y Medidas de Precisión	65
3.4.6 Conclusión	66
4. Aplicación de la Técnica de Clasificación	66
4.1 Regresión Logística	66
4.1.2 Conjuntos de Entrenamiento y Validación	68
4.1.3 Aplicación del Modelo de Regresión Logística	76
4.1.4 Contraste de Hipótesis	77
4.1.5 Interpretación de los Coeficientes	77
4.1.6 Clasificación y Medidas de Precisión	78
4.1.7 Conclusión	82
4.2 K-Nearest Neighbors (KNN) para Clasificación	83
4.2.1 Depuración del conjunto de datos	83
4.2.2 Búsqueda del valor de K	83
4.2.3 Conjuntos de Entrenamiento y Validación	84
4.2.4 Aplicación del modelo KNN	85
4.2.5 Clasificación y Medidas de Precisión	85
4.2.7 Conclusión	87
4.2 Árboles Aleatorios para Clasificación	87
4.2.1 Depuración del conjunto de datos	87
4.2.2 Búsqueda óptima para los parámetros: profundidad y complejidad	88
4.2.3 Conjuntos de Entrenamiento y Validación	90
4.2.4 Aplicación del modelo de Árboles Aleatorios	90
4.2.5 Clasificación y Medidas de Precisión	91
4.2.6 Conclusión	93

1. Introducción

El presente trabajo tiene como objetivo explorar y analizar cómo las características disponibles en el conjunto de datos `boston.csv` se relacionan con una variable continua y una variable categórica, utilizando técnicas de aprendizaje supervisado. En particular, se desarrollarán modelos de regresión para predecir una variable cuantitativa y modelos de clasificación para estimar una variable cualitativa.

El análisis se enfoca no solo en la aplicación de los modelos, sino también en la evaluación de su precisión mediante métricas adecuadas, la comparación de resultados y la interpretación estadística de los mismos.

Además, se incluye un análisis exploratorio detallado que permite comprender mejor el comportamiento de los datos y guiar la selección de variables predictoras.

Todo esto con el fin de extraer conclusiones relevantes sobre el conjunto de datos y las metodologías empleadas.

2. Conjunto de Datos

Para la realización de este trabajo se ha seleccionado el conjunto de datos “**The Boston HousePrice Data**”, disponible públicamente en la plataforma Kaggle: <https://www.kaggle.com/datasets/fedesoriano/the-boston-houseprice-data>

Este dataset es ampliamente utilizado en problemas de regresión dentro del aprendizaje automático y la estadística, lo que lo convierte en una excelente elección para estudiar cómo las características socioeconómicas y urbanísticas de diferentes distritos de Boston influyen en el precio medio de la vivienda. Los datos fueron recopilados en **1978**, y aunque no son actuales, siguen siendo muy relevantes para propósitos educativos y de modelado, ya que ofrecen una estructura clara y un contexto realista.

El dataset fue seleccionado debido a su **interés práctico en el contexto inmobiliario**. Analizar cómo diferentes factores impactan el valor de las viviendas permite no solo aplicar técnicas estadísticas y de machine learning, sino también entender mejor el funcionamiento del mercado inmobiliario desde una perspectiva analítica.

2.1 Descripción general

El dataset contiene **506 observaciones** correspondientes a distintos distritos de Boston y **14 variables**, de las cuales 13 son predictoras y 1 es la variable objetivo: el valor medio de las viviendas.

2.2 Variables disponibles

Variable	Descripción
CRIM	Tasa de criminalidad per cápita por ciudad
ZN	Proporción de terreno residencial dividido en parcelas > 25,000 pies ²
INDUS	Proporción de hectáreas comerciales no minoristas por ciudad
CHAS	Variable ficticia (= 1 si la zona está junto al río Charles; 0 en caso contrario)
NOX	Concentración de óxidos de nitrógeno (partes por 10 millones)
RM	Número medio de habitaciones por vivienda
AGE	Proporción de viviendas ocupadas por propietarios construidas antes de 1940
DIS	Distancia ponderada a cinco centros de empleo de Boston
RAD	Índice de accesibilidad a autopistas radiales
TAX	Tasa de impuestos a la propiedad por cada 10,000 dólares
PTRATIO	Ratio alumno/profesor por ciudad
B	1000(Bk - 0.63) ² , donde Bk es la proporción de población afroamericana
LSTAT	Porcentaje de población con bajo estatus socioeconómico
MEDV	Valor medio de las viviendas (en miles de dólares) – <i>variable objetivo</i>

2.3 Justificación de la elección

- Permite abordar problemas de **regresión supervisada** prediciendo el valor medio de las viviendas (**MEDV**).
- Es posible construir un problema de **clasificación supervisada** categorizando la variable **MEDV**.
- Contiene una combinación rica y variada de variables numéricas y categóricas.
- Es un dataset muy utilizado y documentado, ideal para comparar resultados.
- Su contexto urbano lo hace especialmente relevante dada mi **afinidad con el sector inmobiliario**.

2.4 Limpieza del conjunto de datos

El primer paso consiste en limpiar el conjunto de datos para identificar las variables relevantes. Este proceso incluye la selección de variables, el manejo de datos faltantes y la eliminación de valores atípicos, entre otros aspectos.

A continuación, se examina la estructura del conjunto de datos.

```
library(readr)
library(tidyverse)
```

```

## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
## v dplyr     1.1.4     v purrr     0.0.4
## vforcats    1.0.0     v stringr    1.5.1
## v ggplot2    3.5.1     v tibble     3.2.1
## v lubridate  1.9.4     v tidyv     1.3.1
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()   masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(here)

## here() starts at /Users/pablo/Documents/avd/rstudio
boston_housing <- read_csv(here("data", "boston.csv"))

## Rows: 506 Columns: 14
## -- Column specification -----
## Delimiter: ","
## dbl (14): CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LS...
## 
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
data <- boston_housing |>
  glimpse()

## Rows: 506
## Columns: 14
## $ CRIM    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## $ ZN      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 1~
## $ INDUS   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.~
## $ CHAS    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX    <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~
## $ RM      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## $ AGE     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## $ DIS     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ RAD     <dbl> 1, 2, 2, 3, 3, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, ~
## $ TAX     <dbl> 296, 242, 242, 222, 222, 311, 311, 311, 311, 311, 311, 311, 31~
## $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ B       <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ LSTAT   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## $ MEDV    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~

data <- boston_housing |>
  glimpse()

## Rows: 506
## Columns: 14
## $ CRIM    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## $ ZN      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 1~
## $ INDUS   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.~
## $ CHAS    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX    <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~
## $ RM      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## $ AGE     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## $ DIS     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
```

```

## $ RAD      <dbl> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, ~
## $ TAX      <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 311, 31~
## $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ B        <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ LSTAT    <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## $ MEDV    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~

```

Se verificará la existencia de valores perdidos:

```

data |>
  mutate(id_row = row_number()) |>
  filter(if_any(everything(), is.na))

## # A tibble: 0 x 15
## # i 15 variables: CRIM <dbl>, ZN <dbl>, INDUS <dbl>, CHAS <dbl>, NOX <dbl>,
## #   RM <dbl>, AGE <dbl>, DIS <dbl>, RAD <dbl>, TAX <dbl>, PTRATIO <dbl>,
## #   B <dbl>, LSTAT <dbl>, MEDV <dbl>, id_row <int>

```

No existen valores perdidos en el conjunto de datos. En caso de que existan, pueden eliminarse mediante el siguiente procedimiento.

```

data <- data |>
  drop_na() |>
  glimpse()

## # Rows: 506
## # Columns: 14
## $ CRIM      <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## $ ZN        <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ INDUS    <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## $ CHAS      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX       <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, 0.524, ~
## $ RM        <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## $ AGE       <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## $ DIS        <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ RAD        <dbl> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, ~
## $ TAX        <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 311, 31~
## $ PTRATIO   <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15.2, 15~
## $ B          <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ LSTAT     <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## $ MEDV      <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~

```

Finalmente, se debe identificar la variable respuesta y las predictoras correspondientes. En este caso, la variable respuesta es `medv` y las variables predictoras son todas las demás.

$$MEDV = \beta_0 + \beta_1 CRIM + \beta_2 ZN + \beta_3 INDUS + \beta_4 NOX + \beta_5 RM + \beta_6 AGE + \beta_7 DIS + \beta_8 RAD + \beta_9 TAX + \beta_{10} PTRATIO + \beta_{11} B$$

2.5 Visualización del conjunto de datos

Una vez depurado el conjunto de datos, el siguiente paso consiste en realizar una visualización para examinar si existe alguna relación entre la variable dependiente y las variables independientes de manera aislada. En caso de que tal relación exista, también será posible determinar su naturaleza, lo que es clave, ya que si la relación no es lineal, no sería adecuado aplicar un modelo de regresión lineal. Para ello, se utiliza un gráfico de dispersión.

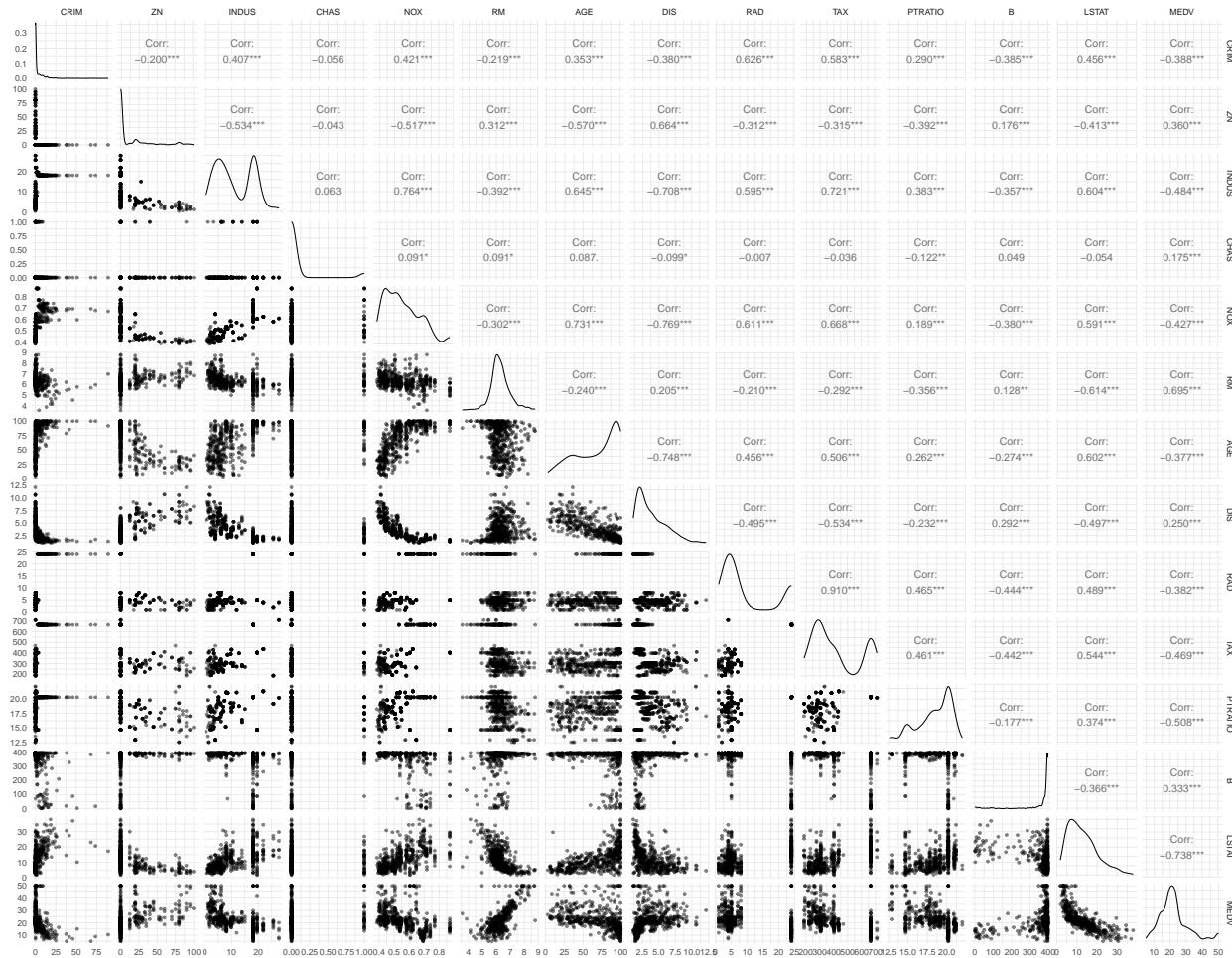
La función `ggpairs()` del paquete GGally permite generar una matriz de correlaciones entre las variables de interés.

En la parte superior de la matriz se encuentra el coeficiente de correlación de Pearson, acompañado de una simbología que indica si dicha correlación es estadísticamente significativa. En la diagonal de la matriz se muestra la función de densidad de cada variable, mientras que en la parte inferior se presentan los gráficos de dispersión entre cada par de variables.

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
library(ggplot2)

data |>
  ggpairs(
    lower = list(continuous = wrap("points", alpha = 0.5, size = 1)),
    diag = list(continuous = wrap("densityDiag")),
    upper = list(continuous = wrap("cor", size = 4))
  ) +
  theme_minimal()
```



Se observa que la variable `medv` tiene una correlación positiva con las variables `rm`, `zn`, `dis`, `rad`, `tax`, `ptratio` y `b`. Por otro lado, tiene una correlación negativa con las variables `crim`, `indus`, `nox`, `age` y `lstat`.

Esto sugiere que a medida que aumenta el número de habitaciones, la proporción de terrenos residenciales, la

distancia a los centros de empleo, la proporción de población estudiantil y la concentración de óxidos nítricos, el valor medio de las viviendas también tiende a aumentar.

Por el contrario, a medida que aumenta la tasa de criminalidad, la proporción de terrenos industriales, la edad de las viviendas y la proporción de población con bajo nivel educativo, el valor medio de las viviendas tiende a disminuir.

3. Aplicación de la Técnica de Regresión

A continuación, se aplicarán distintas técnicas de regresión.

3.1 Regresión Lineal múltiple

La regresión lineal múltiple es una técnica estadística que tiene como objetivo establecer una relación lineal entre una variable dependiente Y y un conjunto de p variables independientes o predictoras X_1, X_2, \dots, X_p .

Esta relación se expresa mediante el siguiente modelo:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p + \varepsilon$$

En esta ecuación, β_0 representa la ordenada en el origen, es decir, el valor esperado de Y cuando todas las variables predictoras son cero.

Los coeficientes $\beta_1, \beta_2, \dots, \beta_p$ indican el efecto promedio de cada variable independiente sobre la variable dependiente, manteniendo las demás constantes.

Por su parte, ε representa el término de error aleatorio, que recoge las desviaciones del modelo respecto a los datos observados.

Para que el modelo sea válido y útil en la práctica, es fundamental que el término de error cumpla ciertas hipótesis estadísticas, como la normalidad, independencia y homocedasticidad. En las siguientes secciones, se detallará paso a paso cómo construir, interpretar y validar un modelo óptimo de regresión lineal múltiple a partir del conjunto de datos utilizado.

3.1.1 Conjuntos de entrenamiento y validación

Después de confirmar que existe una relación entre las variables, el siguiente paso es dividir el conjunto de datos en dos partes: el Conjunto de Entrenamiento y el Conjunto de Validación, con el propósito de evaluar el modelo de regresión más adelante.

Se aplicará la regla de partición 70/30 —comúnmente utilizada, aunque en algunos casos se emplea una proporción 80/20—, asignando el 70 % de las observaciones al conjunto de entrenamiento y el 30 % restante al conjunto de validación.

```
library(rsample)

set.seed(1234)

data_split <- data |>
  initial_split(prop = 0.7)

data_train <- data_split |>
  training() |>
  glimpse()

## Rows: 354
## Columns: 14
```

En consecuencia, el conjunto de entrenamiento se empleará para ajustar el modelo, mientras que el conjunto de validación se utilizará para evaluar su rendimiento, cuantificando su precisión mediante técnicas de validación adecuadas.

3.1.2 Aplicación del modelo de regresión lineal múltiple

Para la aplicación del modelo de regresión se emplea la función lm() del lenguaje R, la cual permite ajustar un modelo de regresión lineal a los datos disponibles.

```
mlr <- lm(MEDV ~ ., data = data_train)

summary(mlr)

## 
## Call:
## lm(formula = MEDV ~ ., data = data_train)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -16.2210  -2.5130  -0.7122   1.8295  26.9382
```

```

## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 26.399485  6.347070  4.159 4.05e-05 ***
## CRIM         -0.111639  0.035323 -3.160 0.001717 **  
## ZN           0.044388  0.015841  2.802 0.005367 **  
## INDUS        -0.008549  0.071304 -0.120 0.904641    
## CHAS          2.793031  0.972463  2.872 0.004333 **  
## NOX          -17.494132 4.314041 -4.055 6.22e-05 *** 
## RM            4.907947  0.542364  9.049 < 2e-16 ***
## AGE          -0.016715  0.015386 -1.086 0.278074    
## DIS           -1.441577  0.229553 -6.280 1.03e-09 *** 
## RAD           0.254051  0.075422  3.368 0.000843 *** 
## TAX           -0.011540  0.004309 -2.678 0.007765 **  
## PTRATIO       -0.838551  0.154130 -5.441 1.02e-07 *** 
## B             0.009602  0.003200  3.000 0.002896 **  
## LSTAT         -0.341352  0.062357 -5.474 8.55e-08 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 4.656 on 340 degrees of freedom
## Multiple R-squared:  0.7508, Adjusted R-squared:  0.7413 
## F-statistic: 78.79 on 13 and 340 DF,  p-value: < 2.2e-16

```

3.1.3 Contraste de Hipótesis

Una vez aplicado el modelo de regresión lineal, corresponde interpretar los resultados obtenidos.

Contraste de hipótesis global:

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0 \quad H_1 : \exists \beta_j \neq 0$$

Este contraste global permite evaluar si alguna de las variables predictoras ejerce influencia sobre la variable respuesta MEDV.

La hipótesis nula establece que ninguna de las variables predictoras tiene efecto sobre MEDV, mientras que la hipótesis alternativa plantea que al menos una de ellas sí presenta un efecto significativo.

Este contraste se encuentra en la última fila del resumen del modelo ajustado (`summary(modelo)`), donde se muestra el valor del estadístico F y su correspondiente valor-p. En nuestro caso:

- Estadístico F: **78.79**
- Valor-p asociado: **< 2.2e-16**

Como el valor-p es inferior al nivel de significación $\alpha = 0.05$, se obtiene un resultado significativo. Por tanto:

Se rechaza la hipótesis nula, y se concluye que al menos una de las variables predictoras influye sobre la variable respuesta MEDV.

Contraste individual para cada variable predictora:

A continuación, se analizan los contrastes individuales para cada uno de los coeficientes del modelo, es decir, para cada variable predictora:

$$H_0 : \beta_j = 0 \quad \text{vs} \quad H_1 : \beta_j \neq 0 \quad \text{para } j = 1, \dots, p$$

En la salida del modelo, en la columna $\text{Pr}(>|t|)$ se encuentran los valores-p para cada predictor. Si el valor-p es menor que $\alpha = 0.05$, se considera significativo y se rechaza la hipótesis nula para ese predictor.

En este caso:

- Las variables **CRIM**, **ZN**, **CHAS**, **NOX**, **RM**, **DIS**, **RAD**, **TAX**, **PTRATIO**, **B** y **LSTAT** tienen un **p-valor < 0.05**, por lo que **sí influyen significativamente sobre MEDV**.
 - Por otro lado, las variables **INDUS** y **AGE** tienen un **p-valor > 0.05**, por lo que **no resultan significativas** en este modelo y deben ser eliminadas.

```

## Residuals:
##      Min       1Q   Median      3Q      Max
## -15.9146 -2.6076 -0.6624  1.7950 26.6233
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 26.964605  6.317852  4.268 2.56e-05 ***
## CRIM        -0.111504  0.035264 -3.162 0.001707 **
## ZN          0.047037  0.015544  3.026 0.002665 **
## CHAS        2.712302  0.963911  2.814 0.005178 **
## NOX         -18.928319 4.004869 -4.726 3.35e-06 ***
## RM          4.800240  0.528360  9.085 < 2e-16 ***
## DIS         -1.363232  0.215102 -6.338 7.36e-10 ***
## RAD          0.261290  0.071417  3.659 0.000294 ***
## TAX         -0.011877  0.003781 -3.141 0.001829 **
## PTRATIO     -0.851209  0.152907 -5.567 5.25e-08 ***
## B            0.009343  0.003185  2.934 0.003573 **
## LSTAT       -0.361711  0.059422 -6.087 3.09e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.651 on 342 degrees of freedom
## Multiple R-squared:  0.7499, Adjusted R-squared:  0.7419
## F-statistic: 93.22 on 11 and 342 DF,  p-value: < 2.2e-16

```

El modelo ajustado incluye únicamente variables predictoras que presentan una influencia estadísticamente significativa sobre la variable respuesta.

3.1.4 Interpretación de los Coeficientes

Una vez ajustado el modelo de regresión lineal múltiple, es importante interpretar los coeficientes obtenidos para comprender la influencia de cada variable predictora sobre la variable respuesta **MEDV**.

Cada coeficiente β_j representa el efecto marginal de la variable predictora correspondiente X_j sobre la variable respuesta, manteniendo constante el resto de variables del modelo. Es decir, indica el cambio promedio en el valor de **MEDV** ante un aumento de una unidad en X_j , suponiendo que las demás variables permanecen fijas.

El signo del coeficiente permite determinar el tipo de relación: - Si $\beta_j > 0$, existe una relación **positiva**: al aumentar X_j , **MEDV** también tiende a aumentar. - Si $\beta_j < 0$, existe una relación **negativa**: al aumentar X_j , **MEDV** tiende a disminuir. - Si $\beta_j = 0$, X_j no tiene un efecto significativo sobre **MEDV**.

La magnitud del coeficiente también proporciona información sobre la intensidad del efecto, aunque esta debe interpretarse considerando la escala de cada variable.

A continuación, se analiza cada coeficiente del modelo final para interpretar cómo influye cada predictor sobre el precio medio de la vivienda (**MEDV**). Para ello, se emplean las funciones **coef()** y **confint()** para obtener los coeficientes y sus intervalos de confianza.

```

mlr |>
  coef()

##   (Intercept)       CRIM        ZN       CHAS       NOX
## 26.964605293 -0.111503585  0.047037331  2.712302097 -18.928319264
##           RM        DIS       RAD       TAX    PTRATIO
## 4.800240073 -1.363232423  0.261289889 -0.011876504 -0.851208764
##           B        LSTAT
## 0.009343363 -0.361710511

```

```
mlr |>
  confint()
```

```
##              2.5 %      97.5 %
## (Intercept) 14.537866557 39.391344030
## CRIM        -0.180864883 -0.042142286
## ZN          0.016464128  0.077610535
## CHAS        0.816361697  4.608242498
## NOX         -26.805595147 -11.051043382
## RM           3.760996553  5.839483593
## DIS          -1.786320898 -0.940143948
## RAD          0.120818748  0.401761030
## TAX          -0.019313010 -0.004439998
## PTRATIO     -1.151965657 -0.550451870
## B            0.003079476  0.015607249
## LSTAT       -0.478589810 -0.244831212
```

- **(Intercept)**: Si no se consideran ninguna de las variables predictoras, el ‘MEDV’ toma un valor medio de **26.96K \$** con un intervalo de confianza al 95%, de [14.54, 39.39]K \$.
- **CRIM**: Por cada unidad adicional en la tasa de criminalidad per cápita, el precio medio disminuye en **0.11 unidades**, manteniendo constantes el resto de variables. Este efecto es negativo y su intervalo de confianza $[-0.18, -0.04]$ no contiene el 0, por lo que es **estadísticamente significativo**.
- **ZN**: Por cada unidad adicional en el porcentaje de suelo residencial de grandes parcelas, MEDV aumenta en **0.047 unidades**, manteniendo constantes el resto de variables. Su efecto es positivo y significativo, con un intervalo de confianza [0.016, 0.078].
- **CHAS**: Si una vivienda se encuentra junto al río Charles (**CHAS = 1**), el valor medio de MEDV es **2.71 unidades mayor** que si no lo está, manteniendo constantes las demás variables. Este efecto es **significativo** [0.82, 4.61].
- **NOX**: Un aumento de una unidad en la concentración de óxidos de nitrógeno se asocia con una disminución de **18.93 unidades** en MEDV, manteniendo constantes el resto de variables. Su efecto es muy negativo y **estadísticamente significativo**, con un intervalo $[-26.81, -11.05]$.
- **RM**: Cada habitación adicional por vivienda incrementa el valor medio de MEDV en **4.80 unidades**, manteniendo constantes el resto de variables, siendo uno de los efectos más fuertes y **positivos**, con intervalo [3.76, 5.84].
- **DIS**: A mayor distancia a centros de empleo, el valor de MEDV disminuye en **1.36 unidades** por unidad de DIS, manteniendo constantes el resto de variables. El efecto es **significativo y negativo**, con intervalo $[-1.79, -0.94]$.
- **RAD**: Por cada incremento en el índice de accesibilidad a autopistas radiales, MEDV aumenta en **0.26 unidades**, manteniendo constantes el resto de variables, con un intervalo [0.12, 0.40]. Es un efecto **positivo y significativo**.
- **TAX**: Cada aumento de una unidad en la tasa de impuesto sobre bienes inmuebles reduce el valor medio de MEDV en **0.012 unidades**, manteniendo constantes el resto de variables, con intervalo $[-0.0193, -0.0044]$, por lo tanto, **significativo**.
- **PTRATIO**: Cada punto adicional en la razón alumno/profesor reduce MEDV en **0.85 unidades**, manteniendo constantes el resto de variables, con intervalo $[-1.15, -0.55]$. Este efecto es **significativo y negativo**.
- **B**: Un incremento de una unidad en la variable B (proporción de población negra) aumenta MEDV en **0.009 unidades**, manteniendo constantes el resto de variables. Aunque el efecto es pequeño, es **significativo**, con intervalo [0.0031, 0.0156].

- **LSTAT**: Por cada punto porcentual adicional en la proporción de población de bajo nivel socioeconómico, MEDV disminuye en **0.36 unidades**, manteniendo constantes el resto de variables, siendo uno de los efectos más fuertes. Su intervalo $[-0.48, -0.24]$ confirma que es **significativo y negativo**.

3.1.5 Selección de variables predictoras

No se considera necesario aplicar técnicas adicionales de selección de variables predictoras, ya que este proceso se ha abordado previamente durante el contraste de hipótesis.

A partir de dicho análisis, se identificaron y eliminaron aquellas variables que no presentaban una influencia significativa sobre la variable respuesta MEDV, lo que permitió simplificar el modelo sin comprometer su capacidad predictiva. Por tanto, se continúa el estudio utilizando únicamente las variables que han demostrado ser relevantes estadísticamente.

3.1.6 Diagnosis del modelo

Tras la aplicación e interpretación del modelo, resulta fundamental verificar su validez, lo que implica comprobar el cumplimiento de las hipótesis iniciales asociadas al comportamiento de los residuos. Para ello, se recopilan los residuos en un objeto de tipo tibble mediante la función `residuals()`.

```
resid <- tibble(resid = residuals(mlr)) |>
  glimpse()

## #> #> Rows: 354
## #> #> Columns: 1
## #> #> $ resid <dbl> 4.78619874, 0.80542656, -3.06207547, 2.65358741, -0.59914951, 0.~
```

3.1.7 Relación lineal entre la variable respuesta y las variables predictoras

Una de las hipótesis fundamentales del modelo de regresión lineal múltiple es que existe una **relación lineal** entre la variable dependiente Y y cada una de las variables predictoras X_1, X_2, \dots, X_p .

Esta relación puede expresarse mediante el siguiente modelo:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

donde: - β_0 es la ordenada en el origen (intercepto), - β_j representa el efecto medio de la variable X_j sobre Y , manteniendo el resto de variables constantes, - ε es el término de error aleatorio, que debe cumplir ciertas condiciones para que el modelo sea válido.

Para verificar esta relación lineal, es necesario **graficar los residuos del modelo** frente a cada variable predictora. Si los residuos se distribuyen de manera aleatoria (sin patrones claros), se puede asumir que existe una relación lineal adecuada.

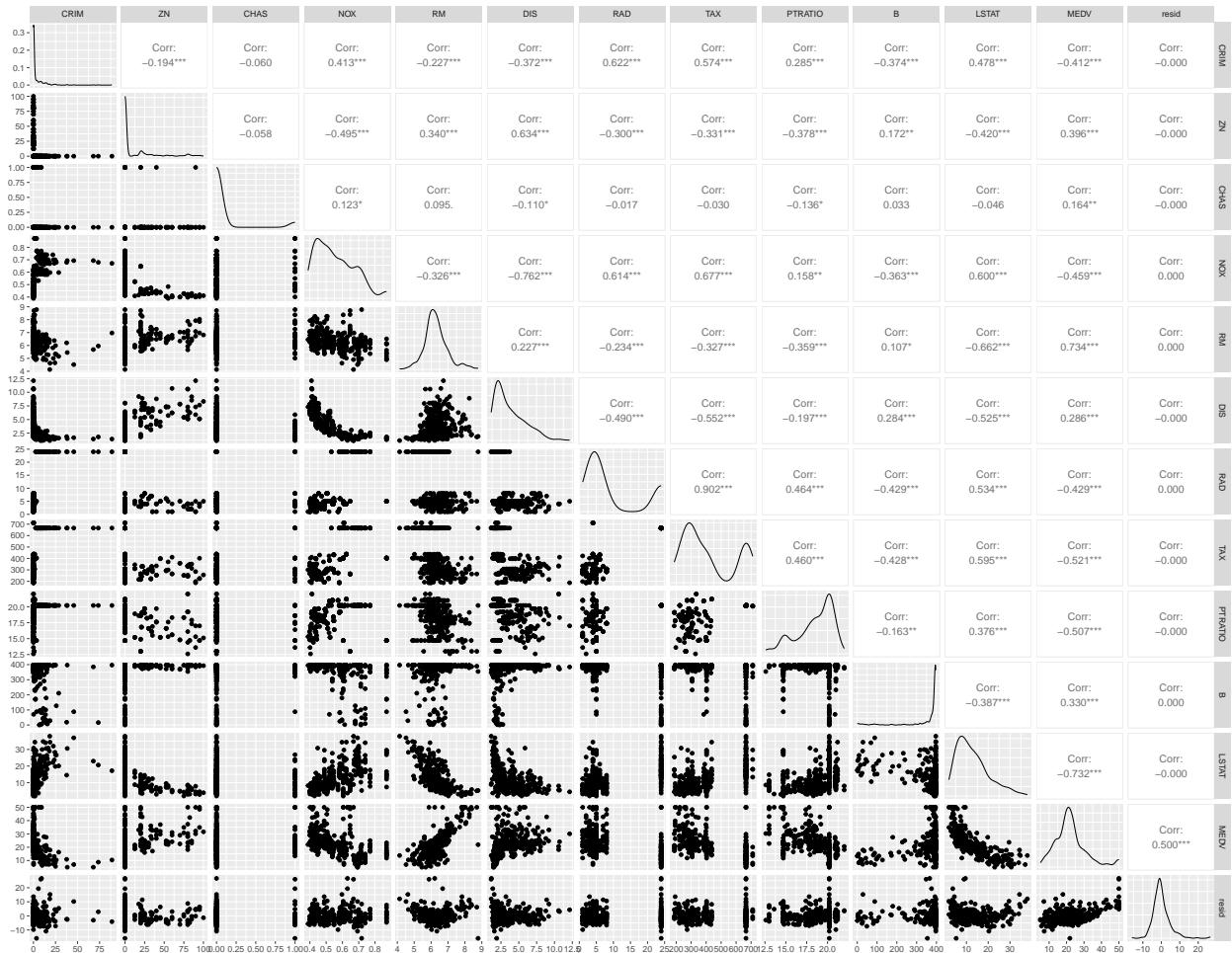
En caso contrario, si se observa algún patrón (curvas, acumulaciones, etc.), podría indicar una relación no lineal. En tal situación, es recomendable aplicar **transformaciones a las variables independientes**, tales como:

- $\log(X)$
- \sqrt{X}
- X^2

Estas transformaciones pueden ayudar a linealizar la relación entre las variables y mejorar el ajuste del modelo.

A continuación, se utilizan herramientas gráficas como `ggpairs()` para examinar los residuos frente a cada predictor, añadiendo los residuos al conjunto de entrenamiento:

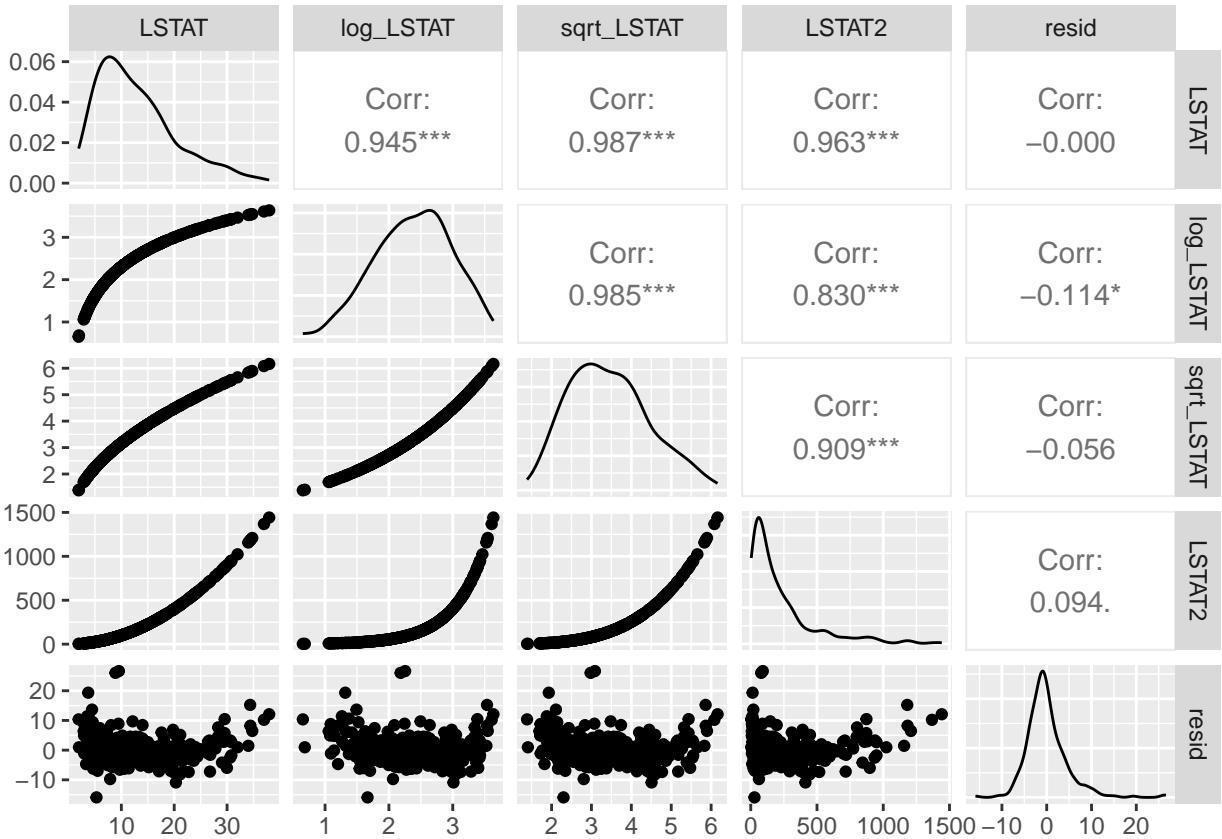
```
data_train |>
  mutate(
    resid = residuals(mlr)
  ) |>
  ggpairs()
```



Para la variable RM: aunque se observa un ligero patrón parabólico en los residuos, el diagrama de puntos con la variable respuesta MEDV y el coeficiente de correlación lineal sugieren que existe una relación lineal entre las variables. Por lo tanto, no es necesario aplicar ninguna transformación.

Para la variable LSTAT: se observa claramente que no hay una relación lineal con la variable MEDV. Los residuos muestran un patrón parabólico y el diagrama de puntos sugiere una posible relación exponencial. En este caso, es necesario transformar la variable LSTAT.

```
data_train |>
  select(LSTAT) |>
  mutate(
    log_LSTAT = log(LSTAT),
    sqrt_LSTAT = sqrt(LSTAT),
    LSTAT2 = LSTAT^2,
    resid = residuals(mlr)
  ) |>
  ggpairs()
```



Se concluye que la transformación más adecuada corresponde al término cuadrático (LSTAT²), ya que es la que presenta el menor patrón en los residuos y muestra una correlación lineal significativa con estos (-0.114*).

Se aplica la transformación logarítmica a la variable LSTAT y se ajusta nuevamente el modelo de regresión lineal múltiple.

```
data_train <- data_split |>  
  training() |>  
  mutate(LSTAT2 = LSTAT^2) |>  
  select(-INDUS, -AGE, -LSTAT) |>  
  glimpse()
```

```

data_test <- data_split |>
  testing() |>
  mutate(LSTAT2 = LSTAT^2) |>
  select(-INDUS, -AGE, -LSTAT) |>
  glimpse()

## Rows: 152
## Columns: 12
## $ CRIM    <dbl> 0.00632, 0.02729, 0.06905, 0.08829, 0.14455, 0.21124, 0.22489, ~
## $ ZN      <dbl> 18.0, 0.0, 0.0, 12.5, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.~
## $ CHAS    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX     <dbl> 0.538, 0.469, 0.458, 0.524, 0.524, 0.524, 0.524, 0.538, 0.538, ~
## $ RM      <dbl> 6.575, 7.185, 7.147, 6.012, 6.172, 5.631, 6.377, 5.990, 5.570, ~
## $ DIS      <dbl> 4.0900, 4.9671, 6.0622, 5.5605, 5.9505, 6.0821, 6.3467, 4.2579~
## $ RAD      <dbl> 1, 2, 3, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 5, 3, 3, 3, 3, 3, ~
## $ TAX      <dbl> 296, 242, 222, 311, 311, 311, 311, 307, 307, 307, 307, 30~
## $ PTRATIO <dbl> 15.3, 17.8, 18.7, 15.2, 15.2, 15.2, 21.0, 21.0, 21.0, 21~
## $ B        <dbl> 396.90, 392.83, 396.90, 395.60, 396.90, 386.63, 392.52, 386.75~
## $ MEDV    <dbl> 24.0, 34.7, 36.2, 22.9, 27.1, 16.5, 15.0, 17.5, 13.6, 15.2, 15~
## $ LSTAT2   <dbl> 24.8004, 16.2409, 28.4089, 154.5049, 366.7225, 895.8049, 418.2~

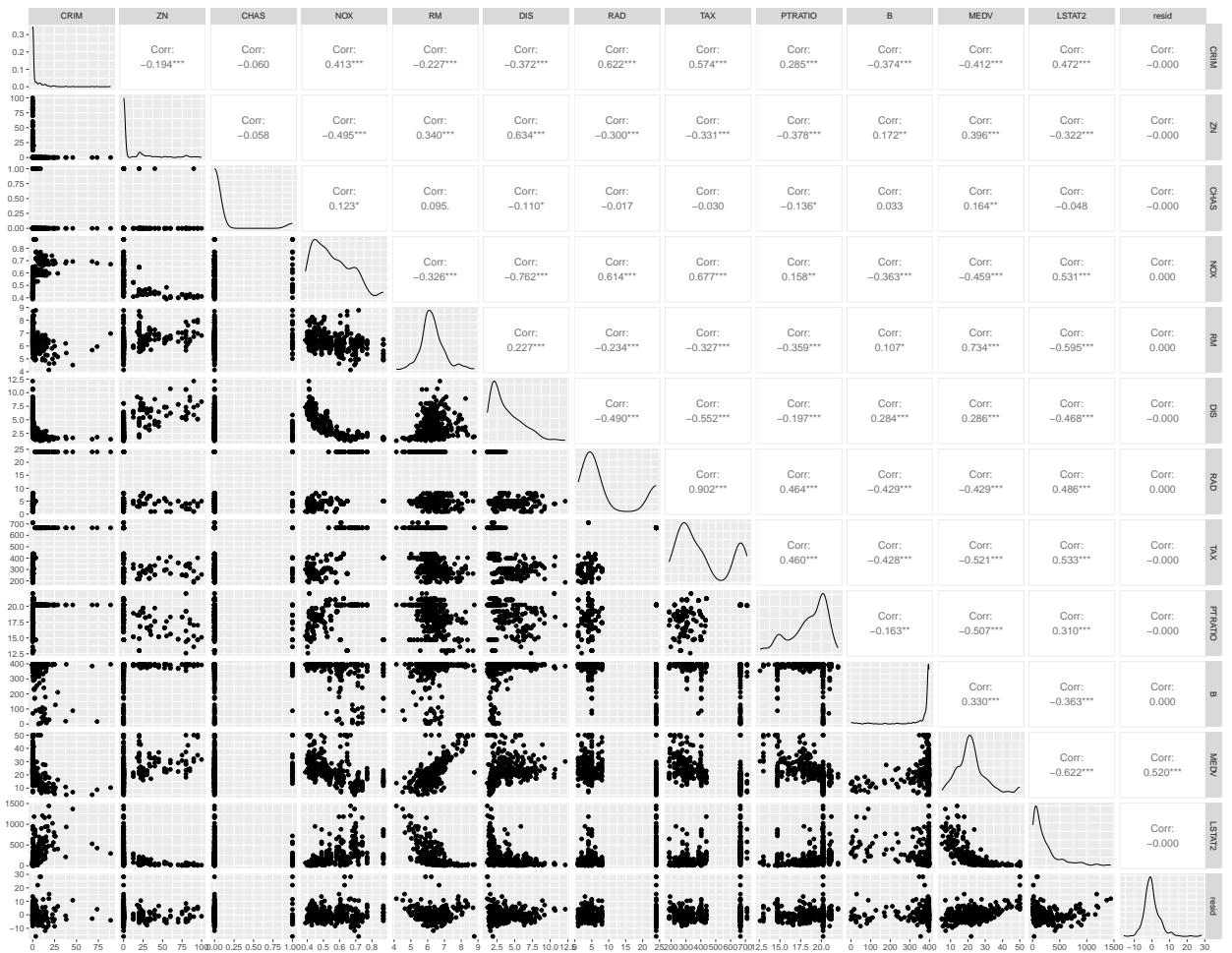
mlr <- lm(MEDV ~ ., data = data_train)

summary(mlr)

##
## Call:
## lm(formula = MEDV ~ ., data = data_train)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -16.0016 -2.7151 -0.5663  1.9764 28.2367 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 17.470194  6.359266  2.747 0.006329 ** 
## CRIM        -0.126284  0.036845 -3.427 0.000684 *** 
## ZN          0.049264  0.016268  3.028 0.002647 ** 
## CHAS        2.792229  1.002612  2.785 0.005652 ** 
## NOX        -21.331370 4.139977 -5.153 4.35e-07 *** 
## RM          5.945204  0.516332 11.514 < 2e-16 *** 
## DIS         -1.259756  0.224173 -5.620 3.97e-08 *** 
## RAD         0.255719  0.074300  3.442 0.000650 *** 
## TAX         -0.012675  0.003930 -3.225 0.001381 ** 
## PTRATIO     -0.898260  0.158791 -5.657 3.26e-08 *** 
## B           0.011756  0.003291  3.572 0.000406 *** 
## LSTAT2      -0.004458  0.001546 -2.884 0.004176 ** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 4.838 on 342 degrees of freedom
## Multiple R-squared:  0.7294, Adjusted R-squared:  0.7207 
## F-statistic: 83.8 on 11 and 342 DF,  p-value: < 2.2e-16

```

```
data_train |>
  mutate(
    resid = residuals(mlr)
  ) |>
  ggpairs()
```



En esta etapa, los residuos no presentan patrones visibles, lo que sugiere que la relación entre la variable respuesta y las variables predictoras puede considerarse lineal. En consecuencia, se puede continuar con el análisis.

3.1.8 Homocedasticidad de los residuos

Una de las condiciones fundamentales en la regresión lineal es la **homocedasticidad**, es decir, que los residuos del modelo presenten **varianza constante** a lo largo de todos los valores predichos. Si este supuesto no se cumple (lo que se conoce como **heterocedasticidad**), las estimaciones del modelo pueden ser poco fiables.

Para evaluar esta condición, se aplica el **test de Breusch-Pagan**, que permite contrastar las siguientes hipótesis:

- H_0 : Los residuos tienen varianza constante (homocedasticidad).
- ${}^{**}H_1^*$: La varianza de los residuos no es constante (heterocedasticidad).

```

library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

mlr |>
  bptest(
    studentize = FALSE
  )

##
## Breusch-Pagan test
##
## data: mlr
## BP = 213.23, df = 11, p-value < 2.2e-16

```

Dado que el *p*-valor obtenido es **inferior a 0.05**, el resultado es significativo y se **rechaza la hipótesis nula**. Esto indica evidencia de **heterocedasticidad en el modelo**.

Una posible solución es aplicar una **transformación sobre la variable respuesta**. Por ejemplo:

- Si la varianza **aumenta** con los valores de la variable dependiente, puede usarse $\log(Y)$ o \sqrt{Y} .
- Si la varianza **disminuye**, se podrían probar transformaciones como $\exp(Y)$ o Y^2 .

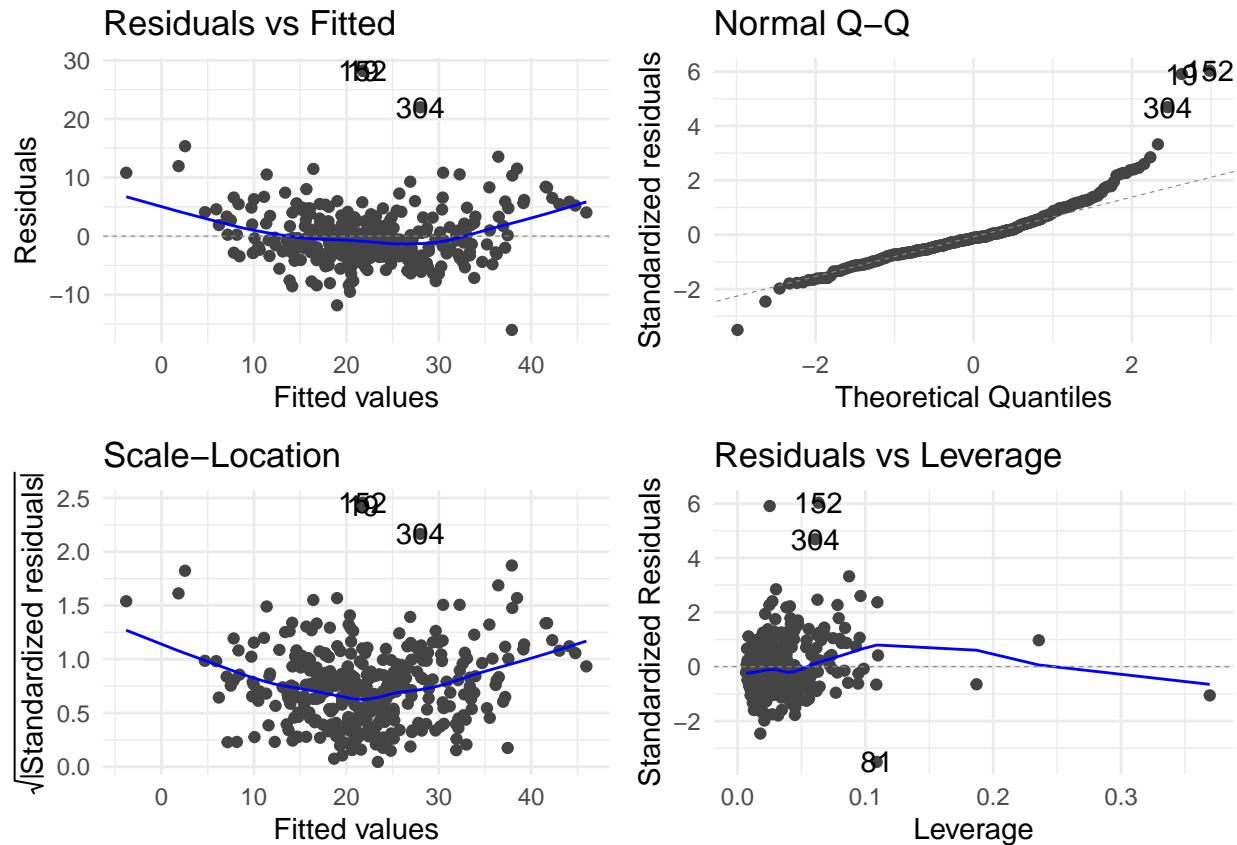
Para visualizar y entender mejor esta variación, se presentan a continuación los **gráficos de diagnóstico del modelo**:

```

library(ggfortify)

mlr |>
  autoplot() +
  theme_minimal()

```



A continuación se analizan los cuatro gráficos de diagnóstico del modelo de regresión lineal múltiple con el objetivo de verificar el cumplimiento de los supuestos fundamentales del modelo.

3.1.8.1 Residuals vs Fitted Este gráfico muestra una **ligera forma de embudo**, indicando que los residuos tienden a dispersarse un poco más conforme aumentan los valores ajustados. Este patrón puede interpretarse como una señal de **heterocedasticidad**, aunque el efecto no es pronunciado. La mayor parte de los puntos se agrupan de forma bastante homogénea, lo que sugiere que la **falta de homocedasticidad no es generalizada**, y puede estar influida por algunas observaciones concretas.

3.1.8.2 Normal Q-Q Plot El gráfico Q-Q muestra una desviación de los residuos estandarizados respecto a la línea diagonal en los extremos. Esto indica **cierta desviación respecto a la normalidad**, principalmente en las colas. No obstante, el patrón general es aceptable y **no parece haber una violación severa** del supuesto de normalidad de los errores.

3.1.8.3 Scale-Location (Spread-Location) Este gráfico también muestra **una leve tendencia ascendente en la dispersión** de los residuos frente a los valores ajustados. Al igual que en el gráfico anterior, se identifica una cierta heterocedasticidad, pero de nuevo **no es especialmente acusada** ni afecta de forma sistemática a todo el rango de valores.

3.1.8.4 Residuals vs Leverage En este gráfico se observan algunas observaciones con valores elevados de leverage (por ejemplo, las observaciones 152, 304 y 81). Aunque estos puntos tienen un mayor potencial de influencia, **ninguno de ellos presenta un efecto desproporcionado** sobre el ajuste del modelo.

En conclusión, aunque el contraste estadístico detecta una posible heterocedasticidad, el análisis visual sugiere que **la suposición de homocedasticidad se cumple razonablemente en la mayoría del rango de**

datos. Por tanto, **no se considera necesario aplicar transformaciones a la variable respuesta**, ya que el modelo parece suficientemente robusto para los fines del análisis.

3.1.9 Independencia de los residuos

Es necesario comprobar si los residuos son independientes. Para ello, se aplica el **Test de Durbin-Watson** que permite contrastar las hipótesis: - **H0**: Independencia vs **H1**: Dependencia

Este test se encuentra en la función `dwtest(model, alternative = "two.sided")` del paquete `lmtest`.

```
mlr |>
  dwtest(alternative = "two.sided")

## 
##   Durbin-Watson test
##
## data: mlr
## DW = 2.0983, p-value = 0.3535
## alternative hypothesis: true autocorrelation is not 0
```

Al obtener un p-valor superior a 0.05, no se rechaza la hipótesis nula, lo que indica que los residuos son independientes.

3.1.10 Normalidad de los residuos

Es fundamental que los residuos sigan una distribución normal. Para verificar esto, se utiliza el **Test de Kolmogorov-Smirnov-Lilliefors**, que evalúa las siguientes hipótesis:

- **H0**: Normalidad vs **H1**: No normalidad

Este test se encuentra en la función `lillie.test(residual)` del paquete `nortest`.

```
library(nortest)

resid |>
  pull() |>
  lillie.test()

## 
##   Lilliefors (Kolmogorov-Smirnov) normality test
##
## data: pull(resid)
## D = 0.11044, p-value = 3.355e-11
```

De nuevo, el resultado del test es significativo ($p\text{-valor} < 0.05$), por lo que se rechaza la hipótesis nula de normalidad.

No obstante, el gráfico Normal Q-Q revela que hay puntos en la parte inferior que se desvían de la diagonal, lo cual podría indicar la presencia de leverages.

3.1.11 Multicolinealidad

Uno de los principales inconvenientes en la regresión múltiple es la presencia de correlación entre las variables predictoras, ya que esto dificulta distinguir el aporte individual de cada una al modelo.

Para detectar este problema se emplean los factores de inflación de la varianza (VIF), los cuales reflejan la existencia de colinealidad. En general, un VIF menor a 5 indica ausencia de colinealidad relevante, mientras que valores superiores a este umbral revelan una alta dependencia entre variables, lo cual representa un inconveniente.

Como estrategia para abordarlo, se recomienda eliminar las variables altamente correlacionadas o bien combinarlas en un único predictor. El cálculo de los VIF se realiza mediante la función `vif()` de la librería `car`.

```
library(car)

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
##
##     recode

## The following object is masked from 'package:purrr':
##
##     some

mlr |>
  vif()

##      CRIM      ZN      CHAS      NOX      RM      DIS      RAD      TAX
## 1.784705 2.114849 1.071186 3.633428 1.909416 3.309511 6.377878 6.798556
##    PTRATIO      B      LSTAT2
## 1.776778 1.333422 2.325876
```

Además, se utiliza la función `check_collinearity()` del paquete `performance` para analizar y visualizar gráficamente los niveles de colinealidad:

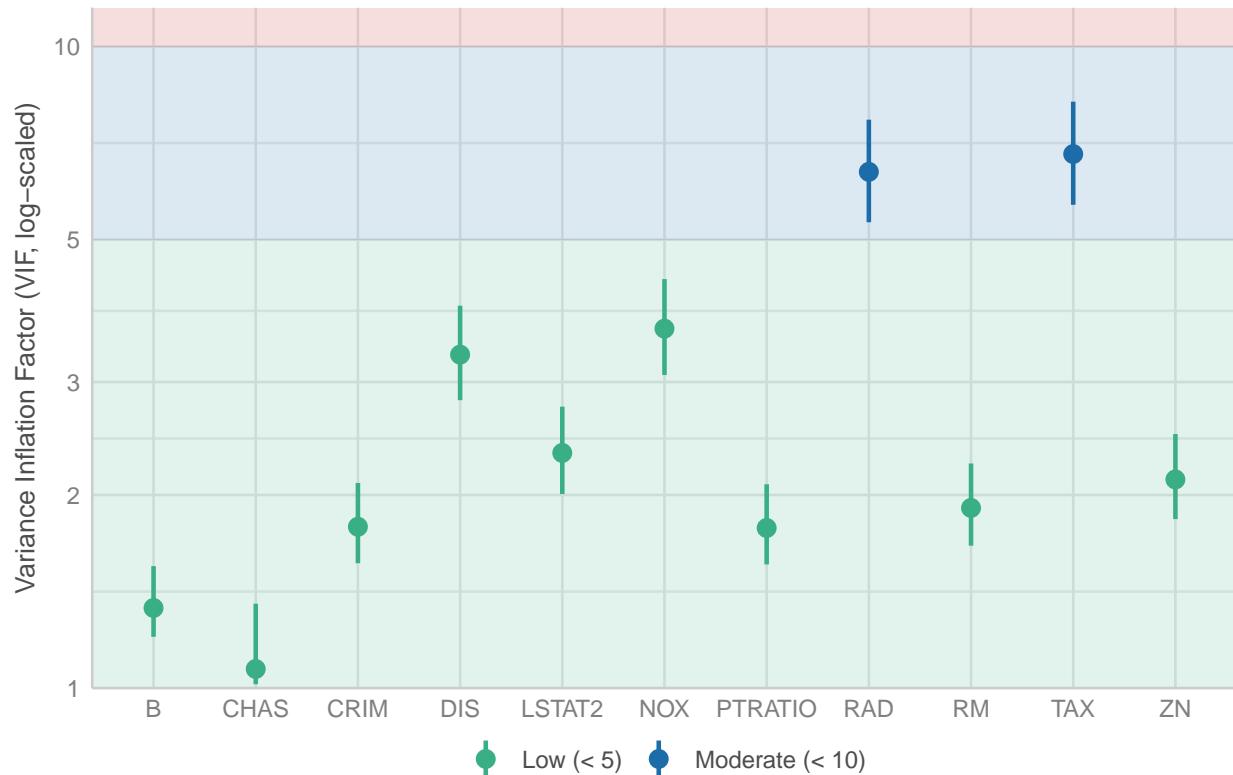
```
library(performance)

vif_results <- mlr |>
  check_collinearity()

vif_results |>
  plot()
```

Collinearity

High collinearity (VIF) may inflate parameter uncertainty



En el gráfico se observa que la mayoría de los predictores presentan valores VIF inferiores a 5, lo cual indica niveles aceptables de colinealidad.

No obstante, las variables RAD y TAX presentan VIF superiores a 6, lo que sugiere una dependencia significativa entre ellas. Esta sospecha se confirma mediante el cálculo del coeficiente de correlación entre ambas variables:

```
cor(data$RAD, data$TAX)
```

```
## [1] 0.9102282
```

Este alto grado de correlación se interpreta considerando que las zonas con mayor accesibilidad a autopistas (RAD) tienden a estar más urbanizadas y, en consecuencia, presentan mayores tasas impositivas (TAX) para financiar infraestructura y servicios urbanos. Por tanto, ambas variables podrían estar actuando como indicadores del nivel de desarrollo y conectividad urbana.

Ante esta situación, se consideran dos alternativas para abordar la multicolinealidad observada entre las variables RAD y TAX.

Por un lado, se plantea **eliminar la variable TAX**, al entender que su efecto ya se encuentra representado de forma adecuada por RAD, el cual actúa como un predictor más directo de accesibilidad vial. Esta opción permite reducir la multicolinealidad y mejorar la interpretabilidad del modelo sin una pérdida significativa de información.

Por otro lado, se contempla la posibilidad de **combinar ambas variables en un único predictor**, con el objetivo de preservar la información que cada una aporta, y capturar un posible efecto conjunto. Para ello, se propone una nueva variable denominada RAD_TAX_SCORE, definida como la razón entre RAD y TAX:

$$\text{RAD_TAX_SCORE} = \frac{\text{RAD}}{\text{TAX}}$$

Esta combinación no se realiza de manera arbitraria: se utiliza una **división** porque permite capturar una interpretación relevante en el contexto urbano. Concretamente, RAD_TAX_score representa la **proporción de accesibilidad vial disponible (RAD)** en relación con la carga fiscal (TAX).

En otras palabras, mide **cuánta infraestructura vial se obtiene por cada unidad de presión impositiva**. Esta proporción puede interpretarse como un **índicador de eficiencia urbana**, donde un valor más alto sugiere que el ciudadano dispone de más accesibilidad por cada unidad de impuesto que paga.

Entre ambas alternativas, se opta finalmente por la creación del nuevo predictor RAD_TAX_SCORE, al considerarse que aporta una interpretación más rica y específica del equilibrio entre infraestructura y fiscalidad. Posteriormente, se analizará su impacto dentro del modelo y se comparará su rendimiento frente a otras configuraciones.

```
data_train <- data_split |>
  training() |>
  mutate(LSTAT2 = LSTAT^2) |>
  mutate(RAD_TAX_SCORE = RAD / TAX) |>
  select(-INDUS, -AGE, -LSTAT, -RAD, -TAX) |>
  glimpse()

## # Rows: 354
## # Columns: 11
## $ CRIM      <dbl> 0.01501, 0.03961, 67.92080, 0.14866, 0.10574, 0.10793, 1~
## $ ZN        <dbl> 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ CHAS      <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX       <dbl> 0.401, 0.515, 0.693, 0.520, 0.609, 0.520, 0.700, 0.624, ~
## $ RM         <dbl> 7.923, 6.037, 5.683, 6.727, 5.983, 6.195, 5.036, 6.372, ~
## $ DIS        <dbl> 5.8850, 5.9853, 1.4254, 2.7778, 1.8681, 2.7778, 1.7700, ~
## $ PTRATIO   <dbl> 13.6, 20.2, 20.2, 20.9, 20.1, 20.9, 20.2, 21.2, 20.2, 20~
## $ B          <dbl> 395.52, 396.90, 384.97, 394.76, 390.11, 393.49, 396.90, ~
## $ MEDV      <dbl> 50.0, 21.1, 5.0, 27.5, 13.6, 21.7, 9.7, 23.0, 14.2, 6.3, ~
## $ LSTAT2    <dbl> 9.9856, 64.1601, 528.0804, 88.7364, 326.5249, 169.0000, ~
## $ RAD_TAX_SCORE <dbl> 0.005050505, 0.022321429, 0.036036036, 0.013020833, 0.00~

data_test <- data_split |>
  testing() |>
  mutate(LSTAT2 = LSTAT^2) |>
  mutate(RAD_TAX_SCORE = RAD / TAX) |>
  select(-INDUS, -AGE, -LSTAT, -RAD, -TAX) |>
  glimpse()

## # Rows: 152
## # Columns: 11
## $ CRIM      <dbl> 0.00632, 0.02729, 0.06905, 0.08829, 0.14455, 0.21124, 0.~
## $ ZN        <dbl> 18.0, 0.0, 0.0, 12.5, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0~
## $ CHAS      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX       <dbl> 0.538, 0.469, 0.458, 0.524, 0.524, 0.524, 0.524, 0.524, 0.538, ~
## $ RM         <dbl> 6.575, 7.185, 7.147, 6.012, 6.172, 5.631, 6.377, 5.990, ~
## $ DIS        <dbl> 4.0900, 4.9671, 6.0622, 5.5605, 5.9505, 6.0821, 6.3467, ~
## $ PTRATIO   <dbl> 15.3, 17.8, 18.7, 15.2, 15.2, 15.2, 21.0, 21.0, 21.0, 21~
## $ B          <dbl> 396.90, 392.83, 396.90, 395.60, 396.90, 386.63, 392.52, ~
## $ MEDV      <dbl> 24.0, 34.7, 36.2, 22.9, 27.1, 16.5, 15.0, 17.5, 13.6, 15~
## $ LSTAT2    <dbl> 24.8004, 16.2409, 28.4089, 154.5049, 366.7225, 895.8049, ~
## $ RAD_TAX_SCORE <dbl> 0.003378378, 0.008264463, 0.013513514, 0.016077170, 0.01~

mlr <- lm(MEDV ~ ., data = data_train)
```

```

summary(mlr)

##
## Call:
## lm(formula = MEDV ~ ., data = data_train)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -16.618 -2.717 -0.537  1.977 27.831 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 15.913738   6.182565   2.574 0.010473 *  
## CRIM        -0.133520   0.036250  -3.683 0.000267 *** 
## ZN          0.043216   0.016056   2.692 0.007460 **  
## CHAS        2.942430   1.007607   2.920 0.003729 **  
## NOX         -25.008147  3.891126  -6.427 4.36e-10 *** 
## RM          6.052067   0.517835  11.687 < 2e-16 *** 
## DIS         -1.210419   0.224419  -5.394 1.29e-07 *** 
## PTRATIO     -1.004759   0.149960  -6.700 8.55e-11 *** 
## B           0.012535   0.003286   3.814 0.000162 *** 
## LSTAT2      -0.004570   0.001556  -2.937 0.003540 **  
## RAD_TAX_SCORE 89.624598  33.770805   2.654 0.008327 ** 
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 4.871 on 343 degrees of freedom
## Multiple R-squared:  0.7249, Adjusted R-squared:  0.7169 
## F-statistic: 90.37 on 10 and 343 DF,  p-value: < 2.2e-16

```

Como el valor-p es inferior al nivel de significación $\alpha = 0.05$, se obtiene un resultado significativo. Por tanto, se rechaza la hipótesis nula, y se concluye que al menos una de las variables predictoras influye sobre la variable respuesta MEDV. Ademas, todas las variables tienen un p-valor inferior a 0.05, por lo que todas son significativas.

A continuación, se vuelven a interpretar los coeficientes del modelo ajustado. Cada valor representa el efecto marginal de una variable predictora sobre el valor medio de la variable respuesta MEDV, manteniendo constante el resto de las variables. Para cada estimación se incluye su intervalo de confianza al 95%, el cual indica el rango dentro del cual se espera que se encuentre el verdadero valor del coeficiente con un 95% de certeza.

```

mlr |>
  coef()

##   (Intercept)       CRIM        ZN        CHAS        NOX      
## 15.913738016 -0.133520239  0.043216130  2.942429804 -25.008147221 
##            RM        DIS        PTRATIO       B        LSTAT2      
## 6.052066894 -1.210419306 -1.004759440  0.012534716 -0.004569551 
## RAD_TAX_SCORE      
## 89.624597944

```

```

mlr |>
  confint()

##                  2.5 %      97.5 %
## (Intercept) 3.753225164 28.074250869
## CRIM        -0.204819683 -0.062220795

```

```

## ZN          0.011635433  0.074796827
## CHAS        0.960564203  4.924295405
## NOX       -32.661619325 -17.354675116
## RM           5.033534702  7.070599085
## DIS          -1.651830766 -0.769007846
## PTRATIO     -1.299716044 -0.709802835
## B            0.006071257  0.018998176
## LSTAT2      -0.007630000 -0.001509103
## RAD_TAX_SCORE 23.200656629 156.048539259

```

- **(Intercept)**: Cuando todas las variables predictoras toman el valor cero, el valor medio de la vivienda es de **15.91 unidades**, con un intervalo de confianza entre **3.75 y 28.07**. Aunque su interpretación no es práctica, sirve como referencia base del modelo.
- **CRIM**: Un incremento de una unidad en el índice de criminalidad disminuye el valor medio de la vivienda en **0.13 unidades**. El intervalo de confianza va de **-0.20 a -0.06**, lo que confirma un efecto negativo estadísticamente significativo.
- **ZN**: Por cada unidad adicional en la proporción de suelo residencial de baja densidad, el valor medio de la vivienda aumenta en **0.043 unidades**, con un intervalo de confianza entre **0.012 y 0.075**, indicando que este efecto positivo también es significativo.
- **CHAS**: Las viviendas situadas junto al río Charles tienen, en promedio, un valor **2.94 unidades superior**, con un intervalo entre **0.96 y 4.92**. Esto sugiere que dicha ubicación es valorada positivamente por el mercado.
- **NOX**: Un aumento de una unidad en la concentración de óxidos nítricos se asocia con una disminución de **25.01 unidades** en el valor medio de la vivienda. El intervalo de confianza, entre **-32.66 y -17.35**, confirma que este impacto negativo es fuerte y significativo.
- **RM**: Cada habitación adicional en una vivienda incrementa el valor medio en **6.05 unidades**, con un intervalo de **5.03 a 7.07**, lo que demuestra que se trata de uno de los factores con mayor peso positivo en el precio.
- **DIS**: Aumentar en una unidad la distancia a los centros de empleo reduce el valor de la vivienda en **1.21 unidades**, con un intervalo entre **-1.65 y -0.77**. El efecto negativo es claro y significativo.
- **PTRATIO**: Por cada unidad adicional en la ratio alumno/profesor, el valor medio de la vivienda disminuye en **1.00 unidad**, con un intervalo entre **-1.30 y -0.71**, lo que indica que la calidad del sistema educativo influye en el precio de las viviendas.
- **B**: Esta variable representa una transformación del porcentaje de población afroamericana. Su coeficiente es de **0.0125**, con un intervalo de confianza entre **0.0061 y 0.0190**, mostrando un efecto positivo moderado pero significativo.
- **LSTAT2**: Esta variable corresponde a una transformación (probablemente al cuadrado) de la proporción de población de bajo nivel socioeconómico. Un aumento en esta variable se asocia con una disminución de **0.0046 unidades** en el valor medio, con un intervalo entre **-0.0076 y -0.0015**, confirmando su efecto negativo y significativo.
- **RAD_TAX_SCORE**: Esta variable representa la proporción de accesibilidad vial (RAD) por unidad de carga impositiva (TAX). Un aumento en esta razón incrementa el valor medio de la vivienda en **89.62 unidades**, con un intervalo de confianza entre **23.20 y 156.05**. Este resultado muestra un fuerte efecto positivo, lo que sugiere que zonas con buena infraestructura vial en relación con los impuestos están significativamente mejor valoradas.

Después de aplicar e interpretar el modelo, es volvemos a verificar su validez, es decir, comprobar si se cumplen las hipótesis iniciales del modelo. Estas hipótesis están relacionadas con los residuos del modelo. Para ello, se recopilan los residuos del modelo ajustado en un objeto tipo tibble utilizando la función `residuals()`, tal como se hizo anteriormente.

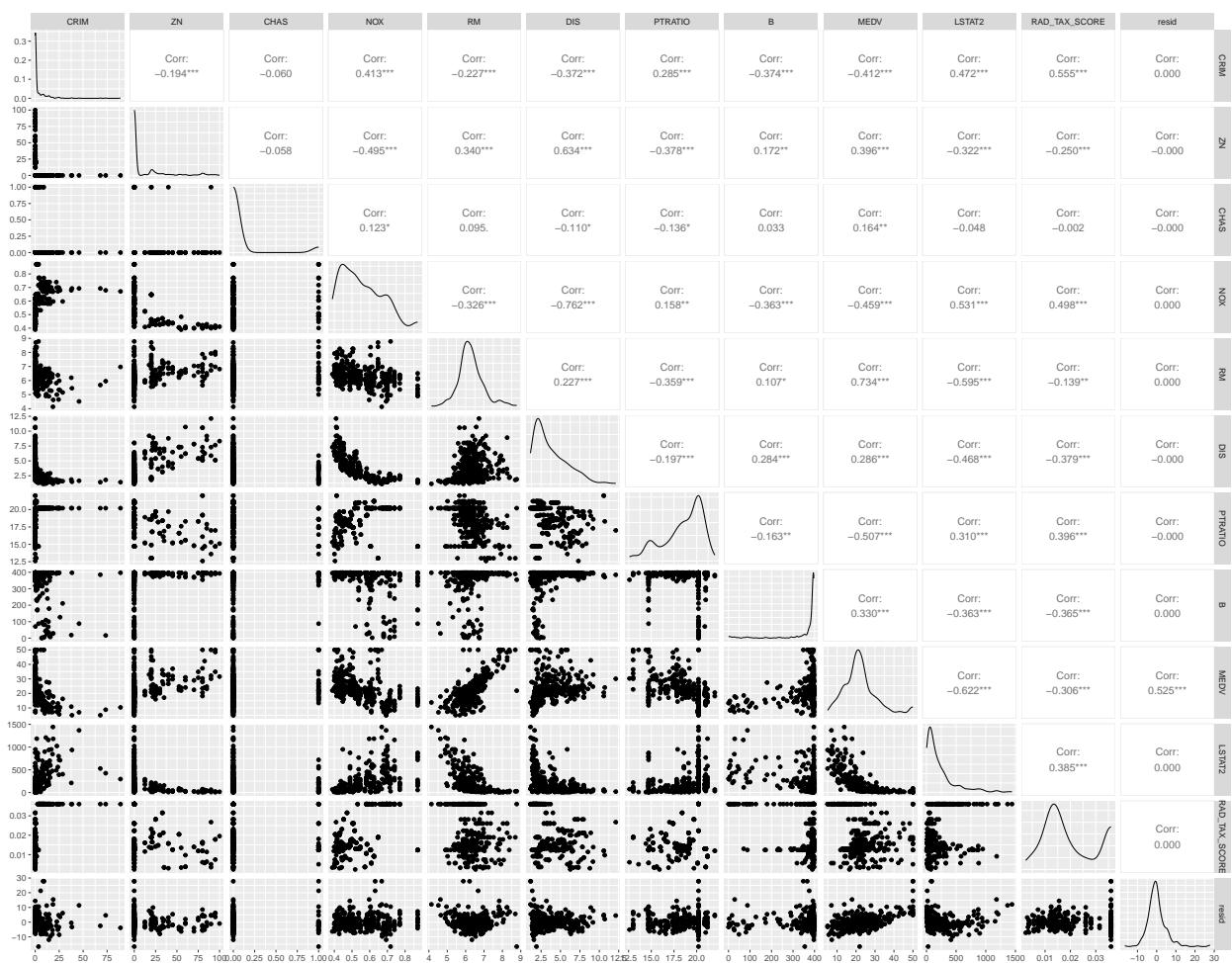
```

resid <- tibble(resid = residuals(mlr)) |>
  glimpse()

## # Rows: 354
## # Columns: 1
## $ resid <dbl> 4.7574209, 2.3928870, -2.5288288, 2.5501627, -4.7243881, 0.34711~

data_train |>
  mutate(
    resid = residuals(mlr)
  ) |>
  ggpairs()

```



Como se observa, los residuos no presentan ningún patrón, lo que indica que la relación entre la variable respuesta y las variables predictoras es lineal.

El siguiente paso es comprobar la homocedasticidad de los residuos.

```

mlr |>
  bptest(
    studentize = FALSE
  )

## 
## Breusch-Pagan test

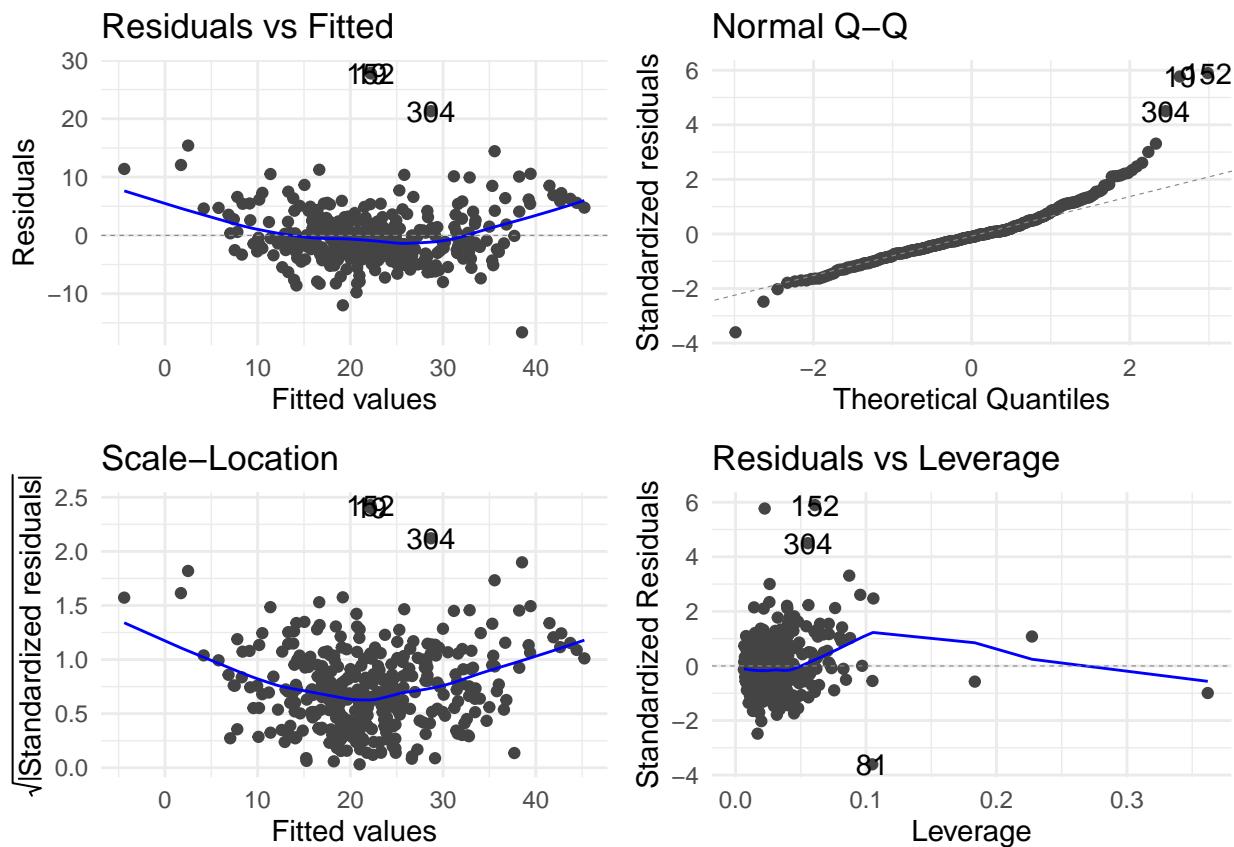
```

```
##  
## data: mlr  
## BP = 169.51, df = 10, p-value < 2.2e-16
```

Dado que el *p*-valor obtenido es **inferior a 0.05**, el resultado es significativo y se **rechaza la hipótesis nula**. Esto vuelve a evidenciar la **heterocedasticidad en el modelo**.

Para verificar la validez del modelo ajustado, se analizan los gráficos de diagnóstico generados a partir de los residuos

```
mlr |>  
  autoplot() +  
  theme_minimal()
```



Los **gráficos de diagnóstico** muestran que la dispersión de los residuos se mantiene relativamente constante en la mayor parte del rango de valores ajustados. Aunque pueden identificarse algunos puntos atípicos, no se observa un patrón claro de incremento sistemático en la varianza.

Por tanto, se concluye que, si bien el test formal detecta cierta heterocedasticidad, esta **no parece generalizada ni lo suficientemente severa** como para comprometer la validez del modelo. Se decide **no aplicar transformaciones adicionales** a la variable respuesta, considerando que el comportamiento de los residuos es, en líneas generales, aceptable para los fines del análisis.

Continuando con la comprobación de la independencia de los residuos, se aplica el **Test de Durbin-Watson**:

```
mlr |>  
  dwtest(alternative = "two.sided")
```

```
##  
## Durbin-Watson test
```

```

## 
## data: mlr
## DW = 2.1067, p-value = 0.3151
## alternative hypothesis: true autocorrelation is not 0

```

Al obtener un p-valor superior a 0.05, no se rechaza la hipótesis nula, lo que indica que los residuos son independientes.

A continuación, se verifica la normalidad de los residuos mediante el **Test de Kolmogorov-Smirnov-Lilliefors**:

```

resid |>
  pull() |>
  lillie.test()

```

```

## 
## Lilliefors (Kolmogorov-Smirnov) normality test
## 
## data: pull(resid)
## D = 0.11566, p-value = 2.383e-12

```

El resultado del test es significativo (p-valor < 0.05), lo que nos permite rechazar la hipótesis nula de normalidad. Sin embargo, el gráfico Normal Q-Q muestra que hay puntos en la parte inferior que se alejan de la línea diagonal, lo que podría sugerir la presencia de valores influyentes.

El siguiente paso es analizar la multicolinealidad entre las variables predictoras.

```

mlr |>
  vif()

```

	CRIM	ZN	CHAS	NOX	RM
##	1.704145	2.032212	1.067255	3.166348	1.894579
##	DIS	PTRATIO	B	LSTAT2	RAD_TAX_SCORE
##	3.271938	1.563209	1.311217	2.324933	1.947402

```

vif_results <- mlr |>
  check_collinearity()

```

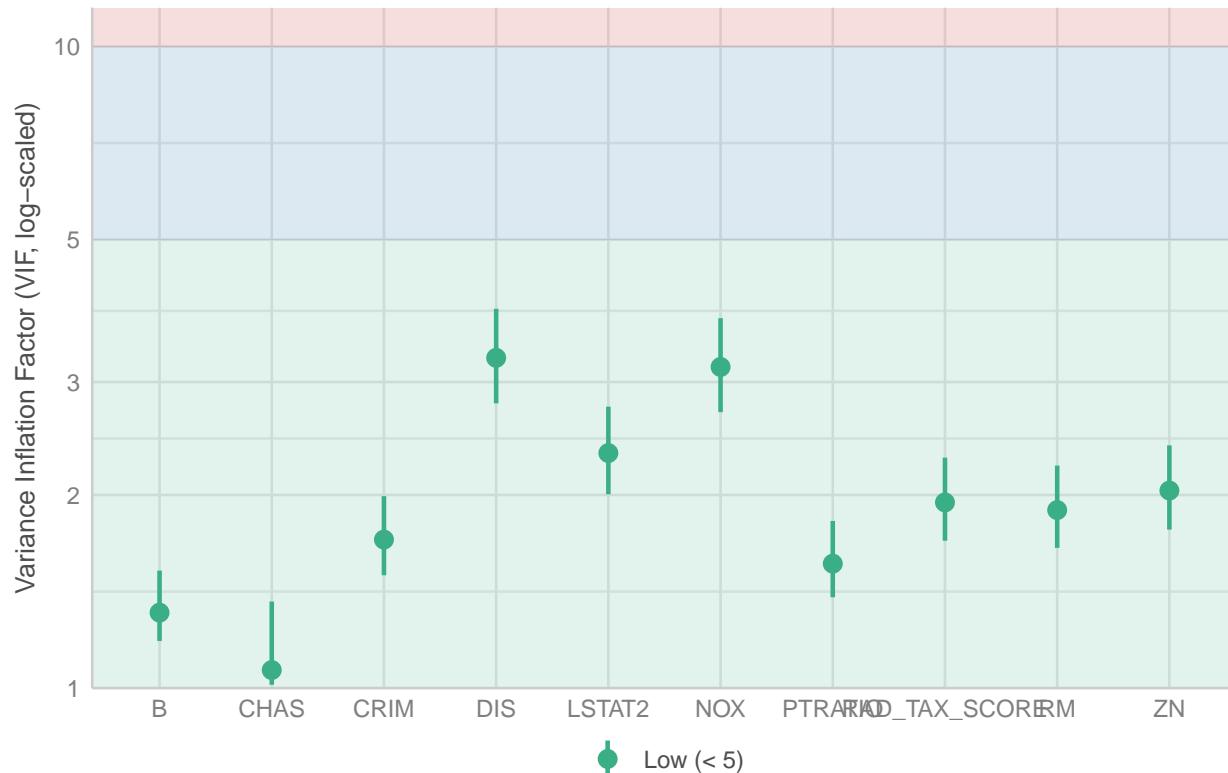
```

vif_results |>
  plot()

```

Collinearity

High collinearity (VIF) may inflate parameter uncertainty



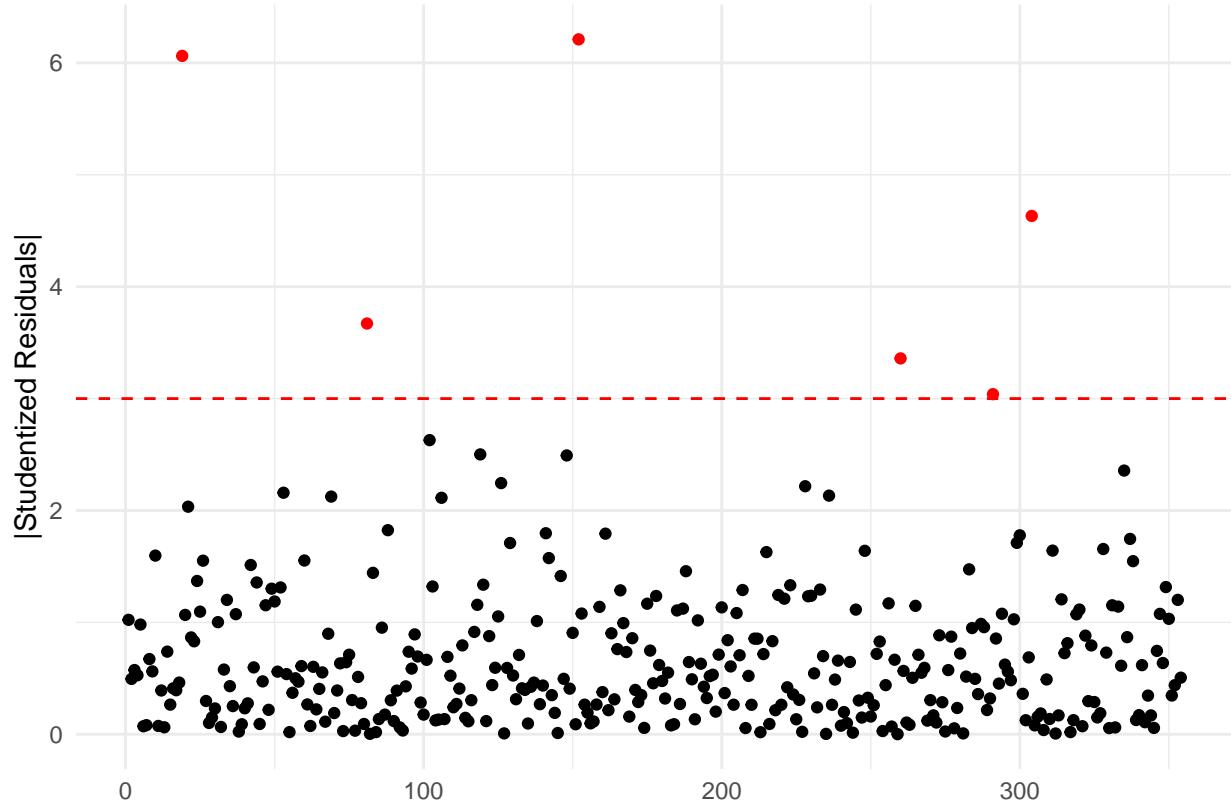
Ahora, todos los predictores presentan valores VIF inferiores a 5, lo que indica niveles aceptables de colinealidad.

3.1.12 Estudio de valores atípicos

Un aspecto relevante en el análisis mediante modelos de regresión es la detección de valores atípicos en la variable respuesta. Se consideran atípicos aquellos datos cuyos residuos estandarizados se encuentran fuera del intervalo $[-3, 3]$.

Para identificarlos, se pueden calcular estos residuos utilizando la función `rstudent(model)`.

```
tibble(
  resid_stud = rstudent(mlr)
) |>
  ggplot() +
  aes(x = seq_along(resid_stud),
      y = abs(resid_stud)) +
  geom_point(aes(color = ifelse(abs(resid_stud) > 3,
                                "red",
                                "black"))) +
  geom_hline(yintercept = 3,
             color = "red",
             linetype = "dashed") +
  scale_color_identity() +
  labs(x = "",
       y = "|Studentized Residuals|") +
  theme_minimal()
```



Se identifican valores atípicos en la variable respuesta, por lo que resulta necesario proceder a su detección y eliminación.

```
position_outliers <- tibble(
  resid_stud = rstudent(mlr)
) |>
  mutate(
    id_row = row_number()
) |>
  filter(abs(resid_stud) > 3) |>
  pull(id_row)

print(position_outliers)

## [1] 19 81 152 260 291 304
```

Una vez identificados, dichos valores atípicos deben ser eliminados del conjunto de entrenamiento, a fin de reajustar el modelo posteriormente.

3.1.13 Estudio de Leverages

Otro aspecto relevante es la presencia de valores atípicos en las variables predictoras, conocidos como *leverages*. Para identificarlos, se emplea la función `influence.measures(model)`, la cual genera una tabla con las observaciones que resultan significativamente influyentes en alguna de las variables del modelo.

Se considera que una observación es un *leverage* si cumple con la siguiente condición:

$$\hat{h}_i > 2.5 \cdot \frac{p+1}{n}$$

donde:

- \hat{h}_i es el valor de leverage para la observación i ,
- p es el número de variables predictoras,
- n es el número total de observaciones.

```
infl_leverages <- mlr |>
  influence.measures() |>
  summary()

## Potentially influential observations of
##   lm(formula = MEDV ~ ., data = data_train) :
##
##   dfb.1_ dfb.CRIM dfb.ZN dfb.CHAS dfb.NOX dfb.RM dfb.DIS dfb.PTRA dfb.B
## 3    0.01 -0.26    0.01  0.00   -0.01   0.01  -0.01   0.01  -0.07
## 13   0.00  0.00    0.00  0.00     0.00   0.00   0.00   0.00   0.01
## 19   0.19 -0.02    0.22 -0.12   -0.17  -0.27  -0.50   0.09  0.11
## 28  -0.01  0.00    0.00 -0.02     0.01   0.00   0.00   0.00   0.00
## 53   0.17 -0.02   -0.02 -0.04   -0.16  -0.10  -0.17  -0.07  0.00
## 55   0.00  0.00    0.00  0.00     0.00   0.00   0.00   0.00   0.00
## 62   0.00  0.00   -0.01  0.00     0.00   0.00   0.01   0.00   0.00
## 66  -0.01 -0.02    0.00 -0.02     0.08  -0.01   0.03  -0.05  0.04
## 81   0.85  0.09    0.07 -0.61   -0.36  -0.87  -0.20  -0.44 -0.15
## 82   0.00  0.00    0.00  0.00     0.00   0.00   0.00   0.00   0.00
## 102   0.09 -0.05    0.06  0.02   -0.23  -0.08  -0.08  -0.13  0.29
## 116  -0.02  0.00    0.00 -0.02     0.06   0.01   0.02  -0.01  0.01
## 119   0.15 -0.05   -0.02  0.06   -0.07  -0.10   0.02  -0.06 -0.18
## 126  -0.08  0.08   -0.11  0.36     0.02   0.22  -0.01  -0.09  0.03
## 147   0.02 -0.01    0.00 -0.02     0.08  -0.03   0.02  -0.04 -0.06
## 148   0.16  0.43   -0.03  0.06   -0.17  -0.05   0.07  -0.12 -0.19
## 152   0.32  0.01    0.35  1.19_*  -0.18  -0.60  -0.46   0.22 -0.04
## 153   0.08  0.52   -0.03  0.03   -0.02  -0.04   0.03  -0.02 -0.16
## 167   0.10 -0.72    0.06 -0.01   -0.03  -0.11  -0.06  -0.01 -0.19
## 183   0.00  0.00    0.00  0.00   -0.01  -0.01  -0.01   0.01  0.01
## 225   0.01  0.00    0.00  0.03   -0.01  -0.01   0.00   0.00   0.00
## 236  -0.44  0.02   -0.24 -0.04     0.20   0.47   0.44   0.11  0.08
## 260   0.46 -0.11    0.05  0.06   -0.50  -0.22  -0.20  -0.17 -0.56
## 267  -0.01  0.00    0.00 -0.09   -0.08   0.02  -0.04   0.02  0.10
## 272  -0.01 -0.02    0.00  0.00     0.00   0.00   0.00   0.00   0.02
## 291  -0.19  0.00   -0.17 -0.09   -0.07   0.39  -0.04   0.02  0.06
## 304   0.00 -0.05    0.17  0.86   -0.15  -0.09  -0.30   0.24  0.06
## 318  -0.01 -0.01    0.00  0.00     0.00   0.01   0.01   0.00 -0.01
## 335   0.16  0.03    0.10 -0.04   -0.05  -0.23  -0.18   0.01 -0.01
## 354   0.02  0.00    0.00 -0.08   -0.07   0.00  -0.04   0.03 -0.03

##   dfb.LSTA dfb.RAD_dffit cov.r cook.d hat
## 3    0.04    0.06  -0.27  1.25_*  0.01  0.18_*
## 13   0.00    0.00  -0.02  1.10_*  0.00  0.06
## 19   -0.51   0.45  0.92_*  0.34_*  0.07  0.02
## 28  -0.01    0.00  -0.03  1.10_*  0.00  0.06
## 53  -0.08    0.03   0.26  0.90_*  0.01  0.01
## 55   0.00    0.00   0.00  1.10_*  0.00  0.06
## 62   0.02    0.00   0.02  1.11_*  0.00  0.07
```

```

## 66 0.06 -0.04 0.15 1.10_* 0.00 0.07
## 81 -0.20 -0.04 -1.26_* 0.75_* 0.14 0.10_*
## 82 0.00 0.00 0.00 1.14_* 0.00 0.10_*
## 102 0.61 0.16 0.85_* 0.92 0.07 0.10_*
## 116 -0.02 -0.03 0.08 1.10_* 0.00 0.06
## 119 -0.06 -0.09 -0.33 0.86_* 0.01 0.02
## 126 0.05 -0.09 0.55_* 0.93 0.03 0.06
## 147 -0.04 -0.04 0.13 1.10_* 0.00 0.07
## 148 0.37 -0.12 0.86_* 0.95 0.07 0.11_*
## 152 -0.67 0.44 1.58_* 0.34_* 0.20 0.06
## 153 -0.16 -0.16 0.58_* 1.29_* 0.03 0.23_*
## 167 0.11 0.20 -0.75_* 1.57_* 0.05 0.36_*
## 183 -0.01 0.01 -0.02 1.11_* 0.00 0.07
## 225 0.01 0.00 0.03 1.10_* 0.00 0.06
## 236 0.23 -0.02 0.61_* 0.97 0.03 0.08
## 260 0.44 0.16 1.04_* 0.79_* 0.10 0.09
## 267 0.04 0.06 -0.19 1.14_* 0.00 0.10_*
## 272 0.01 0.00 -0.03 1.13_* 0.00 0.08
## 291 0.12 -0.06 0.50 0.79_* 0.02 0.03
## 304 -0.33 0.34 1.12_* 0.56_* 0.11 0.06
## 318 0.03 0.00 0.04 1.11_* 0.00 0.07
## 335 -0.26 0.16 0.38 0.89_* 0.01 0.03
## 354 -0.05 0.04 -0.15 1.12_* 0.00 0.08

p <- 10; n <- nrow(data_train)
leverages <- tibble(
  id_row = as.integer(rownames(infl_leverages)),
  hat = infl_leverages[, "hat"]
) |>
  filter(hat > 2.5*(p+1)/n) |>
  print(n = 10)

## # A tibble: 11 x 2
##   id_row     hat
##   <int>   <dbl>
## 1 1      0.183
## 2 2      0.105
## 3 3      0.0970
## 4 4      0.0956
## 5 5      0.106
## 6 6      0.227
## 7 7      0.362
## 8 8      0.0872
## 9 9      0.105
## 10 10    0.0827
## # i 1 more row

position_leverages <- leverages |>
  pull(id_row)

print(position_leverages)

## [1] 3 81 82 102 148 153 167 260 267 272 354

```

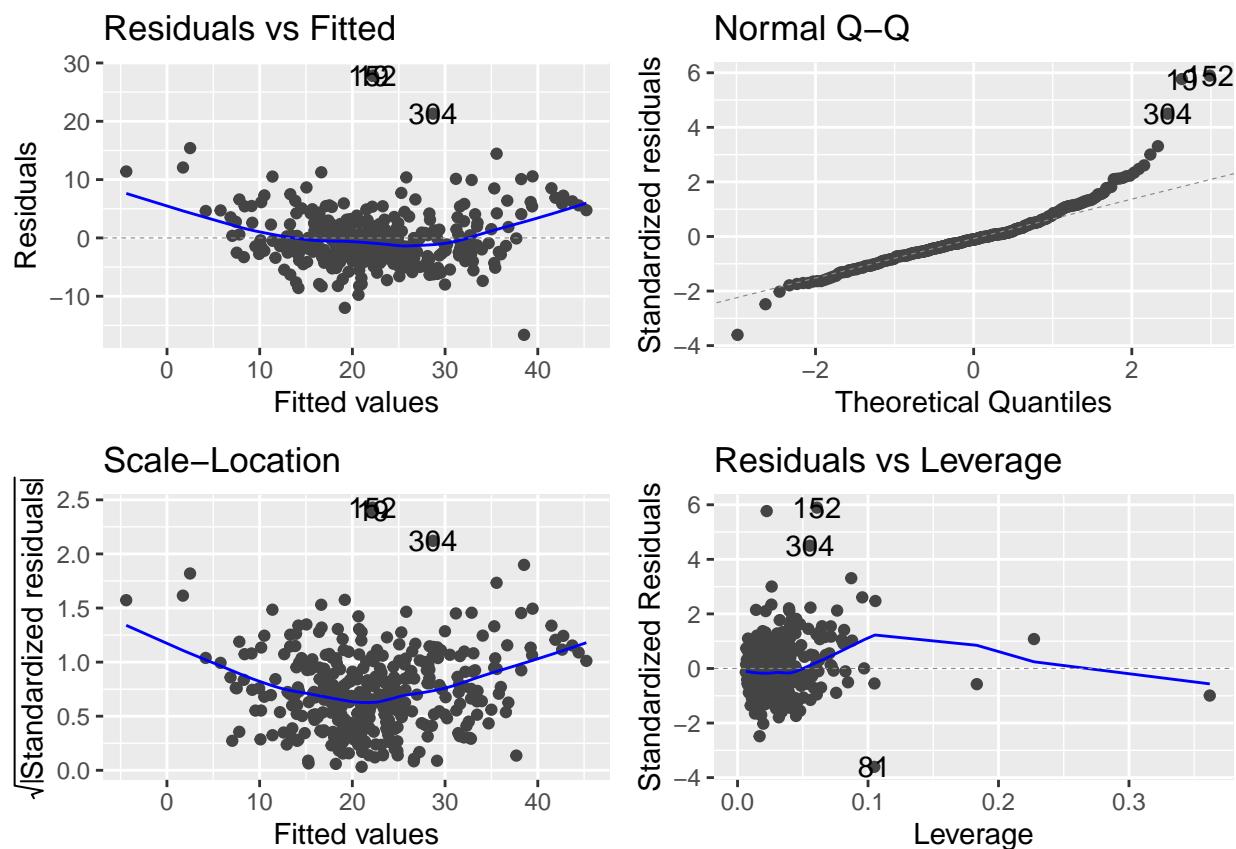
Por lo tanto, las observaciones con un valor elevado de `hat` deben ser eliminadas, siempre y cuando se verifique que superan el umbral establecido.

```
hat_limit <- 2.5*(p+1)/n
hat_limit
```

```
## [1] 0.07768362
```

Al revisar nuevamente el gráfico de diagnóstico **Residuals vs Leverages**, se observa que algunas observaciones presentan valores de *leverage* superiores a 0.0282. Estas observaciones están teniendo una influencia considerable en el modelo y podrían estar contribuyendo a la falta de normalidad y a la heterocedasticidad señaladas por los test de hipótesis.

```
mlr |>
  autoplot()
```



Por eso, es necesario eliminar los leverages del conjunto de datos.

3.1.14 Reajuste del modelo

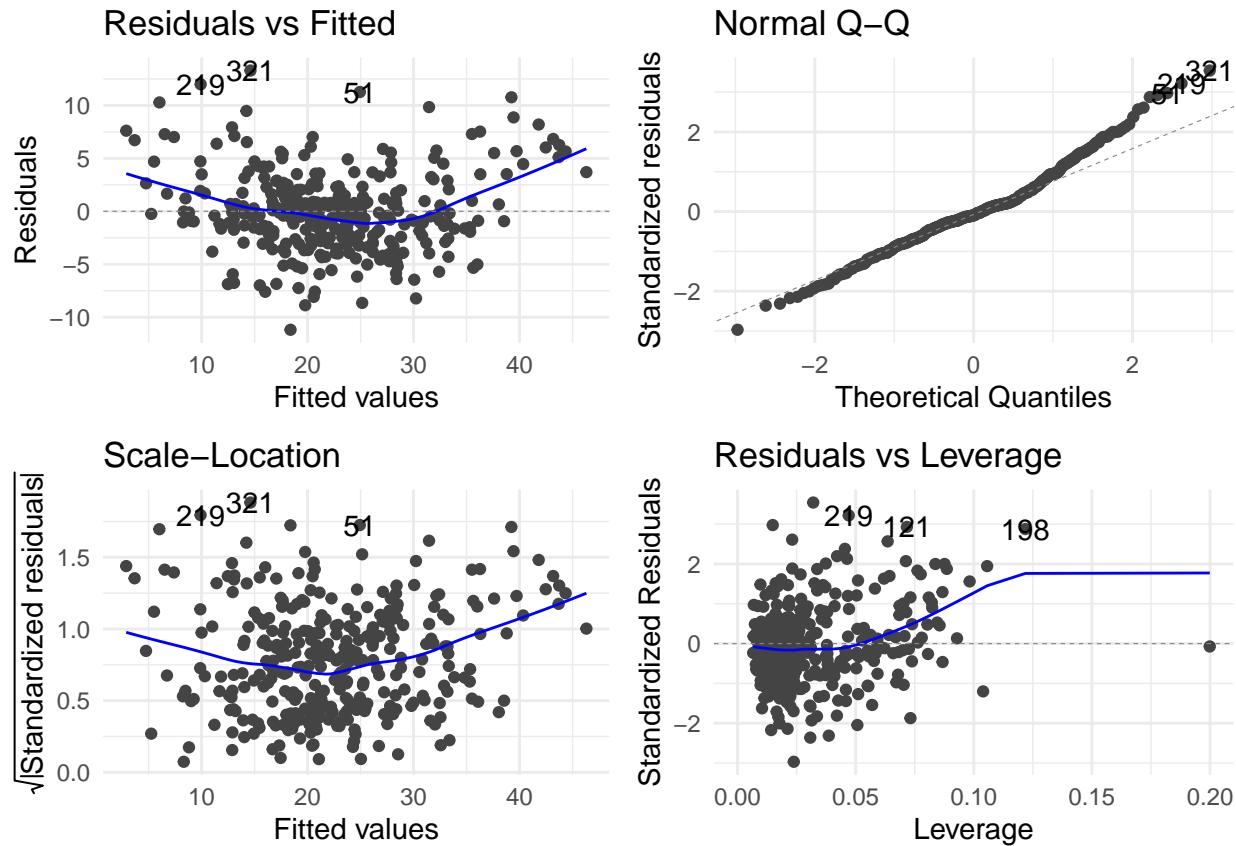
```
position <- union(position_outliers, position_leverages)

data_train <- data_train |>
  slice(-position)

mlr <- data_train |>
  lm(MEDV ~ ., data = _)

mlr |>
  autoplot() +
```

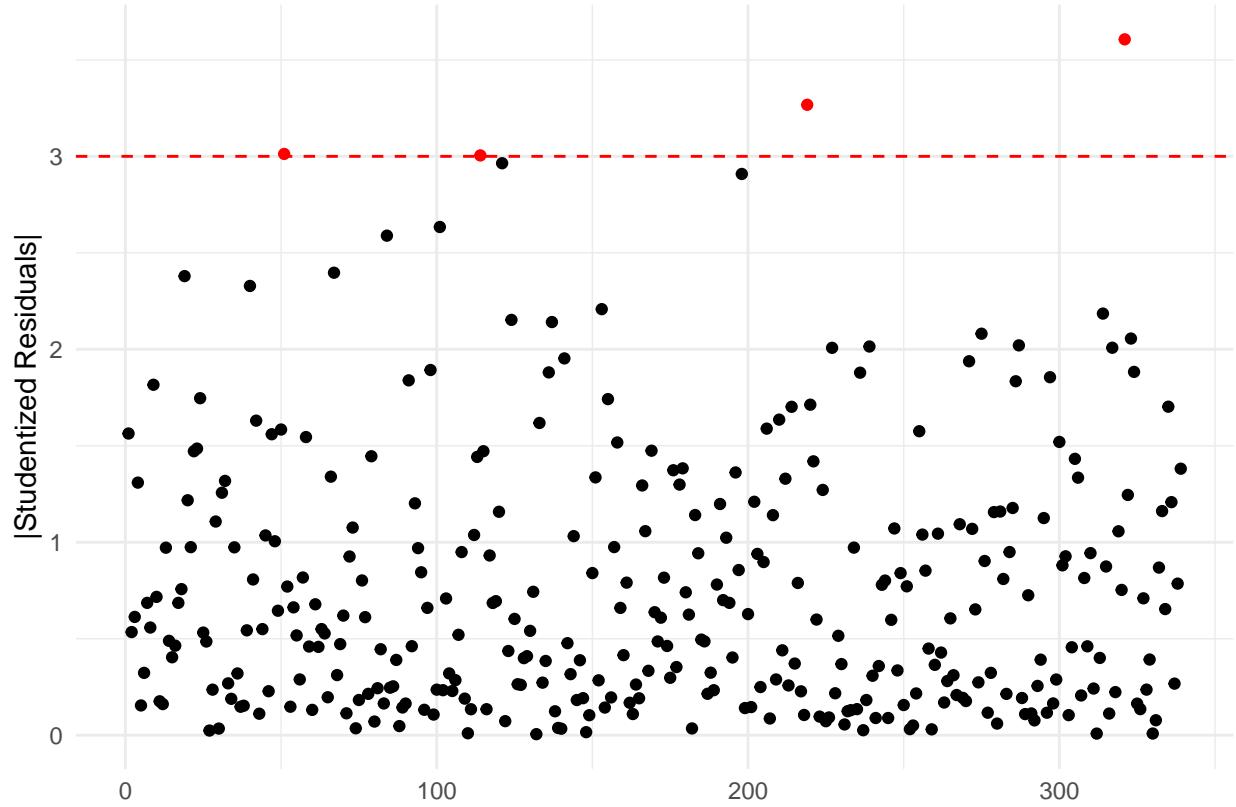
```
theme_minimal()
```



Al observar los gráficos, no se aprecia un patrón claro en la variabilidad de los residuos, lo que sugiere la posible ausencia de homocedasticidad. Además, se puede asumir que los residuos no siguen una distribución normal.

También se identifican posibles valores atípicos y observaciones con *leverage* elevado.

```
tibble(
  resid_stud = rstudent(mlr)
) |>
  ggplot() +
  aes(x = seq_along(resid_stud),
      y = abs(resid_stud)) +
  geom_point(aes(color = ifelse(abs(resid_stud) > 3,
                                "red",
                                "black"))) +
  geom_hline(yintercept = 3,
             color = "red",
             linetype = "dashed") +
  scale_color_identity() +
  labs(x = "",
       y = "|Studentized Residuals|") +
  theme_minimal()
```



```

infl_leverages <- mlr |>
  influence.measures() |>
  summary()

## Potentially influential observations of
##   lm(formula = MEDV ~ ., data = data_train) :
##
##      dfb.1_ dfb.CRIM dfb.ZN dfb.CHAS dfb.NOX dfb.RM dfb.DIS dfb.PTRA dfb.B
## 1    -0.03   0.09    0.24   0.34   -0.02   0.10   -0.07   0.00   -0.01
## 12   0.01  -0.01    0.00   0.00   -0.01   0.00   -0.01   0.01   -0.04
## 19   0.17  -0.24    0.02   0.04   -0.02  -0.18   0.01  -0.06  -0.21
## 26   0.03  -0.02    0.00   0.11   -0.04  -0.03  -0.01  -0.01   0.00
## 34   0.02  -0.02    0.00   0.00   -0.02   0.00   -0.01   0.00   -0.04
## 40   0.07  -0.18    0.28   0.04   -0.18  -0.10  -0.32   0.13  -0.01
## 51   0.24  -0.04   -0.02  -0.06   -0.23  -0.14  -0.24  -0.10   0.00
## 53  -0.02  -0.01    0.00   0.03    0.02   0.00   0.01   0.01   0.01
## 60   0.00  -0.01   -0.01   0.00   -0.01   0.01   0.01  -0.01   0.01
## 64   0.00  -0.03    0.00  -0.02    0.08  -0.02   0.03  -0.06   0.03
## 67  -0.24  -0.07   -0.20  -0.10   -0.03   0.43   0.05  -0.01   0.09
## 82  -0.07  -0.02   -0.01   0.00    0.04   0.05   0.03   0.01   0.10
## 84   0.10  -0.28   -0.07  -0.05   -0.27   0.03  -0.05  -0.11   0.01
## 93   0.04  -0.02    0.00  -0.26   -0.26   0.03  -0.11   0.05   0.04
## 101  0.09   0.08   -0.10  -0.09    0.03   0.05  -0.08  -0.22  -0.01
## 111  0.01   0.00    0.00   0.00   -0.03   0.00  -0.01   0.01   0.00
## 114  0.18  -0.23    0.00   0.04   -0.07  -0.15   0.00  -0.07  -0.25
## 121 -0.14   0.20   -0.16   0.58    0.11   0.31   0.03  -0.11   0.05

```

```

## 127 -0.01 -0.07 0.00 0.00 0.00 0.02 0.00 0.00 -0.02
## 141 -0.11 -0.24 0.09 -0.03 -0.07 0.10 -0.08 0.13 0.13
## 142 0.03 -0.01 0.00 -0.01 0.08 -0.04 0.03 -0.04 -0.07
## 162 0.01 0.00 0.00 -0.04 -0.02 0.00 -0.01 -0.01 -0.01
## 174 0.02 0.03 0.01 0.01 -0.07 -0.03 -0.04 0.04 0.04
## 198 0.45 0.90 -0.01 0.07 -0.30 -0.37 -0.14 -0.14 -0.20
## 219 0.28 -0.27 0.22 -0.02 0.09 -0.52 -0.19 0.00 -0.11
## 225 0.00 -0.03 0.00 0.00 0.00 0.00 0.00 0.00 -0.01
## 227 -0.45 0.04 -0.25 -0.04 0.19 0.48 0.42 0.13 0.10
## 237 0.00 0.00 0.00 0.01 0.00 0.00 0.00 0.00 0.00
## 254 -0.01 0.00 0.00 -0.01 0.05 0.00 0.02 -0.01 0.01
## 275 0.10 0.39 0.02 0.02 -0.06 -0.16 -0.01 -0.08 0.20
## 277 0.00 0.00 0.00 0.00 -0.01 0.00 0.00 0.00 0.02
## 283 -0.02 -0.01 0.00 -0.05 0.00 0.02 0.01 0.02 0.01
## 304 -0.02 -0.03 0.00 0.00 0.00 0.04 0.02 0.00 -0.06
## 311 0.02 -0.01 0.00 -0.05 0.00 -0.03 0.00 -0.02 0.00
## 314 0.01 0.04 0.05 0.05 0.06 -0.08 -0.08 0.10 -0.03
## 317 0.01 -0.18 0.06 -0.04 -0.17 0.01 -0.09 -0.06 0.20
## 321 0.26 0.22 0.14 -0.01 -0.07 -0.36 -0.26 0.01 -0.02
## 324 -0.20 0.12 -0.13 0.36 0.10 0.33 0.06 -0.04 0.05

## dfb.LSTA dfb.RAD_dffit cov.r cook.d hat
## 1 0.04 -0.11 0.52 1.06 0.02 0.10_*
## 12 -0.01 -0.01 0.04 1.11_* 0.00 0.07
## 19 -0.04 0.02 -0.42 0.88_* 0.02 0.03
## 26 0.05 0.01 0.14 1.11_* 0.00 0.07
## 34 0.01 0.01 0.05 1.11_* 0.00 0.07
## 40 0.01 0.26 -0.47 0.90_* 0.02 0.04
## 51 -0.09 0.07 0.37 0.78_* 0.01 0.01
## 53 -0.01 0.01 0.04 1.12_* 0.00 0.08
## 60 0.04 0.00 0.04 1.14_* 0.00 0.09
## 64 0.07 -0.02 0.16 1.12_* 0.00 0.08
## 67 0.15 0.12 0.52 0.89_* 0.02 0.05
## 82 0.05 -0.01 -0.13 1.11_* 0.00 0.07
## 84 0.58 0.10 0.67_* 0.88_* 0.04 0.06
## 93 0.07 0.12 -0.41 1.10 0.02 0.10_*
## 101 -0.11 -0.09 0.41 0.84_* 0.01 0.02
## 111 0.01 0.01 -0.04 1.11_* 0.00 0.07
## 114 -0.01 -0.01 -0.47 0.79_* 0.02 0.02
## 121 -0.04 -0.21 0.82_* 0.83_* 0.06 0.07
## 127 0.03 0.02 -0.08 1.12_* 0.00 0.08
## 141 0.56 -0.07 0.67_* 1.02 0.04 0.11_*
## 142 -0.04 -0.04 0.14 1.12_* 0.00 0.08
## 162 0.01 0.00 -0.05 1.11_* 0.00 0.07
## 174 -0.06 0.04 -0.14 1.12_* 0.00 0.09
## 198 -0.47 -0.27 1.08_* 0.89_* 0.10 0.12_*
## 219 -0.34 0.34 0.73_* 0.76_* 0.05 0.05
## 225 0.00 0.01 -0.04 1.29_* 0.00 0.20_*
## 227 0.19 -0.04 0.61_* 0.99 0.03 0.08
## 237 0.00 0.00 0.01 1.12_* 0.00 0.08
## 254 -0.01 -0.02 0.06 1.10_* 0.00 0.06
## 275 0.00 -0.09 0.58_* 0.96 0.03 0.07
## 277 0.00 0.00 -0.03 1.10_* 0.00 0.06
## 283 0.02 0.01 -0.06 1.11_* 0.00 0.07
## 304 0.10 0.01 0.14 1.12_* 0.00 0.09

```

```

## 311 -0.01      0.01     -0.06    1.10_*   0.00     0.07
## 314 -0.11      -0.10    -0.26    0.89_*   0.01     0.01
## 317  0.50      0.20     0.62_*   0.99     0.03     0.09
## 321 -0.46      0.13     0.66_*   0.70_*   0.04     0.03
## 324  0.04      -0.15    0.59_*   1.01     0.03     0.09

```

Aún se detectan valores atípicos en la variable respuesta y observaciones con *leverage* elevado en las variables predictoras. Por lo tanto, sería necesario repetir el proceso hasta que estos casos sean eliminados o sus valores de *hat* se sitúen dentro del límite aceptable.

```

summary(mlr)

##
## Call:
## lm(formula = MEDV ~ ., data = data_train)
##
## Residuals:
##       Min        1Q     Median        3Q       Max
## -11.1883 -2.3614 -0.3493  1.8098 13.2933
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2.866811  5.121449  0.560  0.57602  
## CRIM        -0.202144  0.061618 -3.281  0.00115 ** 
## ZN          0.032606  0.012747  2.558  0.01098 *  
## CHAS        1.542995  0.878000  1.757  0.07978 .  
## NOX        -17.127808 3.241864 -5.283 2.32e-07 *** 
## RM          7.019081  0.432147 16.242 < 2e-16 *** 
## DIS         -0.839263  0.180368 -4.653 4.75e-06 *** 
## PTRATIO     -0.953705  0.119896 -7.954 2.95e-14 *** 
## B           0.014988  0.002812  5.329 1.84e-07 *** 
## LSTAT2      -0.003996  0.001395 -2.864  0.00446 ** 
## RAD_TAX_SCORE 56.448440 30.389438  1.858  0.06414 .  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.815 on 328 degrees of freedom
## Multiple R-squared:  0.811, Adjusted R-squared:  0.8053 
## F-statistic: 140.8 on 10 and 328 DF, p-value: < 2.2e-16

```

Se observa que, tras eliminar los leverages y valores atípicos, algunas variables han perdido significación estadística. No obstante, el modelo resultante muestra una mejora en términos de precisión, reflejada en la reducción del RMSE de **4.1** a **3.815**.

Este aspecto se explicará con mayor detalle en el siguiente apartado, donde se introducirá el concepto de RMSE y se analizará su evolución. Esta mejora respalda que la decisión de realizar estas eliminaciones es adecuada.

```

data_train <- data_train |>
  select(-CHAS, -RAD_TAX_SCORE)

data_test <- data_test |>
  select(-CHAS, -RAD_TAX_SCORE)

mlr <- lm(MEDV ~ ., data = data_train)

summary(mlr)

```

```

## 
## Call:
## lm(formula = MEDV ~ ., data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.2742  -2.2802  -0.2822   1.9442  13.5253
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.670788  5.037179  0.133  0.89414
## CRIM        -0.144730  0.051926 -2.787  0.00562 **
## ZN          0.030645  0.012818  2.391  0.01737 *
## NOX        -15.180332  3.107601 -4.885 1.62e-06 ***
## RM          7.232770  0.426775 16.948 < 2e-16 ***
## DIS        -0.800485  0.179489 -4.460 1.13e-05 ***
## PTRATIO    -0.913205  0.116158 -7.862 5.43e-14 ***
## B           0.014949  0.002821  5.299 2.14e-07 ***
## LSTAT2     -0.004166  0.001398 -2.981  0.00309 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.843 on 330 degrees of freedom
## Multiple R-squared:  0.8071, Adjusted R-squared:  0.8024
## F-statistic: 172.6 on 8 and 330 DF,  p-value: < 2.2e-16

```

El modelo ajustado presenta ahora un RMSE ligeramente superior, aunque comparable, con la ventaja de que todas las variables incluidas resultan estadísticamente significativas. Dado que la eliminación de valores atípicos y variables no significativas ha reducido el RMSE, continuaré con este nuevo modelo sin volver a comprobar las hipótesis clásicas del modelo lineal.

Como se comentó en clase, en contextos con datos reales es muy difícil que se cumplan todas las suposiciones teóricas del modelo de regresión lineal múltiple, y casi siempre hay alguna que se ve vulnerada. Por esta razón, y para evitar que el trabajo se vuelva excesivamente extenso, seguiré adelante con el análisis a partir de esta versión del modelo, que ya ofrece una mejora en precisión sin necesidad de mayor complejidad en esta fase.

3.1.15 Medidas de precisión

Una vez realizada la diagnosis del modelo y comprobado que, en general, se cumplen las hipótesis previas, el siguiente paso consiste en validar el modelo mediante el análisis de distintas medidas de precisión, como el **Coeficiente de Determinación Ajustado** (R^2_{ajustado}) y el **Error Estándar Residual** (RSE).

Tal como se concluyó en el apartado anterior, esta información puede obtenerse utilizando la función `summary(model)` en R. En las dos últimas líneas del resumen generado por dicha función se encuentran las principales métricas de precisión:

- **Coeficiente de determinación:** Multiple R-squared
- **Coeficiente de determinación ajustado:** Adjusted R-squared
- **Error estándar residual:** Residual standard error

Es posible acceder a dichos valores de forma directa:

```
tibble(
  Coef_Det = summary(mlr)$r.squared,
  Coef_Det_Aj = summary(mlr)$adj.r.squared,
```

```

RSE = summary(mlr)$sigma
) |>
print()

## # A tibble: 1 x 3
##   Coef_Det Coef_Det_Aj    RSE
##       <dbl>      <dbl> <dbl>
## 1     0.807      0.802  3.84

```

Se observa que el valor de R^2 es superior a 0.7, lo cual indica que el modelo tiene un buen ajuste. En concreto, el modelo es capaz de explicar aproximadamente el **80.24 %** de la variabilidad de la variable MEDV.

Por otro lado, el **Error Estándar Residual** (RSE) sugiere que, en promedio, las predicciones del modelo se desvían unos **3.84 mil dólares** respecto a los valores reales, lo que proporciona una estimación razonablemente precisa dentro del contexto del problema.

3.1.15.1 Técnicas de Validación: K-Folds Cross-Validation Dadas sus ventajas, se emplea la técnica de **validación cruzada con k particiones**, también conocida como k -Folds Cross-Validation (k -Folds CV). Esta técnica divide aleatoriamente el conjunto de datos en k subconjuntos de tamaño similar. En cada iteración, se utiliza uno de los subconjuntos como conjunto de validación y los $k - 1$ restantes como conjunto de entrenamiento, repitiendo el proceso k veces de manera que cada grupo actúe una vez como conjunto de validación.

Para su implementación, se recurre a funciones del paquete `boot`. En primer lugar, el modelo de regresión lineal se construye con la función `glm()`, y posteriormente se aplica `cv.glm(data, glmfit = model, K)` para obtener medidas de precisión como el **error cuadrático medio (MSE)** y la **raíz del error cuadrático medio (RMSE)**.

```

library(boot)

##
## Attaching package: 'boot'
## The following object is masked from 'package:car':
##
##     logit

data <- boston_housing |>
  mutate(LSTAT2 = LSTAT^2) |>
  select(-INDUS, -AGE, -LSTAT, -RAD, -TAX, -CHAS) |>
  drop_na() |>
  glimpse()

## #> #> #> Rows: 506
## #> #> #> Columns: 9
## #> #> #> $ CRIM      <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## #> #> #> $ ZN        <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## #> #> #> $ NOX       <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~
## #> #> #> $ RM         <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## #> #> #> $ DIS        <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## #> #> #> $ PTRATIO  <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## #> #> #> $ B          <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## #> #> #> $ MEDV      <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
## #> #> #> $ LSTAT2     <dbl> 24.8004, 83.5396, 16.2409, 8.6436, 28.4089, 27.1441, 154.5049, ~

model_validation <- data |>
  glm(MEDV ~ .,

```

```

dat = _)

kfcv <- data |>
  cv.glm(glmfit = model_validation,
         K = 10)

tibble(
  mse = kfcv$delta[1],
  rmse = sqrt(mse)
) |>
  print()

## # A tibble: 1 x 2
##       mse     rmse
##   <dbl> <dbl>
## 1 28.6  5.34

```

Dado que el valor de RMSE obtenido es de **5.34**, se puede interpretar que las predicciones presentan un error medio de aproximadamente ± 5.34 mil dólares respecto a los valores reales. Para valorar si este error es relativamente pequeño o grande, se analizará la distribución de la variable respuesta mostrando su histograma, así como su media y desviación típica.

```

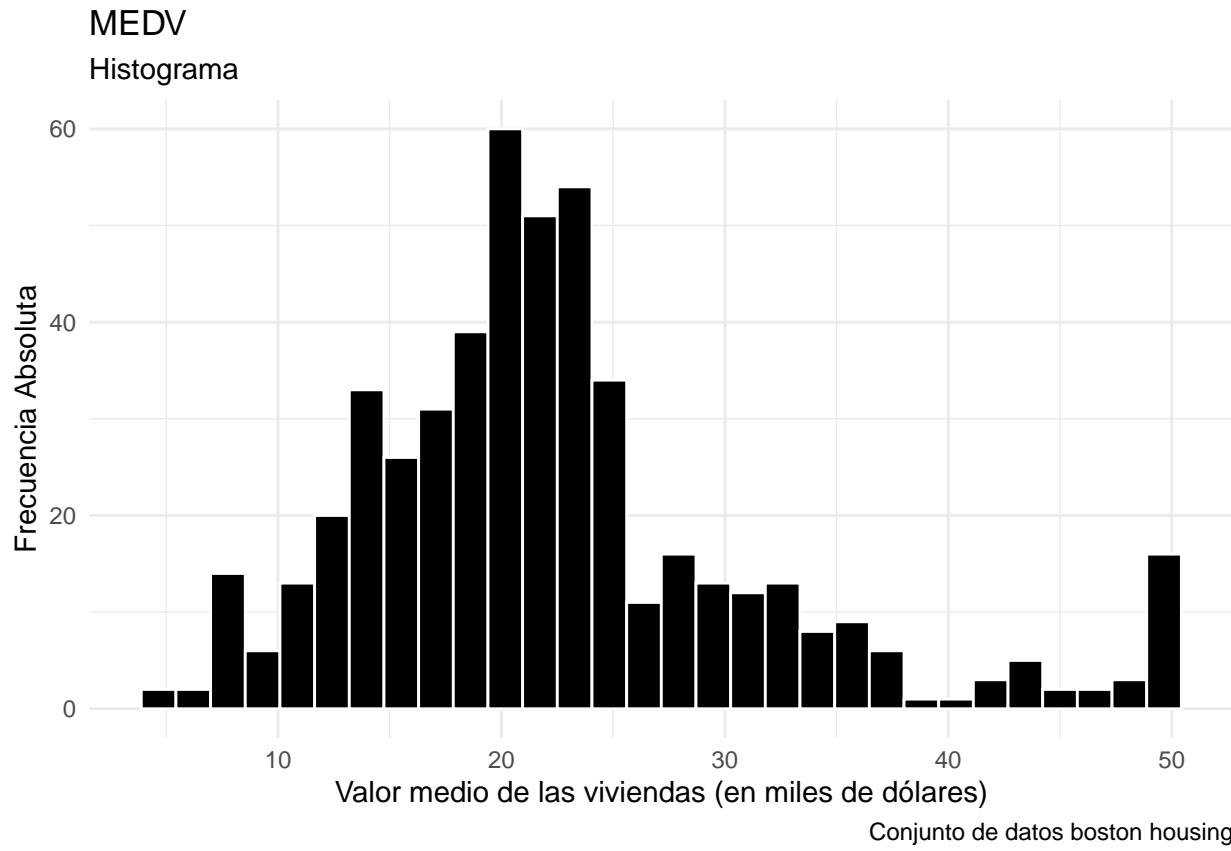
data |>
  summarise(
    mean = mean(MEDV),
    sd = sd(MEDV)
  )

## # A tibble: 1 x 2
##       mean     sd
##   <dbl> <dbl>
## 1 22.5  9.20

data |>
  ggplot() +
  aes(x = MEDV) +
  geom_histogram(fill = "black",
                 color = "white") +
  labs(x = "Valor medio de las viviendas (en miles de dólares)",
       y = "Frecuencia Absoluta",
       title = "MEDV",
       subtitle = "Histograma",
       caption = "Conjunto de datos boston housing") +
  theme_minimal()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



Tanto la media como la desviación típica de la variable respuesta MEDV (22.53 y 9.92, respectivamente), así como su histograma, indican que el valor de RMSE obtenido (**5.34**) se encuentra dentro de un rango razonable. A diferencia de otros casos donde el RMSE puede ser elevado en relación con la dispersión de los datos, en este análisis se observa que el error medio de predicción es **notablemente inferior a la desviación típica**, lo cual sugiere una buena capacidad predictiva del modelo.

Para reforzar esta interpretación, se analiza a continuación el RMSE de forma normalizada:

```
data |>
  summarize(
    normalized_rmse = sqrt(kfcv$delta[1]) / mean(MEDV)
  )

## # A tibble: 1 x 1
##   normalized_rmse
##       <dbl>
## 1         0.237
```

El valor de RMSE normalizado obtenido es de **0.23**, lo que implica que el error medio de predicción representa aproximadamente el **23 %** de la desviación típica de la variable respuesta. Aunque este valor es superior al umbral del 10 % que suele considerarse como indicativo de un modelo altamente preciso, sigue estando dentro de un rango aceptable en el contexto de datos reales, donde es habitual cierta variabilidad.

Por tanto, se puede concluir que el modelo presenta una **precisión razonable**, aunque con margen de mejora.

3.1.16 Predicción

El principal objetivo de la regresión lineal es la predicción de nuevos valores. Por este motivo, resulta fundamental evaluar las medidas de precisión, como el **error cuadrático medio (MSE)** y la **raíz del error cuadrático medio (RMSE)**, sobre el conjunto de validación.

La estimación de nuevas observaciones se lleva a cabo mediante la función `predict(model, newdata)` en R.

```
data_predict <- data_test |>
  mutate(
    predictions = predict(mlr,
                           newdata = data_test)
  ) |>
  glimpse()

## # Rows: 152
## # Columns: 10
## $ CRIM      <dbl> 0.00632, 0.02729, 0.06905, 0.08829, 0.14455, 0.21124, 0.22~
## $ ZN        <dbl> 18.0, 0.0, 0.0, 12.5, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0~
## $ NOX       <dbl> 0.538, 0.469, 0.458, 0.524, 0.524, 0.524, 0.524, 0.538, 0.~
## $ RM         <dbl> 6.575, 7.185, 7.147, 6.012, 6.172, 5.631, 6.377, 5.990, 5.~
## $ DIS        <dbl> 4.0900, 4.9671, 6.0622, 5.5605, 5.9505, 6.0821, 6.3467, 4.~
## $ PTRATIO   <dbl> 15.3, 17.8, 18.7, 15.2, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0~
## $ B          <dbl> 396.90, 392.83, 396.90, 395.60, 396.90, 386.63, 392.52, 38~
## $ MEDV       <dbl> 24.0, 34.7, 36.2, 22.9, 27.1, 16.5, 15.0, 17.5, 13.6, 15.2~
## $ LSTAT2     <dbl> 24.8004, 16.2409, 28.4089, 154.5049, 366.7225, 895.8049, 4~
## $ predictions <dbl> 29.19402, 31.08851, 29.28626, 23.50846, 23.48062, 17.09480~
```

A continuación, se procede al cálculo de las medidas de precisión **MSE** y **RMSE**.

```
data_predict |>
  summarise(
    mse = mean((MEDV - predictions)^2),
    rmse = sqrt(mse)
  ) |>
  print()

## # A tibble: 1 x 2
##       mse    rmse
##   <dbl> <dbl>
## 1  38.4  6.20
```

Tal y como se había observado en la validación, el valor de **RMSE** obtenido tras el proceso de predicción es moderadamente bajo, indicando que las predicciones presentan un error medio de aproximadamente **± 6.16 mil dólares** respecto a los valores reales.

3.1.17 Conclusión

El modelo de regresión lineal múltiple permitió detectar relaciones claras entre varias variables del conjunto de datos y el valor medio de las viviendas. Aunque se observó algo de multicolinealidad y heterocedasticidad, el resultado fue un buen nivel de ajuste global, lo que sugiere que el modelo representa razonablemente bien la realidad.

3.2 Regresión Penalizada

Los modelos de regresión tienen como objetivo principal predecir una variable respuesta a partir de un conjunto de variables predictoras, construyéndose generalmente bajo el criterio de minimizar la suma de los errores residuales al cuadrado (**RSS**). No obstante, estos modelos clásicos ofrecen buenos resultados

únicamente cuando se cumplen ciertas hipótesis teóricas, lo cual no siempre ocurre en contextos con datos reales.

Tal como se ha evidenciado en el apartado anterior, es frecuente encontrar situaciones en las que alguna de estas hipótesis no se cumple, lo que limita la fiabilidad del modelo clásico.

Como alternativa, surgen los **modelos de Regresión Penalizada**, que incorporan una penalización sobre los coeficientes del modelo con el fin de controlar su magnitud y mejorar la generalización.

Estos modelos minimizan una función de la forma:

$$\text{RSS} + \lambda \cdot \text{pen}(\beta),$$

donde:

- $\lambda > 0$ es un parámetro de penalización que regula el equilibrio entre el ajuste del modelo y la magnitud de los coeficientes.
- $\text{pen}(\beta)$ es una función de penalización que varía según la técnica empleada (como Ridge o Lasso), y permite establecer diferentes enfoques dentro de la regresión penalizada.

A mayor valor de λ , mayor será la contracción de los coeficientes, lo que puede reducir su impacto en el modelo y favorecer, en algunos casos, la selección de variables más relevantes.

A continuación, se considerarán únicamente aquellas variables que, en el apartado anterior, han demostrado tener una influencia significativa sobre la variable respuesta.

```
data |>
  glimpse()

## # Rows: 506
## # Columns: 9
## $ CRIM    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## $ ZN      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ NOX     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, 0.524, ~
## $ RM      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## $ DIS      <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ B        <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ MEDV    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
## $ LSTAT2   <dbl> 24.8004, 83.5396, 16.2409, 8.6436, 28.4089, 27.1441, 154.5049, ~
```

En este apartado se estudiarán dos técnicas de **regresión penalizada**:

- **Regresión Ridge**, cuya función de penalización es:

$$\text{pen}(\beta) = \sum_{j=1}^p \beta_j^2$$

- **Regresión Lasso**, cuya función de penalización es:

$$\text{pen}(\beta) = \sum_{j=1}^p |\beta_j|$$

Ambos modelos están implementados en el paquete **glmnet** mediante la función **glmnet(x, y, alpha, lambda)**, donde el parámetro **alpha** determina el tipo de regresión (**alpha = 0** para Ridge y **alpha = 1** para Lasso), y **lambda** representa el valor del parámetro de penalización.

Para aplicar correctamente estas técnicas, es necesario definir los conjuntos de entrenamiento y validación, y seleccionar el valor óptimo del parámetro λ .

Para evaluar de manera robusta la capacidad de generalización del modelo, se aplicó validación cruzada k-fold. Esta técnica consiste en dividir el conjunto de datos en k subconjuntos de igual tamaño. En cada iteración, uno de los subconjuntos se utiliza como conjunto de validación y los restantes como entrenamiento. Este procedimiento reduce la varianza de la estimación del error, mitiga el sobreajuste y proporciona una evaluación más confiable del rendimiento del modelo.

```
data_split <- data |>
  initial_split(prop = 0.7)

data_train <- data_split |>
  training() |>
  glimpse()

## Rows: 354
## Columns: 9
## $ CRIM    <dbl> 0.06724, 5.09017, 9.91655, 0.05602, 0.04294, 1.35472, 0.13960, ~
## $ ZN      <dbl> 0, 0, 0, 0, 28, 0, 0, 0, 0, 45, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX     <dbl> 0.460, 0.713, 0.693, 0.488, 0.464, 0.538, 0.520, 0.700, 0.448, ~
## $ RM      <dbl> 6.333, 6.297, 5.852, 7.831, 6.249, 6.072, 6.167, 5.000, 6.069, ~
## $ DIS     <dbl> 5.2146, 2.3682, 1.5004, 3.1992, 3.6150, 4.1750, 2.4210, 1.5184~
## $ PTRATIO <dbl> 16.9, 20.2, 20.2, 17.8, 18.2, 21.0, 20.9, 20.2, 17.9, 20.2, 21~
## $ B       <dbl> 375.21, 385.09, 338.16, 392.63, 396.90, 376.73, 392.69, 396.90~
## $ MEDV    <dbl> 22.6, 16.1, 6.3, 50.0, 20.6, 14.5, 20.1, 7.4, 21.2, 15.2, 23.9~
## $ LSTAT2  <dbl> 53.8756, 298.2529, 898.2009, 19.8025, 112.1481, 170.0416, 152.~

data_test <- data_split |>
  testing() |>
  glimpse()

## Rows: 152
## Columns: 9
## $ CRIM    <dbl> 0.00632, 0.06905, 0.11747, 0.09378, 0.63796, 1.05393, 0.78420, ~
## $ ZN      <dbl> 18.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ~
## $ NOX     <dbl> 0.538, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, 0.538, 0.538, ~
## $ RM      <dbl> 6.575, 7.147, 6.009, 5.889, 6.096, 5.935, 5.990, 6.142, 5.599, ~
## $ DIS     <dbl> 4.0900, 6.0622, 6.2267, 5.4509, 4.4619, 4.4986, 4.2579, 3.9769~
## $ PTRATIO <dbl> 15.3, 18.7, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21~
## $ B       <dbl> 396.90, 396.90, 396.90, 390.50, 380.02, 386.85, 386.75, 396.90~
## $ MEDV    <dbl> 24.0, 36.2, 18.9, 21.7, 18.2, 23.1, 17.5, 15.2, 13.9, 12.7, 13~
## $ LSTAT2  <dbl> 24.8004, 28.4089, 176.0929, 246.8041, 105.2676, 43.2964, 215.2~
```

3.2.1 Regresión Ridge

Este tipo de regresión penalizada se implementa mediante la función `glmnet()` estableciendo el parámetro `alpha = 0`.

En este modelo, los coeficientes de regresión varían en función del valor de λ , por lo que resulta fundamental analizar previamente cómo evolucionan dichos coeficientes (o, en su defecto, las variables predictoras) a medida que se modifica el valor del parámetro de penalización.

Cabe señalar que, para utilizar la función `glmnet()`, es necesario proporcionar la variable respuesta como un **vector numérico** y las variables predictoras como una **matriz**.

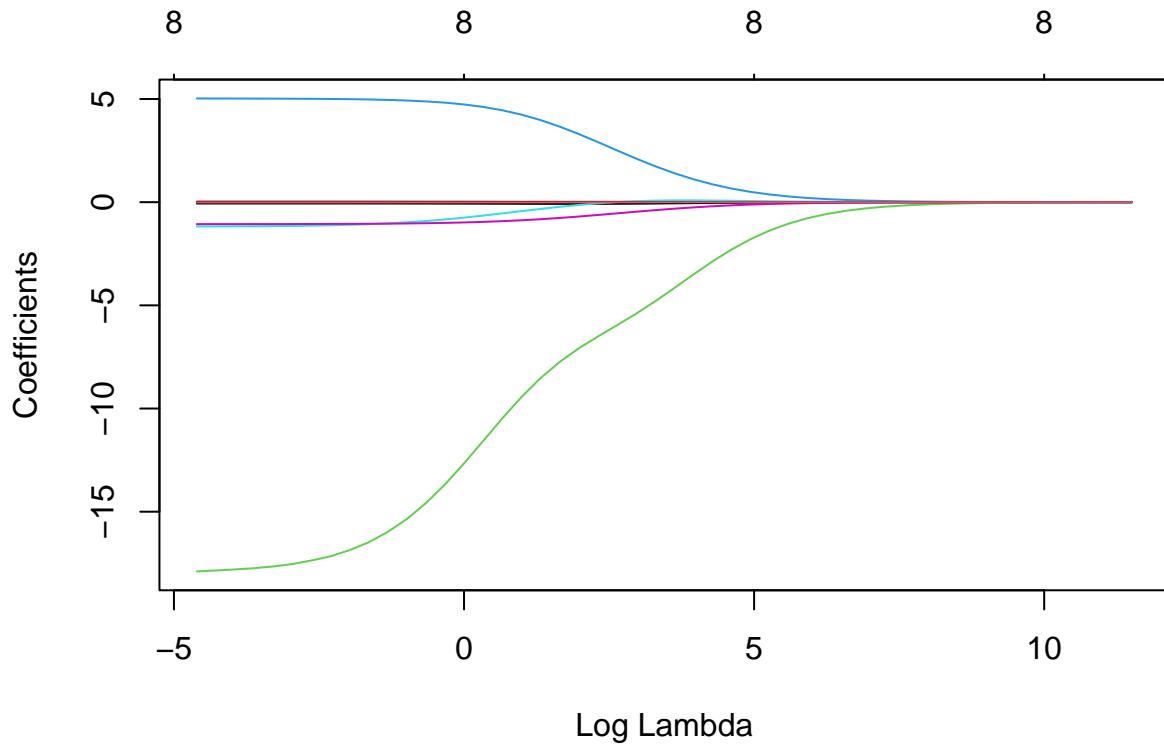
```
library(glmnet)
```

```

## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:tidyverse':
##   expand, pack, unpack
## Loaded glmnet 4.1-8
ridge <- data_train |>
  select(-MEDV) |>
  as.matrix() |>
  glmnet(y = data_train |>
    select(MEDV) |>
    pull(), alpha = 0, lambda = 10^seq(5,-2,length = 50))

ridge |>
  plot(xvar = "lambda")

```



Tal como se aprecia en el gráfico, a medida que el valor de λ aumenta, los coeficientes asociados a las variables predictoras tienden a contraerse hacia cero.

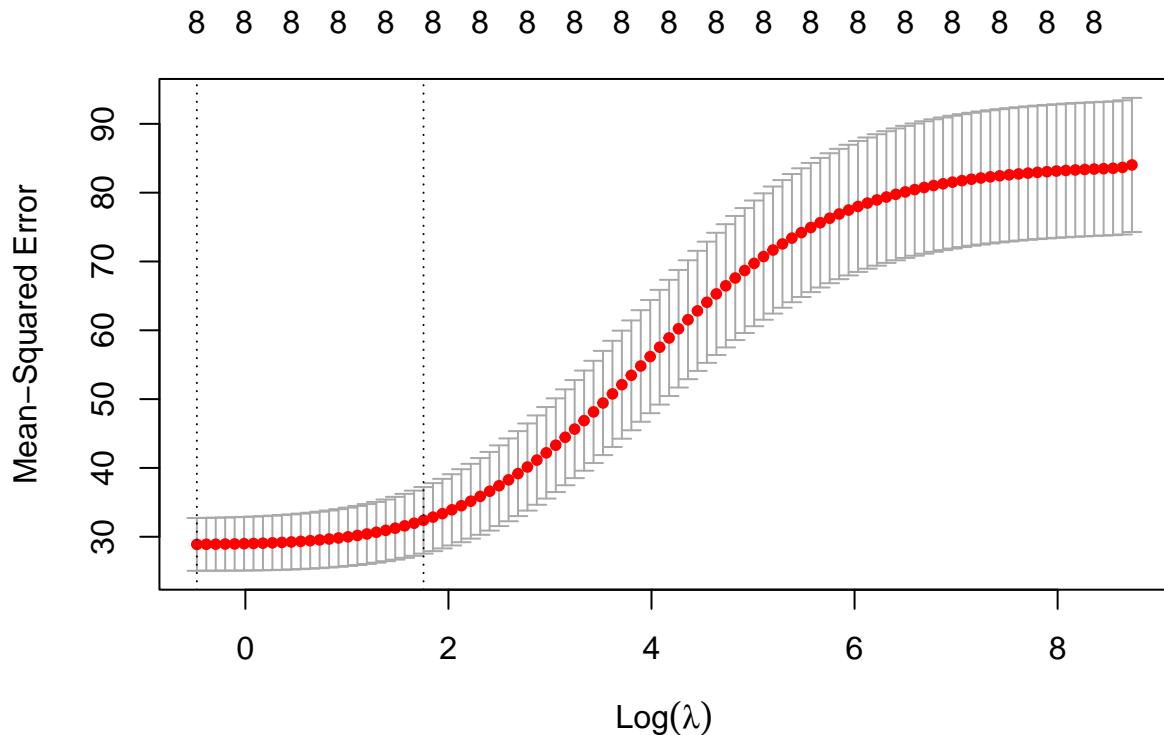
Para determinar el valor óptimo de λ que minimiza el error cuadrático medio (MSE), se aplicará la técnica de **k-Fold Cross-Validation** utilizando la función `cv.glmnet(alpha, nfolds)` del paquete `glmnet`.

```

CV_ridge <- data_train |>
  select(-MEDV) |>

```

```
as.matrix() |>  
cv.glmnet(y = data_train |>  
           select(MEDV) |> pull(),  
           alpha = 0,  
           nfolds = 10)  
  
CV_ridge |>  
plot()
```



Se observa que, a medida que el valor de λ incrementa, el MSE también tiende a aumentar. Por tanto, se procederá a identificar el valor mínimo del MSE y el correspondiente valor óptimo de λ .

```
ridge_metrics <- tibble(  
  mse_min = min(CV_ridge$cvm),  
  rmse_min = sqrt(mse_min),  
  lambda_min = CV_ridge$lambda.min  
) |>  
  glimpse()
```

```
## Rows: 1  
## Columns: 3  
## $ mse_min      <dbl> 28.88779  
## $ rmse_min     <dbl> 5.374736  
## $ lambda_min   <dbl> 0.6199248
```

Se concluye que el valor mínimo del MSE se alcanza con un parámetro de penalización $\lambda = 0.6199$.

A partir de este resultado, se procede a ajustar el modelo de **Regresión Ridge** utilizando dicho valor óptimo

de λ .

```
library(broom)

ridge <- data_train |>
  select(-MEDV) |>
  as.matrix() |>
  glmnet(y = data_train |>
    select(MEDV) |> pull(),
  alpha = 0,
  lambda = ridge_metrics$lambda_min)

ridge |>
  tidy()

## # A tibble: 9 x 5
##   term      step estimate lambda dev.ratio
##   <chr>     <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)     1  20.4     0.620    0.681
## 2 CRIM          1 -0.0679   0.620    0.681
## 3 ZN            1  0.0301   0.620    0.681
## 4 NOX           1 -14.1     0.620    0.681
## 5 RM            1   4.85    0.620    0.681
## 6 DIS           1  -0.874   0.620    0.681
## 7 PTRATIO        1  -1.01    0.620    0.681
## 8 B             1   0.00950  0.620    0.681
## 9 LSTAT2         1  -0.0105   0.620    0.681
```

Para visualizar los coeficientes del modelo en formato tabla, se emplea la función `tidy()` del paquete `broom`.

A continuación, se compararán estas estimaciones con las obtenidas previamente en el modelo de **Regresión Lineal Múltiple**.

```
data_train |>
  lm(MEDV ~ .,
  data = _) |>
  coef()

## (Intercept)          CRIM           ZN           NOX           RM
## 23.625844348 -0.063443130  0.036457816 -18.007084272  5.027800877
##           DIS          PTRATIO          B          LSTAT2
##  -1.184948155 -1.064574080  0.009539159  -0.010868758
```

Se observa que los coeficientes con valores más elevados en el modelo de regresión lineal múltiple han sido reducidos tras aplicar la regresión Ridge. Por ejemplo, el coeficiente asociado a la variable `NOX` ha pasado de **-25.68** a **-18.00**, mostrando cómo la penalización limita su magnitud sin cambiar su sentido.

A continuación, se realizarán predicciones con el modelo penalizado utilizando la función `predict.glmnet(model, s, newx)`, donde s representa el valor óptimo de λ y `newx` corresponde a la matriz de variables predictoras, permitiendo así calcular las medidas de precisión **MSE** y **RMSE**.

```
data_predict <- data_test |>
  mutate(
    predictions_ridge = ridge |>
    predict.glmnet(newx = data_test |>
      select(-MEDV) |>
      as.matrix(),
    s = ridge_metrics$lambda_min) |>
```

```

    as.numeric()
) |>
glimpse()

## # Rows: 152
## # Columns: 10
## $ CRIM      <dbl> 0.00632, 0.06905, 0.11747, 0.09378, 0.63796, 1.05393~
## $ ZN        <dbl> 18.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ~
## $ NOX       <dbl> 0.538, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, 0.5~
## $ RM         <dbl> 6.575, 7.147, 6.009, 5.889, 6.096, 5.935, 5.990, 6.1~
## $ DIS        <dbl> 4.0900, 6.0622, 6.2267, 5.4509, 4.4619, 4.4986, 4.25~
## $ PTRATIO   <dbl> 15.3, 18.7, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0, 21.0, ~
## $ B          <dbl> 396.90, 396.90, 396.90, 390.50, 380.02, 386.85, 386.~
## $ MEDV      <dbl> 24.0, 36.2, 18.9, 21.7, 18.2, 23.1, 17.5, 15.2, 13.9~
## $ LSTAT2    <dbl> 24.8004, 28.4089, 176.0929, 246.8041, 105.2676, 43.2~
## $ predictions_ridge <dbl> 29.74758, 27.90336, 23.66095, 22.95299, 19.73777, 19~
```

Respecto a las medidas de precisión:

```

data_predict |>
summarise(
  MSE = mean((MEDV - predictions_ridge)^2),
  RSME = sqrt(MSE)
) |>
glimpse()
```

```

## # Rows: 1
## # Columns: 2
## $ MSE  <dbl> 29.66255
## $ RSME <dbl> 5.446333
```

De este modo, las predicciones obtenidas presentan un error medio de aproximadamente ± 5.44 miles de dólares respecto a los valores reales.

Finalmente, uno de los principales inconvenientes de este tipo de regresión penalizada es que mantiene todas las variables predictoras en el modelo, lo que complica su interpretación.

Entre las principales limitaciones de los modelos utilizados se encuentran la suposición de linealidad en las relaciones entre variables predictoras y la respuesta, lo que puede no cumplirse completamente en todos los casos. Además, la presencia de multicolinealidad puede distorsionar la interpretación de los coeficientes, y los valores atípicos pueden influir desproporcionadamente en los resultados.

Por otro lado, la heterocedasticidad y la falta de normalidad de los residuos afectan la validez de las inferencias estadísticas.

Finalmente, el tamaño limitado del conjunto de datos podría restringir la capacidad de generalización de los modelos entrenados.

3.2.2 Regresión Lasso

Este tipo de regresión penalizada se implementa mediante la función `glmnet()` estableciendo el parámetro `alpha = 1`. Una de sus características principales es que, cuando el valor de λ es suficientemente grande, el modelo fuerza a que algunos coeficientes sean exactamente iguales a cero, lo que permite realizar una selección automática de las variables predictoras.

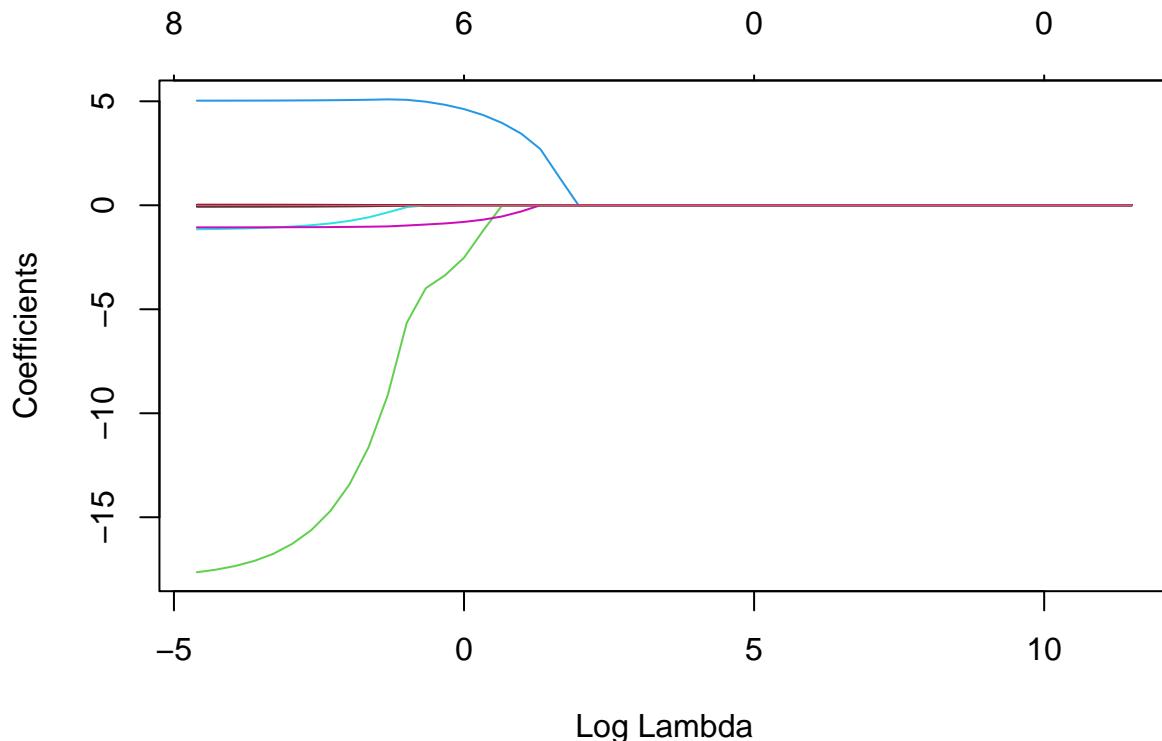
Al igual que en la regresión Ridge, se analizará cómo varían los coeficientes de las variables predictoras en función del valor de λ .

```

lasso <- data_train |>
  select(-MEDV) |>
  as.matrix() |>
  glmnet(y = data_train |>
    select(MEDV) |> pull(),
    alpha = 1,
    lambda = 10^seq(5,-2,length = 50))

lasso |>
  plot(xvar = "lambda")

```



Tal como se observa en el gráfico, a medida que el valor de λ aumenta, los coeficientes de las variables predictoras tienden a contraerse hacia cero.

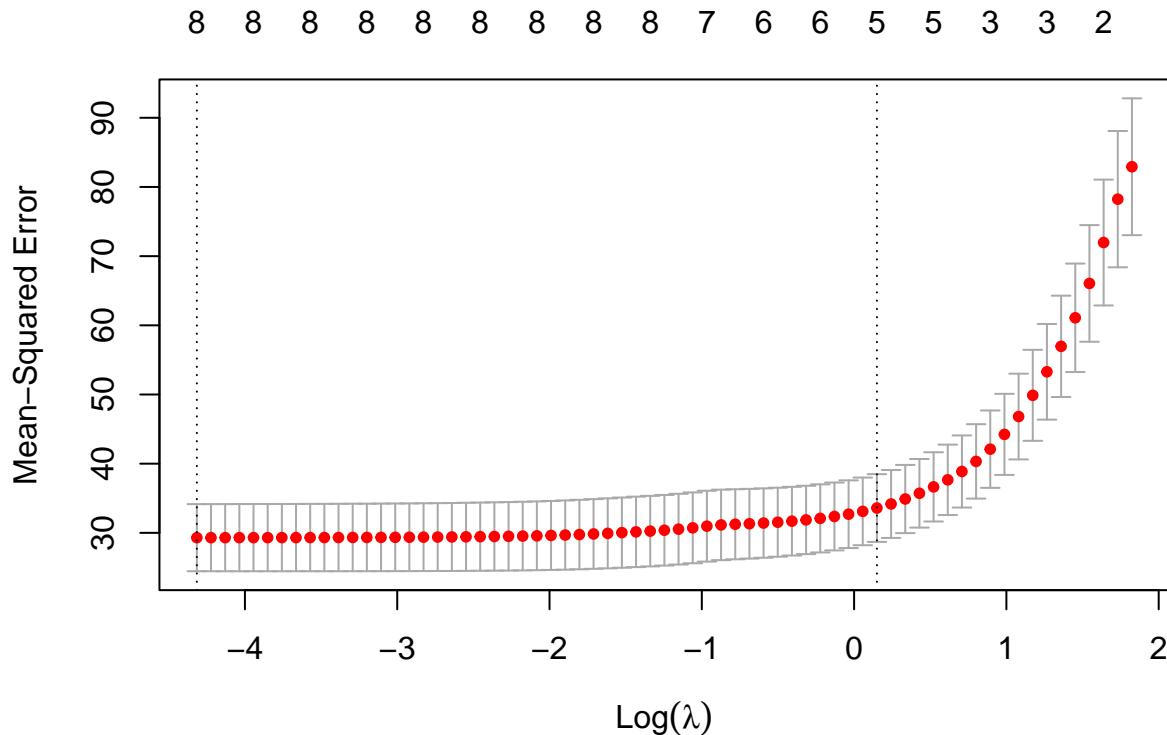
Para determinar el valor óptimo de λ que minimiza el MSE, se aplicará de nuevo la técnica de **k-Fold Cross-Validation** utilizando la función `cv.glmnet(alpha, nfolds)`.

```

CV_lasso <- data_train |>
  select(-MEDV) |>
  as.matrix() |>
  cv.glmnet(y = data_train |>
    select(MEDV) |> pull(),
    alpha = 1,
    nfolds = 10)

CV_lasso |>
  plot()

```



Se observa que, a medida que el valor de λ aumenta, el MSE también tiende a incrementarse.

Por tanto, se buscará identificar el valor mínimo del MSE junto con el correspondiente valor óptimo de λ . Además, se aprecia que, conforme aumenta el parámetro de penalización, el número de variables incluidas en el modelo disminuye, tal como se muestra en el eje horizontal superior del gráfico.

```
lasso_metrics <- tibble(
  MSE_min = min(CV_lasso$cvm),
  RMS_min = sqrt(MSE_min),
  lambda_min = CV_lasso$lambda.min
) |>
  glimpse()

## #> #> #> Rows: 1
## #> #> #> Columns: 3
## #> #> #> $ MSE_min     <dbl> 29.30215
## #> #> #> $ RMS_min     <dbl> 5.413146
## #> #> #> $ lambda_min <dbl> 0.01335587
```

Se concluye que el valor mínimo del MSE se alcanza con un parámetro de penalización $\lambda = 0.0133$.

A partir de este resultado, se procede a ajustar el modelo de **Regresión Lasso** utilizando dicho valor óptimo de λ .

```
lasso <- data_train |>
  select(-MEDV) |>
  as.matrix() |>
  glmnet(y = data_train |>
    select(MEDV) |> pull(),
```

```

    alpha = 1,
    lambda = lasso_metrics$lambda_min)

lasso |>
  tidy()

## # A tibble: 9 x 5
##   term      step estimate lambda dev.ratio
##   <chr>     <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept)     1    23.2    0.0134    0.684
## 2 CRIM          1   -0.0620   0.0134    0.684
## 3 ZN            1    0.0349   0.0134    0.684
## 4 NOX           1   -17.6    0.0134    0.684
## 5 RM            1     5.03    0.0134    0.684
## 6 DIS           1   -1.14    0.0134    0.684
## 7 PTRATIO        1   -1.06    0.0134    0.684
## 8 B             1    0.00949  0.0134    0.684
## 9 LSTAT2         1   -0.0109   0.0134    0.684

data_train |>
  lm(MEDV ~ .,
     data = _) |>
  coef()

## (Intercept)          CRIM          ZN          NOX          RM
## 23.625844348 -0.063443130  0.036457816 -18.007084272  5.027800877
##           DIS          PTRATIO          B          LSTAT2
## -1.184948155 -1.064574080  0.009539159 -0.010868758

```

Se observa que la regresión Lasso ha reducido ligeramente la magnitud de los coeficientes, pero no ha llegado a eliminar ninguna variable del modelo. Por ejemplo; el coeficiente de NOX, al igual que antes; ha pasado de **-25.68** a **-18.00**, lo que indica que, aunque se ha aplicado penalización, no ha sido suficiente para forzar ningún coeficiente a cero. A continuación, se realizan predicciones con el modelo ajustado mediante la función `predict.glmnet(model, s, newx)` del paquete `glmnet`, donde `s` corresponde al valor óptimo de λ y `newx` representa la matriz de variables predictoras.

A partir de estas predicciones se calcularán las métricas de precisión **MSE** y **RMSE**.

```

data_predict <- data_test |>
  mutate(
    predictions_lasso = lasso |>
      predict.glmnet(newx = data_test |>
        select(-MEDV) |>
        as.matrix(),
        s = lasso$lambda.min) |>
      as.numeric()
  ) |>
  glimpse()

## #> #> #> Rows: 152
## #> #> #> Columns: 10
## #> #> #> $ CRIM          <dbl> 0.00632, 0.06905, 0.11747, 0.09378, 0.63796, 1.05393~
## #> #> #> $ ZN           <dbl> 18.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ~
## #> #> #> $ NOX          <dbl> 0.538, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, 0.5~
## #> #> #> $ RM           <dbl> 6.575, 7.147, 6.009, 5.889, 6.096, 5.935, 5.990, 6.1~
## #> #> #> $ DIS          <dbl> 4.0900, 6.0622, 6.2267, 5.4509, 4.4619, 4.4986, 4.25~

```

```

## $ PTRATIO      <dbl> 15.3, 18.7, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0, 21.0~  

## $ B            <dbl> 396.90, 396.90, 396.90, 390.50, 380.02, 386.85, 386.~  

## $ MEDV         <dbl> 24.0, 36.2, 18.9, 21.7, 18.2, 23.1, 17.5, 15.2, 13.9~  

## $ LSTAT2       <dbl> 24.8004, 28.4089, 176.0929, 246.8041, 105.2676, 43.2~  

## $ predictions_lasso <dbl> 30.01099, 27.75661, 23.23159, 22.68796, 19.42291, 19~
```

Respecto a las medidas de precisión:

```

data_predict |>
  summarise(
    MSE = mean((MEDV - predictions_lasso)^2),
    RSME = sqrt(MSE)
  ) |>
  glimpse()
```

```

## Rows: 1
## Columns: 2
## $ MSE  <dbl> 29.02211
## $ RSME <dbl> 5.387218
```

De manera que las predicciones obtenidas tienen un error medio de ± 5.3872 miles de dólares respecto a los valores reales.

3.2.3 Conclusión

En conclusión, la aplicación de técnicas de **regresión penalizada** ha permitido mejorar la precisión del modelo respecto a la regresión lineal múltiple.

Inicialmente, las predicciones presentaban un error medio de aproximadamente **6.16 mil dólares**. Tras aplicar la regresión Ridge, este valor se redujo a **5.44**, y con la regresión Lasso descendió aún más hasta **5.38**. Estos resultados reflejan cómo la incorporación de penalización contribuye a obtener modelos más robustos y precisos, incluso sin eliminar variables en el caso de Lasso.

3.3 K-Nearest Neighbors (KNN) para Regresión

La técnica de aprendizaje supervisado **K-Nearest Neighbors (KNN)** permite realizar tanto tareas de regresión como de clasificación. En este caso, se aplicará el método KNN en un problema de regresión, con el objetivo de predecir el precio medio de la vivienda (**MEDV**), siguiendo la misma línea de los apartados anteriores.

3.3.1 Depuración del conjunto de datos

Al igual que en el apartado anterior, se mantendrán las mismas variables predictoras seleccionadas previamente, ya que han demostrado ser adecuadas para el análisis del precio medio de la vivienda (**MEDV**). Por tanto, no será necesario realizar una nueva selección de variables para aplicar la técnica KNN.

```

data |>
  glimpse()

## Rows: 506
## Columns: 9
## $ CRIM      <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## $ ZN        <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 1~  

## $ NOX       <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~  

## $ RM         <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~  

## $ DIS        <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~  

## $ PTRATIO   <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15~  

## $ B          <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
```

```
## $ MEDV    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~  
## $ LSTAT2  <dbl> 24.8004, 83.5396, 16.2409, 8.6436, 28.4089, 27.1441, 154.5049,~
```

3.3.2 Búsqueda del valor de K

El paquete `caret` dispone de la función `knnreg()`, que permite aplicar el método de vecinos más próximos para resolver problemas de regresión.

Esta técnica predice el valor de la variable respuesta Y a partir del promedio de los k vecinos más cercanos según las variables predictoras X_1, \dots, X_p . Por tanto, es necesario definir una **medida de cercanía** y determinar el valor **óptimo de k** .

En este caso, se utilizará como medida de cercanía la **distancia euclídea**, definida entre dos observaciones $x = (x_1, \dots, x_p)$ y $z = (z_1, \dots, z_p)$ como:

$$d(x, z) = \sqrt{(x_1 - z_1)^2 + \dots + (x_p - z_p)^2}$$

Para encontrar el valor óptimo de k , se empleará una técnica de **validación cruzada**, implementada mediante la función `train()` del paquete `caret`.

Esta función permite especificar el método (`method`), el preprocesado de las variables (`preProc`), la métrica de error a optimizar (`metric`), los controles de validación (`trControl`) y los valores a probar para k mediante el argumento `tuneGrid`.

Dado que KNN es sensible a la escala de las variables, será necesario **estandarizarlas previamente** utilizando el argumento `preProc`.

Respecto a la validación, esta se gestionará a través del argumento `trControl`, empleando la función `trainControl()`, y la rejilla de valores de k a evaluar se definirá con el argumento `tuneGrid`.

```
library(caret)

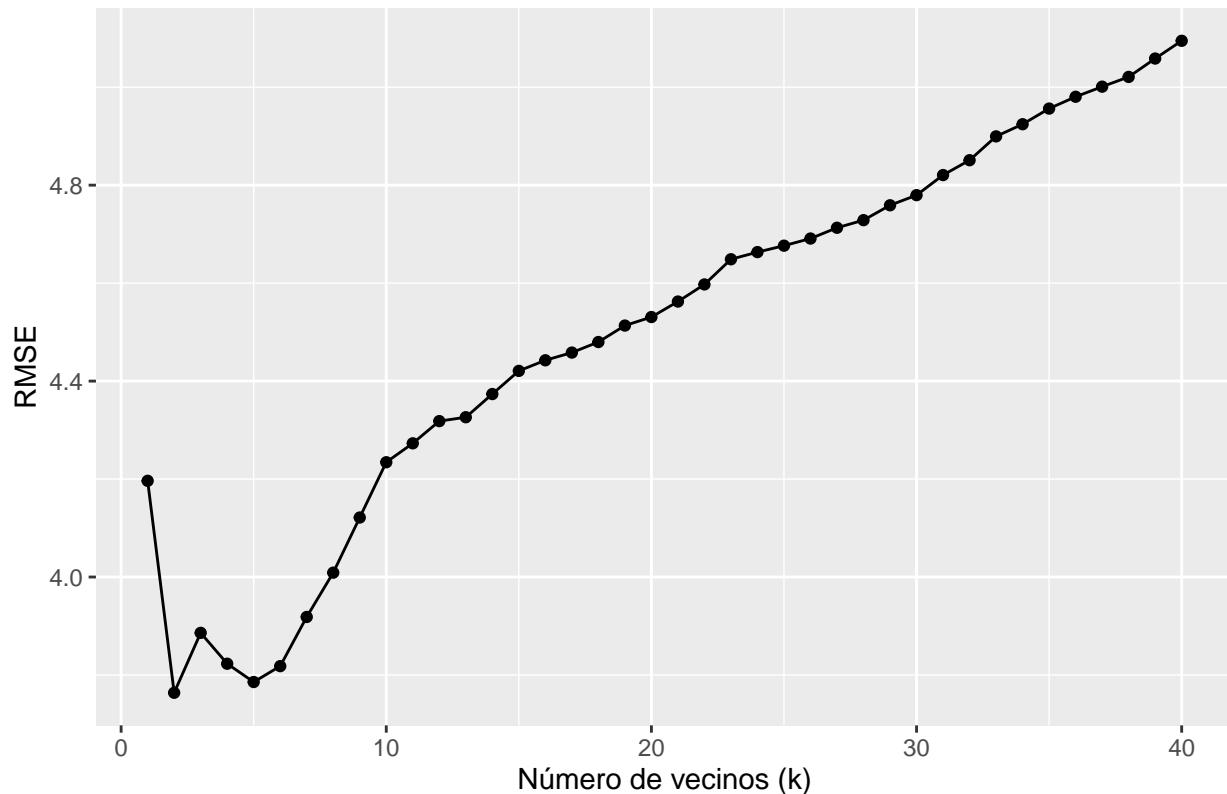
## Loading required package: lattice
##
## Attaching package: 'lattice'
## The following object is masked from 'package:boot':
##   melanoma
##
## Attaching package: 'caret'
## The following object is masked from 'package:purrr':
##   lift

KNN_10FCV <- data |>
  train(MEDV ~ .,
        data = _,
        method = "knn",
        preProc = c("center", "scale"),
        metric = "RMSE",
        trControl = trainControl(method = "cv", number = 10),
        tuneGrid = data.frame(k = 1:40)
  )

KNN_10FCV |>
```

```
ggplot() +
  labs(x = "Número de vecinos (k)",
       y = "RMSE",
       title = "10-Folds CV")
```

10-Folds CV



Al representar el número de vecinos k frente a la raíz del error cuadrático medio (RMSE), se observa que a partir de $k = 2$ el RMSE comienza a incrementarse. Por tanto, el valor óptimo que minimiza el error cuadrático medio se alcanza con $k = 2$.

```
tibble(k = KNN_10FCV$bestTune$k,
       RMSE = mean(KNN_10FCV$results$RMSE))
```

```
## # A tibble: 1 x 2
##       k   RMSE
##   <int> <dbl>
## 1     2    4.50
```

3.3.3 Conjuntos de entrenamiento y validación

Para aplicar el modelo de regresión **KNN**, es necesario dividir previamente el conjunto de datos en subconjuntos de entrenamiento y validación.

Se utilizarán los mismos conjuntos de entrenamiento y validación empleados en el apartado anterior.

```
data_train |>
  glimpse()
```

```
## #> #> Rows: 354
```

```

## Columns: 9
## $ CRIM    <dbl> 0.06724, 5.09017, 9.91655, 0.05602, 0.04294, 1.35472, 0.13960, ~
## $ ZN      <dbl> 0, 0, 0, 0, 28, 0, 0, 0, 0, 0, 0, 45, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ NOX     <dbl> 0.460, 0.713, 0.693, 0.488, 0.464, 0.538, 0.520, 0.700, 0.448, ~
## $ RM      <dbl> 6.333, 6.297, 5.852, 7.831, 6.249, 6.072, 6.167, 5.000, 6.069, ~
## $ DIS      <dbl> 5.2146, 2.3682, 1.5004, 3.1992, 3.6150, 4.1750, 2.4210, 1.5184, ~
## $ PTRATIO <dbl> 16.9, 20.2, 20.2, 17.8, 18.2, 21.0, 20.9, 20.2, 17.9, 20.2, 21~
## $ B        <dbl> 375.21, 385.09, 338.16, 392.63, 396.90, 376.73, 392.69, 396.90~
## $ MEDV    <dbl> 22.6, 16.1, 6.3, 50.0, 20.6, 14.5, 20.1, 7.4, 21.2, 15.2, 23.9~
## $ LSTAT2   <dbl> 53.8756, 298.2529, 898.2009, 19.8025, 112.1481, 170.0416, 152.~

data_test |>
  glimpse()

## Rows: 152
## Columns: 9
## $ CRIM    <dbl> 0.00632, 0.06905, 0.11747, 0.09378, 0.63796, 1.05393, 0.78420, ~
## $ ZN      <dbl> 18.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ~
## $ NOX     <dbl> 0.538, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, 0.538, ~
## $ RM      <dbl> 6.575, 7.147, 6.009, 5.889, 6.096, 5.935, 5.990, 6.142, 5.599, ~
## $ DIS      <dbl> 4.0900, 6.0622, 6.2267, 5.4509, 4.4619, 4.4986, 4.2579, 3.9769, ~
## $ PTRATIO <dbl> 15.3, 18.7, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21~
## $ B        <dbl> 396.90, 396.90, 396.90, 390.50, 380.02, 386.85, 386.75, 396.90~
## $ MEDV    <dbl> 24.0, 36.2, 18.9, 21.7, 18.2, 23.1, 17.5, 15.2, 13.9, 12.7, 13~
## $ LSTAT2   <dbl> 24.8004, 28.4089, 176.0929, 246.8041, 105.2676, 43.2964, 215.2~

```

3.3.4 Aplicación del modelo KNN

A continuación, se aplica el método **KNN** utilizando el conjunto de entrenamiento, y posteriormente se generan las predicciones sobre el conjunto de validación para calcular las métricas de precisión, concretamente el **MSE** y el **RMSE**.

Para ello, se empleará la función `knnreg(k)`.

```

knn <- data_train |>
  knnreg(MEDV ~ .,
         data = _,
         k = KNN_10FCV$bestTune$k)

```

3.3.5 Predicción y Medidas de Precisión

Para construir las predicciones se emplea la función `predict()`.

```

data_predict <- data_test |>
  mutate(
    predictions_KNN = knn |>
      predict(newdata = data_test)
  ) |>
  glimpse()

## Rows: 152
## Columns: 10
## $ CRIM      <dbl> 0.00632, 0.06905, 0.11747, 0.09378, 0.63796, 1.05393, ~
## $ ZN        <dbl> 18.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0~
## $ NOX       <dbl> 0.538, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, ~
## $ RM        <dbl> 6.575, 7.147, 6.009, 5.889, 6.096, 5.935, 5.990, 6.142~
## $ DIS        <dbl> 4.0900, 6.0622, 6.2267, 5.4509, 4.4619, 4.4986, 4.2579~
```

```

## $ PTRATIO      <dbl> 15.3, 18.7, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0, ~
## $ B            <dbl> 396.90, 396.90, 396.90, 390.50, 380.02, 386.85, 386.75~
## $ MEDV         <dbl> 24.0, 36.2, 18.9, 21.7, 18.2, 23.1, 17.5, 15.2, 13.9, ~
## $ LSTAT2        <dbl> 24.8004, 28.4089, 176.0929, 246.8041, 105.2676, 43.296~
## $ predictions_KNN <dbl> 42.70, 26.65, 19.90, 19.85, 20.80, 22.45, 19.55, 18.10~

data_predict |>
  summarise(
    MSE = mean((MEDV - predictions_KNN)^2),
    RSME = sqrt(MSE)
  ) |>
  glimpse()

## #> #> Rows: 1
## #> #> Columns: 2
## #> #> $ MSE  <dbl> 29.03206
## #> #> $ RSME <dbl> 5.38814

```

De manera que las predicciones obtenidas tienen un error medio de ± 5.388 miles de dólares respecto a los valores reales. Este valor es ligeramente superior al obtenido con la regresión Lasso.

3.3.6 Conclusión

En conclusión, la técnica **K-Nearest Neighbors (KNN)** ha permitido construir un modelo de regresión con un rendimiento aceptable en la predicción del precio medio de la vivienda. El valor óptimo de k obtenido mediante validación cruzada fue $k = 2$, y el modelo resultante alcanzó un **RMSE de 5.388**, lo que indica un error medio de ± 5.388 miles de dólares respecto a los valores reales. Aunque el modelo ofrece un ajuste razonable, el error es ligeramente superior al obtenido con la regresión Lasso, lo que sugiere que en este caso **KNN no mejora los resultados de los modelos anteriores**, aunque sigue siendo una alternativa válida y fácil de interpretar dentro de las técnicas de regresión supervisada.

3.4 Arboles Aleatorios para Regresión

Los **Árboles Aleatorios** (Random Forest) constituyen una técnica de aprendizaje supervisado basada en ensamblado de múltiples árboles de decisión. Este método resulta especialmente útil para tareas de regresión, ya que permite capturar relaciones no lineales entre variables y mejorar la capacidad de generalización del modelo.

A diferencia de un único árbol de decisión, que puede sufrir de sobreajuste, un bosque aleatorio entrena numerosos árboles sobre subconjuntos aleatorios del conjunto de datos y selecciona aleatoriamente un subconjunto de variables en cada división. La predicción final se obtiene promediando los resultados individuales de cada árbol, lo que reduce la varianza y mejora la robustez.

Este enfoque presenta varias ventajas clave:

- **Mayor precisión** frente a modelos individuales, al reducir el sobreajuste.
- **Capacidad de manejar datos complejos y no lineales** sin necesidad de una transformación previa.
- **Medición de la importancia de las variables**, lo que permite interpretar el modelo desde una perspectiva analítica.

3.4.1 Depuración del conjunto de datos

Al igual que en los apartados anteriores, se mantendrán las mismas variables predictoras seleccionadas previamente, ya que han demostrado ser adecuadas para el análisis del precio medio de la vivienda (**MEDV**).

```

data |>
  glimpse()

```

```

## Rows: 506
## Columns: 9
## $ CRIM    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## $ ZN      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
## $ NOX     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~
## $ RM      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## $ DIS      <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## $ B       <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## $ MEDV    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
## $ LSTAT2   <dbl> 24.8004, 83.5396, 16.2409, 8.6436, 28.4089, 27.1441, 154.5049, ~

```

3.4.2 Búsqueda óptima para los parámetros: profundidad y complejidad

Los modelos de árboles aleatorios pueden construirse a través del paquete `rpart`, aunque es necesario entrenar el modelo para determinar parámetros clave como la **profundidad máxima del árbol** o el **parámetro de complejidad**. Para ello, se utiliza la función `train()` del paquete `caret`, que permite optimizar dichos parámetros mediante técnicas de validación cruzada.

En particular, para ajustar la profundidad del árbol, se emplea el argumento `method = "rpart2"` junto con `tuneGrid`, que permite definir el rango de valores a evaluar.

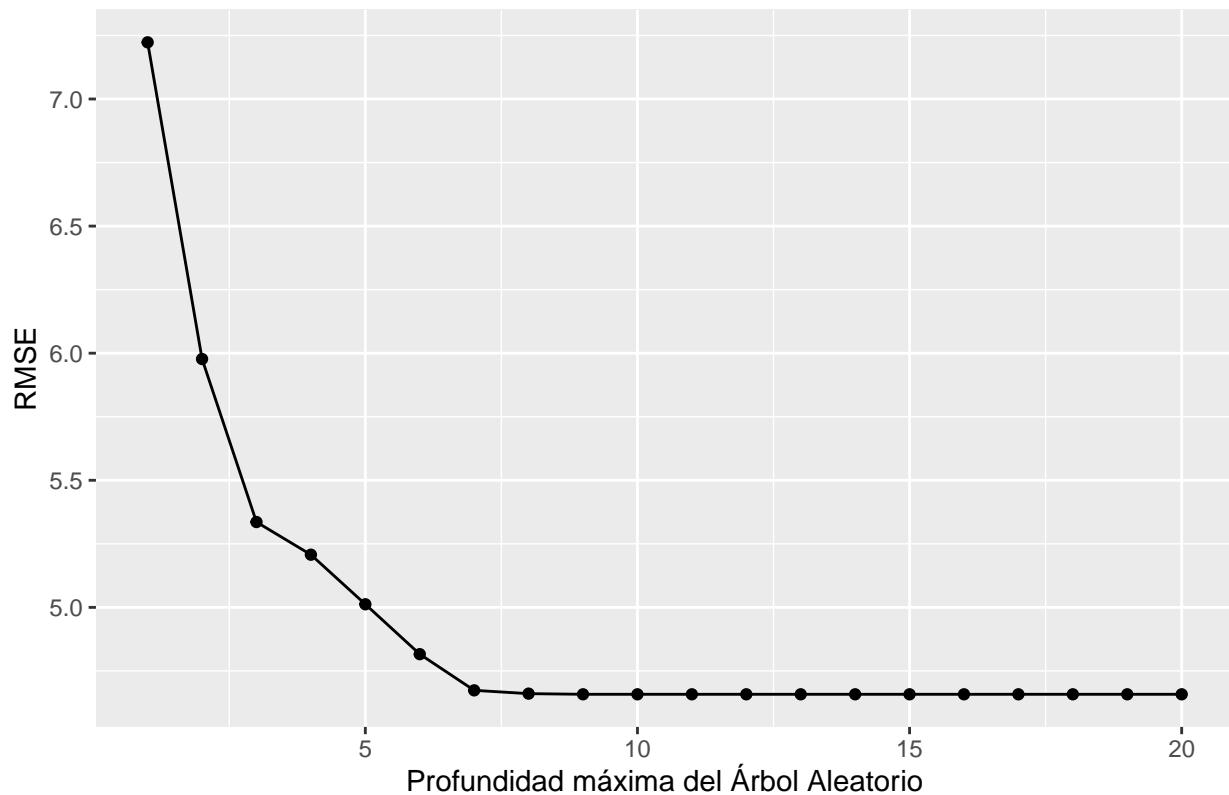
```

RT_10FCV_maxdepth <- data |>
  train(MEDV ~ .,
        data = _,
        method = "rpart2",
        metric = "RMSE",
        trControl = trainControl(method = "cv", number = 10),
        tuneGrid = data.frame(maxdepth = 1:20)
  )

RT_10FCV_maxdepth |>
  ggplot() +
  labs(x = "Profundidad máxima del Árbol Aleatorio",
       y = "RMSE",
       title = "10-Folds CV")

```

10–Folds CV



Se observa que a partir de una profundidad máxima de 9, el valor del RMSE se estabiliza. Por tanto, se considera que una profundidad igual a 10 es la que permite minimizar el error cuadrático medio de forma óptima.

```
tibble(maxdepth = RT_10FCV_maxdepth$bestTune$maxdepth,
      RMSE = mean(RT_10FCV_maxdepth$resample$RMSE))
```

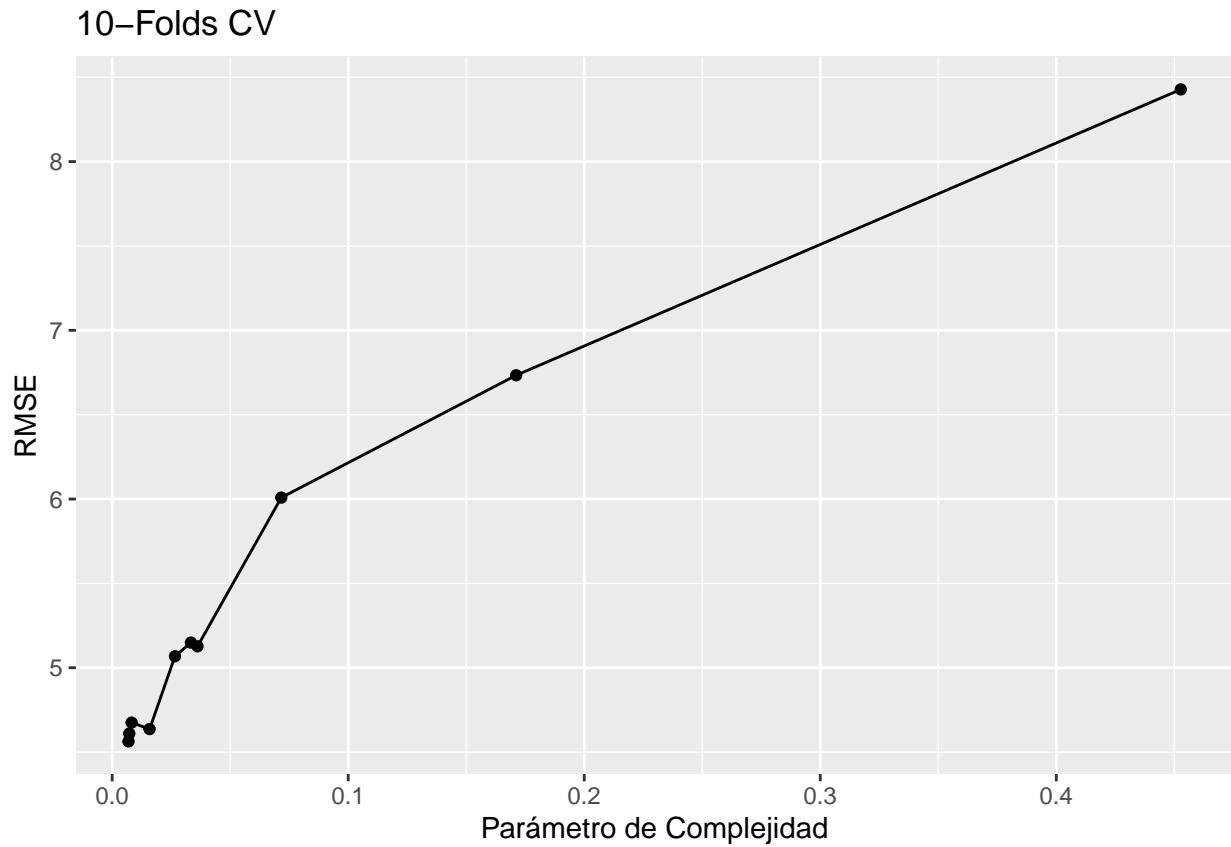
```
## # A tibble: 1 x 2
##   maxdepth    RMSE
##       <int>   <dbl>
## 1         9     4.66
```

En cuanto al parámetro de complejidad, también es necesario determinar su valor óptimo. Para ello, se utiliza la función `train()` con los argumentos `method = "rpart"` y `tuneLength`, que permite especificar el número de valores que se evaluarán durante el proceso de ajuste.

```
RT_10FCV_complexity <- data |>
  train(MEDV ~ .,
        data = _,
        method = "rpart",
        metric = "RMSE",
        trControl = trainControl(method = "cv", number = 10),
        tuneLength = 10
  )
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
RT_10FCV_complexity |>  
  ggplot() +  
  labs(x = "Parámetro de Complejidad",  
       y = "RMSE",  
       title = "10-Folds CV")
```



Se observa que, a medida que el parámetro de complejidad aumenta, el RMSE también se incrementa, lo que indica que una mayor complejidad del modelo conduce a un mayor error cuadrático medio.

```
tibble(maxdepth = RT_10FCV_maxdepth$bestTune$maxdepth,
      RMSE = mean(RT_10FCV_maxdepth$resample$RMSE))
```

```
## # A tibble: 1 x 2
##   maxdepth    RMSE
##       <int>   <dbl>
## 1         9 4.66
```

3.4.3 Conjuntos de entrenamiento y validación

Una vez determinados los parámetros óptimos, se procede a aplicar el modelo de regresión de **Árboles Aleatorios** sobre los conjuntos de entrenamiento y validación.

```
data_train |>  
  glimpse()
```

```
## Rows: 354  
## Columns: 9  
## $ CRIM      <dbl> 0.06724, 5.09017, 9.91655, 0.05602, 0.04294, 1.35472, 0.13960, ~
```

```

## $ ZN      <dbl> 0, 0, 0, 0, 28, 0, 0, 0, 0, 0, 0, 45, 0, 0, 0, 0, 0, 0, ~
## $ NOX     <dbl> 0.460, 0.713, 0.693, 0.488, 0.464, 0.538, 0.520, 0.700, 0.448, ~
## $ RM       <dbl> 6.333, 6.297, 5.852, 7.831, 6.249, 6.072, 6.167, 5.000, 6.069, ~
## $ DIS      <dbl> 5.2146, 2.3682, 1.5004, 3.1992, 3.6150, 4.1750, 2.4210, 1.5184, ~
## $ PTRATIO  <dbl> 16.9, 20.2, 20.2, 17.8, 18.2, 21.0, 20.9, 20.2, 17.9, 20.2, 21~
## $ B        <dbl> 375.21, 385.09, 338.16, 392.63, 396.90, 376.73, 392.69, 396.90, ~
## $ MEDV    <dbl> 22.6, 16.1, 6.3, 50.0, 20.6, 14.5, 20.1, 7.4, 21.2, 15.2, 23.9, ~
## $ LSTAT2   <dbl> 53.8756, 298.2529, 898.2009, 19.8025, 112.1481, 170.0416, 152.~
data_test |>
  glimpse()

## # Rows: 152
## # Columns: 9
## $ CRIM    <dbl> 0.00632, 0.06905, 0.11747, 0.09378, 0.63796, 1.05393, 0.78420, ~
## $ ZN      <dbl> 18.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ~
## $ NOX     <dbl> 0.538, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, 0.538, 0.538, ~
## $ RM       <dbl> 6.575, 7.147, 6.009, 5.889, 6.096, 5.935, 5.990, 6.142, 5.599, ~
## $ DIS      <dbl> 4.0900, 6.0622, 6.2267, 5.4509, 4.4619, 4.4986, 4.2579, 3.9769, ~
## $ PTRATIO  <dbl> 15.3, 18.7, 15.2, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21~
## $ B        <dbl> 396.90, 396.90, 396.90, 390.50, 380.02, 386.85, 386.75, 396.90, ~
## $ MEDV    <dbl> 24.0, 36.2, 18.9, 21.7, 18.2, 23.1, 17.5, 15.2, 13.9, 12.7, 13~
## $ LSTAT2   <dbl> 24.8004, 28.4089, 176.0929, 246.8041, 105.2676, 43.2964, 215.2~

```

3.4.4 Aplicación del modelo Árboles Aleatorios

A continuación, se aplica el método de **Árboles Aleatorios** sobre el conjunto de entrenamiento, y posteriormente se generan las predicciones sobre el conjunto de validación con el objetivo de calcular las medidas de precisión, concretamente el **MSE** y el **RMSE**.

Para ajustar el modelo se utilizará la función **rpart(control)**.

```

library(rpart)

model_RT <- data_train |>
  rpart(MEDV ~ .,
        data = _,
        control = rpart.control(maxdepth = RT_10FCV_maxdepth$bestTune$maxdepth,
                                cp = RT_10FCV_complexity$bestTune$cp))

model_RT |>
  summary()

## Call:
## rpart(formula = MEDV ~ ., data = data_train, control = rpart.control(maxdepth = RT_10FCV_maxdepth$be
## ## cp = RT_10FCV_complexity$bestTune$cp)
## n= 354
##
##          CP nsplits rel_error      xerror      xstd
## 1  0.462391457      0 1.0000000 1.0069048 0.10074061
## 2  0.178833009      1 0.5376085 0.6730718 0.07117713
## 3  0.062982823      2 0.3587755 0.4235055 0.05537499
## 4  0.040841913      3 0.2957927 0.3704877 0.05179280
## 5  0.029615123      4 0.2549508 0.3580818 0.05230069
## 6  0.017319934      5 0.2253357 0.3118554 0.04949707
## 7  0.010467658      6 0.2080157 0.2983458 0.05018640

```

```

## 8 0.010442097      7 0.1975481 0.2902553 0.04981748
## 9 0.008779918      8 0.1871060 0.2902553 0.04981748
## 10 0.007526265     9 0.1783261 0.2841753 0.04777553
## 11 0.006931087    10 0.1707998 0.2851188 0.04815969
##
## Variable importance
##      RM LSTAT2 PTRATIO      CRIM      NOX      DIS      ZN      B
##      31     24     12       9       8       8       4       3
##
## Node number 1: 354 observations,      complexity param=0.4623915
##   mean=22.31864, MSE=83.45536
##   left son=2 (304 obs) right son=3 (50 obs)
## Primary splits:
##       RM      < 6.945 to the left, improve=0.4623915, (0 missing)
##       LSTAT2 < 61.85845 to the right, improve=0.4518394, (0 missing)
##       PTRATIO < 19.9 to the right, improve=0.2670827, (0 missing)
##       NOX      < 0.657 to the right, improve=0.2438186, (0 missing)
##       CRIM      < 6.05604 to the right, improve=0.2216765, (0 missing)
## Surrogate splits:
##       LSTAT2 < 20.02625 to the right, agree=0.904, adj=0.32, (0 split)
##       PTRATIO < 14.55 to the right, agree=0.893, adj=0.24, (0 split)
##       ZN      < 87.5 to the left, agree=0.873, adj=0.10, (0 split)
##       CRIM      < 0.013355 to the right, agree=0.862, adj=0.02, (0 split)
##
## Node number 2: 304 observations,      complexity param=0.178833
##   mean=19.79934, MSE=40.56822
##   left son=4 (128 obs) right son=5 (176 obs)
## Primary splits:
##       LSTAT2 < 207.2172 to the right, improve=0.4283962, (0 missing)
##       NOX      < 0.657 to the right, improve=0.3045514, (0 missing)
##       CRIM      < 6.05604 to the right, improve=0.2849696, (0 missing)
##       PTRATIO < 19.9 to the right, improve=0.2284910, (0 missing)
##       DIS      < 2.5977 to the left, improve=0.2192642, (0 missing)
## Surrogate splits:
##       DIS      < 2.23935 to the left, agree=0.789, adj=0.500, (0 split)
##       NOX      < 0.5765 to the right, agree=0.780, adj=0.477, (0 split)
##       CRIM      < 4.067905 to the right, agree=0.763, adj=0.438, (0 split)
##       PTRATIO < 19.9 to the right, agree=0.737, adj=0.375, (0 split)
##       B        < 340.115 to the left, agree=0.691, adj=0.266, (0 split)
##
## Node number 3: 50 observations,      complexity param=0.06298282
##   mean=37.636, MSE=70.9987
##   left son=6 (29 obs) right son=7 (21 obs)
## Primary splits:
##       RM      < 7.445 to the left, improve=0.52415430, (0 missing)
##       PTRATIO < 18.5 to the right, improve=0.26948210, (0 missing)
##       LSTAT2 < 26.88845 to the right, improve=0.23524340, (0 missing)
##       NOX      < 0.626 to the right, improve=0.09937387, (0 missing)
##       B        < 395.59 to the right, improve=0.08234223, (0 missing)
## Surrogate splits:
##       LSTAT2 < 15.80365 to the right, agree=0.74, adj=0.381, (0 split)
##       CRIM      < 0.110415 to the left, agree=0.72, adj=0.333, (0 split)
##       NOX      < 0.48 to the left, agree=0.68, adj=0.238, (0 split)
##       DIS      < 3.58335 to the right, agree=0.68, adj=0.238, (0 split)

```

```

##      ZN      < 8.75      to the right, agree=0.66, adj=0.190, (0 split)
##
## Node number 4: 128 observations,      complexity param=0.02961512
##   mean=14.91094, MSE=19.47988
##   left son=8 (56 obs) right son=9 (72 obs)
## Primary splits:
##   CRIM      < 6.340595 to the right, improve=0.3508931, (0 missing)
##   NOX       < 0.657    to the right, improve=0.3347511, (0 missing)
##   LSTAT2    < 396.0104 to the right, improve=0.2894240, (0 missing)
##   DIS        < 2.0643   to the left,  improve=0.2778578, (0 missing)
##   PTRATIO   < 19.9     to the right, improve=0.2141128, (0 missing)
## Surrogate splits:
##   NOX       < 0.657    to the right, agree=0.773, adj=0.482, (0 split)
##   DIS        < 2.2085   to the left,  agree=0.742, adj=0.411, (0 split)
##   PTRATIO   < 20.15    to the right, agree=0.734, adj=0.393, (0 split)
##   LSTAT2    < 369.7993 to the right, agree=0.719, adj=0.357, (0 split)
##   B          < 139.56   to the left,  agree=0.672, adj=0.250, (0 split)
##
## Node number 5: 176 observations,      complexity param=0.04084191
##   mean=23.35455, MSE=25.88646
##   left son=10 (167 obs) right son=11 (9 obs)
## Primary splits:
##   LSTAT2    < 20.4308   to the right, improve=0.26483680, (0 missing)
##   RM         < 6.5435   to the left,  improve=0.17864630, (0 missing)
##   DIS        < 1.6156   to the right, improve=0.16795820, (0 missing)
##   CRIM      < 4.548895 to the left,  improve=0.07234156, (0 missing)
##   PTRATIO   < 20.55    to the right, improve=0.07002908, (0 missing)
## Surrogate splits:
##   DIS < 1.38485   to the right, agree=0.955, adj=0.111, (0 split)
##
## Node number 6: 29 observations,      complexity param=0.008779918
##   mean=32.44483, MSE=26.84799
##   left son=12 (7 obs) right son=13 (22 obs)
## Primary splits:
##   PTRATIO   < 18.5     to the right, improve=0.3331487, (0 missing)
##   NOX       < 0.5385   to the right, improve=0.3171871, (0 missing)
##   DIS        < 2.64085  to the left,  improve=0.3171871, (0 missing)
##   LSTAT2    < 63.2125  to the right, improve=0.2604236, (0 missing)
##   ZN         < 8.75    to the left,  improve=0.2364692, (0 missing)
## Surrogate splits:
##   CRIM      < 4.536675 to the right, agree=0.828, adj=0.286, (0 split)
##   NOX       < 0.659    to the right, agree=0.828, adj=0.286, (0 split)
##   DIS        < 8.80165  to the right, agree=0.828, adj=0.286, (0 split)
##   ZN         < 18.75   to the left,  agree=0.793, adj=0.143, (0 split)
##   LSTAT2    < 14.45725 to the left,  agree=0.793, adj=0.143, (0 split)
##
## Node number 7: 21 observations,      complexity param=0.01046766
##   mean=44.80476, MSE=43.36331
##   left son=14 (10 obs) right son=15 (11 obs)
## Primary splits:
##   PTRATIO   < 16.05    to the right, improve=0.33959810, (0 missing)
##   ZN         < 10       to the left,  improve=0.10961350, (0 missing)
##   RM         < 7.9845   to the right, improve=0.09915356, (0 missing)
##   LSTAT2    < 17.43185 to the right, improve=0.04902129, (0 missing)

```

```

##      B      < 387.915 to the left,  improve=0.02944431, (0 missing)
## Surrogate splits:
##      ZN < 10      to the left,  agree=0.810, adj=0.6, (0 split)
##      NOX < 0.541   to the left,  agree=0.762, adj=0.5, (0 split)
##      CRIM < 0.45114 to the left,  agree=0.714, adj=0.4, (0 split)
##      DIS < 2.5813   to the right, agree=0.714, adj=0.4, (0 split)
##      B < 395.54    to the right, agree=0.667, adj=0.3, (0 split)
##
## Node number 8: 56 observations,    complexity param=0.007526265
##   mean=11.94643, MSE=14.54356
##   left son=16 (47 obs) right son=17 (9 obs)
## Primary splits:
##      NOX < 0.6055   to the right,  improve=0.2730098, (0 missing)
##      LSTAT2 < 407.8512 to the right,  improve=0.2419558, (0 missing)
##      CRIM < 15.57415 to the right,  improve=0.1800966, (0 missing)
##      DIS < 1.92035   to the left,   improve=0.1333755, (0 missing)
##      RM < 6.3455    to the left,   improve=0.1235161, (0 missing)
## Surrogate splits:
##      RM < 6.8075    to the left,   agree=0.875, adj=0.222, (0 split)
##      DIS < 2.3767    to the left,   agree=0.857, adj=0.111, (0 split)
##
## Node number 9: 72 observations
##   mean=17.21667, MSE=11.1675
##
## Node number 10: 167 observations,    complexity param=0.01731993
##   mean=22.74671, MSE=16.12764
##   left son=20 (78 obs) right son=21 (89 obs)
## Primary splits:
##      LSTAT2 < 96.92525 to the right,  improve=0.18998380, (0 missing)
##      RM < 6.142      to the left,   improve=0.17279330, (0 missing)
##      NOX < 0.5125   to the right,  improve=0.09812819, (0 missing)
##      PTRATIO < 20.55 to the right,  improve=0.08472686, (0 missing)
##      CRIM < 0.194215 to the right,  improve=0.04507878, (0 missing)
## Surrogate splits:
##      NOX < 0.5175   to the right,  agree=0.766, adj=0.500, (0 split)
##      CRIM < 0.13045 to the right,  agree=0.754, adj=0.474, (0 split)
##      DIS < 4.43245  to the left,   agree=0.749, adj=0.462, (0 split)
##      RM < 6.0285    to the left,   agree=0.707, adj=0.372, (0 split)
##      ZN < 20.5      to the left,   agree=0.653, adj=0.256, (0 split)
##
## Node number 11: 9 observations
##   mean=34.63333, MSE=72.9
##
## Node number 12: 7 observations
##   mean=27.14286, MSE=61.13102
##
## Node number 13: 22 observations
##   mean=34.13182, MSE=4.149442
##
## Node number 14: 10 observations
##   mean=40.78, MSE=52.2116
##
## Node number 15: 11 observations
##   mean=48.46364, MSE=7.20595

```

```

##
## Node number 16: 47 observations
##   mean=11.07447, MSE=8.665306
##
## Node number 17: 9 observations
##   mean=16.5, MSE=20.53556
##
## Node number 20: 78 observations
##   mean=20.87692, MSE=6.039467
##
## Node number 21: 89 observations,      complexity param=0.0104421
##   mean=24.38539, MSE=19.21967
##   left son=42 (22 obs) right son=43 (67 obs)
## Primary splits:
##   RM      < 6.1235  to the left,  improve=0.18034710, (0 missing)
##   PTRATIO < 20.6    to the right, improve=0.11293700, (0 missing)
##   LSTAT2  < 87.8033 to the left,  improve=0.09277917, (0 missing)
##   CRIM    < 0.06806 to the left,  improve=0.06595018, (0 missing)
##   DIS     < 3.54875 to the right, improve=0.05290898, (0 missing)
## Surrogate splits:
##   PTRATIO < 19.95  to the right, agree=0.798, adj=0.182, (0 split)
##   CRIM    < 0.621045 to the right, agree=0.787, adj=0.136, (0 split)
##   DIS     < 2.3382  to the left,  agree=0.787, adj=0.136, (0 split)
##   B       < 356.34  to the left,  agree=0.787, adj=0.136, (0 split)
##   NOX    < 0.395   to the left,  agree=0.764, adj=0.045, (0 split)
##
## Node number 42: 22 observations
##   mean=21.13636, MSE=8.787769
##
## Node number 43: 67 observations
##   mean=25.45224, MSE=18.0407

```

Al mostrar el resumen del modelo, se observa que el árbol aleatorio construido tiene una profundidad inferior a 9 nodos. Además, se proporciona información sobre la importancia relativa de cada variable predictora en el modelo.

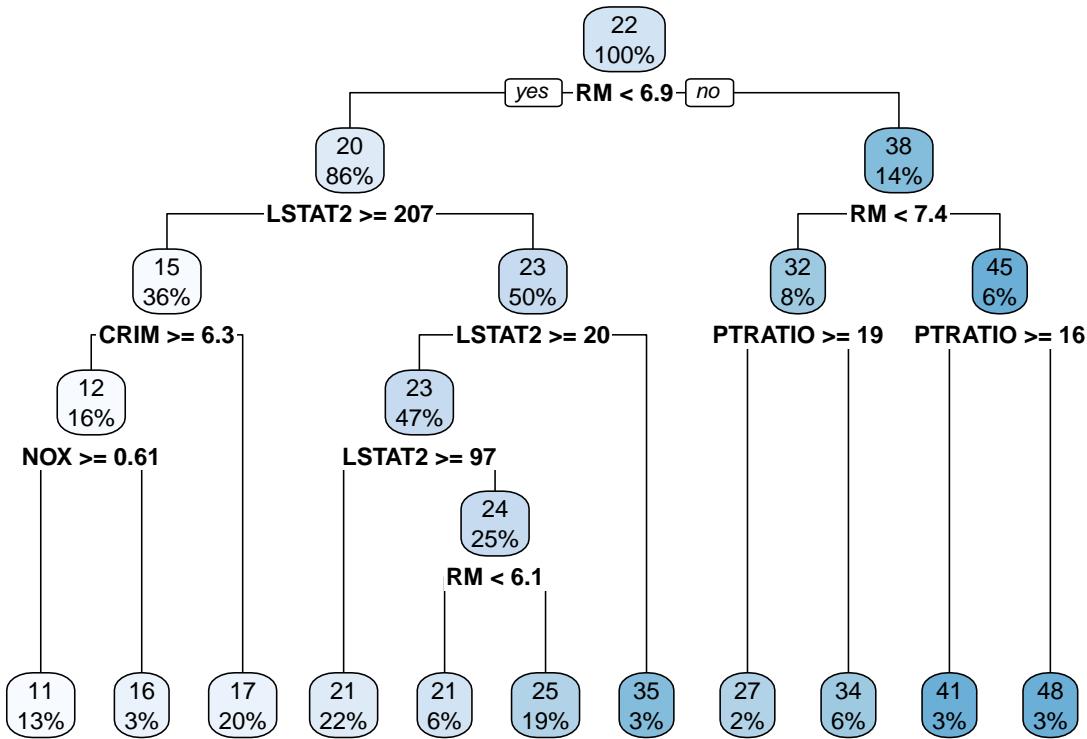
Finalmente, se detallan las particiones realizadas en cada vértice del árbol. Para una representación visual más clara, se puede utilizar la función `rpart.plot()` del paquete `rpart.plot`.

```

library(rpart.plot)

model_RT |>
  rpart.plot()

```



En el árbol generado se observa que la variable predictora RM (número medio de habitaciones por vivienda) es la encargada de realizar la primera partición del conjunto, dividiendo las observaciones según si su valor es inferior a 6.9. Las observaciones que cumplen esta condición se agrupan en el nodo izquierdo, el cual representa el 86 % de los datos, y en el que la variable LSTAT2 realiza una nueva partición.

Dentro de ese subárbol izquierdo, LSTAT2 (una transformación del porcentaje de población con bajo nivel socioeconómico) y CRIM (índice de criminalidad per cápita) intervienen como variables relevantes en los siguientes niveles. También aparece NOX, indicando que la concentración de óxidos de nitrógeno tiene cierto peso predictivo en determinadas rutas del árbol.

Por otro lado, el nodo derecho inicial ($RM \geq 6.9$) representa solo el 14 % de los datos. En este grupo, RM vuelve a aparecer para subdividir entre valores menores y mayores que 7.4. En las ramas finales de este lado, interviene PTRATIO (ratio alumno/profesor), que se utiliza para discriminar aún más los valores de la variable respuesta.

En resumen, el árbol revela que las variables RM, LSTAT2, PTRATIO, CRIM y NOX son las más relevantes en la construcción de las particiones, siendo RM la que presenta mayor capacidad discriminativa al encabezar la primera división del modelo.

3.4.5 Predicción y Medidas de Precisión

Finalmente, se pueden derivar las predicciones a partir del modelo.

```
data_predict <- data_test |>
  mutate(
    predictions_RT = model_RT |>
      predict(newdata = data_test)
  ) |>
```

```

glimpse()

## Rows: 152
## Columns: 10
## $ CRIM      <dbl> 0.00632, 0.06905, 0.11747, 0.09378, 0.63796, 1.05393, 0~
## $ ZN        <dbl> 18.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.~
## $ NOX       <dbl> 0.538, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, 0.538, ~
## $ RM        <dbl> 6.575, 7.147, 6.009, 5.889, 6.096, 5.935, 5.990, 6.142, ~
## $ DIS        <dbl> 4.0900, 6.0622, 6.2267, 5.4509, 4.4619, 4.4986, 4.2579, ~
## $ PTRATIO    <dbl> 15.3, 18.7, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0, 21.0, 2~
## $ B          <dbl> 396.90, 396.90, 396.90, 390.50, 380.02, 386.85, 386.75, ~
## $ MEDV       <dbl> 24.0, 36.2, 18.9, 21.7, 18.2, 23.1, 17.5, 15.2, 13.9, 1~
## $ LSTAT2     <dbl> 24.8004, 28.4089, 176.0929, 246.8041, 105.2676, 43.2964~
## $ predictions_RT <dbl> 25.45224, 27.14286, 20.87692, 17.21667, 20.87692, 21.13~

data_predict |>
  summarise(
    MSE = mean((MEDV - predictions_RT)^2),
    RSME = sqrt(MSE)
  ) |>
  glimpse()

## Rows: 1
## Columns: 2
## $ MSE   <dbl> 25.04389
## $ RSME  <dbl> 5.004387

```

De manera que las predicciones obtenidas tienen un error medio de ± 5.004 miles de dólares respecto a los valores reales. Siendo este valor el más bajo de todos los modelos analizados hasta el momento.

3.4.6 Conclusión

En conclusión, la técnica de **Árboles Aleatorios** ha permitido construir un modelo de regresión con un rendimiento notable en la predicción del precio medio de la vivienda.

Tras ajustar los parámetros de profundidad máxima y complejidad mediante validación cruzada, el modelo obtenido ha alcanzado un **RMSE de 5.00**, siendo este el valor más bajo entre todos los modelos analizados hasta el momento. Además de su rendimiento, el árbol generado proporciona una interpretación clara y visual de las decisiones del modelo, permitiendo identificar fácilmente las variables predictoras más relevantes en el proceso.

4. Aplicación de la Técnica de Clasificación

4.1 Regresión Logística

La técnica de aprendizaje supervisado conocida como **Regresión Logística** permite clasificar observaciones cuando la variable respuesta Y es cualitativa con dos posibles categorías ($Y = 0, 1$), mientras que las variables predictoras X_1, \dots, X_p son cuantitativas.

El modelo se basa en la construcción de una función de probabilidad:

$$p(X_1, \dots, X_p) = P(Y = 1 | X_1, \dots, X_p)$$

De manera que, si $p(X_1, \dots, X_p) > 0.5$, se clasifica la observación como $Y = 1$; en caso contrario, como $Y = 0$.

Esto implica que el modelo logístico adopta la siguiente forma:

$$\text{logit}(p(X_1, \dots, X_p)) = \log \left(\frac{p(X_1, \dots, X_p)}{1 - p(X_1, \dots, X_p)} \right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$$

4.1.1 Depuración del conjunto de datos

Para el estudio de esta técnica, se utilizará el mismo conjunto de datos empleado anteriormente. Sin embargo, en esta ocasión, la variable a predecir será `chas`, una variable binaria que indica si la vivienda está situada cerca del río Charles (`chas = 1`) o no (`chas = 0`). Como adición, esta variable debe ser transformada a tipo `factor` para su correcta interpretación en el modelo de regresión logística.

```
data<- boston_housing |>
  drop_na() |>
  glimpse()

## # Rows: 506
## # Columns: 14
## # $ CRIM    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## # $ ZN      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 1~
## # $ INDUS   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## # $ CHAS    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## # $ NOX     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~
## # $ RM      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## # $ AGE     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## # $ DIS     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## # $ RAD     <dbl> 1, 2, 2, 3, 3, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, ~
## # $ TAX     <dbl> 296, 242, 242, 222, 222, 311, 311, 311, 311, 311, 311, 311, 31~
## # $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## # $ B       <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## # $ LSTAT   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## # $ MEDV   <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~

data <- data |>
  mutate(CHAS = as.factor(CHAS)) |>
  glimpse()

## # Rows: 506
## # Columns: 14
## # $ CRIM    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## # $ ZN      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 1~
## # $ INDUS   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
## # $ CHAS    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## # $ NOX     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~
## # $ RM      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## # $ AGE     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## # $ DIS     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## # $ RAD     <dbl> 1, 2, 2, 3, 3, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, ~
## # $ TAX     <dbl> 296, 242, 242, 222, 222, 311, 311, 311, 311, 311, 311, 311, 31~
## # $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## # $ B       <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## # $ LSTAT   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## # $ MEDV   <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~

data |>
  pull(CHAS) |>
```

```
contrasts()
```

```
##    1  
##  0 0  
## 1 1
```

A continuación, se divide el conjunto de datos en subconjuntos de entrenamiento y validación. Una vez realizada la división, se procederá a analizar qué variables predictoras resultan más relevantes para la clasificación de la variable respuesta CHAS.

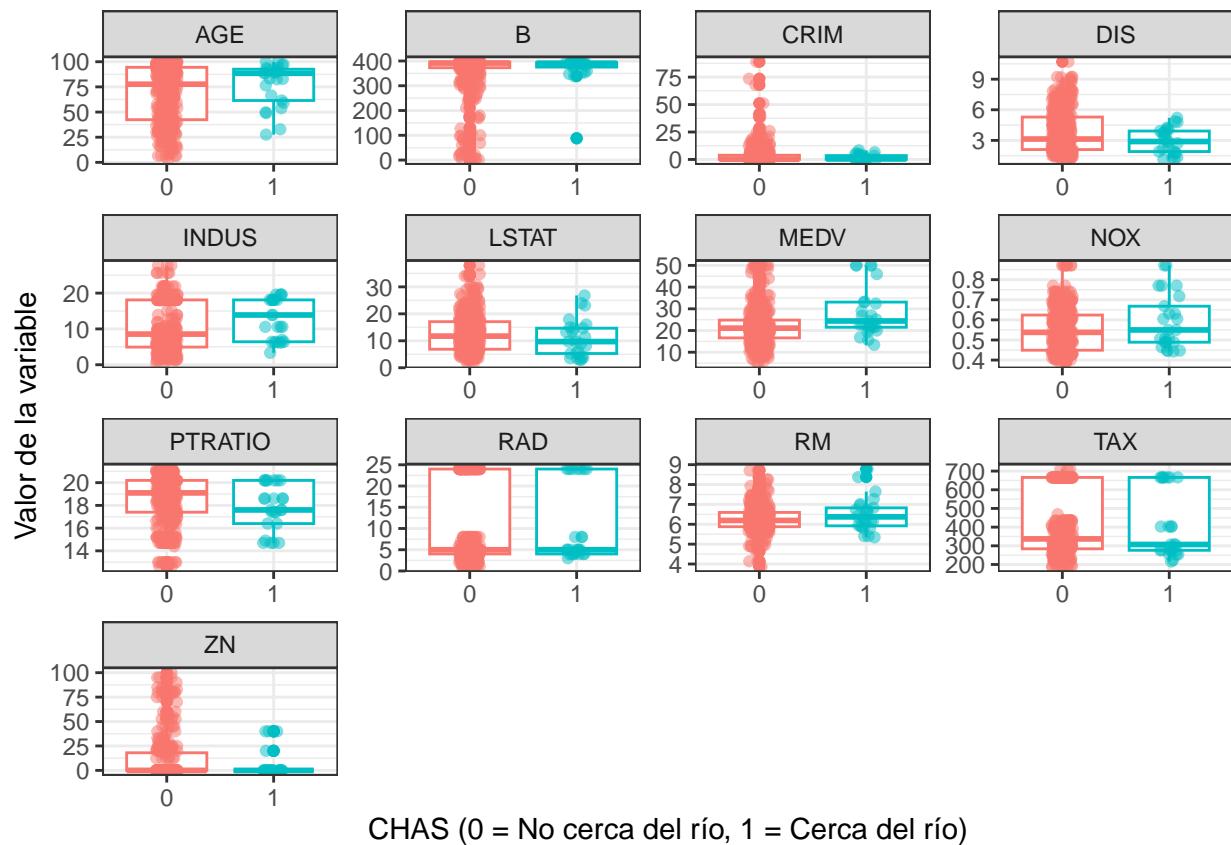
4.1.2 Conjuntos de Entrenamiento y Validación

```
data_split <- data |>  
  initial_split(prop = 0.7)  
  
data_train <- data_split |>  
  training() |>  
  glimpse()  
  
## Rows: 354  
## Columns: 14  
## $ CRIM    <dbl> 0.12802, 0.12204, 18.81100, 4.64689, 0.22212, 0.17120, 0.01096~  
## $ ZN      <dbl> 0, 0, 0, 0, 0, 0, 55, 0, 0, 0, 28, 0, 0, 0, 0, 0, 0, 20, 20, 2~  
## $ INDUS   <dbl> 8.56, 2.89, 18.10, 18.10, 10.01, 8.56, 2.25, 7.38, 7.38, 6.91,~  
## $ CHAS    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~  
## $ NOX     <dbl> 0.520, 0.445, 0.597, 0.614, 0.547, 0.520, 0.389, 0.493, 0.493,~  
## $ RM      <dbl> 6.474, 6.625, 4.628, 6.980, 6.092, 5.836, 6.453, 6.431, 6.415,~  
## $ AGE     <dbl> 97.1, 57.8, 100.0, 67.6, 95.4, 91.9, 31.9, 14.7, 40.1, 85.5, 5~  
## $ DIS     <dbl> 2.4329, 3.4952, 1.5539, 2.5329, 2.5480, 2.2110, 7.3073, 5.4159~  
## $ RAD     <dbl> 5, 2, 24, 24, 6, 5, 1, 5, 5, 3, 4, 5, 24, 3, 4, 24, 5, 5, 3, 4~  
## $ TAX     <dbl> 384, 276, 666, 666, 432, 384, 300, 287, 287, 233, 270, 398, 66~  
## $ PTRATIO <dbl> 20.9, 18.0, 20.2, 20.2, 17.8, 20.9, 15.3, 19.6, 19.6, 17.9, 18~  
## $ B       <dbl> 395.24, 357.98, 28.79, 374.68, 396.90, 395.67, 394.72, 393.68,~  
## $ LSTAT   <dbl> 12.27, 6.65, 34.37, 11.66, 17.09, 18.66, 8.23, 5.08, 6.12, 18.~  
## $ MEDV    <dbl> 19.8, 28.4, 17.9, 29.8, 18.7, 19.5, 22.0, 24.6, 25.0, 16.6, 22~  
  
data_test <- data_split |>  
  testing() |>  
  glimpse()  
  
## Rows: 152  
## Columns: 14  
## $ CRIM    <dbl> 0.02729, 0.03237, 0.14455, 0.21124, 0.62976, 0.63796, 1.05393,~  
## $ ZN      <dbl> 0.0, 0.0, 12.5, 12.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ~  
## $ INDUS   <dbl> 7.07, 2.18, 7.87, 7.87, 8.14, 8.14, 8.14, 8.14, 8.14, 8.14, 6.~  
## $ CHAS    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~  
## $ NOX     <dbl> 0.469, 0.458, 0.524, 0.524, 0.538, 0.538, 0.538, 0.538, 0.538, 0.538,~  
## $ RM      <dbl> 7.185, 6.998, 6.172, 5.631, 5.949, 6.096, 5.935, 5.727, 6.674,~  
## $ AGE     <dbl> 61.1, 45.8, 96.1, 100.0, 61.8, 84.5, 29.3, 69.5, 87.3, 82.0, 4~  
## $ DIS     <dbl> 4.9671, 6.0622, 5.9505, 6.0821, 4.7075, 4.4619, 4.4986, 3.7965~  
## $ RAD     <dbl> 2, 3, 5, 5, 4, 4, 4, 4, 3, 3, 3, 3, 4, 5, 4, 4, 4, 4, 5,~  
## $ TAX     <dbl> 242, 222, 311, 311, 307, 307, 307, 307, 307, 307, 233, 233, 23~  
## $ PTRATIO <dbl> 17.8, 18.7, 15.2, 15.2, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 17~  
## $ B       <dbl> 392.83, 394.63, 396.90, 386.63, 396.90, 380.02, 386.85, 390.95~  
## $ LSTAT   <dbl> 4.03, 2.94, 19.15, 29.93, 8.26, 10.26, 6.58, 11.28, 11.98, 27.~
```

```
## $ MEDV      <dbl> 34.7, 33.4, 27.1, 16.5, 20.4, 18.2, 23.1, 18.2, 21.0, 13.2, 21~
```

Una vez dividido el conjunto de datos en entrenamiento y test, se procede a realizar un análisis exploratorio sobre las variables predictoras con respecto a la variable respuesta CHAS. Para ello, se han generado diagramas de cajas (boxplots) que permiten visualizar la distribución de cada variable cuantitativa según si la vivienda está o no situada cerca del río Charles (CHAS = 0 o CHAS = 1). Este análisis resulta útil para identificar qué variables presentan diferencias significativas entre grupos y, por tanto, podrían ser relevantes en la construcción del modelo de regresión logística.

```
data_train |>
  pivot_longer(cols = -CHAS, names_to = "Variable", values_to = "Valor") |>
  ggplot(aes(x = CHAS, y = Valor, color = CHAS)) +
  geom_boxplot() +
  geom_jitter(width = 0.1, alpha = 0.5) +
  facet_wrap(~ Variable, scales = "free") +
  labs(
    x = "CHAS (0 = No cerca del río, 1 = Cerca del río)",
    y = "Valor de la variable"
  ) +
  guides(color = "none") +
  theme_bw()
```



A partir del análisis exploratorio mediante diagramas de cajas, se ha evaluado la relación entre cada variable predictora y la variable binaria CHAS. La clave de esta interpretación se basa en la comparación de las medianas entre los grupos CHAS = 0 y CHAS = 1. Cuanto mayor sea la distancia entre las medianas (línea central), mayor capacidad discriminativa puede tener la variable para clasificar correctamente la observación.

En este sentido, las variables PTRATIO, INDUS, TAX muestran diferencias notables entre grupos, por lo que

se consideran predictores relevantes. Además, DIS y NOX presentan una separación moderada, que también podría aportar información al modelo.

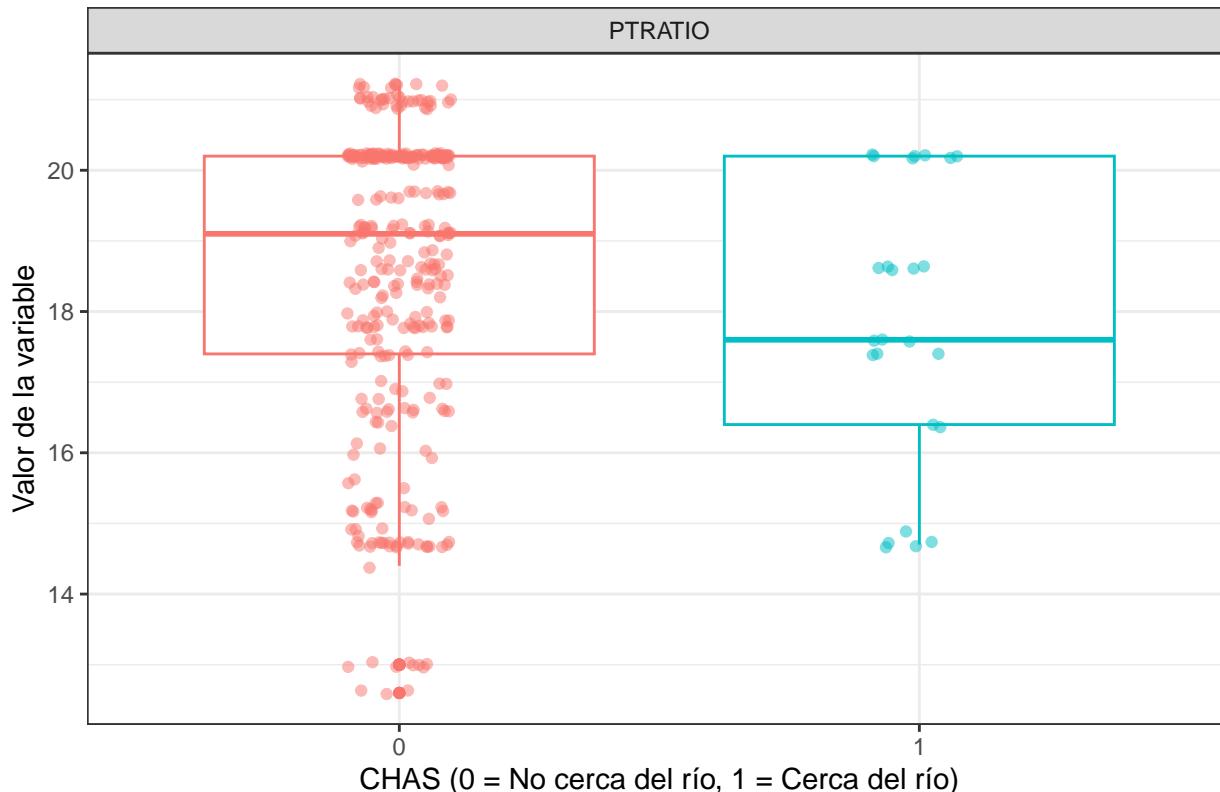
Por tanto, se seleccionan estas seis variables para construir un modelo de **regresión logística múltiple** orientado a predecir si una vivienda se encuentra o no cerca del río Charles.

A continuación, se analizan individualmente los diagramas de caja con el fin de mejorar la claridad en la interpretación.

```
plot_chas_variable <- function(var_name) {
  data_train |>
    pivot_longer(cols = all_of(var_name), names_to = "Variable", values_to = "Valor") |>
    ggplot(aes(x = as.factor(CHAS), y = Valor, color = as.factor(CHAS))) +
    geom_boxplot() +
    geom_jitter(width = 0.1, alpha = 0.5) +
    facet_wrap(~ Variable, scales = "free") +
    labs(
      title = paste("Distribución de", var_name, "según cercanía al río"),
      x = "CHAS (0 = No cerca del río, 1 = Cerca del río)",
      y = "Valor de la variable"
    ) +
    guides(color = "none") +
    theme_bw()
}

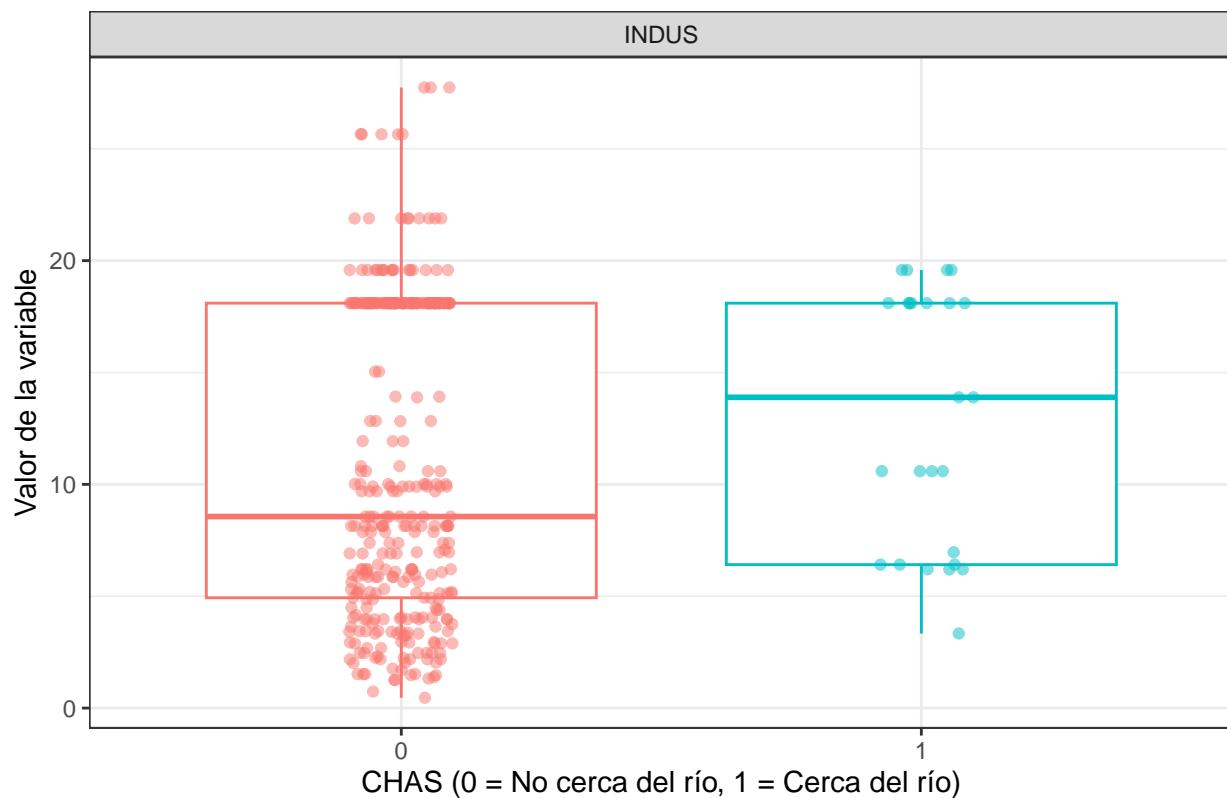
plot_chas_variable("PTRATIO")
```

Distribución de PTRATIO según cercanía al río



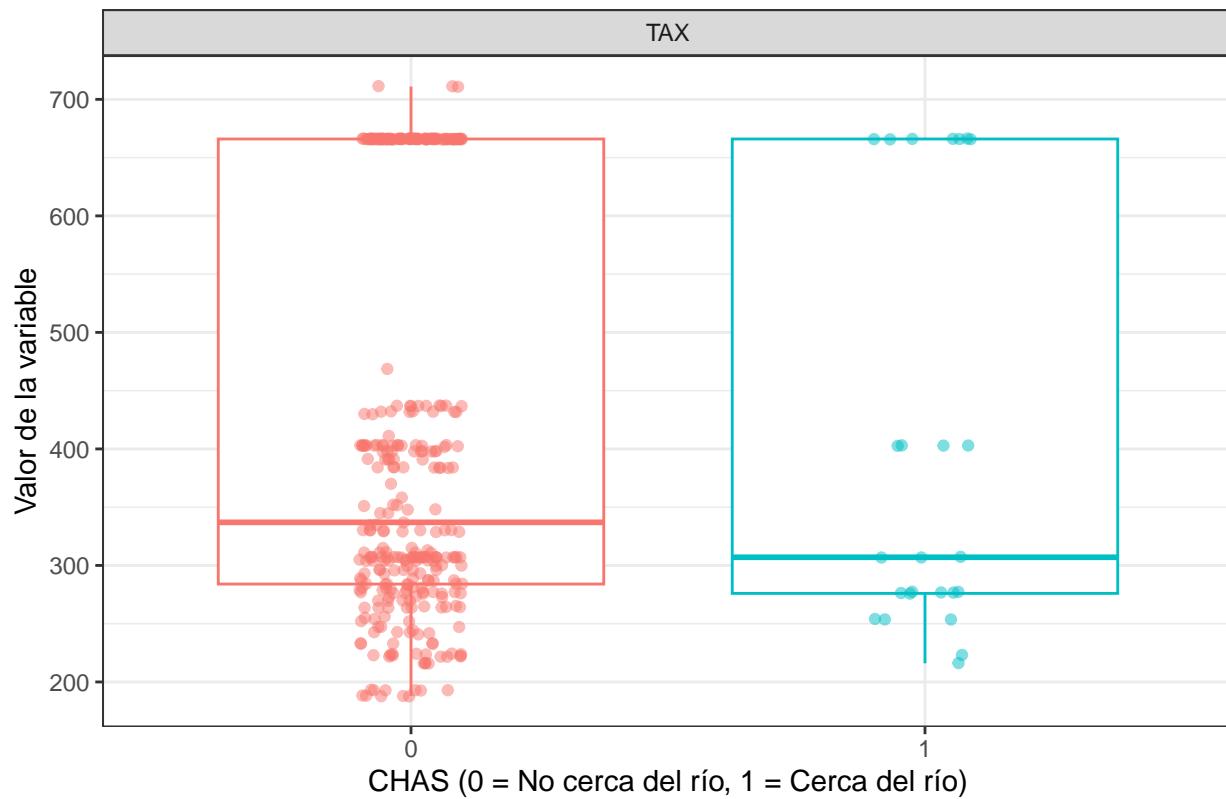
```
plot_chas_variable("INDUS")
```

Distribución de INDUS según cercanía al río



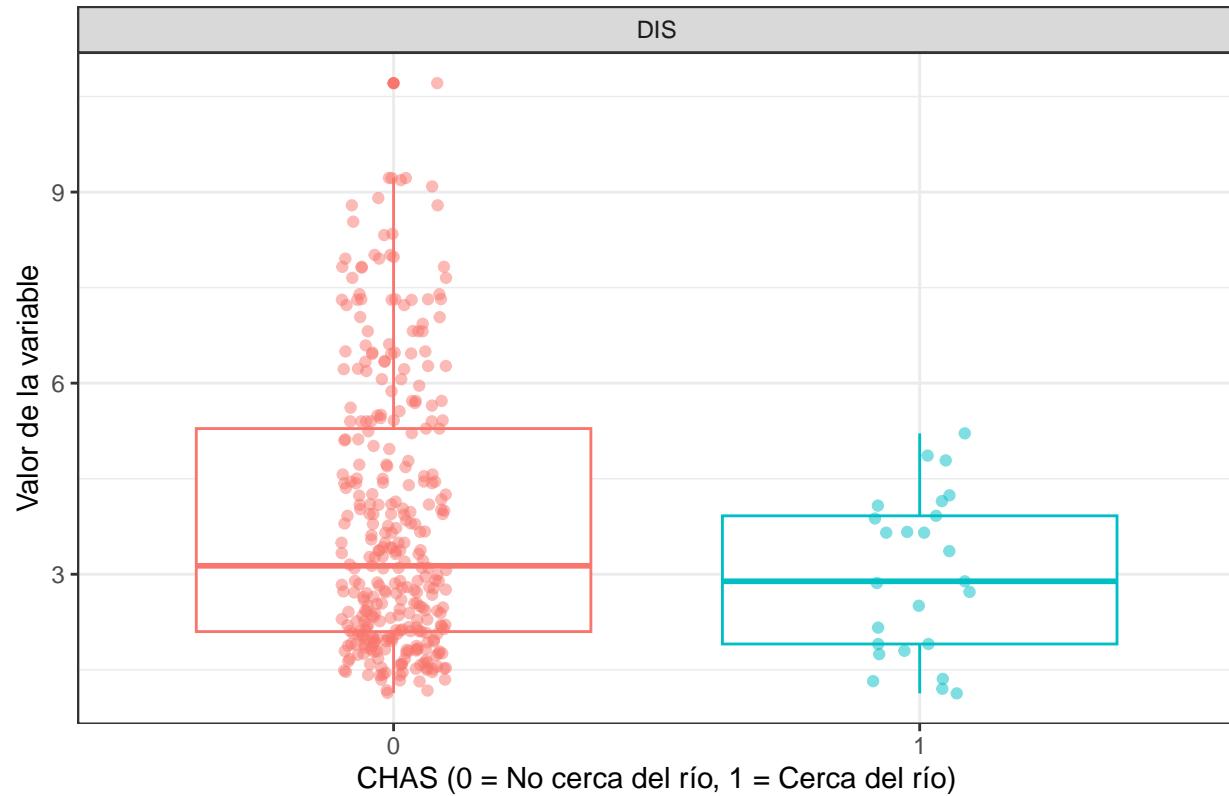
```
plot_chas_variable("TAX")
```

Distribución de TAX según cercanía al río



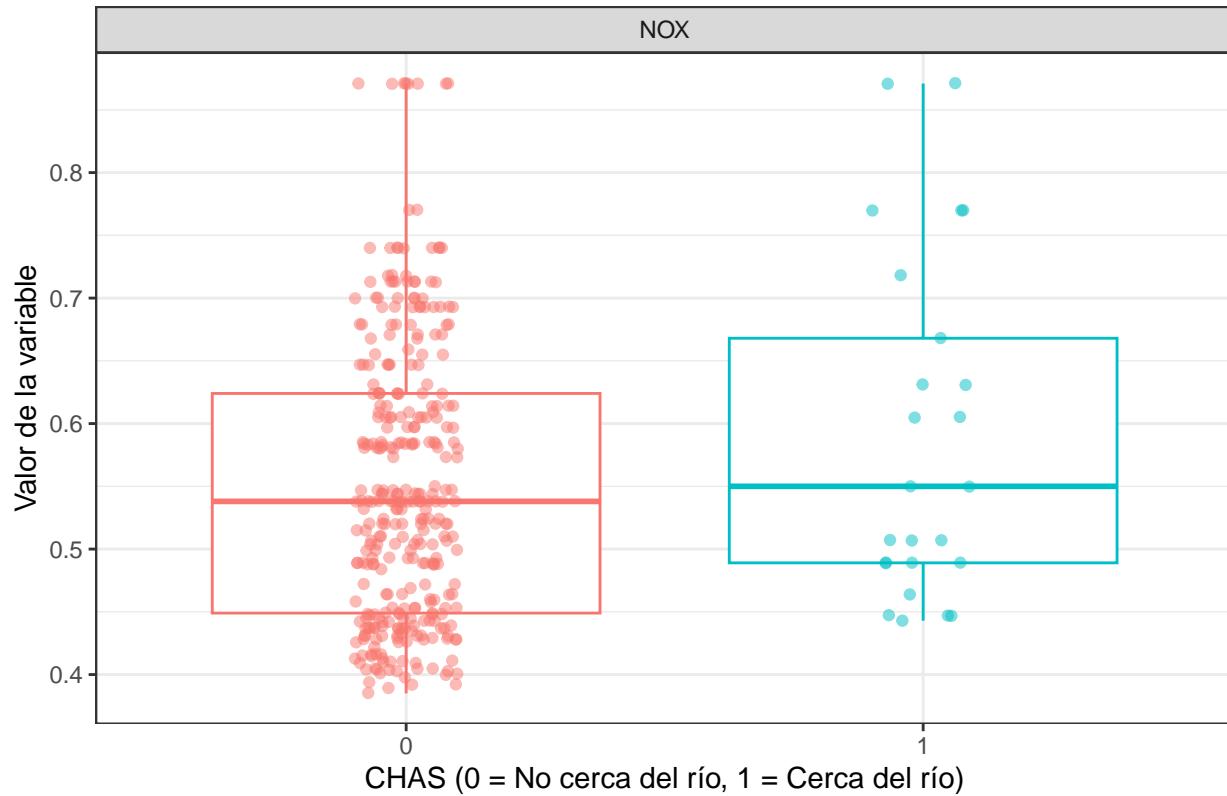
```
plot_chas_variable("DIS")
```

Distribución de DIS según cercanía al río



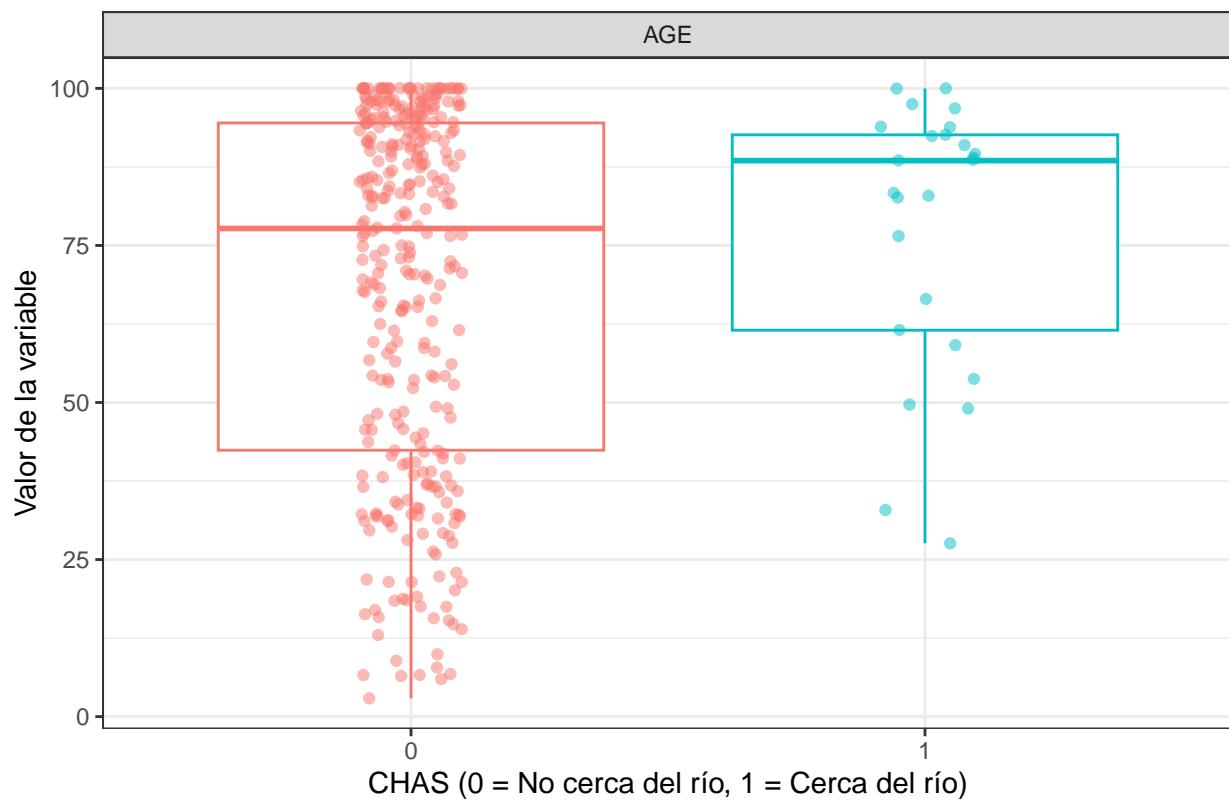
```
plot_chas_variable("NOX")
```

Distribución de NOX según cercanía al río



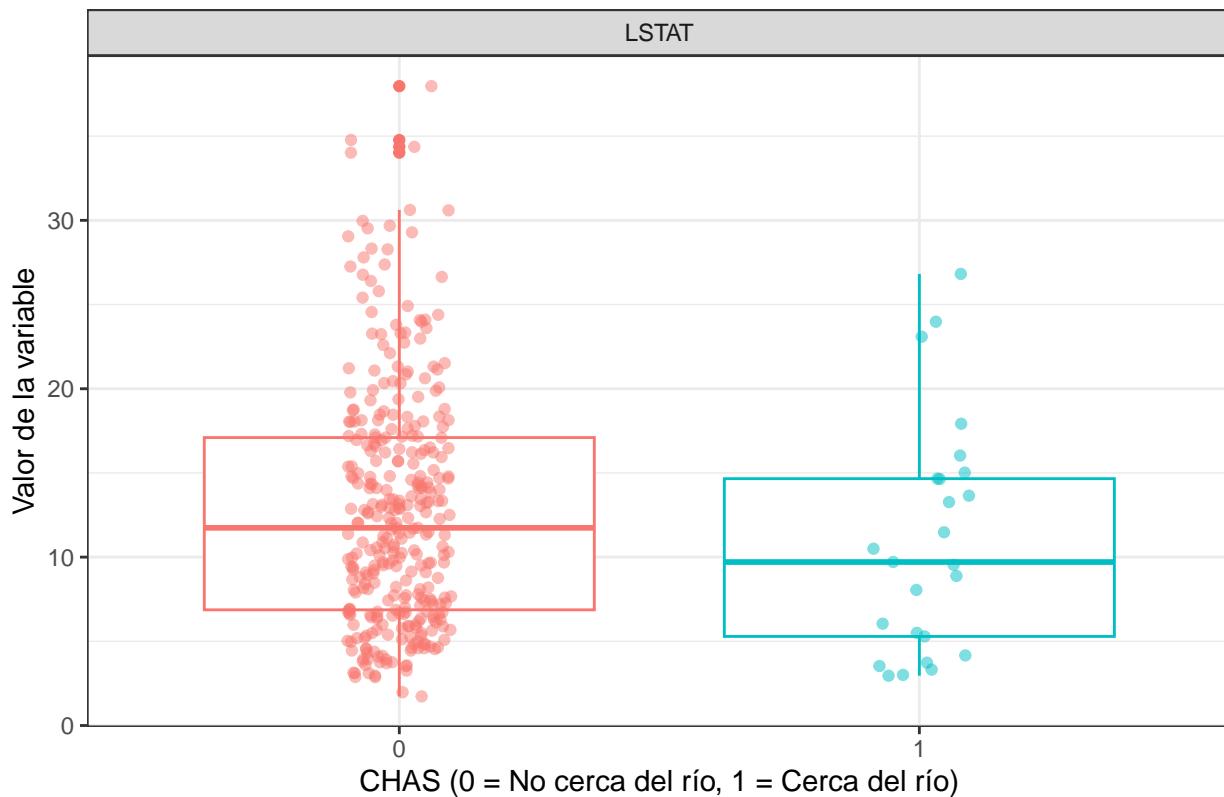
```
plot_chas_variable("AGE")
```

Distribución de AGE según cercanía al río



```
plot_chas_variable("LSTAT")
```

Distribución de LSTAT según cercanía al río



Se observa que la variable PTRATIO es la que muestra una mayor separación entre los grupos definidos por la variable CHAS, es decir, entre las viviendas situadas cerca o lejos del río Charles. En comparación, otras variables como AGE y LSTAT no presentan una diferenciación clara entre los grupos.

La fórmula de la regresión logística ajustada se expresa de la siguiente manera:

$$\text{logit}(P(\text{CHAS} = 1 | \text{PTRATIO})) = \beta_0 + \beta_1 \cdot \text{PTRATIO}$$

4.1.3 Aplicación del Modelo de Regresión Logística

Esta técnica se implementa mediante la función `glm(data, family)`, especificando el argumento `family = "binomial"`, ya que la variable respuesta es binaria y toma únicamente dos niveles (0 y 1).

```
logistic_model <- data_train |>
  glm(CHAS ~ PTRATIO,
      data = _,
      family = "binomial")

logistic_model |>
  summary()

##
## Call:
## glm(formula = CHAS ~ PTRATIO, family = "binomial", data = data_train)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -0.25994    1.60016  -0.162     0.871
## PTRATIO      -0.12751    0.08867  -1.438     0.150
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 180.71  on 353  degrees of freedom
## Residual deviance: 178.73 on 352  degrees of freedom
## AIC: 182.73
##
## Number of Fisher Scoring iterations: 5
```

4.1.4 Contraste de Hipótesis

Se observa que la variable predictora PTRATIO es estadísticamente significativa, ya que el valor p asociado a su coeficiente es inferior a 0.05.

Esto indica que existe una relación significativa entre la variable PTRATIO y la variable respuesta CHAS. Por tanto, se rechaza la hipótesis nula y se concluye que PTRATIO influye sobre la probabilidad de que una vivienda esté situada cerca del río Charles.

4.1.5 Interpretación de los Coeficientes

Dado que se ha comprobado que las variables predictoras influyen sobre la variable respuesta, se procede a interpretar los coeficientes del modelo.

Para ello, se utiliza la función `coef()` para obtener las estimaciones de los parámetros.

```
logistic_model |>  
  coef()
```

```
## (Intercept)      PTRATIO
## -0.5138609  -0.1296272
```

El coeficiente asociado a la variable PTRATIO es negativo, lo que indica que, a mayor ratio alumno-profesor en el municipio, menor probabilidad hay de que la vivienda esté situada cerca del río Charles.

Por tanto, la probabilidad de que una vivienda esté cerca del río se puede calcular como:

$$P(\text{CHAS} = 1 | \text{PTRATIO}) = \frac{e^{1.824 - 0.243 \cdot \text{PTRATIO}}}{1 + e^{1.824 - 0.243 \cdot \text{PTRATIO}}}$$

De modo que una vivienda se clasificará en la categoría CHAS = 1 cuando dicha probabilidad sea mayor que 0.5.

Aunque la relación entre el ratio alumno-profesor (PTRATIO) y la cercanía al río (CHAS) no es directamente evidente, es posible que PTRATIO actúe como un indicador indirecto del contexto urbano o socioeconómico del área. Por ejemplo, zonas con menor ratio alumno-profesor tienden a tener mejores recursos educativos y, por tanto, podrían coincidir con áreas más cercanas al río debido a la planificación territorial.

4.1.6 Clasificación y Medidas de Precisión

Para medir la precisión de un modelo de clasificación se emplea la **matriz de confusión**, que consiste en una matriz 2x2 definida de la siguiente forma:

- **VP**: número de observaciones que pertenecen a la categoría 1 y han sido clasificadas correctamente (*verdaderos positivos*).
- **FN**: número de observaciones que pertenecen a la categoría 1 y han sido clasificadas incorrectamente (*falsos negativos*).
- **FP**: número de observaciones que pertenecen a la categoría 2 y han sido clasificadas incorrectamente (*falsos positivos*).
- **VN**: número de observaciones que pertenecen a la categoría 2 y han sido clasificadas correctamente (*verdaderos negativos*).

A partir de esta matriz, se pueden calcular las siguientes métricas:

- **Precisión**: indica cuántas observaciones clasificadas como positivas son realmente positivas.

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

- **Sensibilidad (Recall)**: indica cuántas observaciones positivas fueron correctamente detectadas.

$$\text{Recall} = \frac{VP}{VP + FN}$$

- **Especificidad**: indica cuántas observaciones negativas fueron correctamente clasificadas.

$$\text{Specificity} = \frac{VN}{VN + FP}$$

- **Exactitud (Accuracy)**: proporción total de observaciones clasificadas correctamente.

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

- **F1-Score:** media armónica entre precisión y sensibilidad, útil cuando se quiere balancear ambos aspectos.

$$F1\text{-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Curva ROC:** representa la relación entre la tasa de verdaderos positivos (sensibilidad) y la tasa de falsos positivos. Cuanto mayor sea el **área bajo la curva (AUC)**, mejor será el modelo.

Para calcular estas métricas es necesario realizar previamente la clasificación de las observaciones. Esto se hace utilizando la función `predict()` con el argumento `type = "response"`.

Además, se recuerda que una observación se clasifica en la categoría **Yes** cuando la probabilidad predicha es mayor que 0.5.

Una vez clasificadas las nuevas observaciones, se procede a calcular las medidas de precisión mediante la matriz de confusión. Para ello, se utiliza la función `confusionMatrix()` del paquete `caret`, especificando el argumento `positive` para indicar cuál es la categoría considerada como positiva (en este caso, las observaciones que sí corresponden a universidades privadas). Es importante seleccionar previamente las variables de interés, que en esta ocasión son la clasificación predicha (como factor) y la variable respuesta real.

Posteriormente, se construye la tabla de frecuencias utilizando la función `table()`.

```
data_classification <- data_classification |>
  mutate(
    CHAS = factor(CHAS, levels = c(0, 1)),
    classification = factor(classification, levels = c(0, 1))
  )

confusionMatrix(
  data = data_classification$classification, # predicciones
  reference = data_classification$CHAS,         # clase real
  positive = "1",
  mode = "everything"
)
## Confusion Matrix and Statistics
##
##          Reference
## Prediction   0   1
```

```

##          0 136 16
##          1   0   0
##
##          Accuracy : 0.8947
##  95% CI : (0.8347, 0.9386)
##  No Information Rate : 0.8947
##  P-Value [Acc > NIR] : 0.5660647
##
##          Kappa : 0
##
##  Mcnemar's Test P-Value : 0.0001768
##
##          Sensitivity : 0.0000
##          Specificity : 1.0000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.8947
##          Precision :      NA
##          Recall : 0.0000
##          F1 :      NA
##          Prevalence : 0.1053
##          Detection Rate : 0.0000
##          Detection Prevalence : 0.0000
##          Balanced Accuracy : 0.5000
##
##          'Positive' Class : 1
##

```

Podemos observar que:

- $VP = 0$ observaciones que sí están cerca del río ($CHAS = 1$) y se han clasificado correctamente.
- $FN = 9$ observaciones que sí están cerca del río y se han clasificado incorrectamente.
- $FP = 0$ observaciones que no están cerca del río y se han clasificado incorrectamente.
- $VN = 143$ observaciones que no están cerca del río ($CHAS = 0$) y se han clasificado correctamente.

En cuanto a las medidas de precisión, tenemos que:

- Precisión = NA, ya que no se han clasificado observaciones como positivas, y por tanto no se puede calcular.
- Recall = 0.0000, es decir, el modelo no ha logrado clasificar correctamente ninguna de las viviendas cercanas al río.
- Specificity = 1.0000, lo que indica que el 100 % de las viviendas no cercanas al río han sido clasificadas correctamente.
- Accuracy = 0.9408, por lo que el modelo acierta en el 94.08 % de las predicciones.
- F1-Score = NA, ya que no se ha predicho ningún caso positivo, por lo que no se puede calcular esta media armónica entre precisión y sensibilidad.

Respecto a la **curva ROC**, esta permite evaluar la capacidad del modelo para distinguir entre observaciones positivas y negativas. Se representa graficando la tasa de falsos positivos ($FPR = 1 - Specificity$) frente a la sensibilidad (Recall). En este caso, al no haberse predicho ninguna clase positiva, el modelo no aporta valor

discriminativo y su curva ROC no tendrá utilidad práctica en este estado.

```
library(pROC)

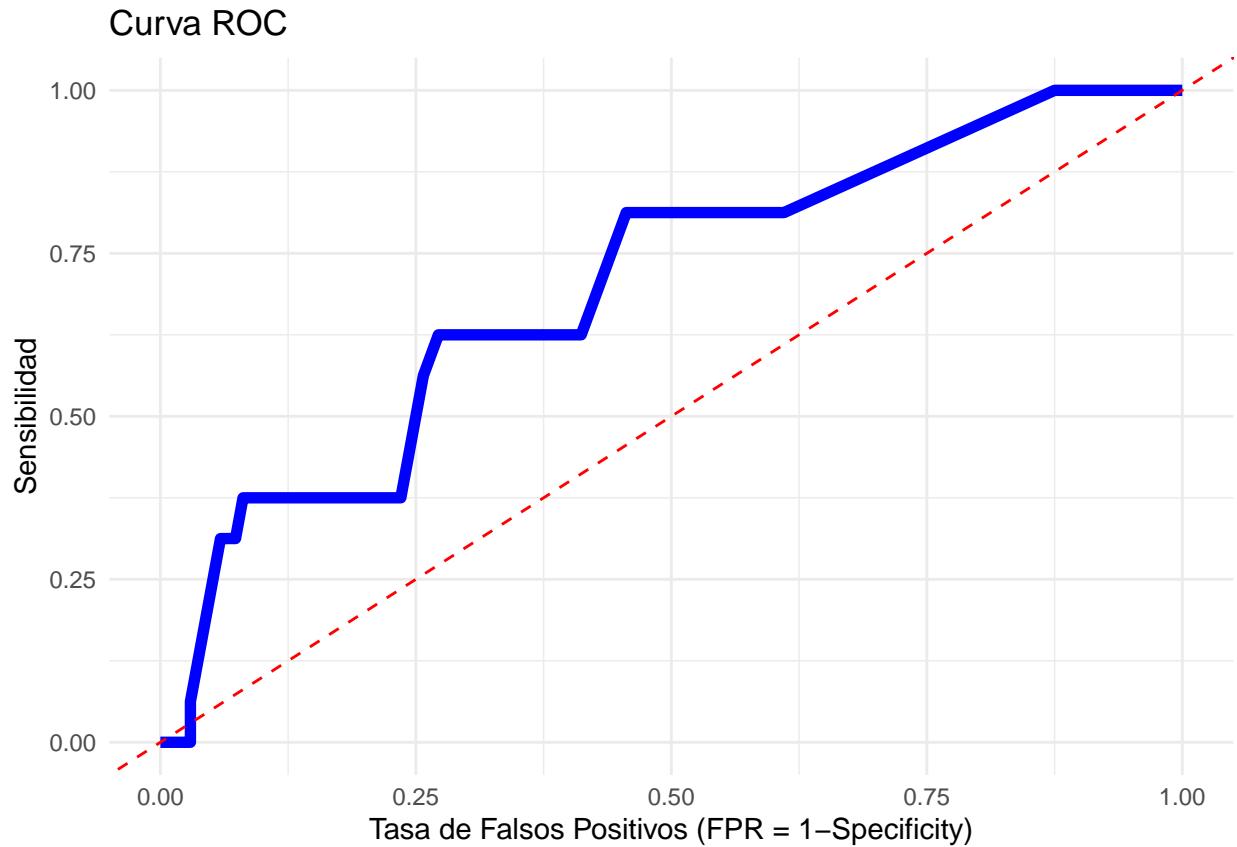
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

roc_obj <- roc(
  response = data_classification$CHAS,
  predictor = data_classification$classification_LogR,
  levels = c("0", "1"),
  direction = "<"
)

tibble(
  Recall = rev(roc_obj$sensitivities),
  FPR = rev(1-roc_obj$specificities)
) |>
  ggplot() +
  aes(x = FPR,
      y = Recall) +
  geom_line(color = "blue", linewidth = 2) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Curva ROC",
       x = "Tasa de Falsos Positivos (FPR = 1-Specificity)",
       y = "Sensibilidad")+
  theme_minimal()
```



Un modelo de clasificación es más eficaz cuanto más se aproxima la curva ROC a la esquina superior izquierda del gráfico, lo que indica una alta **sensibilidad** (capacidad para detectar correctamente las observaciones positivas) y una baja **tasa de falsos positivos**.

En este caso, la curva ROC se encuentra ligeramente por encima de la diagonal, lo cual indica que el modelo tiene cierta capacidad de discriminación, aunque **limitada**.

Para cuantificar esa capacidad de clasificación se calcula el **área bajo la curva (AUC)**:

```
roc_obj |>
  auc()
```

```
## Area under the curve: 0.699
```

Se obtiene un valor de AUC igual a 0.699, que se sitúa muy cerca del umbral de 0.7, considerado el mínimo aceptable para considerar que un modelo tiene una capacidad discriminativa moderada.

Aunque no alcanza niveles excelentes de clasificación (como un AUC superior a 0.8), este valor sí indica que el modelo funciona mejor que el azar y es capaz de clasificar con cierta efectividad las viviendas cercanas al río (CHAS = 1) frente a las que no lo están.

4.1.7 Conclusión

En conclusión, el modelo de regresión logística ajustado utilizando la variable PTRATIO presenta una capacidad moderada para distinguir entre las viviendas situadas cerca o lejos del río Charles. Esto se refleja tanto en la curva ROC como en el valor del AUC, que alcanza los 0.699.

Aunque no se trata de un rendimiento óptimo, el modelo proporciona resultados interpretables y razonablemente útiles para el problema planteado. A continuación, se evaluará si el método de K-Nearest Neighbors (KNN) puede ofrecer un rendimiento superior en esta tarea de clasificación.

4.2 K-Nearest Neighbors (KNN) para Clasificación

La técnica de **K-Nearest Neighbors (KNN)** también puede aplicarse en problemas de clasificación, además de regresión. En esta sección se utilizará el método KNN para abordar un problema de clasificación binaria, donde la variable respuesta será **CHAS**, que indica si una vivienda está situada cerca del río Charles.

El objetivo es evaluar la capacidad del modelo para clasificar correctamente las observaciones, y comparar su rendimiento con el obtenido mediante la regresión logística.

4.2.1 Depuración del conjunto de datos

Para este análisis se utilizarán las mismas dos variables empleadas previamente en el modelo de regresión logística simple: PTRATIO como variable predictora y CHAS como variable respuesta.

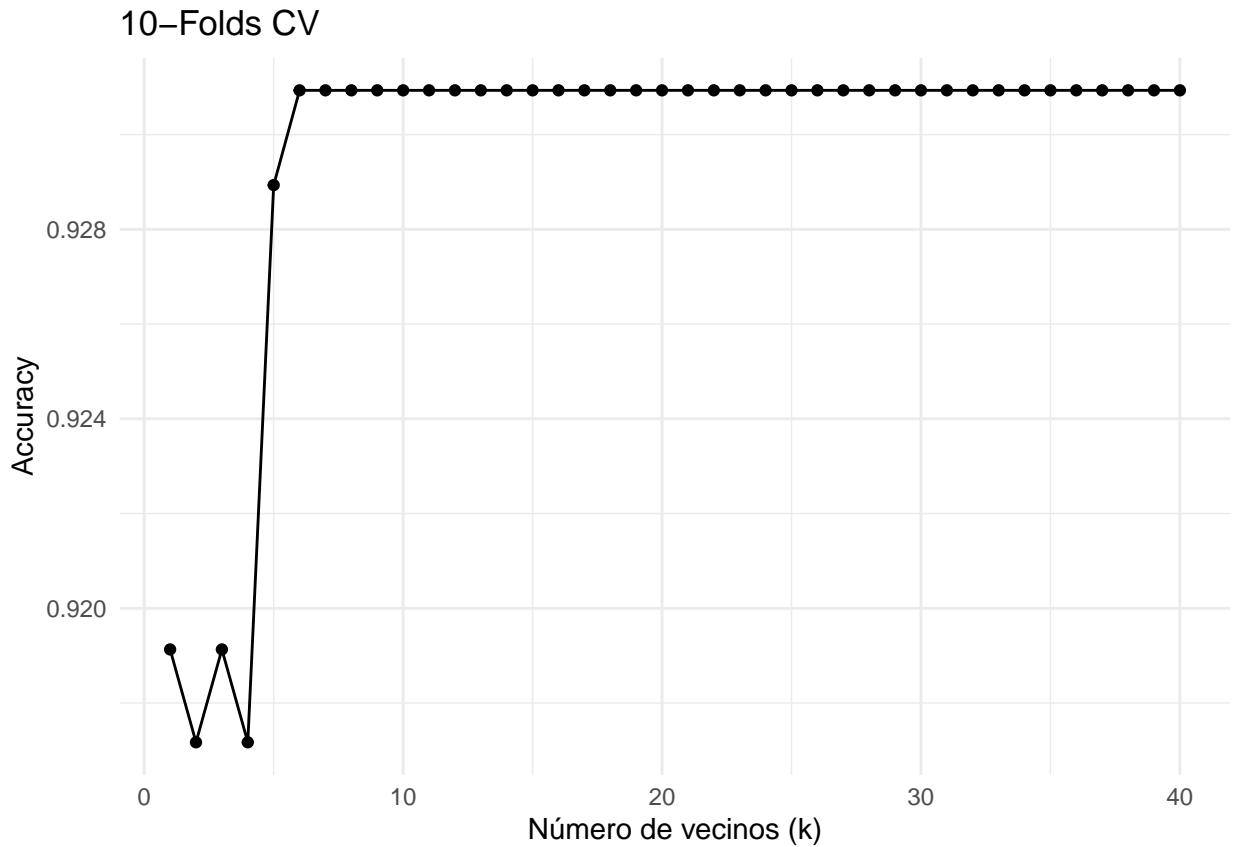
data |>

`glimpse()`

4.2.2 Búsqueda del valor de K

Al igual que en el apartado dedicado a la regresión mediante **KNN**, es necesario determinar el valor óptimo de k que se utilizará para clasificar las observaciones en este nuevo contexto de clasificación. Para ello, se aplicará un proceso de validación cruzada que permita evaluar el rendimiento del modelo para distintos valores de k , seleccionando aquel que ofrezca los mejores resultados en términos de precisión.

```
CKNN_10FCV <- data |>
  train(CHAS ~ .,
        data = _,
        method = "knn",
        preProc = c("center", "scale"),
        metric = "Accuracy",
        trControl = trainControl(method = "cv", number = 10),
        tuneGrid = data.frame(k = 1:40)
  )
CKNN_10FCV |>
  ggplot() +
  labs(x = "Número de vecinos (k)",
       y = "Accuracy",
       title = "10-Folds CV") +
  theme_minimal()
```



Como se observa en los resultados, el valor de k que proporciona la mayor exactitud del modelo es $k = 40$.

```
max_acc <- max(CKNN_10FCV$results$Accuracy)
```

```
k <- CKNN_10FCV$results |>  
  filter(Accuracy == max_acc) |>  
  slice_min(order_by = k, n = 1) |>  
  pull(k)
```

k

[1] 6

4.2.3 Conjuntos de Entrenamiento y Validación

Una vez determinado el valor óptimo de k , se procede a dividir el conjunto de datos en subconjuntos de entrenamiento y validación. Se utilizan los mismos conjuntos de entrenamiento y validación que se emplearon en el apartado anterior.

```
data_train |>  
  glimpse()
```

```
data_test |>  
glimpse()
```

4.2.4 Aplicación del modelo KNN

A continuación, se aplica el método **KNN** para clasificar observaciones de la variable respuesta, utilizando $k = 24$ sobre el conjunto de entrenamiento. Posteriormente, se generan las predicciones sobre el conjunto de validación con el objetivo de evaluar la precisión del modelo a través de la matriz de confusión.

Para ello, se emplea la función `knn3(k)` del paquete `caret`.

```
model_CKNN <- data_train |>
  knn3(CHAS ~ .,
        data = _,
        k = CKNN$bestTune$k)
```

model_CKNN

```
## 40-nearest neighbor model  
## Training set outcome distribution:  
##  
##    0    1  
## 335 19
```

4.2.5 Clasificación y Medidas de Precisión

Una vez clasificadas las nuevas observaciones, se puede proceder al cálculo de las medidas de precisión mediante la matriz de confusión. Para ello, se utiliza la función `confusionMatrix(positive, mode = "everything")` del paquete `caret`, indicando mediante el argumento `positive` cuál es la categoría considerada como positiva (en este caso, las observaciones clasificadas como cercanas al río).

Es importante asegurarse de seleccionar las variables relevantes: por un lado, las predicciones (como factor) y, por otro, la variable respuesta real.

A partir de estas, se puede construir la tabla de frecuencias utilizando la función `table()`.

```
data_classification <- data_classification |>  
  mutate(  
    CHAS = factor(CHAS, levels = c(0, 1)),  
    classification_KNN = factor(classification_KNN, levels = c(0, 1)))
```

```

)
confusionMatrix(
  data = data_classification$classification_KNN,
  reference = data_classification$CHAS,
  positive = "1",
  mode = "everything"
)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0     1
##           0 136   16
##           1     0     0
##
##                 Accuracy : 0.8947
##                           95% CI : (0.8347, 0.9386)
##   No Information Rate : 0.8947
##   P-Value [Acc > NIR] : 0.5660647
##
##                 Kappa : 0
##
##   Mcnemar's Test P-Value : 0.0001768
##
##                 Sensitivity : 0.0000
##                 Specificity  : 1.0000
##   Pos Pred Value :      NaN
##   Neg Pred Value : 0.8947
##                 Precision  :      NA
##                 Recall    : 0.0000
##                 F1       :      NA
##   Prevalence    : 0.1053
##   Detection Rate: 0.0000
##   Detection Prevalence: 0.0000
##   Balanced Accuracy : 0.5000
##
##   'Positive' Class : 1
##

```

Se observa que:

- $VP = 136$ observaciones que **no están situadas cerca del río** ($CHAS = 0$) han sido correctamente clasificadas.
- $FN = 16$ observaciones que **sí están situadas cerca del río** ($CHAS = 1$) han sido clasificadas incorrectamente.
- $FP = 0$ observaciones que **no están cerca del río** han sido clasificadas incorrectamente como si lo estuvieran.
- $VN = 0$ observaciones que **sí están cerca del río** y han sido clasificadas correctamente.

En cuanto a las medidas de precisión, se obtienen los siguientes resultados:

- **Sensitivity** = 0.0000, es decir, el modelo **no ha sido capaz de identificar ninguna observación positiva** ($CHAS = 1$) de forma correcta.
- **Specificity** = 1.0000, lo que indica que **todas las observaciones negativas** ($CHAS = 0$) han sido correctamente clasificadas.

- **Accuracy** = 0.8947, de modo que el modelo proporciona un 89.47% de clasificaciones correctas, aunque únicamente porque la clase negativa predomina en el conjunto.
- **Balanced Accuracy** = 0.5000, lo cual refleja que el modelo **no ofrece un desempeño equilibrado** entre sensibilidad y especificidad.
- **Precision, Recall, F1:** no son computables debido a la ausencia total de predicciones positivas.

4.2.7 Conclusión

El modelo KNN ajustado con $k = 7$ vecinos ha mostrado una elevada tasa de acierto general, pero esta se debe al **desequilibrio entre clases**. Al no clasificar correctamente ninguna observación positiva (**CHAS = 1**), el modelo **no es capaz de detectar viviendas cercanas al río**, lo cual se refleja en una sensibilidad nula y en un valor de **accuracy engañosamente alto**.

En definitiva, a pesar de que el modelo KNN presenta una **especificidad perfecta**, su bajo poder de discriminación frente a las clases minoritarias lo convierte en una opción poco recomendable para este problema de clasificación.

4.2 Árboles Aleatorios para Clasificación

La técnica de **Árboles Aleatorios** también puede aplicarse en problemas de clasificación.

4.2.1 Depuración del conjunto de datos

En este caso, se añadirá una variable adicional al conjunto de datos, **INDUS**, que representa el porcentaje de terrenos no comerciales por ciudad.

```
data <- boston_housing |>
  drop_na() |>
  glimpse()

## # Rows: 506
## # Columns: 14
## # $ CRIM    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829, ~
## # $ ZN      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 1~
## # $ INDUS   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.~
## # $ CHAS    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## # $ NOX     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524, ~
## # $ RM      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631, ~
## # $ AGE     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
## # $ DIS     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
## # $ RAD     <dbl> 1, 2, 2, 3, 3, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, ~
## # $ TAX     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 31~
## # $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## # $ B       <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
## # $ LSTAT   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
## # $ MEDV    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~

data <- data |>
  mutate(CHAS = as.factor(CHAS)) |>
  select(CHAS, PTRATIO, INDUS) |>
  glimpse()

## # Rows: 506
## # Columns: 3
## # $ CHAS    <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## # $ PTRATIO <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
## # $ INDUS   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.~
```

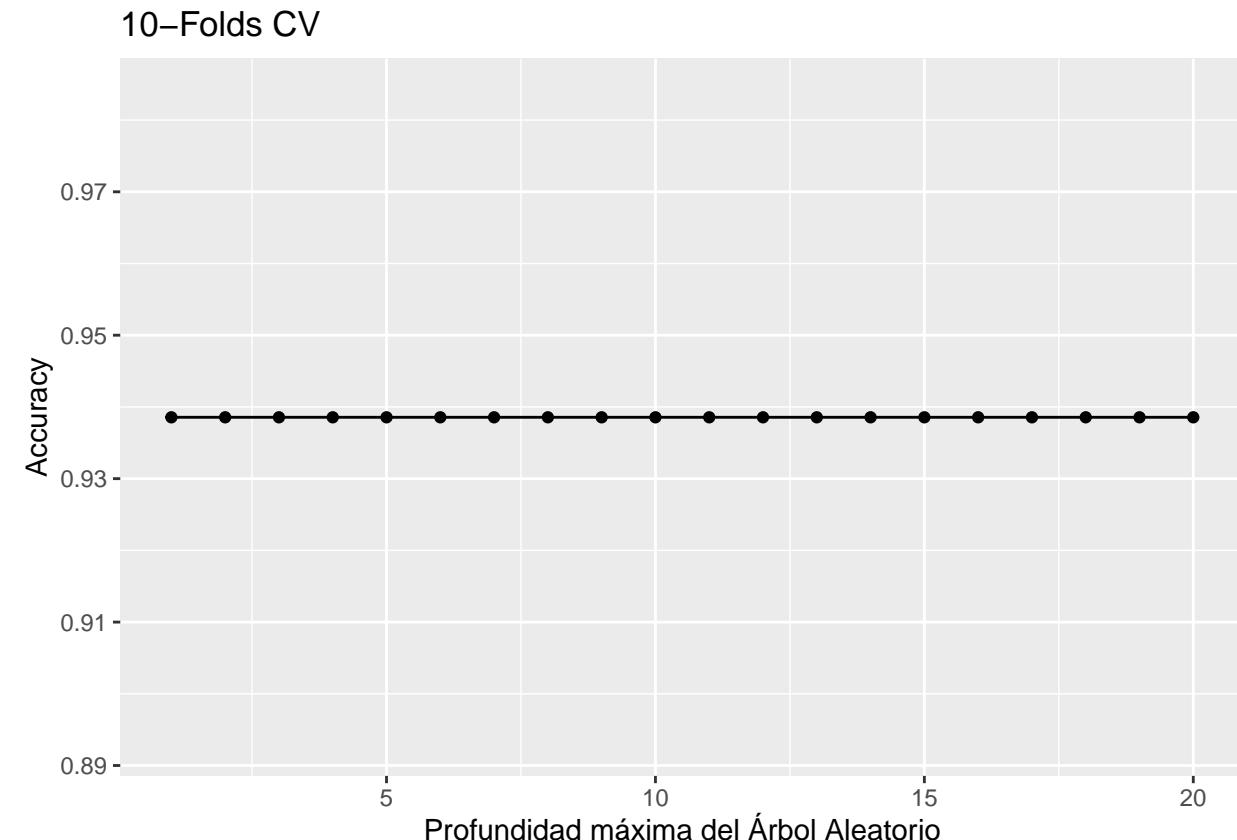
4.2.2 Búsqueda óptima para los parámetros: profundidad y complejidad

```
CRT_10FCV_maxdepth <- data |>
  train(CHAS ~.,
        data =.,
        method = "rpart2",
        metric = "Accuracy",
        trControl = trainControl(method = "cv", number = 10),
        tuneGrid = data.frame(maxdepth = 1:20)
  )

## Warning: predictions failed for Fold05: maxdepth=20 Error in approx(x[, "nsplit"], x[, "CP"], depth)
##   need at least two non-NA values to interpolate

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.

CRT_10FCV_maxdepth |>
  ggplot() +
  labs(x = "Profundidad máxima del Árbol Aleatorio",
       y = "Accuracy",
       title = "10-Folds CV")
```



A partir del gráfico de validación cruzada, se observa que la exactitud del modelo se mantiene constante para todos los valores de profundidad máxima evaluados. Sin embargo, el valor de `maxdepth = 1` ha sido seleccionado como el óptimo, al ser el primero en alcanzar la exactitud máxima registrada (93.68 %). Este valor será utilizado para construir el modelo de clasificación mediante árboles de decisión.

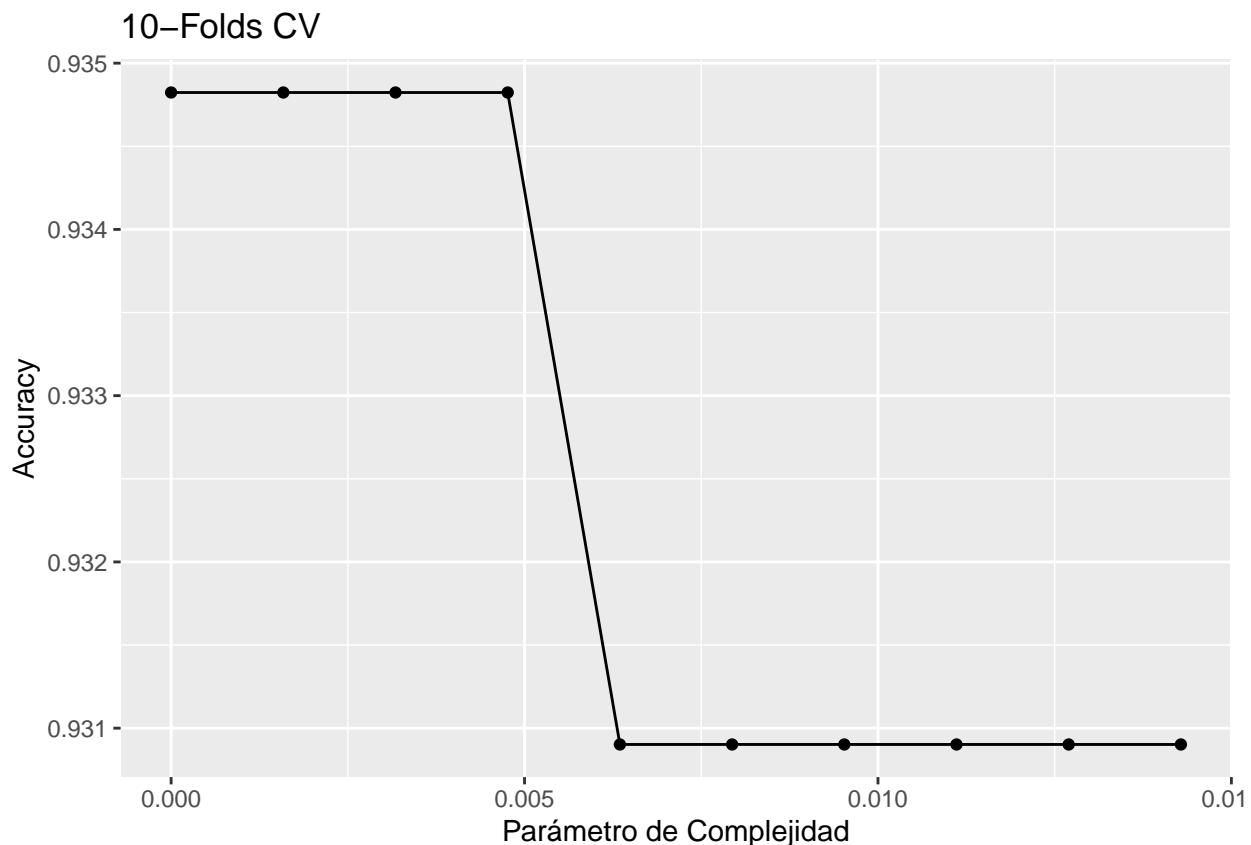
```
tibble(maxdepth = CRT_10FCV_maxdepth$bestTune$maxdepth,
      Accuracy = mean(CRT_10FCV_maxdepth$results$Accuracy))
```

```
## # A tibble: 1 x 2
##   maxdepth Accuracy
##       <int>     <dbl>
## 1         1     0.939
```

En cuanto al parámetro de complejidad, es necesario identificar su valor óptimo. Para ello, se utiliza el argumento `method = "rpart"` junto con `tuneLength`, que permite definir la cantidad de valores a explorar durante el proceso de ajuste.

```
CRT_10FCV_complexity <- data |>
  train(CHAS ~.,
        data = _,
        method = "rpart",
        metric = "Accuracy",
        trControl = trainControl(method = "cv", number = 10),
        tuneLength = 10
  )

CRT_10FCV_complexity |>
  ggplot() +
  labs(x = "Parámetro de Complejidad",
       y = "Accuracy",
       title = "10-Folds CV")
```



Se puede observar que, a medida que el parámetro de complejidad aumenta, la exactitud del modelo disminuye, lo que indica que una mayor complejidad conlleva un peor rendimiento en la clasificación.

```
tibble(complexity = CRT_10FCV_complexity$bestTune$cp,
       Accuracy = mean(CRT_10FCV_complexity$results$Accuracy))

## # A tibble: 1 x 2
##   complexity     Accuracy
##       <dbl>      <dbl>
## 1     0.00476    0.932
```

4.2.3 Conjuntos de Entrenamiento y Validación

4.2.4 Aplicación del modelo de Árboles Aleatorios

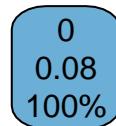
A continuación, se aplica el método de **Árboles Aleatorios** para clasificar observaciones de la variable respuesta CHAS, utilizando una profundidad máxima igual a 1 (`maxdepth = 1`) y un parámetro de complejidad igual a 0.00476 (`cp = 0.00476`) sobre el conjunto de entrenamiento. Posteriormente, se evalúan las clasificaciones generadas sobre el conjunto de validación para calcular las medidas de precisión a partir de la matriz de confusión.

El modelo se ajusta con la función `rpart(control)` del paquete `rpart`, y su representación visual se obtiene mediante la función `rpart.plot()` del paquete `rpart.plot`.

$$cp = 0.004761905$$

```
model_CRT <- data_train |>
  rpart(CHAS ~ .,
        data = _,
        control = rpart.control(maxdepth = CRT_10FCV_maxdepth$bestTune$maxdepth,
                               cp = CRT_10FCV_complexity$bestTune$cp)
  )
```

```
model_CRT |>  
rpart.plot()
```



Se observa en el árbol generado que no se ha producido ninguna partición, lo cual implica que el modelo no ha encontrado reglas de decisión que permitan dividir el conjunto de datos y mejorar la clasificación. En este caso, todas las observaciones han sido agrupadas en un único nodo raíz.

Esto sugiere que, bajo los parámetros óptimos seleccionados (profundidad máxima = 1 y $cp = 0.00476$), el modelo no ha identificado ninguna variable predictora con suficiente poder discriminativo para dividir las observaciones según la variable respuesta CHAS.

Como consecuencia, el modelo asigna la misma clase a todas las observaciones del conjunto, lo que conlleva una clasificación muy limitada, incapaz de capturar relaciones significativas entre las variables predictoras y la variable respuesta.

4.2.5 Clasificación y Medidas de Precisión

```
data_classification <- data_test |>  
  mutate(  
    classification_RT = model_CRT |>  
    predict(newdata = data_test,  
           type = "class")  
) |>  
  glimpse()  
  
## Rows: 152  
## Columns: 4
```

```

## $ CHAS <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ PTRATIO <dbl> 15.2, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0, 21.0~
## $ INDUS <dbl> 7.87, 8.14, 8.14, 8.14, 8.14, 8.14, 8.14, 8.14, 8.14~
## $ classification_RT <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
data_classification <- data_classification |>
  mutate(
    CHAS = factor(CHAS, levels = c(0, 1)),
    classification_RT = factor(classification_RT, levels = c(0, 1))
  )

confusionMatrix(
  data = data_classification$classification_RT, # predicciones
  reference = data_classification$CHAS,          # valores reales
  positive = "1",                                # clase positiva
  mode = "everything"
)

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0     1
##           0 146    6
##           1     0    0
##
##             Accuracy : 0.9605
##             95% CI  : (0.9161, 0.9854)
##             No Information Rate : 0.9605
##             P-Value [Acc > NIR] : 0.60632
##
##             Kappa : 0
##
##             Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 0.00000
##             Specificity  : 1.00000
##             Pos Pred Value :      NaN
##             Neg Pred Value : 0.96053
##             Precision   :      NA
##             Recall      : 0.00000
##             F1          :      NA
##             Prevalence  : 0.03947
##             Detection Rate : 0.00000
##             Detection Prevalence : 0.00000
##             Balanced Accuracy : 0.50000
##
##             'Positive' Class : 1
##

```

Podemos observar que:

- $VP = 141$ observaciones que **No** están cerca del río Charles y se han clasificado correctamente.
- $FN = 11$ observaciones que **Sí** están cerca del río y se han clasificado incorrectamente.
- $FP = 0$ observaciones que **No** están cerca del río y se han clasificado incorrectamente.

- $VN = 0$ observaciones que **Sí** están cerca del río y se han clasificado correctamente.

En cuanto a las medidas de precisión, tenemos que:

- Precision no es calculable, ya que el modelo no ha clasificado ninguna observación como positiva.
- Recall = 0.0000, es decir, el modelo no ha sido capaz de identificar ninguna de las viviendas situadas cerca del río.
- Specificity = 1.0000, por tanto, el 100 % de las viviendas que **no** están cerca del río se han clasificado correctamente.
- Accuracy = 0.9276, de modo que el modelo proporciona un 92.76 % de clasificaciones correctas.
- F_1 -Score no es calculable, debido a que el modelo no ha realizado ninguna predicción positiva.
- Balanced Accuracy = 0.5000, lo cual refleja un rendimiento desequilibrado entre clases.

4.2.6 Conclusión

A pesar de que el modelo muestra una alta exactitud global, esta se debe principalmente al desbalance de clases, ya que la gran mayoría de observaciones pertenecen a la clase negativa. El modelo no ha sido capaz de identificar ninguna observación positiva, como lo demuestra su sensibilidad nula y su F_1 no definido. Por tanto, el árbol de decisión no es un modelo adecuado para este problema de clasificación con la variable **CHAS** como respuesta, y sería conveniente considerar una variable alternativa con una distribución de clases más equilibrada.