Modelo predictivo de precios de coches

Grupo 8 - FID

Introducción

La venta y compra de coches es un mercado que genera muchos datos interesantes que pueden ser tratados usando distintas tecnologías de la información para obtener conocimiento. En el presente proyecto realizamos un estudio de los datos de coches usados que se encuentran en venta.

El objetivo principal de este proyecto es crear modelos que sean capaces de predecir el precio de un coche dados sus datos. Aplicaremos distintos algoritmos y compararemos sus resultados. Además, realizaremos un análisis descriptivo de los datos que tenemos.

A continuación, procederemos a analizar una serie de datasets de coches usados con el objetivo de sacar conclusiones y crear los modelos. Los datasets usados los podemos encontrar en https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho (https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho).

Los datos de este dataset han sido analizados por separado y preprocesados para formar una única fuente de datos. No sólo nos hemos centrado en sacar conclusiones sobre el análisis de los vehículos, sino también buscamos construir diferentes modelos con diferentes métodos y evaluar cuáles obtienen mejores resultados y por qué.

Importación de librerias

En primer lugar, importaremos las librerias que necesitaremos durante la ejecución del presente proyecto.

```
# Instalación de librerías
options(warn = -1)
suppressWarnings(suppressMessages({
if (!requireNamespace("caret", quietly = TRUE)) {
  install.packages("caret")
}
if (!requireNamespace("tidyverse", quietly = TRUE)) {
  install.packages("tidyverse")
}
if (!requireNamespace("dplyr", quietly = TRUE)) {
  install.packages("dplyr")
}
if (!requireNamespace("ggplot2", quietly = TRUE)) {
  install.packages("ggplot2")
}
if (!requireNamespace("corrplot", quietly = TRUE)) {
  install.packages("corrplot")
}
if (!requireNamespace("randomForest", quietly = TRUE)) {
  install.packages("randomForest")
}
if (!requireNamespace("cluster", quietly = TRUE)) {
  install.packages("cluster")
}
if (!requireNamespace("factoextra", quietly = TRUE)) {
  install.packages("factoextra")
}

# Importación de librerías
library(caret)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(corrplot)
library(randomForest)
library(cluster)
library(factoextra)
}))
```

Carga de datos

A continuación, cargaremos los datos de los datasets. Es importante mencionar que, con el fin de trabajar con unos nombres más normalizados, hemos renombrado los datasets encontrados en la web Kaggle de la siguiente manera:

- car data.csv -> car_data_1.csv
- CAR DETAILS FROM CAR DEKHO.csv -> car_data_2.csv
- Car details v3.csv -> car_data_3.csv
- car details v4.csv -> car_data_4.csv

```
# Carga de ficheros
car_data_1 <- read.csv("datasets/car_data_1.csv")
car_data_2 <- read.csv("datasets/car_data_2.csv")
car_data_3 <- read.csv("datasets/car_data_3.csv")
car_data_4 <- read.csv("datasets/car_data_4.csv")

verboseIter <- FALSE
```

Visualización inicial de los datos

Tenemos un total de 4 datasets distintos, por lo que procederemos a analizar cada uno de ellos, teniendo en cuenta los datos y atributos que tiene. Para ello, crearemos un par de funciones que nos serán de ayuda.

```
# Función para conocer el número de valores distintos de cada atributo de un dataset
numero_valores_distintos <- function(car_dataset) {
  nombres_atributos <- names(car_dataset)
  num_valores_distintos_por_atributo <- numeric(length = length(nombres_atributos))
  for (i in seq_along(nombres_atributos)) {
    atributo_actual <- nombres_atributos[i]
    valores_distintos <- unique(car_dataset[, atributo_actual])
    num_valores_distintos_por_atributo[i] <- length(valores_distintos)
  }
  resultados <- data.frame(Atributo = nombres_atributos, NumValoresDistintos = num_valores_distintos_por_atributo
)
  print(resultados)
}

# Función para conocer los posibles valores de unos atributos proporcionados de un dataset
valores_distintos <- function(car_dataset, nombres_atributos) {
  valores_distintos_por_atributo <- list()
  for (i in seq_along(nombres_atributos)) {
    atributo_actual <- nombres_atributos[i]
    valores_distintos <- unique(car_dataset[, atributo_actual])
    valores_distintos_por_atributo[[i]] <- valores_distintos
  }
  resultados <- data.frame(
    Atributo = nombres_atributos,
    ValoresDistintos = sapply(valores_distintos_por_atributo, function(x) toString(x)),
    stringsAsFactors = FALSE
  )
  print(resultados)
}
```

Empezaremos con el dataset “car_data_1”.

```
# Resumen del dataset
summary(car_data_1)
```

```
##    Car_Name              Year      Selling_Price    Present_Price
##  Length:301         Min.   :2003   Min.   : 0.100   Min.   : 0.320
##  Class :character   1st Qu.:2012   1st Qu.: 0.900   1st Qu.: 1.200
##  Mode  :character   Median :2014   Median : 3.600   Median : 6.400
##                     Mean   :2014   Mean   : 4.661   Mean   : 7.628
##                     3rd Qu.:2016   3rd Qu.: 6.000   3rd Qu.: 9.900
##                     Max.   :2018   Max.   :35.000   Max.   :92.600
##    Kms_Driven      Fuel_Type         Seller_Type        Transmission
##  Min.   :   500   Length:301         Length:301         Length:301
##  1st Qu.: 15000   Class :character   Class :character   Class :character
##  Median : 32000   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 36947
##  3rd Qu.: 48767
##  Max.   :500000
##      Owner
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.04319
##  3rd Qu.:0.00000
##  Max.   :3.00000
```

Como podemos observar, hay un total de 301 entradas en este dataset. Hay 9 atributos distintos. Veamos el número de valores distintos que tiene cada atributo.

```
numero_valores_distintos(car_data_1)
```

```
##        Atributo NumValoresDistintos
## 1      Car_Name                 98
## 2          Year                 16
## 3 Selling_Price                156
## 4 Present_Price                147
## 5    Kms_Driven                206
## 6     Fuel_Type                  3
## 7   Seller_Type                  2
## 8  Transmission                  2
## 9         Owner                  3
```

En el caso del atributo “name”, que indica el modelo del vehículo, podemos observar que existen 98 coches distintos. El año de fabricación tiene un total de 16 valores distintos (desde 2003 hasta 2018, como hemos visto en la función summary). El precio de venta y el precio actual tienen 156 y 147 valores distintos, respectivamente. Existen 206 valores de kilometraje diferentes.

Es interesante saber que existen 3 tipos de combustibles, 2 tipos de vendedores, 2 tipos de transmisiones y 3 tipos de propietarios. A continuación, mostramos cada uno para observar los posibles valores.

```
# Obtenemos los últimos atributos del dataset
nombres_atributos <- names(car_data_1)[(ncol(car_data_1) - 3):ncol(car_data_1)]

# Llamamos a la función
valores_distintos(car_data_1, nombres_atributos)
```

```
##       Atributo    ValoresDistintos
## 1    Fuel_Type Petrol, Diesel, CNG
## 2  Seller_Type  Dealer, Individual
## 3 Transmission   Manual, Automatic
## 4        Owner             0, 1, 3
```

Como podemos observar, el tipo de combustible, el tipo de vendedor y el tipo de transmisión tienen valores de texto, mientras que el tipo de propietario tiene los valores numéricos 0, 1 y 3.

Analicemos a continuación el dataset “car_data_2”. Procederemos de manera similar a como se ha hecho con el primer dataset.

```
# Resumen del dataset
summary(car_data_2)
```

```
##      name                year      selling_price       km_driven
##  Length:4340        Min.   :1992   Min.   :  20000   Min.   :     1
##  Class :character   1st Qu.:2011   1st Qu.: 208750   1st Qu.: 35000
##  Mode  :character   Median :2014   Median : 350000   Median : 60000
##                     Mean   :2013   Mean   : 504127   Mean   : 66216
##                     3rd Qu.:2016   3rd Qu.: 600000   3rd Qu.: 90000
##                     Max.   :2020   Max.   :8900000   Max.   :806599
##      fuel           seller_type        transmission          owner
##  Length:4340        Length:4340        Length:4340        Length:4340
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

Existen un total de 4340 entradas en el dataset. Hay 8 atributos distintos que, como podemos observar, son similares a los del primer dataset, a excepción del precio actual que no se encuentra en el dataset “car_data_2”. Veamos el número de valores distintos que tiene cada atributo.

```
numero_valores_distintos(car_data_2)
```

```
##        Atributo NumValoresDistintos
## 1          name                1491
## 2          year                  27
## 3 selling_price                 445
## 4     km_driven                 770
## 5          fuel                   5
## 6   seller_type                   3
## 7  transmission                   2
## 8         owner                   5
```

A diferencia del dataset anterior, el atributo de modelo del coche tiene un número de valores distintos más elevado al de los atributos año de fabricación, precio de venta y kilometraje.

Volvemos a encontrarnos con un pequeño número de valores posibles en el tipo de combustible, tipo de vendedor, tipo de transmisión y tipo de propietario. Veamos estos valores.

```
# Obtenemos los últimos atributos del dataset
nombres_atributos <- names(car_data_2)[(ncol(car_data_2) - 3):ncol(car_data_2)]

# Llamamos a la función
valores_distintos(car_data_2, nombres_atributos)
```

```
##       Atributo
## 1         fuel
## 2  seller_type
## 3 transmission
## 4        owner
##                                                                ValoresDistintos
## 1                                        Petrol, Diesel, CNG, LPG, Electric
## 2                                    Individual, Dealer, Trustmark Dealer
## 3                                                           Manual, Automatic
## 4 First Owner, Second Owner, Fourth & Above Owner, Third Owner, Test Drive Car
```

Podemos observar varias diferencias entre los valores de este dataset y del anterior. El tipo de combustible tiene dos nuevos valores posibles con respecto al dataset “car_data_1”. Así mismo, el tipo de vendedor incluye también el valor “Trustmark Dealer”. Los tipos de transmisiones son iguales en ambos dataset. La mayor diferencia se encuentra en el tipo de propietarios, ya que en este dataset aparecen 5 posibles valores en formato texto, mientras que en el anterior solo aparecían 3 valores en formato numérico.

Proseguimos con el dataset “car_data_3”.

```
# Resumen del dataset
summary(car_data_3)
```

```
##      name               year      selling_price        km_driven
##  Length:8128        Min.   :1983   Min.   :   29999   Min.   :      1
##  Class :character   1st Qu.:2011   1st Qu.:  254999   1st Qu.:  35000
##  Mode  :character   Median :2015   Median :  450000   Median :  60000
##                     Mean   :2014   Mean   :  638272   Mean   :  69820
##                     3rd Qu.:2017   3rd Qu.:  675000   3rd Qu.:  98000
##                     Max.   :2020   Max.   :10000000   Max.   :2360457
##
##      fuel           seller_type        transmission         owner
##  Length:8128        Length:8128        Length:8128        Length:8128
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##    mileage            engine           max_power           torque
##  Length:8128        Length:8128        Length:8128        Length:8128
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##      seats
##  Min.   : 2.000
##  1st Qu.: 5.000
##  Median : 5.000
##  Mean   : 5.417
##  3rd Qu.: 5.000
##  Max.   :14.000
##  NA's   :221
```

Existen un total de 8128 entradas en este dataset. Hay un total de 13 atributos distintos que suponen diferencias con el resto de los datasets usados. Veamos el número de valores distintos que tiene cada atributo.

```
numero_valores_distintos(car_data_3)
```

```
##         Atributo NumValoresDistintos
## 1           name                2058
## 2           year                  29
## 3  selling_price                 677
## 4      km_driven                 921
## 5           fuel                   4
## 6    seller_type                   3
## 7   transmission                   2
## 8          owner                   5
## 9        mileage                 394
## 10        engine                 122
## 11     max_power                 323
## 12        torque                 442
## 13         seats                  10
```

Existen un total de 2058 modelos de coche distintos en este dataset. Como anteriormente, procedemos a analizar los posibles valores de los atributos tipo de combustible, tipo de vendedor, tipo de transmisión y tipo de propietario, que pueden ser interesantes para nuestro estudio.

```
# Obtenemos los atributos del dataset que nos interesan
nombres_atributos <- names(car_data_3)[(ncol(car_data_3) - 8):(ncol(car_data_3) - 5)]

# Llamamos a la función
valores_distintos(car_data_3, nombres_atributos)
```

```
##       Atributo
## 1         fuel
## 2  seller_type
## 3 transmission
## 4        owner
##                                                                ValoresDistintos
## 1                                                  Diesel, Petrol, LPG, CNG
## 2                                  Individual, Dealer, Trustmark Dealer
## 3                                                 Manual, Automatic
## 4 First Owner, Second Owner, Third Owner, Fourth & Above Owner, Test Drive Car
```

Observamos que los valores de los atributos son iguales a los del dataset “car_data_2”, a excepción del tipo de combustible, que en el dataset anterior también contiene el valor “Electric”.

Seguimos el análisis con el último dataset, “car_data_4”.

```
# Resumen del dataset
summary(car_data_4)
```

```
##      Make              Model               Price               Year
##  Length:2059        Length:2059        Min.   :   49000   Min.   :1988
##  Class :character   Class :character   1st Qu.:  484999   1st Qu.:2014
##  Mode  :character   Mode  :character   Median :  825000   Median :2017
##                                        Mean   : 1702992   Mean   :2016
##                                        3rd Qu.: 1925000   3rd Qu.:2019
##                                        Max.   :35000000   Max.   :2022
##
##    Kilometer        Fuel.Type         Transmission         Location
##  Min.   :      0   Length:2059        Length:2059        Length:2059
##  1st Qu.:  29000   Class :character   Class :character   Class :character
##  Median :  50000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :  54225
##  3rd Qu.:  72000
##  Max.   :2000000
##
##     Color              Owner           Seller.Type           Engine
##  Length:2059        Length:2059        Length:2059        Length:2059
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   Max.Power          Max.Torque         Drivetrain            Length
##  Length:2059        Length:2059        Length:2059        Min.   :3099
##  Class :character   Class :character   Class :character   1st Qu.:3985
##  Mode  :character   Mode  :character   Mode  :character   Median :4370
##                                                           Mean   :4281
##                                                           3rd Qu.:4629
##                                                           Max.   :5569
##                                                           NA's   :64
##      Width           Height     Seating.Capacity Fuel.Tank.Capacity
##  Min.   :1475   Min.   :1165   Min.   :2.000    Min.   : 15.00
##  1st Qu.:1695   1st Qu.:1485   1st Qu.:5.000    1st Qu.: 41.25
##  Median :1770   Median :1545   Median :5.000    Median : 50.00
##  Mean   :1768   Mean   :1592   Mean   :5.306    Mean   : 52.00
##  3rd Qu.:1832   3rd Qu.:1675   3rd Qu.:5.000    3rd Qu.: 60.00
##  Max.   :2220   Max.   :1995   Max.   :8.000    Max.   :105.00
##  NA's   :64     NA's   :64     NA's   :64       NA's   :113
```

Existen un total de 2059 entradas en este dataset. Además, este dataset es el que mayor número de atributos tiene, con un total de 20. Los atributos que hemos visto en común en los datasets anteriores también aparecen en este, aunque en el caso del modelo del coche se muestra en dos atributos diferenciados, “Make” (marca) y “Model” (modelo). Veamos el número de valores distintos que tiene cada atributo.

```
numero_valores_distintos(car_data_4)
```

```
##              Atributo NumValoresDistintos
## 1                Make                  33
## 2               Model                1050
## 3               Price                 619
## 4                Year                  22
## 5           Kilometer                 847
## 6           Fuel.Type                   9
## 7        Transmission                   2
## 8            Location                  77
## 9               Color                  17
## 10              Owner                   6
## 11        Seller.Type                   3
## 12             Engine                 109
## 13          Max.Power                 336
## 14         Max.Torque                 291
## 15         Drivetrain                   4
## 16             Length                 249
## 17              Width                 171
## 18             Height                 197
## 19   Seating.Capacity                   7
## 20 Fuel.Tank.Capacity                  56
```

Como podemos observar, existen 33 marcas de coches distintas en el dataset, y un total de 1050 modelos. Procedemos a analizar los posibles valores de los atributos tipo de combustible, tipo de vendedor, tipo de transmisión y tipo de propietario, como hemos hecho con los anteriores datasets.

```
# Obtenemos los atributos del dataset que nos interesan
nombres_atributos <- c("Fuel.Type", "Transmission", "Owner", "Seller.Type")

# Llamamos a la función
valores_distintos(car_data_4, nombres_atributos)
```

```
##       Atributo
## 1    Fuel.Type
## 2 Transmission
## 3        Owner
## 4  Seller.Type
##                                                                  ValoresDistintos
## 1 Petrol, Diesel, CNG, LPG, Electric, CNG + CNG, Hybrid, Petrol + CNG, Petrol + LPG
## 2                                                                Manual, Automatic
## 3                        First, Second, Third, Fourth, UnRegistered Car, 4 or More
## 4                                    Corporate, Individual, Commercial Registration
```

Los valores de estos atributos en el dataset “car_data_4” tienen bastantes diferencias con el resto de datasets. Este posee un mayor número de valores en el tipo de combustible. Además, los tipos de propietario son diferentes a los que hemos observado en los datasets anteriores, aunque reflejan lo mismo en varios casos (por ejemplo, “First” refleja el mismo valor que “First Owner”). El tipo de vendedor solo comparte el valor “Individual”, aunque “Corporate” podría reflejar lo mismo que “Dealer” y “Commercial Registration” podría reflejar el mismo valor que “Trustmark Dealer”.

Preprocesado de datos

Dado el objetivo del presente trabajo, se ha tomado la decisión de desestimar el primer dataset “car_data_1”, dado que se ha detectado una gran diferencia con el resto de datos de los datasets restantes, lo que dificultaría las tareas de preprocesado, algo que no se pretende en el trabajo desarrollado.

Uno de los datos más relevantes a la hora de realizar un predicción del precio es la marca y el modelo del vehículo. Como hemos podido observar, los datasets “car_data_2” y “car_data_3” tienen una estructura similar en la que la marca y el modelo del coche aparecen en el mismo atributo “name”. Sin embargo, en el dataset “car_data_4”, aparecen dos aributos “Make” y “Model” (marca y modelo, respectivamente). Dado que ambos son valores a tener en cuenta de forma individual, se ha optado por separar el atributo “name” de los datasets “car_data_2” y “car_data_3” en dos atributos diferentes.

```
# Usando la función mutate, creamos dos nuevos atributos a partir del atributo name
car_data_2 <- car_data_2 %>%
  mutate(
    make = sapply(strsplit(as.character(name), " "), function(x) x[1]),
    model = sapply(strsplit(as.character(name), " "), function(x) paste(x[-1], collapse = " "))
  )

# Eliminamos el atributo name, ya que no lo usaremos
car_data_2 <- car_data_2 %>% select(-name)

# Reorganizamos los atributos, poniendo primero los atributos make y model
car_data_2 <- car_data_2 %>% select(make, model, everything())

# Comprobamos que los atributos se han creado, borrado y reorganizado de manera correcta
head(car_data_2)
```

```
##      make             model year selling_price km_driven   fuel seller_type
## 1  Maruti            800 AC 2007         60000     70000 Petrol  Individual
## 2  Maruti Wagon R LXI Minor 2007        135000     50000 Petrol  Individual
## 3 Hyundai      Verna 1.6 SX 2012        600000    100000 Diesel  Individual
## 4  Datsun   RediGO T Option 2017        250000     46000 Petrol  Individual
## 5   Honda   Amaze VX i-DTEC 2014        450000    141000 Diesel  Individual
## 6  Maruti     Alto LX BSIII 2007        140000    125000 Petrol  Individual
##   transmission        owner
## 1       Manual  First Owner
## 2       Manual  First Owner
## 3       Manual  First Owner
## 4       Manual  First Owner
## 5       Manual Second Owner
## 6       Manual  First Owner
```

A continuación, realizamos las mismas operaciones con el dataset “car_data_3”.

```
# Usando la función mutate, creamos dos nuevos atributos a partir del atributo name
car_data_3 <- car_data_3 %>%
  mutate(
    make = sapply(strsplit(as.character(name), " "), function(x) x[1]),
    model = sapply(strsplit(as.character(name), " "), function(x) paste(x[-1], collapse = " "))
  )

# Eliminamos el atributo name, ya que no lo usaremos
car_data_3 <- car_data_3 %>% select(-name)

# Reorganizamos los atributos, poniendo primero los atributos make y model
car_data_3 <- car_data_3 %>% select(make, model, everything())

# Comprobamos que los atributos se han creado, borrado y reorganizado de manera correcta
head(car_data_3)
```

```
##      make                  model year selling_price km_driven   fuel
## 1  Maruti        Swift Dzire VDI 2014        450000    145500 Diesel
## 2   Skoda Rapid 1.5 TDI Ambition 2014        370000    120000 Diesel
## 3   Honda     City 2017-2020 EXi 2006        158000    140000 Petrol
## 4 Hyundai      i20 Sportz Diesel 2010        225000    127000 Diesel
## 5  Maruti        Swift VXI BSIII 2007        130000    120000 Petrol
## 6 Hyundai  Xcent 1.2 VTVT E Plus 2017        440000     45000 Petrol
##   seller_type transmission        owner    mileage  engine  max_power
## 1  Individual       Manual  First Owner  23.4 kmpl 1248 CC     74 bhp
## 2  Individual       Manual Second Owner 21.14 kmpl 1498 CC 103.52 bhp
## 3  Individual       Manual  Third Owner  17.7 kmpl 1497 CC     78 bhp
## 4  Individual       Manual  First Owner  23.0 kmpl 1396 CC     90 bhp
## 5  Individual       Manual  First Owner  16.1 kmpl 1298 CC   88.2 bhp
## 6  Individual       Manual  First Owner 20.14 kmpl 1197 CC  81.86 bhp
##                     torque seats
## 1           190Nm@ 2000rpm     5
## 2      250Nm@ 1500-2500rpm     5
## 3    12.7@ 2,700(kgm@ rpm)     5
## 4 22.4 kgm at 1750-2750rpm     5
## 5    11.5@ 4,500(kgm@ rpm)     5
## 6        113.75nm@ 4000rpm     5
```

A continuación, proprocesaremos el dataset “car_data_4”. En primer lugar, hemos observado que la marca de coches Maruti se define en este dataset como Maruti Suzuki. Con el fin de poder unificar los datos de todos los datasets, procedemos a eliminar la palabra “Suzuki” para que la marca del coche figure solo como “Maruti”, al igual que ocurre en los datasets “car_data_2” y “car_data_3”.

```
# Usando la función mutate, nos quedamos con la primera palabra de cada fila
car_data_4 <- car_data_4 %>%
  mutate(
    Make = word(Make, 1)
  )

# Comprobamos que el atributo se ha cambiado de manera correcta
head(car_data_4)
```

```
##      Make                           Model   Price Year Kilometer Fuel.Type
## 1   Honda             Amaze 1.2 VX i-VTEC  505000 2017     87150    Petrol
## 2  Maruti                 Swift DZire VDI  450000 2014     75000    Diesel
## 3 Hyundai            i10 Magna 1.2 Kappa2  220000 2011     67000    Petrol
## 4  Toyota                         Glanza G  799000 2019     37500    Petrol
## 5  Toyota Innova 2.4 VX 7 STR [2016-2020] 1950000 2018     69000    Diesel
## 6  Maruti                         Ciaz ZXi  675000 2017     73315    Petrol
##   Transmission  Location  Color  Owner Seller.Type  Engine          Max.Power
## 1       Manual      Pune   Grey  First   Corporate 1198 cc  87 bhp @ 6000 rpm
## 2       Manual  Ludhiana  White Second  Individual 1248 cc  74 bhp @ 4000 rpm
## 3       Manual   Lucknow Maroon  First  Individual 1197 cc  79 bhp @ 6000 rpm
## 4       Manual Mangalore    Red  First  Individual 1197 cc  82 bhp @ 6000 rpm
## 5       Manual    Mumbai   Grey  First  Individual 2393 cc 148 bhp @ 3400 rpm
## 6       Manual      Pune   Grey  First  Individual 1373 cc  91 bhp @ 6000 rpm
##               Max.Torque Drivetrain Length Width Height Seating.Capacity
## 1      109 Nm @ 4500 rpm        FWD   3990  1680   1505                5
## 2      190 Nm @ 2000 rpm        FWD   3995  1695   1555                5
## 3 112.7619 Nm @ 4000 rpm        FWD   3585  1595   1550                5
## 4      113 Nm @ 4200 rpm        FWD   3995  1745   1510                5
## 5      343 Nm @ 1400 rpm        RWD   4735  1830   1795                7
## 6      130 Nm @ 4000 rpm        FWD   4490  1730   1485                5
##   Fuel.Tank.Capacity
## 1                 35
## 2                 42
## 3                 35
## 4                 37
## 5                 55
## 6                 43
```

Ahora buscamos quedarnos con los atributos que nos puedan ser de ayuda en el presente trabajo. Estos atributos son la marca, el modelo, el año de fabricación, el precio de venta, el kilometraje, el tipo de combustible, el tipo de vendedor, el tipo de transmisión y el propietario del coche. Para ello, debemos ajustar el nombre de estos atributos en todos los datasets, de manera que queden como “make”, “model”, “year”, “selling_price”, “km_driven”, “fuel”, “seller_type”, “transmission” y “owner”. Podemos observar que el “car_data_2” y el “car_data_3” ya poseen estos atributos, por lo que procesaremos el “car_data_4”.

```
# Vamos a renombrar cada atributo con el nombre que hemos definido
car_data_4 <- car_data_4 %>%
  rename_with(~"make", .cols = "Make") %>%
  rename_with(~"model", .cols = "Model") %>%
  rename_with(~"selling_price", .cols = "Price") %>%
  rename_with(~"year", .cols = "Year") %>%
  rename_with(~"km_driven", .cols = "Kilometer") %>%
  rename_with(~"fuel", .cols = "Fuel.Type") %>%
  rename_with(~"transmission", .cols = "Transmission") %>%
  rename_with(~"owner", .cols = "Owner") %>%
  rename_with(~"seller_type", .cols = "Seller.Type")

# Comprobamos que los atributos se han cambiado de manera correcta
head(car_data_4)
```

```
##      make                           model selling_price year km_driven   fuel
## 1   Honda             Amaze 1.2 VX i-VTEC        505000 2017     87150 Petrol
## 2  Maruti                 Swift DZire VDI        450000 2014     75000 Diesel
## 3 Hyundai            i10 Magna 1.2 Kappa2        220000 2011     67000 Petrol
## 4  Toyota                         Glanza G        799000 2019     37500 Petrol
## 5  Toyota Innova 2.4 VX 7 STR [2016-2020]       1950000 2018     69000 Diesel
## 6  Maruti                         Ciaz ZXi        675000 2017     73315 Petrol
##   transmission  Location  Color  owner seller_type  Engine          Max.Power
## 1       Manual      Pune   Grey  First   Corporate 1198 cc  87 bhp @ 6000 rpm
## 2       Manual  Ludhiana  White Second  Individual 1248 cc  74 bhp @ 4000 rpm
## 3       Manual   Lucknow Maroon  First  Individual 1197 cc  79 bhp @ 6000 rpm
## 4       Manual Mangalore    Red  First  Individual 1197 cc  82 bhp @ 6000 rpm
## 5       Manual    Mumbai   Grey  First  Individual 2393 cc 148 bhp @ 3400 rpm
## 6       Manual      Pune   Grey  First  Individual 1373 cc  91 bhp @ 6000 rpm
##               Max.Torque Drivetrain Length Width Height Seating.Capacity
## 1      109 Nm @ 4500 rpm        FWD   3990  1680   1505                5
## 2      190 Nm @ 2000 rpm        FWD   3995  1695   1555                5
## 3 112.7619 Nm @ 4000 rpm        FWD   3585  1595   1550                5
## 4      113 Nm @ 4200 rpm        FWD   3995  1745   1510                5
## 5      343 Nm @ 1400 rpm        RWD   4735  1830   1795                7
## 6      130 Nm @ 4000 rpm        FWD   4490  1730   1485                5
##   Fuel.Tank.Capacity
## 1                 35
## 2                 42
## 3                 35
## 4                 37
## 5                 55
## 6                 43
```

Procederemos a seleccionar los atributos que usaremos de cada uno de los datasets que usaremos en el proyecto. El dataset “car_data_2” ya posee solo los atributos normalizados, así que procesaremos los datasets “car_data_3” y “car_data_4”.

```
# Seleccionamos los atributos del car_data_3
car_data_3 <- car_data_3 %>% select(make, model, year, selling_price, km_driven, fuel, seller_type, transmission,
owner)

# Comprobamos que los atributos se han seleccionado de manera correcta
head(car_data_3)
```

```
##      make                  model year selling_price km_driven   fuel
## 1  Maruti        Swift Dzire VDI 2014        450000    145500 Diesel
## 2   Skoda Rapid 1.5 TDI Ambition 2014        370000    120000 Diesel
## 3   Honda     City 2017-2020 EXi 2006        158000    140000 Petrol
## 4 Hyundai      i20 Sportz Diesel 2010        225000    127000 Diesel
## 5  Maruti        Swift VXI BSIII 2007        130000    120000 Petrol
## 6 Hyundai  Xcent 1.2 VTVT E Plus 2017        440000     45000 Petrol
##   seller_type transmission        owner
## 1  Individual       Manual  First Owner
## 2  Individual       Manual Second Owner
## 3  Individual       Manual  Third Owner
## 4  Individual       Manual  First Owner
## 5  Individual       Manual  First Owner
## 6  Individual       Manual  First Owner
```

```
# Seleccionamos los atributos del car_data_4
car_data_4 <- car_data_4 %>% select(make, model, year, selling_price, km_driven, fuel, seller_type, transmission,
owner)

# Comprobamos que los atributos se han seleccionado de manera correcta
head(car_data_4)
```

```
##      make                           model year selling_price km_driven   fuel
## 1   Honda             Amaze 1.2 VX i-VTEC 2017        505000     87150 Petrol
## 2  Maruti                 Swift DZire VDI 2014        450000     75000 Diesel
## 3 Hyundai            i10 Magna 1.2 Kappa2 2011        220000     67000 Petrol
## 4  Toyota                         Glanza G 2019        799000     37500 Petrol
## 5  Toyota Innova 2.4 VX 7 STR [2016-2020] 2018       1950000     69000 Diesel
## 6  Maruti                         Ciaz ZXi 2017        675000     73315 Petrol
##   seller_type transmission  owner
## 1   Corporate       Manual  First
## 2  Individual       Manual Second
## 3  Individual       Manual  First
## 4  Individual       Manual  First
## 5  Individual       Manual  First
## 6  Individual       Manual  First
```

Normalización de valores

El paso previo antes de la integración de los datos en un mismo dataset es la normalización de todos los datos. Hemos observado que los datos están normalizados para los atributos “make”, “model”, “year”, “selling_price” y “km_driven”. El precio de venta que figura en “selling_price” está en rupias, por lo que haremos una conversión para pasarlo a euros, proceso que haremos en todos los datasets. Los atributos “fuel” y “transmission” poseen valores diversos entre los distintos datasets, pero todos ellos están definidos de la misma manera. Por ejemplo, siempre que aparece el atributo de valor “Petrol” aparece escrito de la misma forma.

Los atributos “seller_type” y “owner” son los únicos que difieren entre los datasets. El “car_data_2” y el “car_data_3” tienen los mismos valores, pero el “car_data_4” posee valores distintos. A los valores del atributo “seller_type” del “car_data_4” tendremos que añadirle la palabra “Owner”. Además, hemos definido que el valor “UnRegistered Car” del “car_data_4” pasará a denominarse “Test Drive Car”, como tenemos en los datasets “car_data_2” y “car_data_3”. El valor “4 or More” también se estimará como “Fourth & Above Owner”. En el caso del atributo “seller_type”, “Corporate” se denominará “Dealer” y “Commercial Registration” se denominará “Trustmark Dealer”.

```
# Diccionario con los valores antiguos y los nuevos de seller_type
dicc_valores_seller <- c("Corporate" = "Dealer", "Commercial Registration" = "Trustmark Dealer")

car_data_4 <- car_data_4 %>%
  mutate(seller_type = ifelse(seller_type %in% names(dicc_valores_seller), dicc_valores_seller[seller_type], sell
er_type))

# Diccionario con los valores antiguos y los nuevos de owner
dicc_valores_owner <- c("First" = "First Owner", "Second" = "Second Owner", "Third" = "Third Owner", "Fourth" = "
Fourth & Above Owner", "4 or More" = "Fourth & Above Owner", "UnRegistered Car" = "Test Drive Car")

car_data_4 <- car_data_4 %>%
  mutate(owner = ifelse(owner %in% names(dicc_valores_owner), dicc_valores_owner[owner], owner))

# Comprobamos que los atributos se han cambiado de manera correcta
head(car_data_4)
```

```
##      make                           model year selling_price km_driven   fuel
## 1   Honda             Amaze 1.2 VX i-VTEC 2017        505000     87150 Petrol
## 2  Maruti                 Swift DZire VDI 2014        450000     75000 Diesel
## 3 Hyundai            i10 Magna 1.2 Kappa2 2011        220000     67000 Petrol
## 4  Toyota                        Glanza G 2019        799000     37500 Petrol
## 5  Toyota Innova 2.4 VX 7 STR [2016-2020] 2018       1950000     69000 Diesel
## 6  Maruti                        Ciaz ZXi 2017        675000     73315 Petrol
##   seller_type transmission        owner
## 1      Dealer       Manual  First Owner
## 2  Individual       Manual Second Owner
## 3  Individual       Manual  First Owner
## 4  Individual       Manual  First Owner
## 5  Individual       Manual  First Owner
## 6  Individual       Manual  First Owner
```

A continuación, unificaremos los datos de los 3 datasets seleccionados.

```
# Unificamos los datos
car_dataset_total <- rbind(car_data_2, car_data_3, car_data_4)

# Comprobamos que los datasets se han unido de manera correcta
head(car_dataset_total)
```

```
##      make             model year selling_price km_driven   fuel seller_type
## 1  Maruti            800 AC 2007         60000     70000 Petrol  Individual
## 2  Maruti Wagon R LXI Minor 2007        135000     50000 Petrol  Individual
## 3 Hyundai      Verna 1.6 SX 2012        600000    100000 Diesel  Individual
## 4  Datsun   RediGO T Option 2017        250000     46000 Petrol  Individual
## 5   Honda   Amaze VX i-DTEC 2014        450000    141000 Diesel  Individual
## 6  Maruti     Alto LX BSIII 2007        140000    125000 Petrol  Individual
##   transmission        owner
## 1       Manual  First Owner
## 2       Manual  First Owner
## 3       Manual  First Owner
## 4       Manual  First Owner
## 5       Manual Second Owner
## 6       Manual  First Owner
```

```
# Además, comprobaremos que tiene el mismo número de filas que la suma de los 3 datasets
suma_filas_datasets <- nrow(car_data_2) + nrow(car_data_3) + nrow(car_data_4)

# Suma de las filas de los datasets
print(suma_filas_datasets)
```

```
## [1] 14527
```

```
# Número de filas del dataset unificado
print(nrow(car_dataset_total))
```

```
## [1] 14527
```

Por último, para obtener el dataset con el que trabajaremos finalmente, convertiremos el precio de venta de rupias a euros, multiplicándolo por un factor.

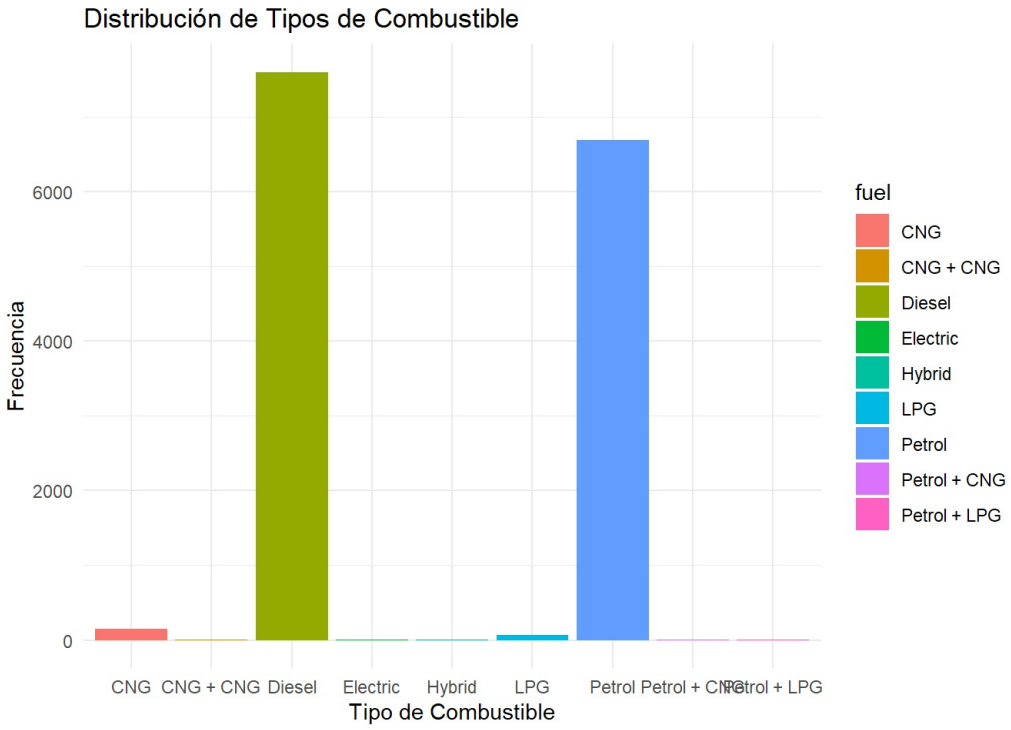
```
# Aplicamos el factor de conversión de 0.011
car_dataset_total <- car_dataset_total %>%
  mutate(selling_price = selling_price * 0.011)

# Comprobamos que el atributo se ha cambiado de manera correcta
head(car_dataset_total)
```

```
##      make             model year selling_price km_driven   fuel seller_type
## 1  Maruti           800 AC 2007           660     70000 Petrol  Individual
## 2  Maruti Wagon R LXI Minor 2007          1485     50000 Petrol  Individual
## 3 Hyundai      Verna 1.6 SX 2012          6600    100000 Diesel  Individual
## 4  Datsun   RediGO T Option 2017          2750     46000 Petrol  Individual
## 5   Honda   Amaze VX i-DTEC 2014          4950    141000 Diesel  Individual
## 6  Maruti     Alto LX BSIII 2007          1540    125000 Petrol  Individual
##   transmission        owner
## 1       Manual  First Owner
## 2       Manual  First Owner
## 3       Manual  First Owner
## 4       Manual  First Owner
## 5       Manual Second Owner
## 6       Manual  First Owner
```

Visualización

Vamos a proceder a visualizar los datos de los que disponemos. Empezaremos mostrando un gráfico de barras para los tipos de combustible existentes en nuestro dataset.

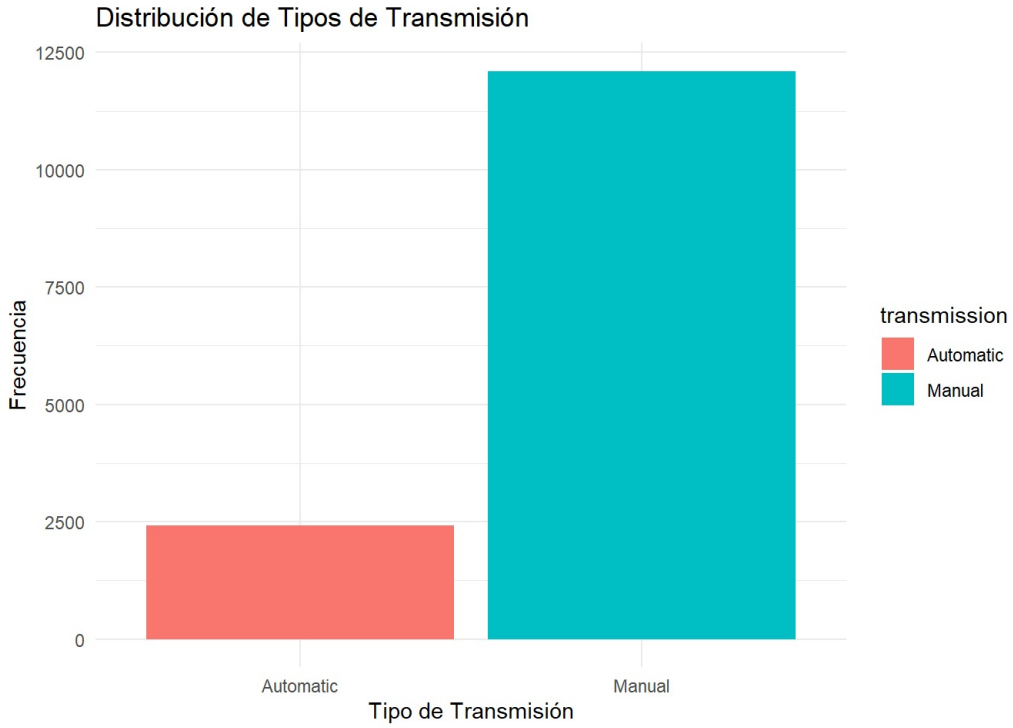
```
# Gráfico de barras para el tipo de combustible
ggplot(car_dataset_total, aes(x = fuel, fill = fuel)) +
  geom_bar() +
  ggtitle("Distribución de Tipos de Combustible") +
  xlab("Tipo de Combustible") +
  ylab("Frecuencia") +
  theme_minimal()
```



Podemos observar que la gran mayoría de coches son diesel o gasolina, con un número muy bajo de los otros tipos de combustible.

A continuación, mostraremos un gráfico de barras para los tipos de transmisión.

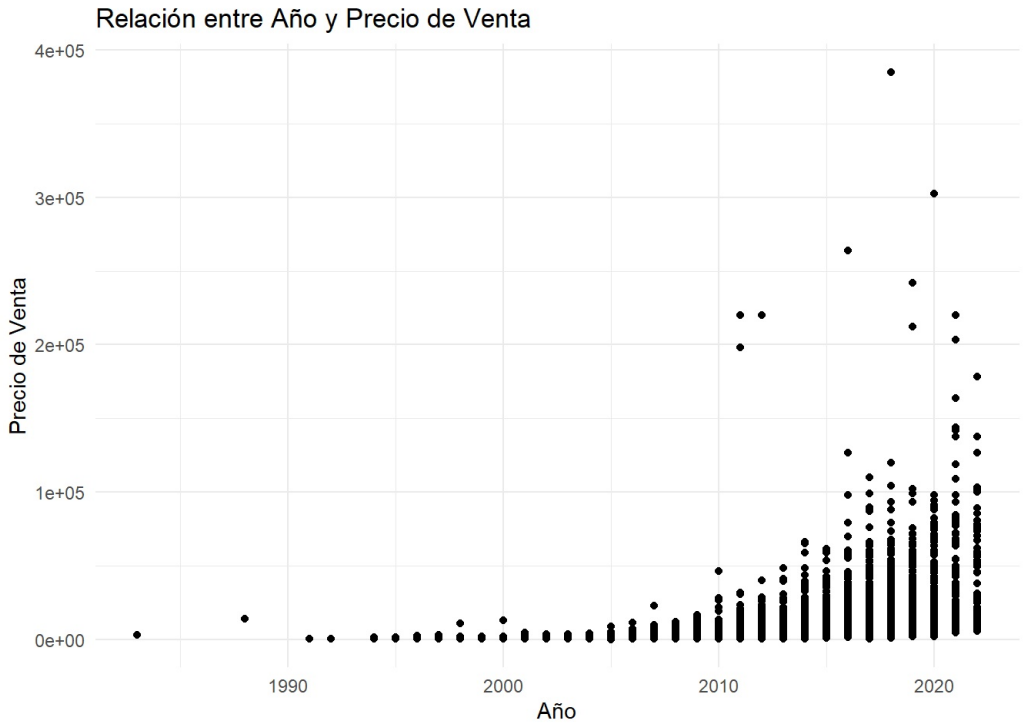
```
# Gráfico de barras para el tipo de transmisión
ggplot(car_dataset_total, aes(x = transmission, fill = transmission)) +
  geom_bar() +
  ggtitle("Distribución de Tipos de Transmisión") +
  xlab("Tipo de Transmisión") +
  ylab("Frecuencia") +
  theme_minimal()
```



A pesar de que la gran mayoría de coches son manuales, existe un número considerable de coches automáticos en el dataset, teniendo en cuenta que contamos con 14527 datos, como hemos visto anteriormente.

Presentamos un gráfico de dispersión para año y precio de venta.

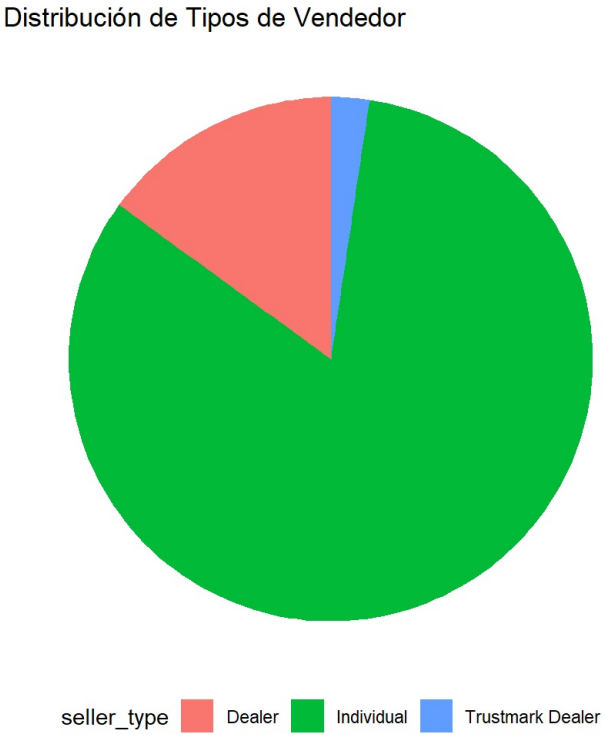
```
# Gráfico de dispersión para año y precio de venta
ggplot(car_dataset_total, aes(x = year, y = selling_price)) +
  geom_point() +
  ggtitle("Relación entre Año y Precio de Venta") +
  xlab("Año") +
  ylab("Precio de Venta") +
  theme_minimal()
```



A simple vista, podemos observar que los coches más nuevos son, por norma general, más caros que los coches más antiguos.

A continuación, vemos un gráfico de torta para el tipo de vendedor.

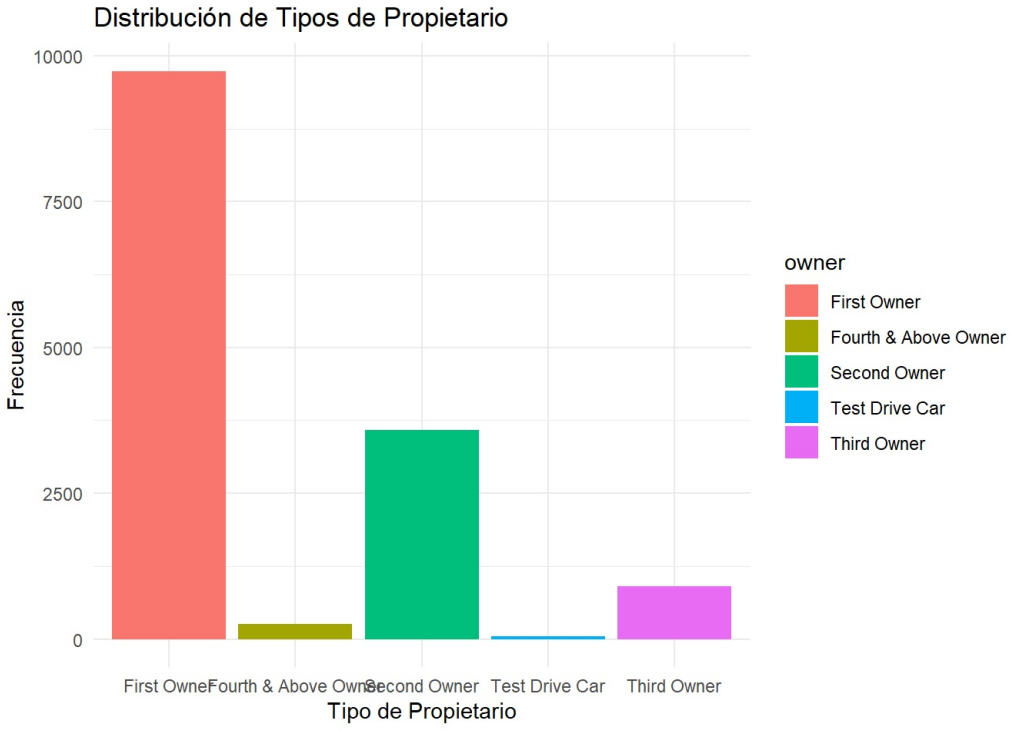
```
# Gráfico circular para el tipo de vendedor
ggplot(car_dataset_total, aes(x = factor(1), fill = seller_type)) +
  geom_bar(width = 1, stat = "count") +
  coord_polar(theta = "y") +
  ggtitle("Distribución de Tipos de Vendedor") +
  theme_void() +
  theme(legend.position = "bottom")
```



La mayoría de los datos representan vendedores individuales.

Veamos un gráfico de barras apiladas para el tipo de propietario.

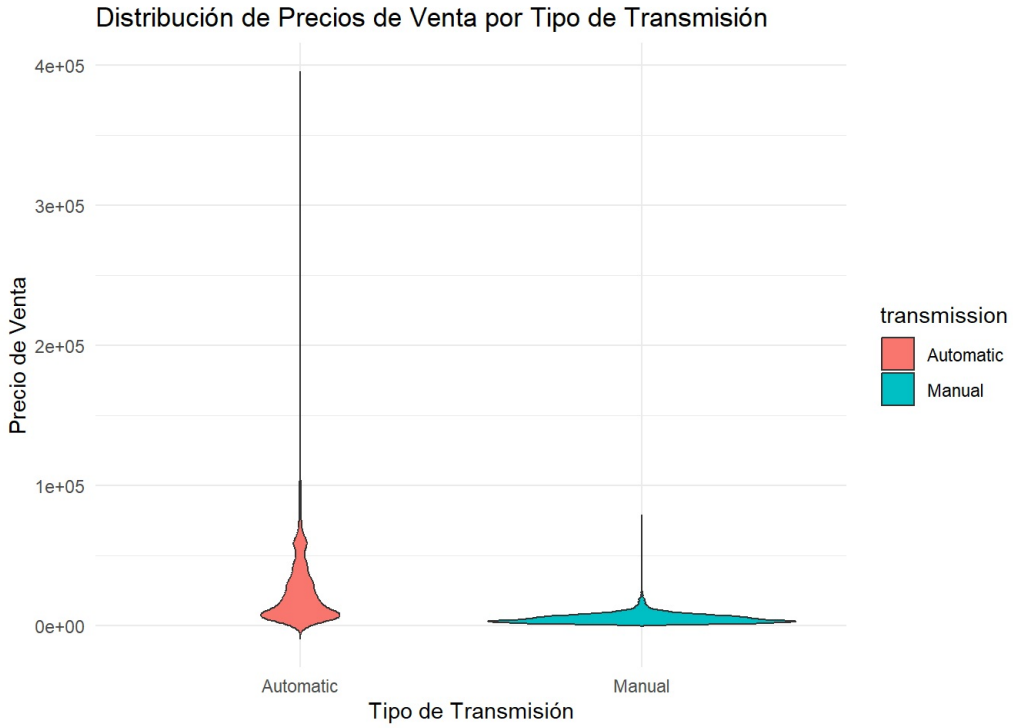
```
# Gráfico de barras apiladas para el tipo de propietario
ggplot(car_dataset_total, aes(x = factor(owner), fill = owner)) +
  geom_bar(position = "stack") +
  ggtitle("Distribución de Tipos de Propietario") +
  xlab("Tipo de Propietario") +
  ylab("Frecuencia") +
  theme_minimal()
```



Podemos ver que la cantidad de vehiculos que han tenido un mayor número de propietarios disminuye cuanto mayor es el número de propietarios. De esta forma, la mayoría de los datos son coches que han tenido un propietario.

Presentamos un gráfico de violín para comparar la distribución de precios por el tipo de transmisión.

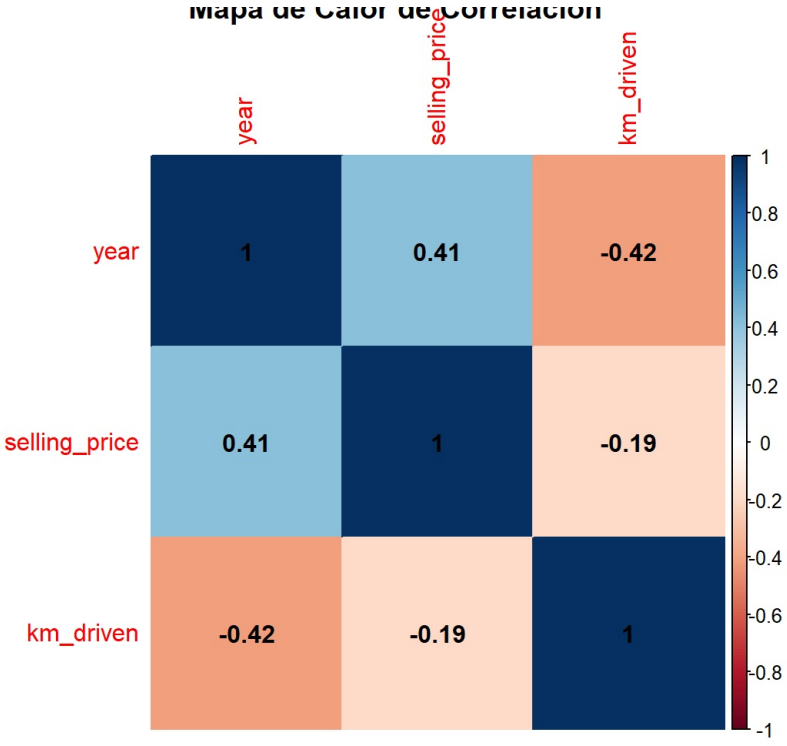
```
# Gráfico de violín para comparar la distribución de precios por tipo de transmisión
ggplot(car_dataset_total, aes(x = transmission, y = selling_price, fill = transmission)) +
  geom_violin(trim = FALSE) +
  ggtitle("Distribución de Precios de Venta por Tipo de Transmisión") +
  xlab("Tipo de Transmisión") +
  ylab("Precio de Venta") +
  theme_minimal()
```



La distribución muestra que los coches automáticos tienden a tener un mayor precio que los manuales.

Por último, presentamos un mapa de calor comparando la correlación entre las variables numéricas en el conjunto de datos.

```
# Selecciona solo las variables numéricas
numeric_vars <- sapply(car_data_2, is.numeric)
numeric_data <- car_data_2[, numeric_vars]

# Calcula la matriz de correlación
correlation_matrix <- cor(numeric_data)

# Crea un mapa de calor con la correlación
corrplot(correlation_matrix, method = "color", addCoef.col = "black", title = "Mapa de Calor de Correlación")
```

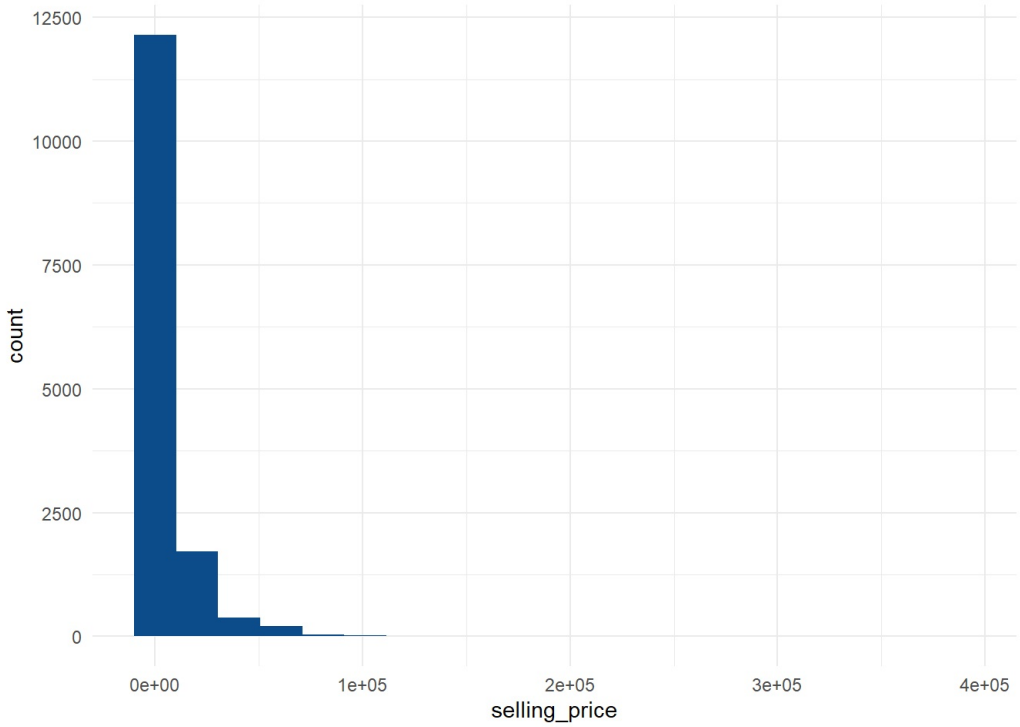


Este mapa muestra la relacion entre las variables representadas en las que vemos como el precio de venta tiene cierta relación con el año, mientras que el kilometraje muestra una relación similar pero inversa con el año.

Predicción

En esta sección del proyecto vamos a plantear modelos predictivos que puedan predecir el precio del vehículo atendiendo a sus características. Hacemos un último tratamiento simple de los datos antes de plantear los modelos y buscamos outliers.

```
# Tratamiento previo de los datos de combustible
car_dataset_total$fuel <- as.factor(car_dataset_total$fuel)

# Realizamos histograma del precio
ggplot(car_dataset_total) +
  aes(x = selling_price) +
  geom_histogram(bins = 20L, fill = "#0c4c8a") +
  theme_minimal()
```




```
# Eliminamos los coches que tengan un precio igual o mayor al valor que consideremos para los outliers
car_dataset_total <- car_dataset_total[car_dataset_total$selling_price <= 50000, ]
```

Como podemos observar, hemos tomado los datos del dataset cuyo precio es inferior a 50000, para evitar posibles outliers.

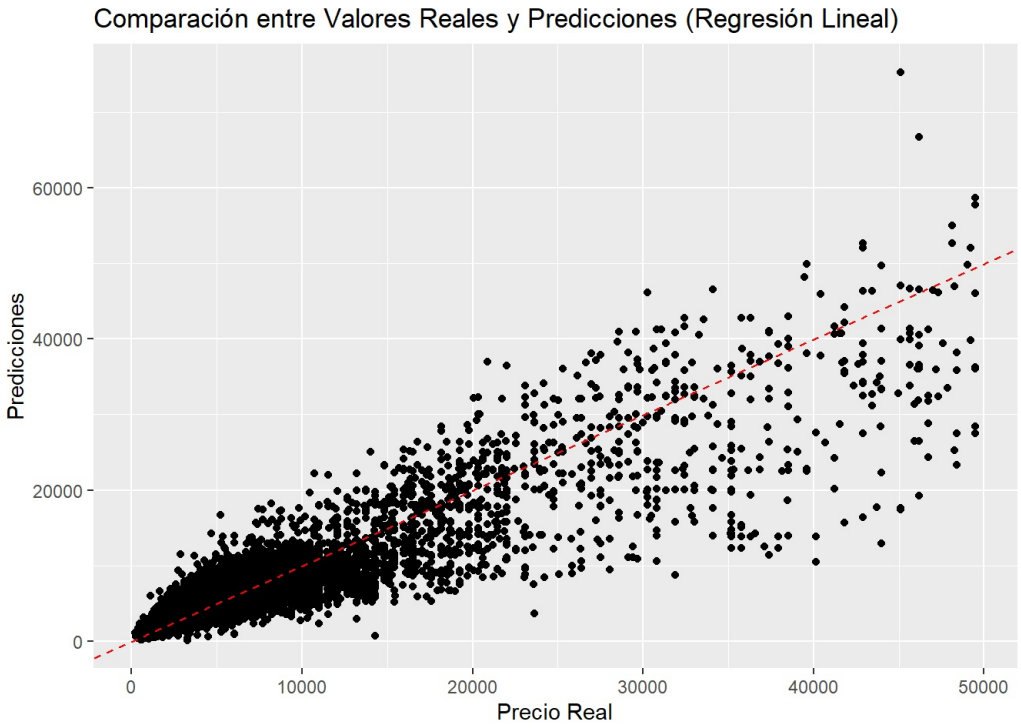
A continuación, presentamos los modelos realizados. Cabe destacar que los atributos seleccionados para el proyecto excluyen el modelo del vehículo, ya que implicaría un gran coste computacional, desembocando en la necesidad de un elevado tiempo de ejecución.

Empezaremos por un modelo basado en regresión lineal.

```
# Usamos una función logarítmica para el precio, evitando errores o valores incoherentes en la predicción, como v
alores negativos
car_dataset_total$selling_price_log <- log1p(car_dataset_total$selling_price)

# Usamos validación cruzada
ctrl <- trainControl(method = "cv", number = 10, verboseIter = verboseIter)

# Modelado (Regresión Lineal con validación cruzada k-fold)
model_CV <- train(selling_price_log ~ make + km_driven + year + fuel + transmission + owner + seller_type,
  data = car_dataset_total,
  method = "lm",
  trControl = ctrl
)
# Realizar predicciones durante la validación cruzada
predictions_cv <- predict(model_CV)
predictions_original <- expm1(predictions_cv)

# Comparar las predicciones con los valores reales en cada iteración
comparison_cv <- data.frame(Real = car_dataset_total$selling_price, Predicciones = predictions_original)

# Visualizar las primeras filas de la comparación
head(comparison_cv)
```

```
##   Real Predicciones
## 1  660     1421.223
## 2 1485     1426.945
## 3 6600     4286.703
## 4 2750     3291.667
## 5 4950     6081.798
## 6 1540     1405.605
```

```
# Crear un gráfico de dispersión para visualizar las predicciones en cada iteración
ggplot(comparison_cv, aes(x = Real, y = Predicciones)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  ggtitle("Comparación entre Valores Reales y Predicciones (Regresión Lineal)") +
  xlab("Precio Real") +
  ylab("Predicciones")
```




```
mse_lm <- mean((predictions_cv - car_dataset_total$selling_price)^2)
r_squared_lm <- cor(predictions_cv, car_dataset_total$selling_price)^2
mae_lm <- mean(abs(predictions_cv - car_dataset_total$selling_price))

cat("Error Cuadrático Medio (MSE) (Regresión Lineal):", mse_lm, "\n")
```

```
## Error Cuadrático Medio (MSE) (Regresión Lineal): 98221616
```

```
cat("Coeficiente de Determinación (R²) (Regresión Lineal):", r_squared_lm, "\n")
```

```
## Coeficiente de Determinación (R²) (Regresión Lineal): 0.6114293
```

```
cat("Error Absoluto Medio (MAE) (Regresión Lineal):", mae_lm, "\n")
```

```
## Error Absoluto Medio (MAE) (Regresión Lineal): 6827.256
```

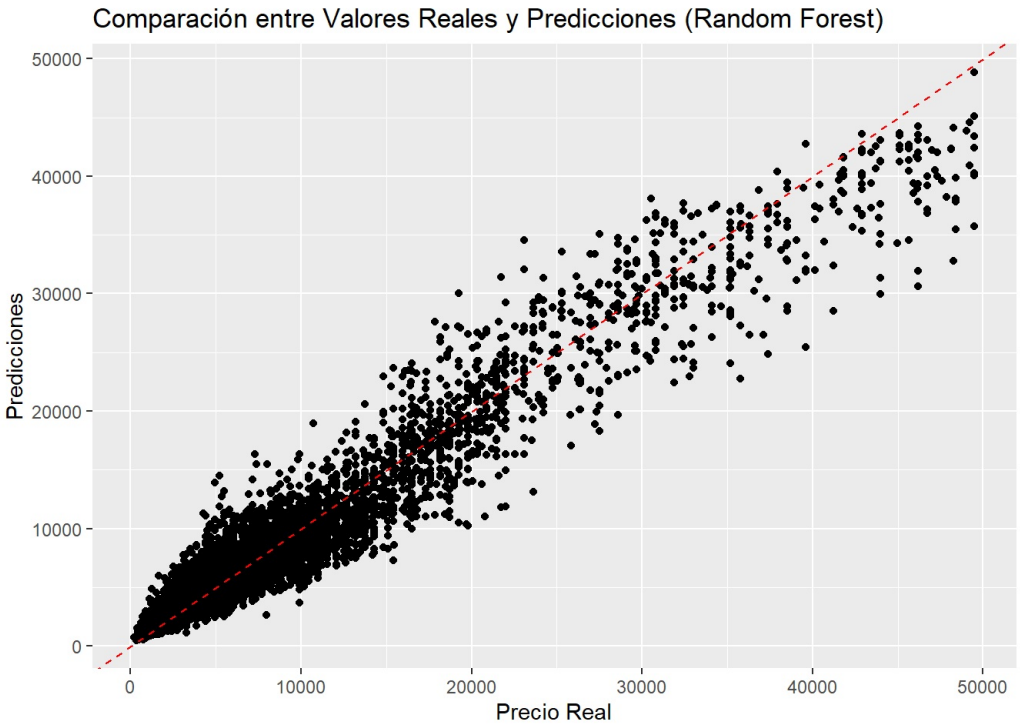
Podemos observar que la predicción tiende a ser correcta para precios inferiores, pero aumenta la dispersión en valores mayores. El coheficiente de determinación (R^2) de 0,61 indica que el modelo no es muy preciso, pero a continuación lo compararemos con otros modelos para observar su desempeño.

Procedemos con el modelo basado en random forest.

```
# Usamos validación cruzada
ctrl <- trainControl(method = "cv", number = 2, verboseIter = verboseIter)

# Modelado (Random Forest con validación cruzada k-fold)
model_CV <- train(selling_price ~ make + km_driven + year + fuel + transmission + owner + seller_type,
  data = car_dataset_total,
  method = "rf",
  trControl = ctrl
)
# Realizar predicciones durante la validación cruzada
predictions_cv <- predict(model_CV)

# Comparar las predicciones con los valores reales en cada iteración
comparison_cv <- data.frame(Real = car_dataset_total$selling_price, Predicciones = predictions_cv)

# Visualizar las primeras filas de la comparación
head(comparison_cv)
```

```
##   Real Predicciones
## 1  660     1017.719
## 2 1485     1415.216
## 3 6600     5083.183
## 4 2750     2826.854
## 5 4950     5575.359
## 6 1540     1385.921
```

```
# Crear un gráfico de dispersión para visualizar las predicciones en cada iteración
ggplot(comparison_cv, aes(x = Real, y = Predicciones)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  ggtitle("Comparación entre Valores Reales y Predicciones (Random Forest)") +
  xlab("Precio Real") +
  ylab("Predicciones")
```




```
mse_rf <- mean((predictions_cv - car_dataset_total$selling_price)^2)
r_squared_rf <- cor(predictions_cv, car_dataset_total$selling_price)^2
mae_rf <- mean(abs(predictions_cv - car_dataset_total$selling_price))

cat("Error Cuadrático Medio (MSE) (Random Forest):", mse_rf, "\n")
```

```
## Error Cuadrático Medio (MSE) (Random Forest): 2709704
```

```
cat("Coeficiente de Determinación (R²) (Random Forest):", r_squared_rf, "\n")
```

```
## Coeficiente de Determinación (R²) (Random Forest): 0.9483367
```

```
cat("Error Absoluto Medio (MAE) (Random Forest):", mae_rf, "\n")
```

```
## Error Absoluto Medio (MAE) (Random Forest): 1005.756
```

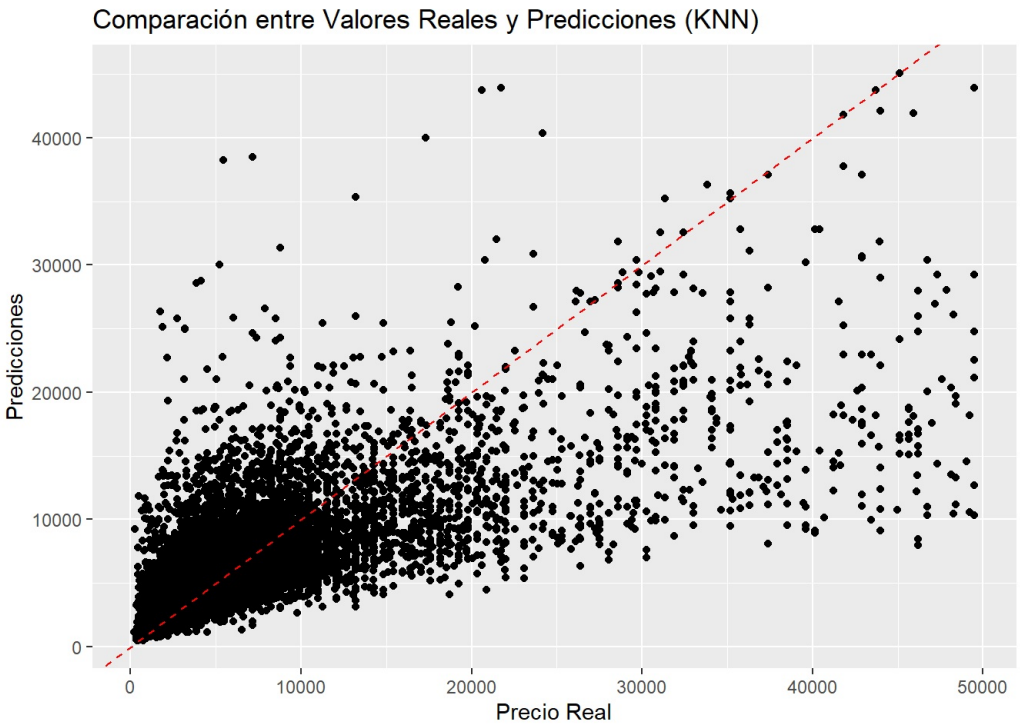
Podemos observar una mejora notable en este modelo con respecto al basado en regresión lineal. Aunque el tiempo de ejecución es bastante superior (cerca de 15 minutos) los resultados son mejores, obeteniendo un coheficiente de determinación (R^2) de 0,94, además de obtener errores notablemente mejores.

Por último, veamos un modelo basado en KNN.

```
# Usamos validación cruzada
ctrl <- trainControl(method = "cv", number = 10, verboseIter = verboseIter)

# Modelado (KNN con validación cruzada k-fold)
model_CV <- train(selling_price ~ make + km_driven + year + fuel + transmission + owner + seller_type,
  data = car_dataset_total,
  method = "knn",
  trControl = ctrl
)
# Realizar predicciones durante la validación cruzada
predictions_cv <- predict(model_CV)

# Comparar las predicciones con los valores reales en cada iteración
comparison_cv <- data.frame(Real = car_dataset_total$selling_price, Predicciones = predictions_cv)

# Visualizar las primeras filas de la comparación
head(comparison_cv)
```

```
##   Real Predicciones
## 1  660     861.6667
## 2 1485    1694.0000
## 3 6600    4667.3838
## 4 2750    4564.9982
## 5 4950    5390.0000
## 6 1540    3069.0000
```

```
# Crear un gráfico de dispersión para visualizar las predicciones en cada iteración
ggplot(comparison_cv, aes(x = Real, y = Predicciones)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  ggtitle("Comparación entre Valores Reales y Predicciones (KNN)") +
  xlab("Precio Real") +
  ylab("Predicciones")
```



```
mse_knn <- mean((predictions_cv - car_dataset_total$selling_price)^2)
r_squared_knn <- cor(predictions_cv, car_dataset_total$selling_price)^2
mae_knn <- mean(abs(predictions_cv - car_dataset_total$selling_price))

cat("Error Cuadrático Medio (MSE) (KNN):", mse_knn, "\n")
```

```
## Error Cuadrático Medio (MSE) (KNN): 19710622
```

```
cat("Coeficiente de Determinación (R²) (KNN):", r_squared_knn, "\n")
```

```
## Coeficiente de Determinación (R²) (KNN): 0.6225287
```

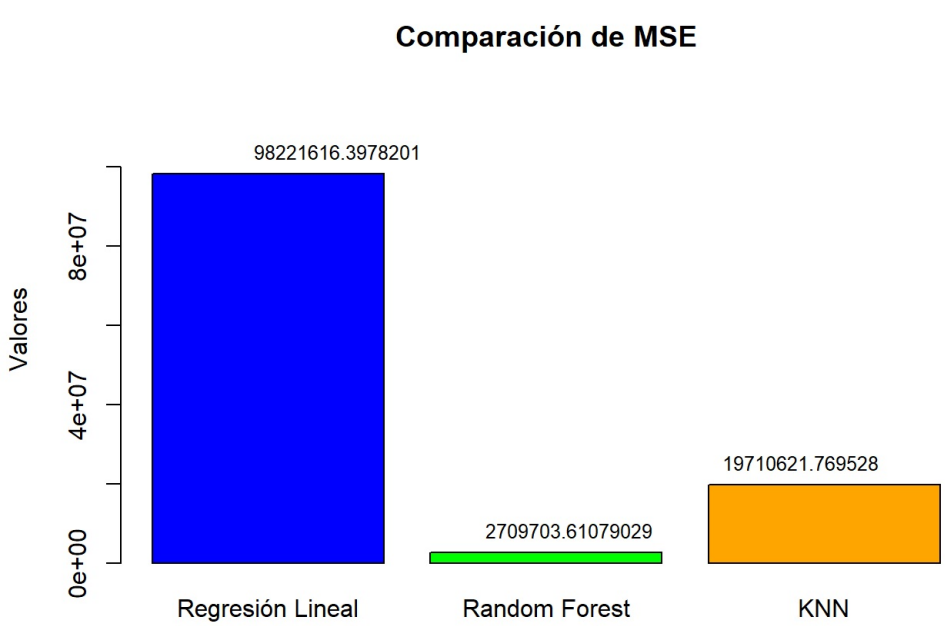
```
cat("Error Absoluto Medio (MAE) (KNN):", mae_knn, "\n")
```

```
## Error Absoluto Medio (MAE) (KNN): 2270.495
```

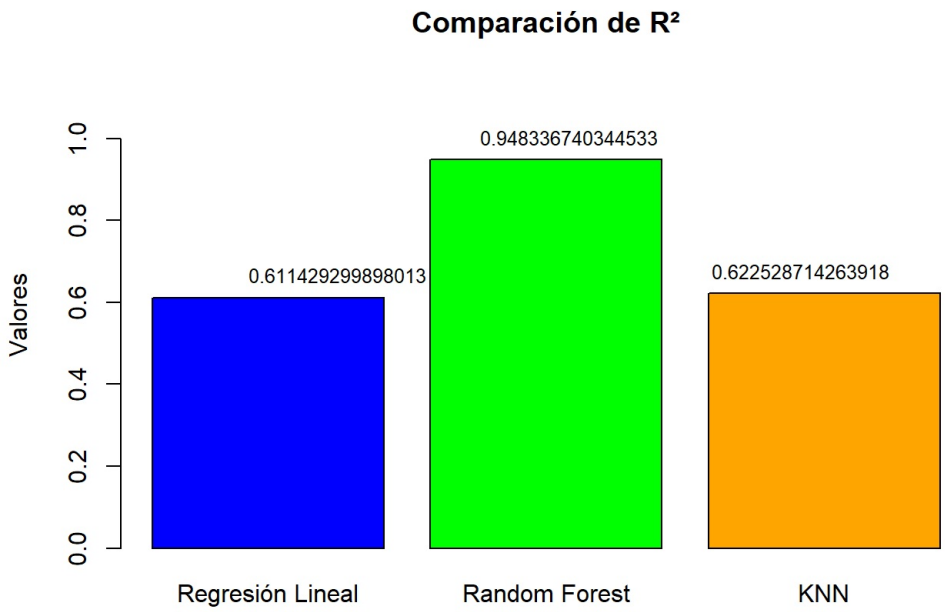
De nuevo, este modelo también funciona relativamente bien con valores pequeños, pero aumenta bastante la dispersión en valores elevados de precios. El error cuadrático medio (MSE) y el error absoluto medio (MAE) son bastante inferiores a los vistos en el modelo de regresión lineal, pero su coheficiente de determinación (R^2) es similar.

Por último, realizaremos una comparación de los modelos desarrollados.

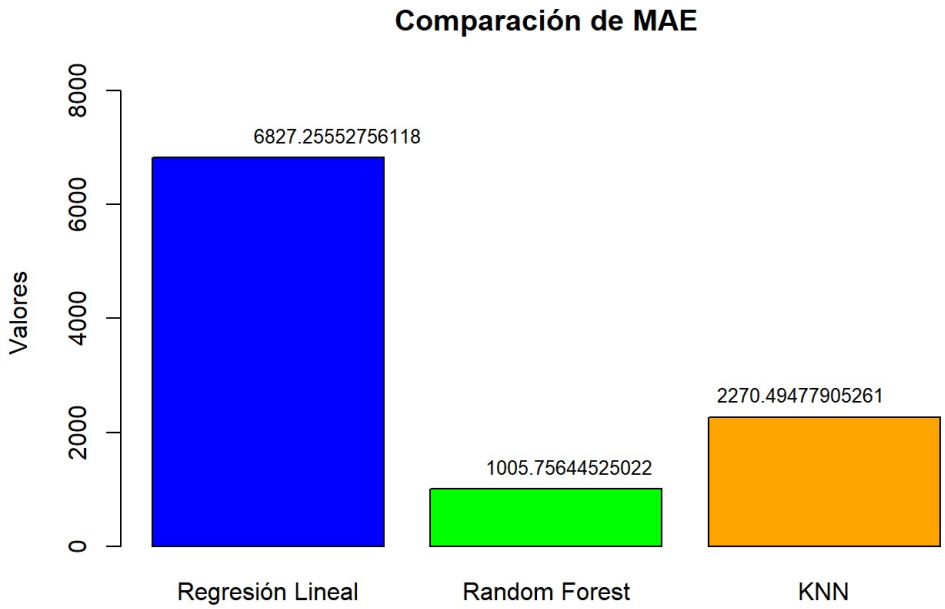
```
algoritmos <- c("Regresión Lineal", "Random Forest", "KNN")

# MSE
mse_group <- c(mse_lm, mse_rf, mse_knn)
barplot(mse_group,
  names.arg = algoritmos, col = c("blue", "green", "orange"),
  main = "Comparación de MSE",
  ylab = "Valores",
  ylim = c(0, max(mse_group) * 1.2)
) # Ajustar el límite y para mayor claridad

# Agregar etiquetas en las barras
text(seq_along(mse_group), mse_group, labels = mse_group, pos = 3, col = "black", cex = 0.8)
```



```
# R²
r_squared_group <- c(r_squared_lm, r_squared_rf, r_squared_knn)
barplot(r_squared_group,
  names.arg = algoritmos, col = c("blue", "green", "orange"),
  main = "Comparación de R²",
  ylab = "Valores",
  ylim = c(0, max(r_squared_group) * 1.2)
) # Ajustar el límite y para mayor claridad

# Agregar etiquetas en las barras
text(seq_along(r_squared_group), r_squared_group, labels = r_squared_group, pos = 3, col = "black", cex = 0.8)
```




```
# MAE
mae_group <- c(mae_lm, mae_rf, mae_knn)
barplot(mae_group,
  names.arg = algoritmos, col = c("blue", "green", "orange"),
  main = "Comparación de MAE",
  ylab = "Valores",
  ylim = c(0, max(mae_group) * 1.2)
) # Ajustar el límite y para mayor claridad

# Agregar etiquetas en las barras
text(seq_along(mae_group), mae_group, labels = mae_group, pos = 3, col = "black", cex = 0.8)
```



En las gráficas se muestra como el modelo basado en random forest es superior en las tres métricas. El error cuadrático medio (MSE) es muy inferior al de los otros dos modelos, de los que la regresión lineal tiene un valor muy superior al resto, lo que indica un mayor error.

Comparando el coheficiente de determinación (R^2), podemos observar similitud entre los modelos de regresión lineal y KNN. Sin embargo, el modelo de random forest destaca, siendo el único que supera el 0,9 en esta métrica.

Por último, vemos la comparativa del error absoluto medio (MAE), en la que se representa una comparación similar a la observada en el error cuadrático medio (MSE). De nuevo, el modelo de random forest obtiene el menor valor de error.

Clustering

A continuación, presentamos un estudio descriptivo usando clustering. Veremos dos técnicas en las que relacionaremos el kilometraje y el año de fabricación del coche, creando clusters con los valores que observamos.

Empezaremos aplicando K-Medias

```
# Seleccionar las variables relevantes para el clustering
variables_clustering <- car_dataset_total[, c("km_driven", "year")]

# Especificar el número de clusters (k). Usaremos k = 3
k <- 3

# Aplicar k-medias
kmeans_result <- kmeans(variables_clustering, centers = k)

# Imprimir resultados
cat("Centros de los clusters:\n")
```

```
## Centros de los clusters:
```

```
print(kmeans_result$centers)
```

```
##   km_driven     year
## 1  26595.65 2016.304
## 2 144049.97 2011.216
## 3  72782.59 2012.821
```

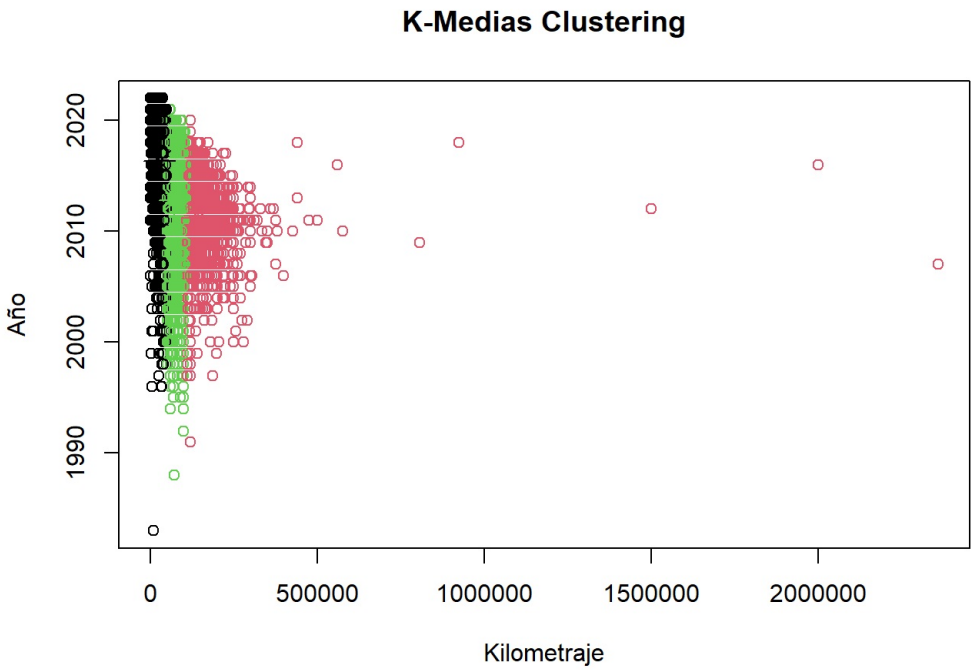
```
cat("Tamaño de los clusters:\n")
```

```
## Tamaño de los clusters:
```

```
print(kmeans_result$size)
```

```
## [1] 5321 2388 6515
```

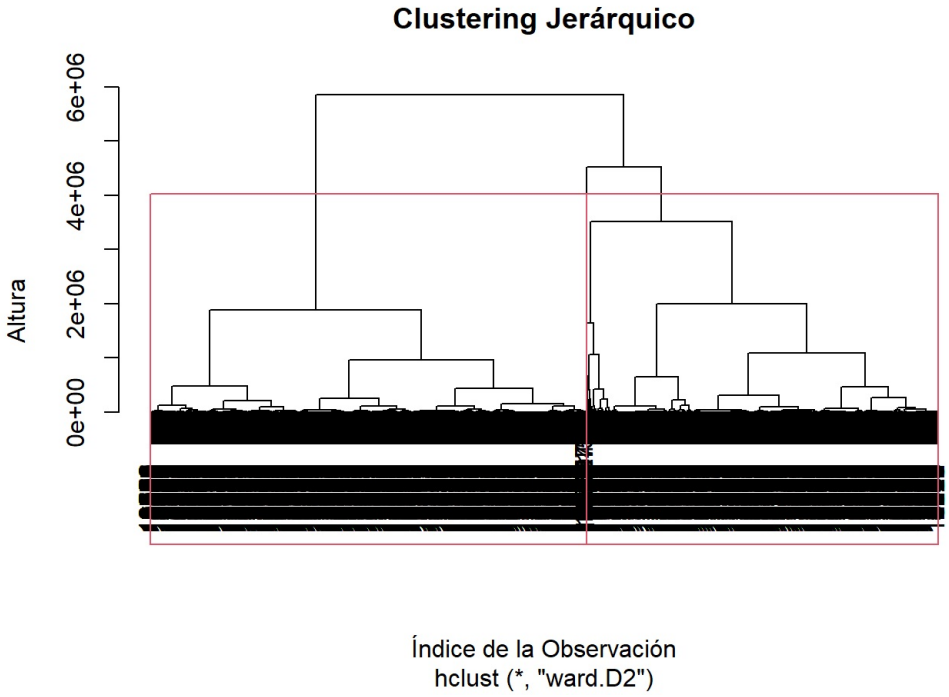
```
# Visualizar los clusters en un gráfico de dispersión
plot(variables_clustering, col = kmeans_result$cluster, main = "K-Medias Clustering", xlab = "Kilometraje", ylab
= "Año")
points(kmeans_result$centers, col = 1:k, pch = 8, cex = 2)
```



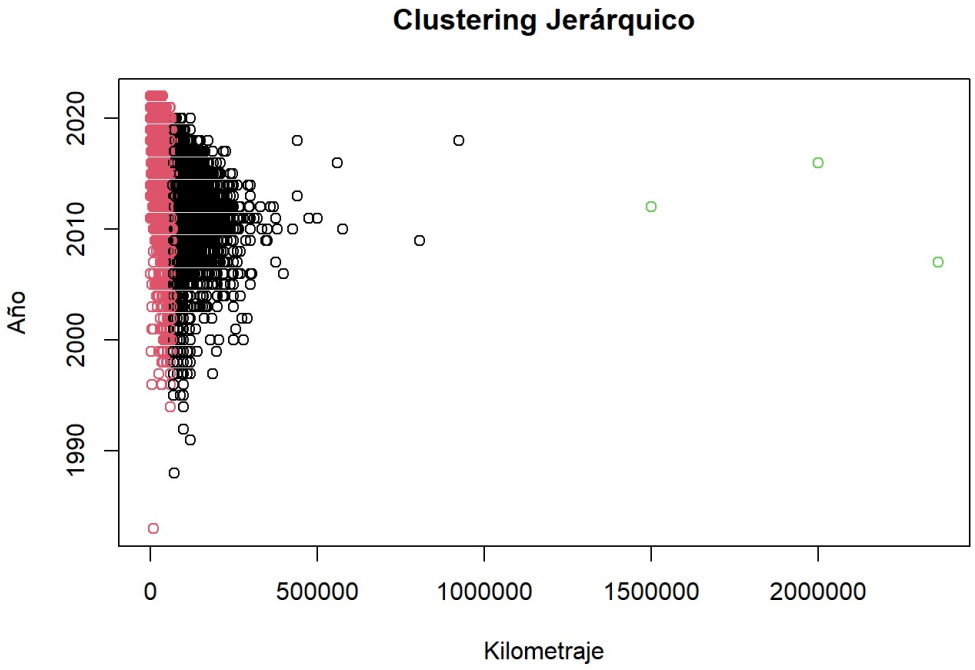
Podemos observar que los clusters se han creado atendiendo a los valores del año y el kilometraje, y los grupos creados son similares. El cluster de los vehiculos con mayor kilometraje muestra los valores más dispersos en comparación con los otros dos clusters.

A continuación, haremos clustering jerárquico usando un árbol.

```
# Seleccionar las variables relevantes para el clustering
variables_clustering <- car_dataset_total[, c("km_driven", "year")]

# Calcular la matriz de distancias
dist_matrix <- dist(variables_clustering)

# Aplicar clustering jerárquico
hierarchical_result <- hclust(dist_matrix, method = "ward.D2")

# Cortar el dendrograma para obtener clusters
k <- 3
clusters <- cutree(hierarchical_result, k)

# Agregar la columna de clusters al conjunto de datos
clusters_mutated <- mutate(variables_clustering, cluster = as.factor(clusters))

plot(hierarchical_result, main = "Clustering Jerárquico", xlab = "Índice de la Observación", ylab = "Altura")
rect.hclust(hierarchical_result, k = k, border = 2:k)
```




```
# Visualizar los clusters en un gráfico de dispersión
plot(variables_clustering, col = clusters_mutated$cluster, main = "Clustering Jerárquico", xlab = "Kilometraje",
ylab = "Año")
points(clusters_mutated$centers, col = 1:k, pch = 8, cex = 2)
```



Como podemos observar, a diferencia del clustering usando K-Medias, el clustering jerárquico representa dos grandes clusters y un tercero que toma valores muy alejados del resto, con kilometrajes muy altos.

Por último, compararemos los clusterings usando su valor de silueta.

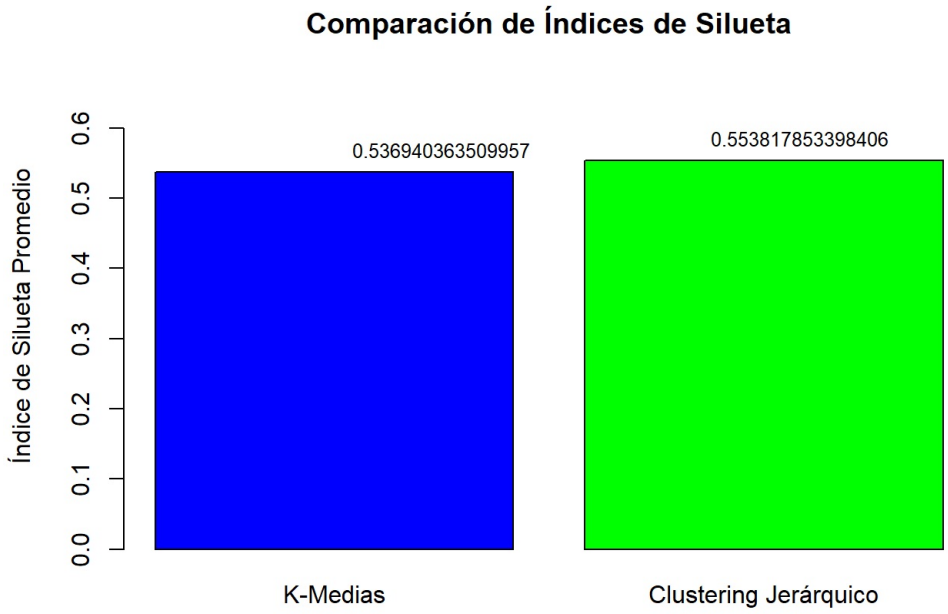
```
# K-Medias
silhouette_kmeans <- silhouette(kmeans_result$cluster, dist(variables_clustering))

# Clustering Jerárquico
silhouette_hierarchical <- silhouette(clusters, dist(variables_clustering))

# Calcular índices de Silueta promedio para cada método
avg_silhouette_kmeans <- mean(silhouette_kmeans[, "sil_width"])
avg_silhouette_hierarchical <- mean(silhouette_hierarchical[, "sil_width"])

cat("Índice promedio de Silueta de K-Means:", avg_silhouette_kmeans, "\n")
```

```
## Índice promedio de Silueta de K-Means: 0.5369404
```

```
cat("Índice promedio de Silueta de Clustering Jerárquico:", avg_silhouette_hierarchical, "\n")
```

```
## Índice promedio de Silueta de Clustering Jerárquico: 0.5538179
```

```
# Visualizar los resultados
avg_silhouettes <- c(avg_silhouette_kmeans, avg_silhouette_hierarchical)
barplot(avg_silhouettes,
  names.arg = c("K-Medias", "Clustering Jerárquico"),
  col = c("blue", "green"),
  main = "Comparación de Índices de Silueta",
  ylab = "Índice de Silueta Promedio",
  ylim = c(0, max(avg_silhouettes) * 1.2)
)

text(seq_along(avg_silhouettes), avg_silhouettes, labels = avg_silhouettes, pos = 3, col = "black", cex = 0.8)
```



El índice de silueta se usa como método para representar la coherencia de los clusterings realizados. Como podemos observar, los valores en ambos casos son similares, siendo ligeramente superior el de K-Medias.

Conclusión

Al finalizar el trabajo realizado en el presente proyecto, podemos concluir que los métodos usados para la predicción proporcionan resultados muy distintos, mientras que el clustering es similar en los dos métodos empleados. La predicción con random forest es bastante superior a la realizada con regresión linear y KNN, obteniendo valores mejores en el error cuadrático medio (MSE), el coheficiente de determinación (R^2) y el error absoluto medio (MAE). Sin embargo, es cierto que este método eleva el tiempo de ejecución notablemente con respecto a los otros dos. Por lo tanto, sería adecuado emplearlo siempre que el tiempo de ejecución no sea una restricción.