Precios de coches

Rodrigo Ponce de Leon

2022-10-30

Contenido:

1. Introducción.

Una empresa automovilística china aspira a entrar en el mercado estadounidense. Desea establecer allí una unidad de fabricación y producir automóviles localmente para competir con sus contrapartes estadounidenses y europeas. Contrataron una empresa de consultoría de automóviles para identificar los principales factores de los que depende el precio de los automóviles, específicamente, en el mercado estadounidense, ya que pueden ser muy diferentes del mercado chino. Esencialmente, la empresa quiere saber:

- Qué variables son significativas para predecir el precio de un automóvil.
- Qué tan bien describen esas variables el precio de un automóvil.

2. Llamado a librerías.

```
library(tidyverse) #manejo, limpieza y visualización de datos
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret) #Machine learning
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

3. Importación de base de datos.

```
df <- read.csv("precios_autos.csv")
```

4. Análisis del dataset.

```
head(df) #primeras cinco observaciones
```

```
##   symboling                  CarName fueltype     carbody drivewheel
## 1         3       alfa-romero giulia      gas convertible        rwd
## 2         3      alfa-romero stelvio      gas convertible        rwd
## 3         1 alfa-romero Quadrifoglio      gas   hatchback        rwd
## 4         2              audi 100 ls      gas       sedan        fwd
## 5         2               audi 100ls      gas       sedan        4wd
## 6         2                 audi fox      gas       sedan        fwd
##   enginelocation wheelbase carlength carwidth carheight curbweight enginetype
## 1          front      88.6     168.8     64.1      48.8       2548       dohc
## 2          front      88.6     168.8     64.1      48.8       2548       dohc
## 3          front      94.5     171.2     65.5      52.4       2823       ohcv
## 4          front      99.8     176.6     66.2      54.3       2337        ohc
## 5          front      99.4     176.6     66.4      54.3       2824        ohc
## 6          front      99.8     177.3     66.3      53.1       2507        ohc
##   cylindernumber enginesize stroke compressionratio horsepower peakrpm citympg
## 1           four        130   2.68              9.0        111    5000      21
## 2           four        130   2.68              9.0        111    5000      21
## 3            six        152   3.47              9.0        154    5000      19
## 4           four        109   3.40             10.0        102    5500      24
## 5           five        136   3.40              8.0        115    5500      18
## 6           five        136   3.40              8.5        110    5500      19
##   highwaympg price
## 1         27 13495
## 2         27 16500
## 3         26 16500
```

```
## 4         30 13950
## 5         22 17450
## 6         25 15250
```

```
str(df)#variables e info
```

```
## 'data.frame':    205 obs. of  21 variables:
##  $ symboling       : int  3 3 1 2 2 2 1 1 1 0 ...
##  $ CarName         : chr  "alfa-romero giulia" "alfa-romero stelvio" "alfa-romero Quadrifoglio" "audi
##  $ fueltype        : chr  "gas" "gas" "gas" "gas" ...
##  $ carbody         : chr  "convertible" "convertible" "hatchback" "sedan" ...
##  $ drivewheel      : chr  "rwd" "rwd" "rwd" "fwd" ...
##  $ enginelocation  : chr  "front" "front" "front" "front" ...
##  $ wheelbase       : num  88.6 88.6 94.5 99.8 99.4 ...
##  $ carlength       : num  169 169 171 177 177 ...
##  $ carwidth        : num  64.1 64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 67.9 ...
##  $ carheight       : num  48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 52 ...
##  $ curbweight      : int  2548 2548 2823 2337 2824 2507 2844 2954 3086 3053 ...
##  $ enginetype      : chr  "dohc" "dohc" "ohcv" "ohc" ...
##  $ cylindernumber  : chr  "four" "four" "six" "four" ...
##  $ enginesize      : int  130 130 152 109 136 136 136 136 131 131 ...
##  $ stroke          : num  2.68 2.68 3.47 3.4 3.4 3.4 3.4 3.4 3.4 3.4 ...
##  $ compressionratio: num  9 9 9 10 8 8.5 8.5 8.5 8.3 7 ...
##  $ horsepower      : int  111 111 154 102 115 110 110 110 140 160 ...
##  $ peakrpm         : int  5000 5000 5000 5500 5500 5500 5500 5500 5500 5500 ...
##  $ citympg         : int  21 21 19 24 18 19 19 19 17 16 ...
##  $ highwaympg      : int  27 27 26 30 22 25 25 25 20 22 ...
##  $ price           : num  13495 16500 16500 13950 17450 ...
```

```
summary(df) #resumen estadístico
```

```
##    symboling         CarName            fueltype           carbody
##  Min.   :-2.0000   Length:205         Length:205         Length:205
##  1st Qu.: 0.0000   Class :character   Class :character   Class :character
##  Median : 1.0000   Mode  :character   Mode  :character   Mode  :character
##  Mean   : 0.8341
##  3rd Qu.: 2.0000
##  Max.   : 3.0000
##   drivewheel        enginelocation       wheelbase        carlength
##  Length:205         Length:205         Min.   : 86.60   Min.   :141.1
##  Class :character   Class :character   1st Qu.: 94.50   1st Qu.:166.3
##  Mode  :character   Mode  :character   Median : 97.00   Median :173.2
##                                        Mean   : 98.76   Mean   :174.0
##                                        3rd Qu.:102.40   3rd Qu.:183.1
##                                        Max.   :120.90   Max.   :208.1
##     carwidth       carheight       curbweight    enginetype
##  Min.   :60.30   Min.   :47.80   Min.   :1488   Length:205
##  1st Qu.:64.10   1st Qu.:52.00   1st Qu.:2145   Class :character
##  Median :65.50   Median :54.10   Median :2414   Mode  :character
##  Mean   :65.91   Mean   :53.72   Mean   :2556
##  3rd Qu.:66.90   3rd Qu.:55.50   3rd Qu.:2935
##  Max.   :72.30   Max.   :59.80   Max.   :4066
##  cylindernumber       enginesize        stroke      compressionratio
```

```
##  Length:205         Min.   : 61.0   Min.   :2.070   Min.   : 7.00
##  Class :character   1st Qu.: 97.0   1st Qu.:3.110   1st Qu.: 8.60
##  Mode  :character   Median :120.0   Median :3.290   Median : 9.00
##                     Mean   :126.9   Mean   :3.255   Mean   :10.14
##                     3rd Qu.:141.0   3rd Qu.:3.410   3rd Qu.: 9.40
##                     Max.   :326.0   Max.   :4.170   Max.   :23.00
##    horsepower       peakrpm        citympg        highwaympg        price
##  Min.   : 48.0   Min.   :4150   Min.   :13.00   Min.   :16.00   Min.   : 5118
##  1st Qu.: 70.0   1st Qu.:4800   1st Qu.:19.00   1st Qu.:25.00   1st Qu.: 7788
##  Median : 95.0   Median :5200   Median :24.00   Median :30.00   Median :10295
##  Mean   :104.1   Mean   :5125   Mean   :25.22   Mean   :30.75   Mean   :13277
##  3rd Qu.:116.0   3rd Qu.:5500   3rd Qu.:30.00   3rd Qu.:34.00   3rd Qu.:16503
##  Max.   :288.0   Max.   :6600   Max.   :49.00   Max.   :54.00   Max.   :45400
```

Se observa que el dataset cuenta con 205 observaciones y 21 variables, de las cuales 8 son categóricas y 13 son numéricas.

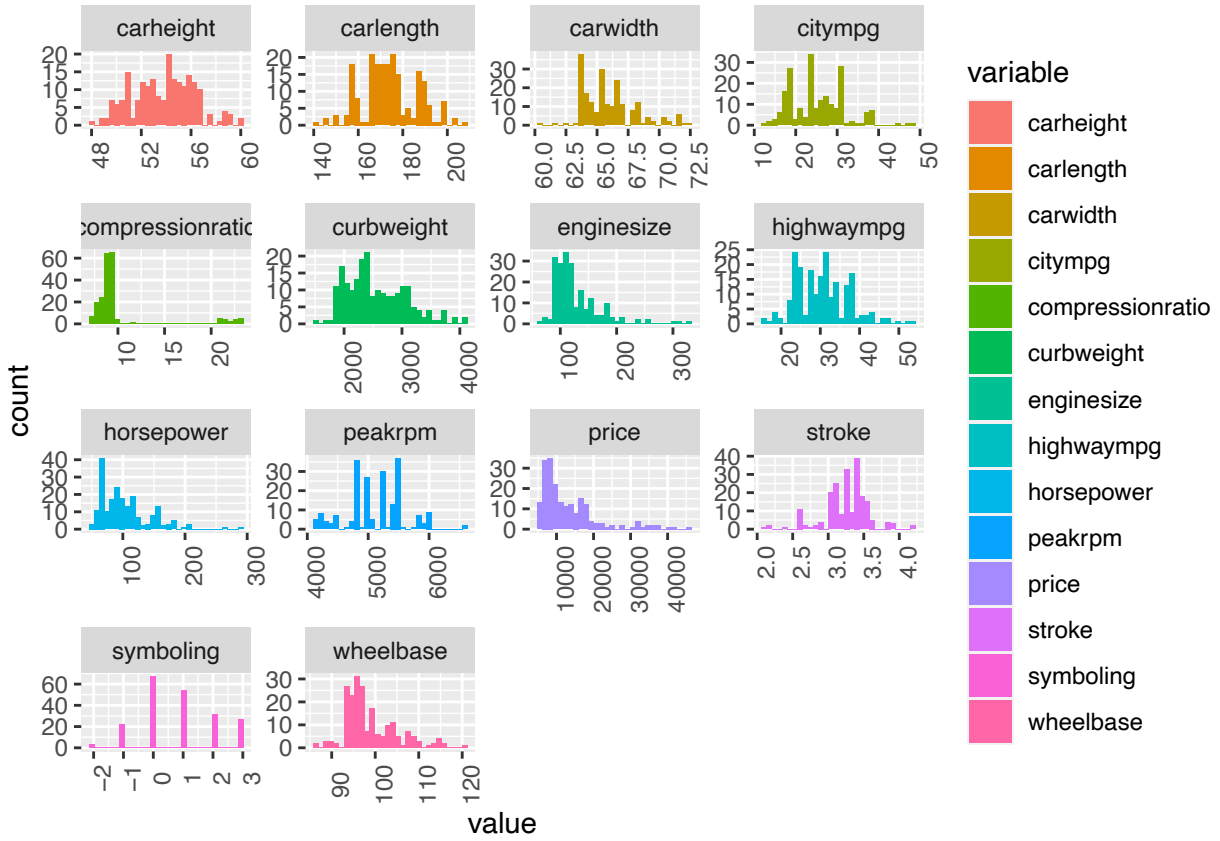
```
sum(duplicated(df)) #número de observaciones duplicadas
```

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

df no cuenta con observaciones duplicadas.

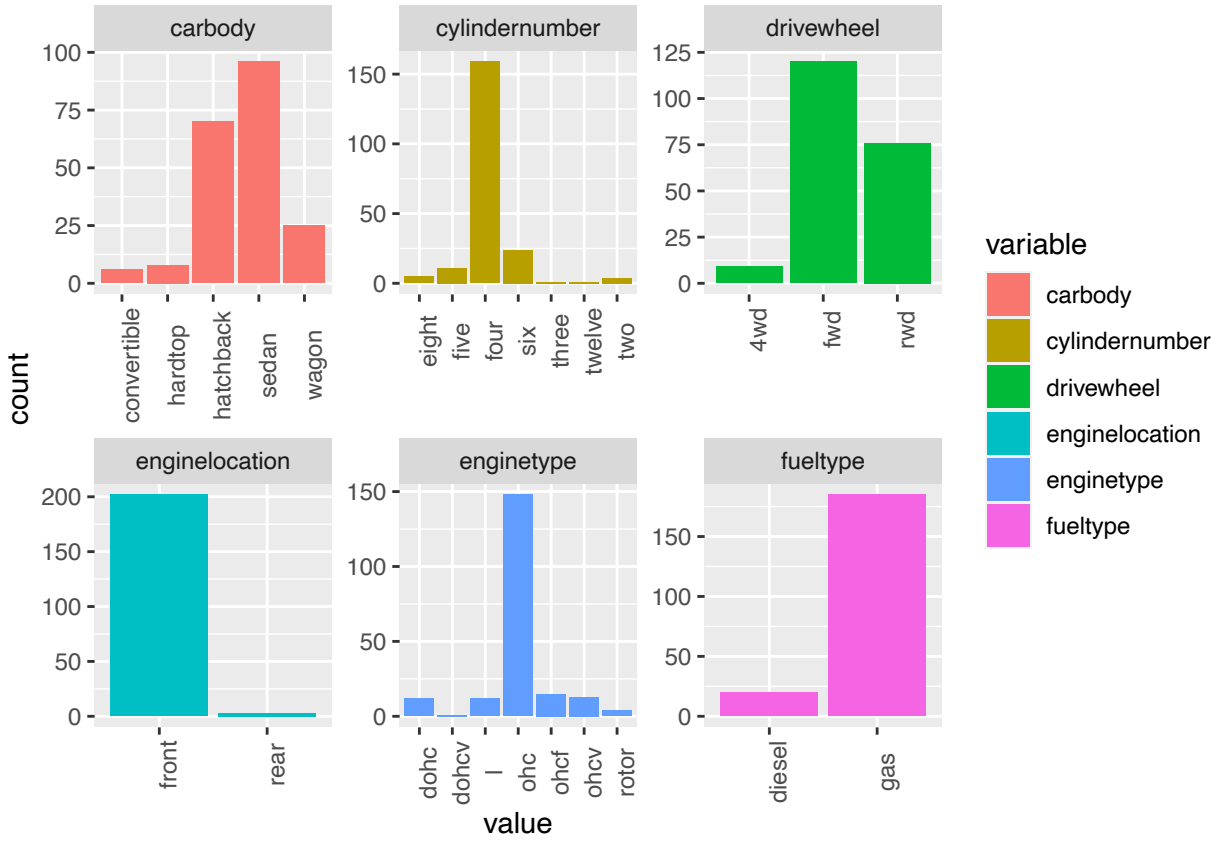
```
nums <- unlist(lapply(df, is.numeric), use.names = F)
df[,nums] %>% gather(key = "variable", value = "value") %>%
  ggplot(aes(value, fill = variable)) + geom_histogram() +
  theme(axis.text.x = element_text(angle = 90)) +
  facet_wrap(~ variable, scales = "free") #histograma para variables numéricas
```

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



Se observa que todas las variables son numéricas, mientras que symboling es una variable categórica no ordinal, dado que se generó un gráfico de barras en lugar de un histograma.

```
cat <- unlist(lapply(df, is.character), use.names = F)
df[,cat] %>% select(-CarName) %>%
  gather(key = "variable", value = "value") %>%
  ggplot(aes(value, fill = variable)) + geom_bar() +
  theme(axis.text.x = element_text(angle = 90)) +
  facet_wrap(~ variable, scales = "free") #histograma para variables no numéricas
```



A continuación se describen los resultados del gráfico anterior:

- El cuerpo de auto más común, es el tipo sedan.
- El número de cilíndros más común es 4.
- La tracción más común es la delantera.
- El más común que el motor se encuentre hacia el frente.
- Los motores ohc son los más comunes.
- La mayoría de los autos utiliza gasolina regular.

La variable CarName con 147 categorías distintas, por lo que no es conveniente visualizarla. Por otro lado, se puede mostrar en una tabla las categorías que cuentan con el mayor número de observaciones:

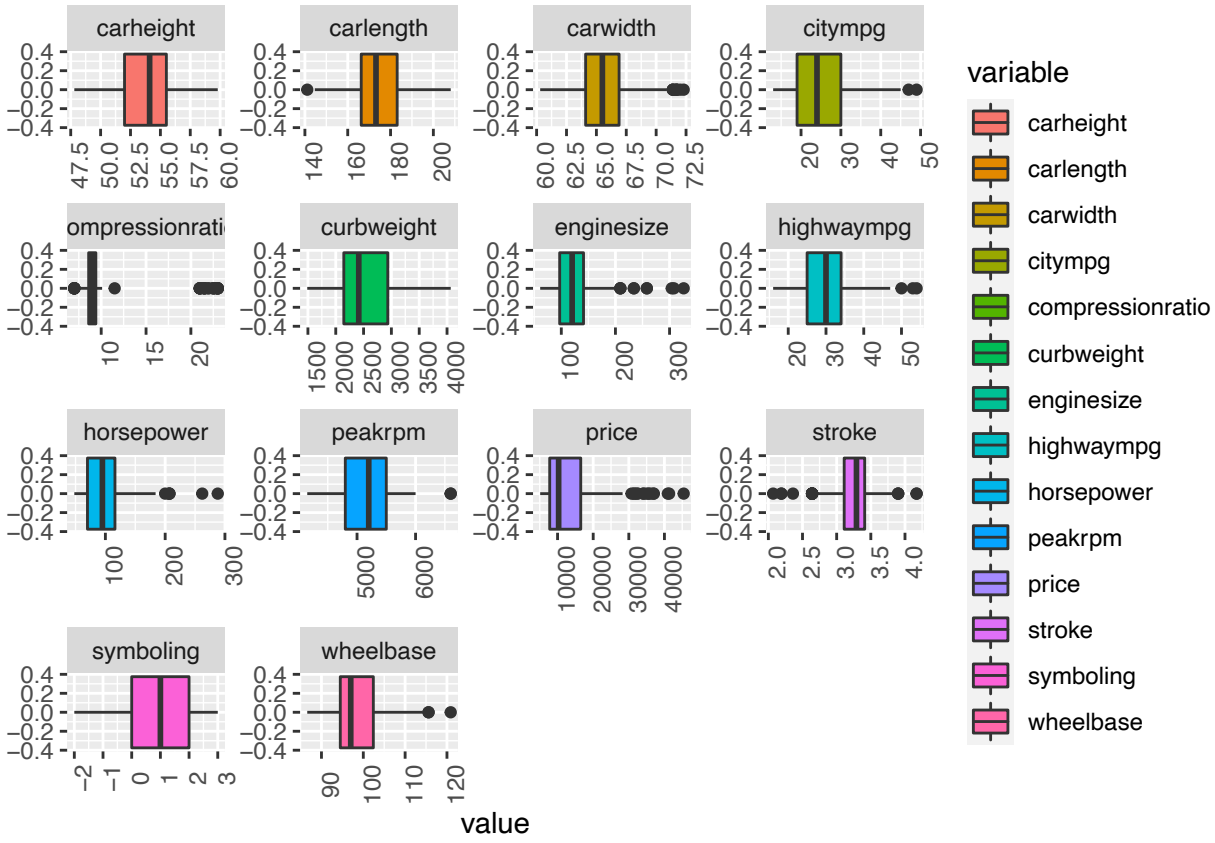
```
df %>% group_by(CarName) %>% summarize(n = n()) %>% arrange(desc(n))
```

```
## # A tibble: 147 x 2
##    CarName                n
##    <chr>              <int>
##  1 peugeot 504            6
##  2 toyota corolla         6
##  3 toyota corona          6
##  4 subaru dl              4
##  5 honda civic            3
```

```
##  6 mazda 626                3
##  7 mitsubishi g4            3
##  8 mitsubishi mirage g4     3
##  9 mitsubishi outlander     3
## 10 toyota mark ii           3
## # ... with 137 more rows
```

Se puede observar que los modelos peugeot 504, toyota corolla, toyota corona, cuentan con la mayor cantidad de observaciones.

```
df[,nums] %>% gather(key = "variable", value = "value") %>%
  ggplot(aes(value, fill = variable)) + geom_boxplot() +
  theme(axis.text.x = element_text(angle = 90)) +
  facet_wrap(~ variable, scales = "free") #boxplots para variables numéricas
```



Se observa que carheight, curbweight y symboling son las únicas variables que no cuentan con valores atípicos.

```
df %>% gather(key = "variable", value = "value") %>%
  group_by(variable) %>% summarise(na_num = sum(is.na(value))) #número de valores nulos por variable
```

```
## # A tibble: 21 x 2
##    variable         na_num
##    <chr>             <int>
##  1 carbody               0
##  2 carheight             0
```

```
##  3 carlength             0
##  4 CarName               0
##  5 carwidth              0
##  6 citympg               0
##  7 compressionratio      0
##  8 curbweight            0
##  9 cylindernumber        0
## 10 drivewheel            0
## # ... with 11 more rows
```

```
sum(is.na(df)) #número total de valores nulos en el dataset
```

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

Se observa que df no cuenta con valores nulos, facilitando el proceso de preparación para el entrenamiento de algoritmos inteligentes.

5. Creación de dataset de entrenamiento.

En este paso se realiza una separación estratificada, utilizando la función createDataPartition. Dada la baja cantidad de observaciones (205), se toma el 30% de las observaciones y se asignan al dataset de prueba.

```
y <- df$price #variable a predecir
set.seed(42, sample.kind = "default") #se ajusta la semilla para reproducibilidad
test_index <- createDataPartition(y, times=1, p=0.3, list=F) #índices de observaciones para test_set
train_data <- df[-test_index,] #datos de entrenamiento
test_data <- df[test_index,] #datos de prueba
```

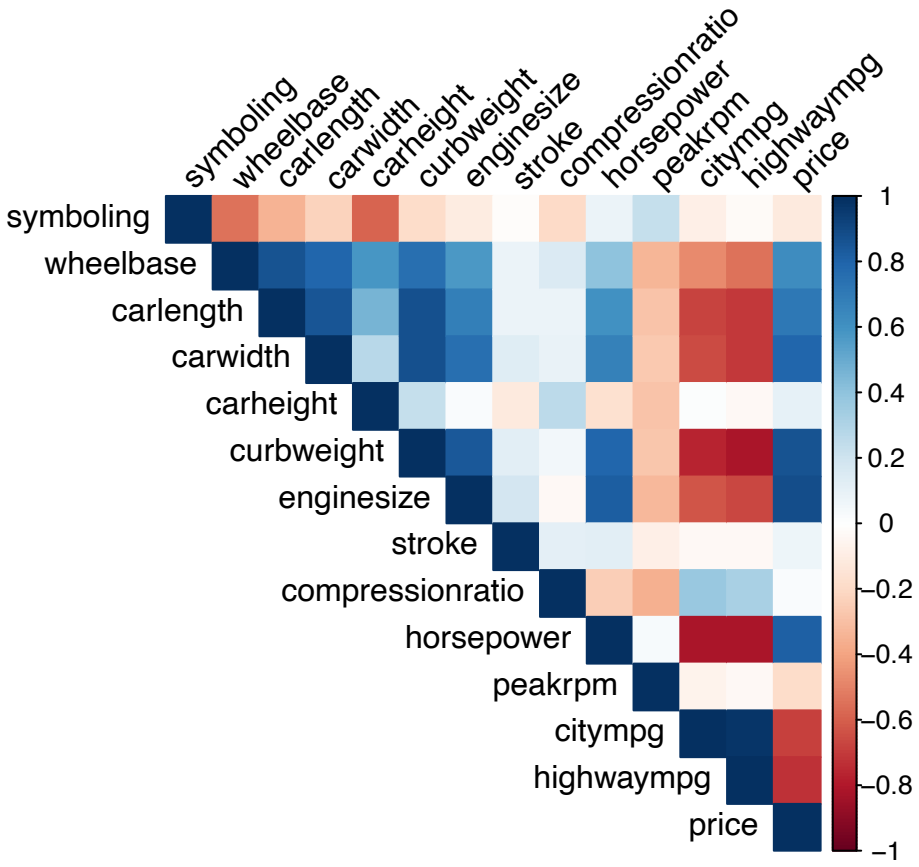
6. Análisis de correlación entre variables y selección de predictores.

Utilizando train_set, se determina la correlación entre las variables numéricas.

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
cor_mat <- cor(train_data[,nums], use = "complete.obs")
corrplot(cor_mat, type = "upper",
         tl.col = "black", tl.srt = 45, method="color")
```



Se puede observar que muchos de los predictores se correlacionan entre sí, lo cual no es bueno dados los problemas de colinealidad. Por lo tanto, se realiza una PCA para determinar los coeficientes de las combinaciones lineales y así elegir las variables más representativas. La función prcomp realiza esta tarea y, además, escala los datos previo al ajuste del modelo no supervisado.

```
pca <- prcomp(train_data[,nums], scale = TRUE,
              center = TRUE, retx = T) #pca
```

```
pca$rotation #matriz a analizar
```

```
##                          PC1          PC2         PC3         PC4         PC5
## symboling        -0.08454431  0.434966341  0.26218081  0.32916614 -0.15469892
## wheelbase         0.30112937 -0.287336601 -0.14268171 -0.12821344 -0.01966354
## carlength         0.33956794 -0.152607416 -0.09421660 -0.04861556 -0.07097619
## carwidth          0.34207192 -0.078366683  0.05343664  0.02000477 -0.13616955
## carheight         0.09609673 -0.468712329 -0.35896368 -0.08350314 -0.02413341
## curbweight        0.36365974 -0.016764222  0.06245596  0.05774949 -0.05879183
## enginesize        0.32782691  0.059715576  0.24000374  0.12096981  0.16569816
## stroke            0.04715772 -0.002064682  0.56978342 -0.78435641  0.07958706
## compressionratio -0.01519120 -0.396701756  0.41975490  0.22941146 -0.65486545
## horsepower        0.31105475  0.275071715  0.08021574  0.02634487 -0.05207406
## peakrpm          -0.09190743  0.340932083 -0.36461347 -0.38722802 -0.68278765
## citympg          -0.30834853 -0.267063261  0.16741249  0.04931796 -0.03398305
## highwaympg       -0.32331224 -0.228186676  0.16619530  0.05181592 -0.01162715
## price             0.33709645  0.051046442  0.11375746  0.15461075 -0.12338342
```

```
##                         PC6           PC7         PC8          PC9
## symboling        -0.61835487  0.3883040548  0.17529780 -0.001025889
## wheelbase        -0.06226823 -0.0703984888  0.42382589 -0.203950075
## carlength        -0.20839924 -0.0541244656  0.14760502  0.548312606
## carwidth         -0.07041540  0.0001490622  0.56710200 -0.075444332
## carheight        -0.31246566  0.5793470699 -0.38304253  0.001279576
## curbweight       -0.04212181 -0.0470900732 -0.07962872  0.116124880
## enginesize        0.34470549  0.3188146049 -0.02248558  0.103754991
## stroke           -0.17058878  0.0973519409 -0.07573194 -0.061671346
## compressionratio -0.04374520 -0.2767185076 -0.26171149 -0.008150479
## horsepower        0.23909894  0.0701192079 -0.23648146  0.476096028
## peakrpm           0.20770614  0.1749228703  0.09439160  0.031024280
## citympg           0.27150374  0.3163630077  0.29366815  0.149170023
## highwaympg        0.22452745  0.2875103522  0.23903693  0.267430875
## price             0.30280235  0.3122791652 -0.10119954 -0.550038492
##                        PC10          PC11         PC12         PC13
## symboling         0.154108899 -0.115771768  0.038629678  0.00589122
## wheelbase         0.460145504 -0.533728481  0.238470044  0.01866396
## carlength         0.336107101  0.562802767  0.105398921 -0.09339706
## carwidth         -0.711428257  0.092332207 -0.014350012 -0.07704231
## carheight        -0.207246113 -0.094701548  0.007658627 -0.04619253
## curbweight        0.059774157 -0.109814436 -0.642484186  0.62779047
## enginesize        0.142812045 -0.127792655 -0.362034706 -0.61048641
## stroke           -0.008801422  0.035960332  0.029083990  0.03234329
## compressionratio  0.004099089 -0.086853837  0.030551937 -0.16934250
## horsepower       -0.210896182 -0.370074296  0.497287089  0.19306221
## peakrpm           0.076470274 -0.007902797 -0.165613235 -0.08262282
## citympg           0.020758741 -0.110574397 -0.122831310  0.16263676
## highwaympg        0.056250292  0.125546344  0.147399979  0.23394856
## price             0.156476970  0.406518517  0.270361688  0.24699187
##                        PC14
## symboling        -0.009104429
## wheelbase         0.075510207
## carlength        -0.163664873
## carwidth          0.044675928
## carheight         0.023479502
## curbweight        0.102087790
## enginesize        0.130567405
## stroke           -0.013800864
## compressionratio  0.022529882
## horsepower       -0.084051216
## peakrpm           0.049540633
## citympg          -0.682272646
## highwaympg        0.675160508
## price            -0.076534453
```

```r
#función para determinar valor máximo (en valor absoluto) por componente principal
get_max_pca <- function(m){
  df_m <- data.frame(m)
  df_m[, "index"] <- df_m %>% row.names
  res <- data.frame(matrix(nrow=0, ncol=0))
  c <- 1
  for (i in df_m %>% select(-index) %>% names()){
    res[c, "val"] <- df_m[abs(df_m[i])==max(abs(df_m[i])), i]
```

```
    res[c, "PC"] <- i
    res[c, "index"] <- df_m[abs(df_m[i])==max(abs(df_m[i])), "index"]
    c <- c + 1
  }
  return(res)
}
```

```
#resultados
re <- get_max_pca(pca$rotation)
```

```
re #dataset con resultados
```

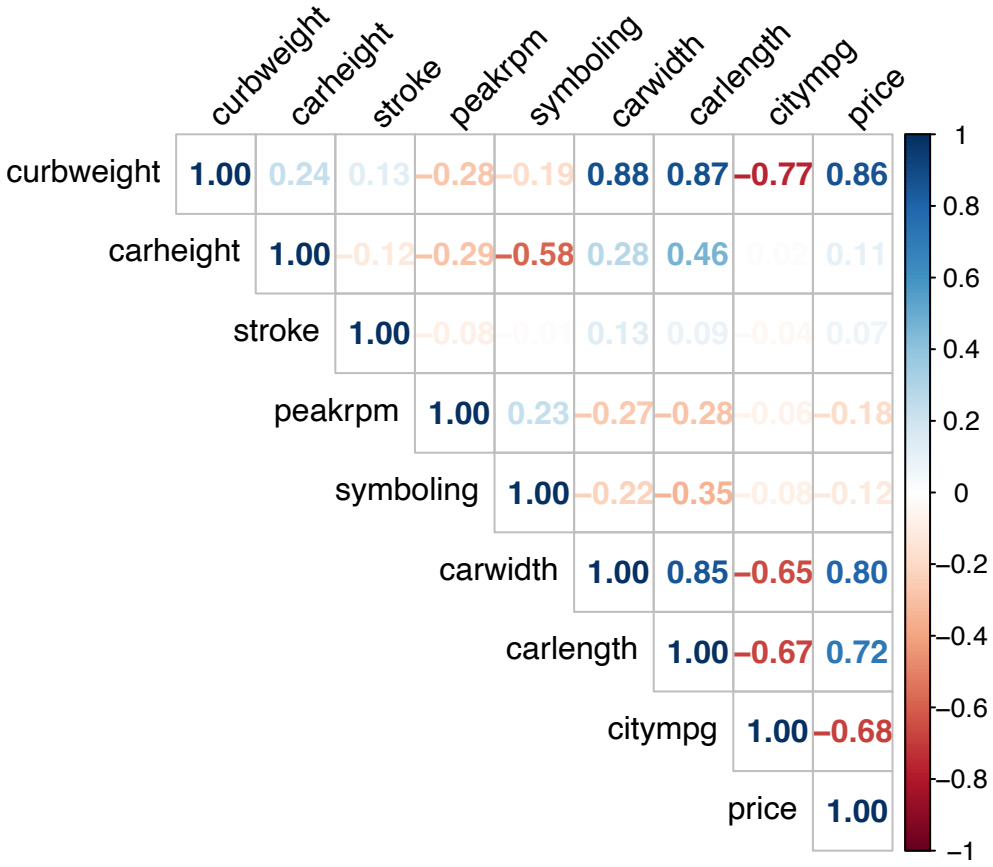
```
##           val   PC      index
## 1   0.3636597  PC1 curbweight
## 2  -0.4687123  PC2  carheight
## 3   0.5697834  PC3     stroke
## 4  -0.7843564  PC4     stroke
## 5  -0.6827877  PC5    peakrpm
## 6  -0.6183549  PC6  symboling
## 7   0.5793471  PC7  carheight
## 8   0.5671020  PC8   carwidth
## 9  -0.5500385  PC9      price
## 10 -0.7114283 PC10   carwidth
## 11  0.5628028 PC11  carlength
## 12 -0.6424842 PC12 curbweight
## 13  0.6277905 PC13 curbweight
## 14 -0.6822726 PC14    citympg
```

```
nums_new <- re$index %>% unique #variables con coeficiente más alto
nums_new <- c(nums_new[nums_new != "price"], nums_new[nums_new == "price"])
nums_new
```

```
## [1] "curbweight" "carheight"  "stroke"     "peakrpm"    "symboling"
## [6] "carwidth"   "carlength"  "citympg"    "price"
```

Una vez teniendo los resultados (nums_new), se visualiza la matriz de correlación con las 9 variables seleccionadas, para determinar si hay correlación entre predictores nuevamente y elegir únicamente a las variables que tengan una correlación más fuerte con price y que no se correlacionen entre ellas mismas.

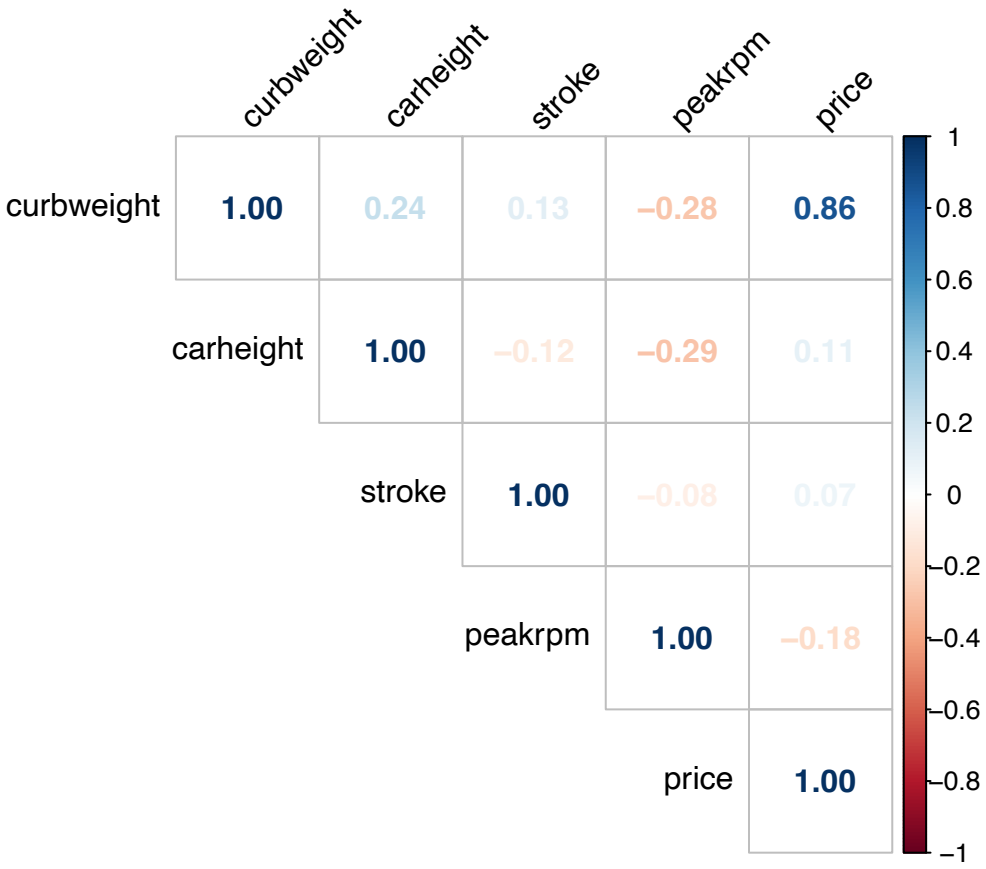
```
cor_mat <- cor(train_data[,nums_new],
               use = "complete.obs")
corrplot(cor_mat, type = "upper",
         tl.col = "black", tl.srt = 45, method="number")
```



Se puede observar que las variables curbweight, carwidth, carlength y citympg se correlacionan entre sí. Por lo tanto, tomando en consideración que curbweight cuenta con la mayor correlación con price, se descartan carwidth, carlength y citympg.

Por otro lado, se mantiene la variable carheight y se descarta la variable symboling debido a que la primera es numérica.

```
cor_mat <- cor(train_data[,nums_new[!nums_new %in% c("carwidth",
                                                     "carlength",
                                                     "citympg", "symboling")]],
               use = "complete.obs")
corrplot(cor_mat, type = "upper",
         tl.col = "black", tl.srt = 45, method="number")
```



Como se puede observar, la nueva matriz de correlación ya no muestra correlación entre los predictores. Sin embargo, ahora se necesita elegir algún otro predictor categórico que ayude a realizar el análisis predictivo.

Observando los gráficos de barras, "enginelocation", "fueltype" pueden fungir como variables categóricas.

7. Preparación de los datos.

Para que los datos numéricos puedan ser comparables, es necesario que se encuentren en la misma escala, pero sin cambiar su distribución. Ergo, se necesita un método de escalamiento, que en este caso es la estandarización.

Por otro lado, las variables categóricas se transforman a dummy dado que ambas cuentan con 2 categorías.

```
#función para escalamiento de datos
ScaleData <- function(df){
  df_pred <- df %>% select(curbweight, carheight, stroke, peakrpm) %>% scale() #dataframe con predictor
  return(data.frame(df_pred))
}
```

```
#función para variable categórica con 2 categorías
DummyData <- function(df){
  df1 <- df
  #df1["fueltype"] <- factor(df1$fueltype %>% factor %>% as.numeric - 1)
  df1["fueltype"] <- df1$fueltype %>% factor %>% as.numeric - 1
  df1["enginelocation"] <- df1$enginelocation %>% factor %>% as.numeric - 1
  #return(df1["fueltype"])
```

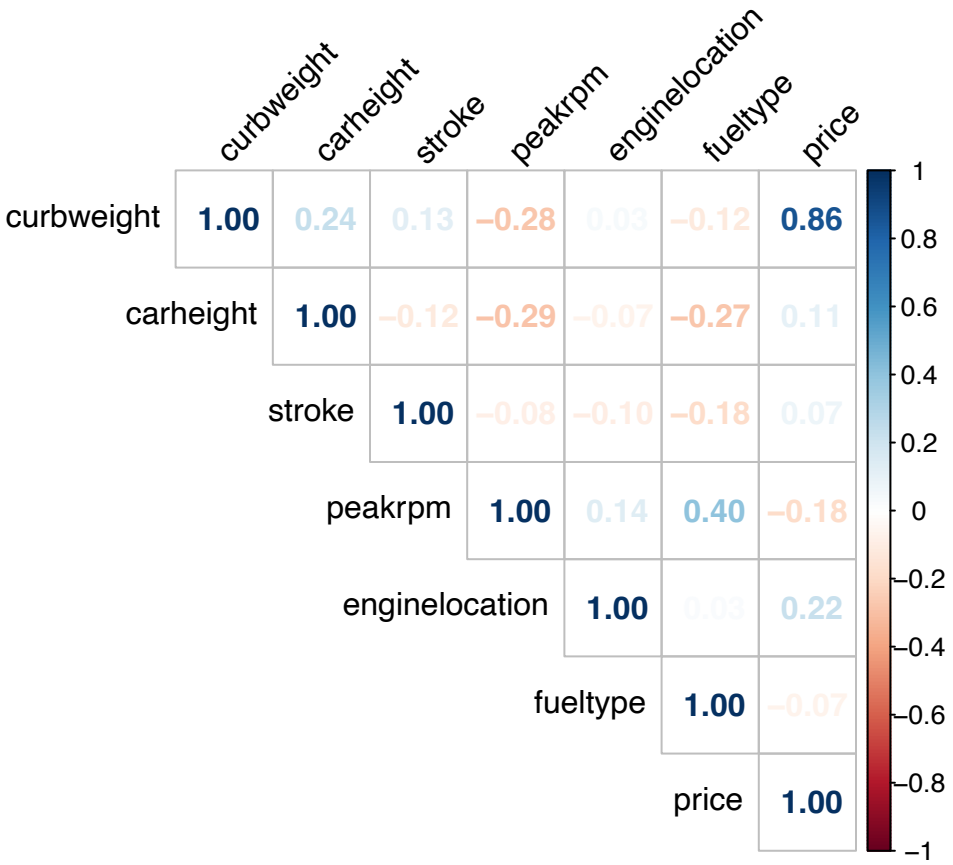
```
  return(df1[c("enginelocation", "fueltype")])
}
```

```
#función para tener dataset preparado
JoinData <- function(df){
  df_num <- ScaleData(df)
  df_cat1 <- DummyData(df)
  #return(cbind(df_num, df_cat, df_cat1,"price"=df["price"]))
  return(cbind(df_num, df_cat1, "price"=df["price"]))
}
```

```
#Datos preparados
train_data_prep <- JoinData(train_data)
test_data_prep <- JoinData(test_data)
```

Teniendo los datos preparados, se debe revisar que no hay correlación entre los predictores. Se observa que los predictores no están correlacionados:

```
cor_mat <- cor(train_data_prep,
               use = "complete.obs")
corrplot(cor_mat, type = "upper",
         tl.col = "black", tl.srt = 45, method="number")
```



8. Entrenamiento de modelos y evaluación.

Teniendo los datos preparados, se entrenan 2 diferentes algoritmos (randomforest regressor y xgboost regressor), realizando validación cruzada con $k = 5$ y tuneo de hiperparámetros, y un modelo de regresión multilineal.

Se evalúa para randomforest regressor y xgboost regressor utilizando RMSE, donde entre más pequeño sea el resultado mejor el modelo.

$$[\frac{(x_i - \bar{x})^2}{n}]^{1/2}$$

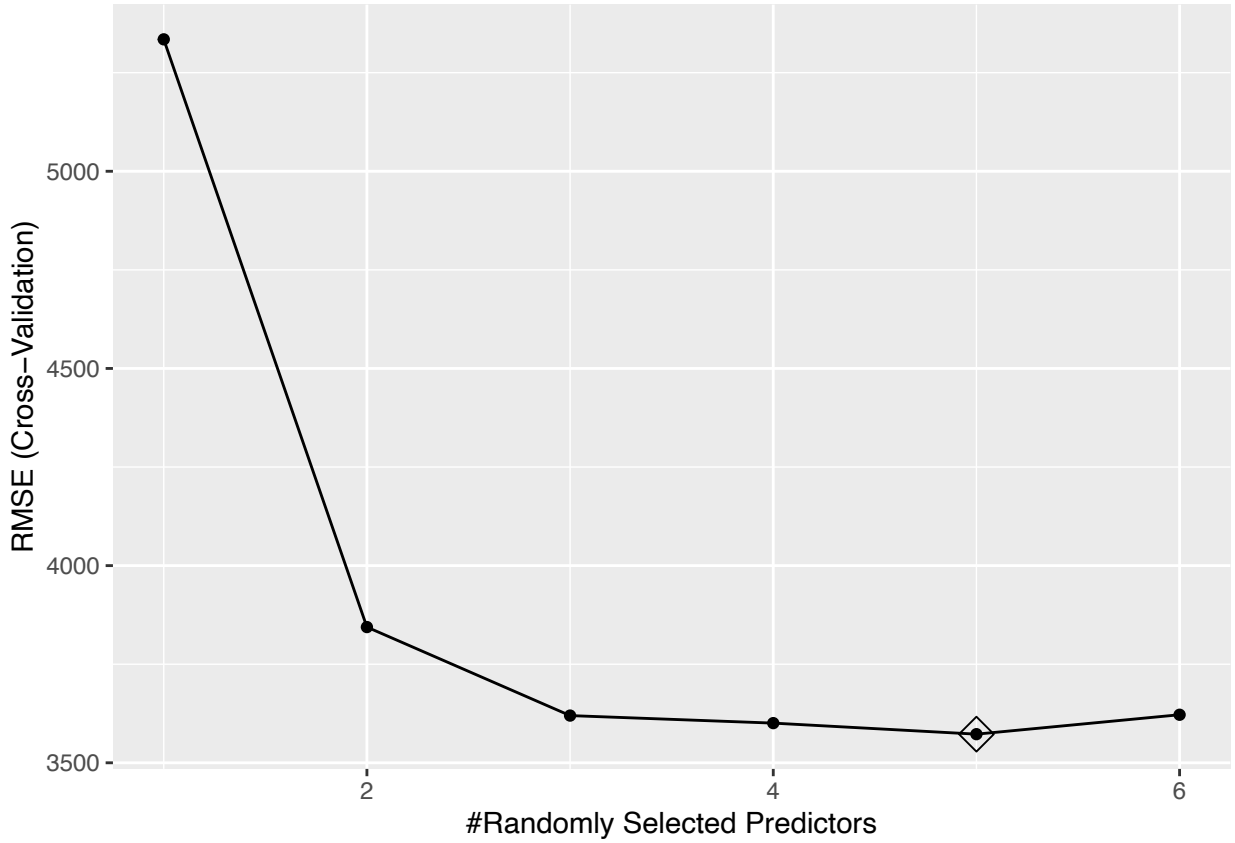
Por otro lado, para el modelo lineal se determina el coeficiente de determinación y se realiza un análisis de residuos para determinar si éste es adecuado o no.

- Random Forest:

```
train_control <- trainControl(method = "cv", number = 5) #para realizar validación cruzada en los algor
```

```
#random forest
train_rf <- train(price ~. , method = "ranger",
                  data = train_data_prep,
                  tuneGrid = expand.grid(
                    mtry = seq(1:6),
                    splitrule = "variance", #dado que es un problema de regresión
                    min.node.size = 5 #para problemas de regresión
                  ),
                  trControl = train_control) #cross-validation
```

```
ggplot(train_rf, highlight = T)#resultados
```




```
pred_rf <- predict(train_rf, test_data_prep) #predicciones
```

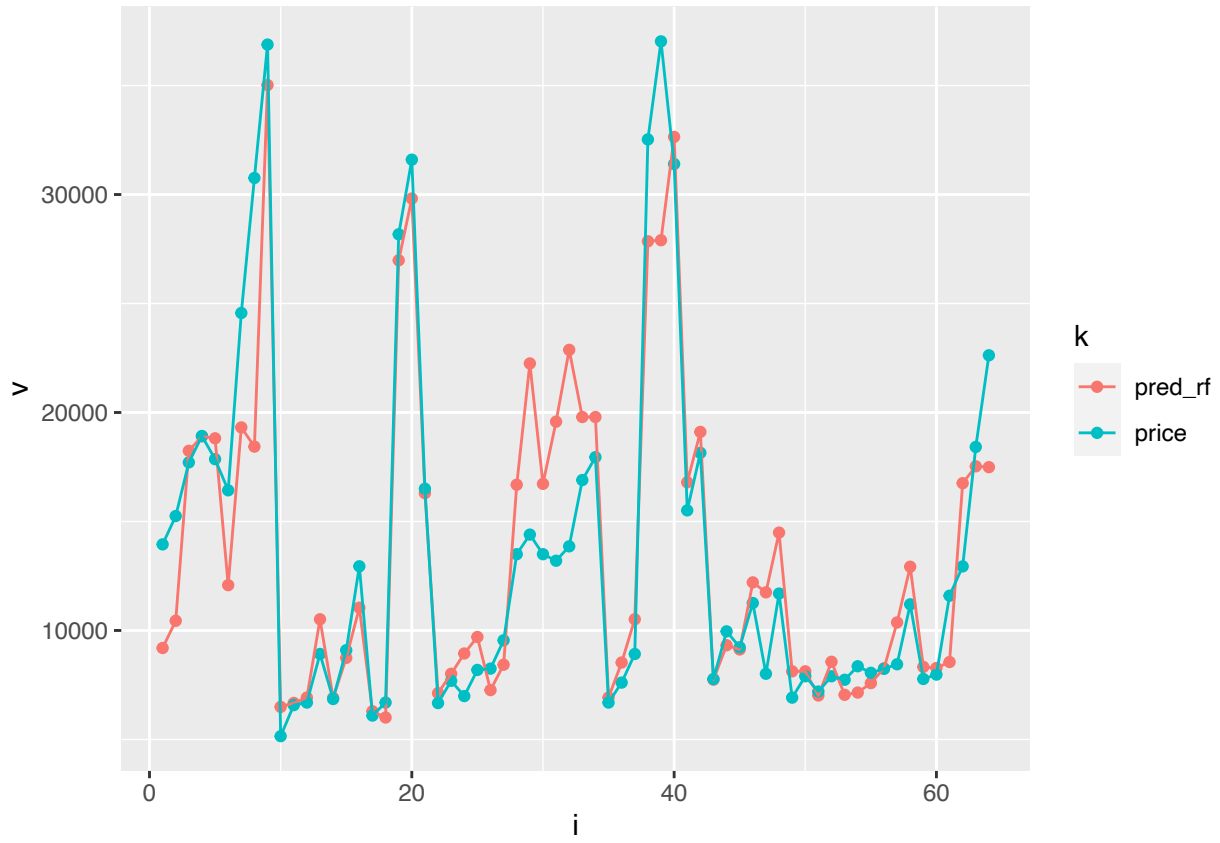
```
RMSE(pred_rf, test_data_prep$price) #métrica
```

```
## [1] 3270.414
```

```
R2(pred_rf, test_data_prep$price) #métrica R^2
```

```
## [1] 0.8338249
```

```
#visualización de resultados
data.frame(i = seq(1:nrow(test_data_prep)),pred_rf,
      price = test_data_prep$price) %>% gather(key = "k",
                                               value = "v",
                                               -i) %>%
  ggplot(aes(i, v, col = k)) +
  geom_point() +
  geom_line()
```



Se observa que el modelo con $mtry = 4$ se obtienen los mejores resultados con un $rmse = 3316.399$ y $r^2 = 0.8299176$. Por otro lado, se observa cómo las predicciones tienen una tendencia aproximada a la de los valores verdaderos.

*Xgboost:

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```
train_xgb <- train(price ~., method = "xgbDART",
                   data = train_data_prep,
                   trControl = train_control,
                   tuneGrid = expand.grid(
                     nrounds = c(11),
                     max_depth = c(6, 7, 8),#
                     eta = c(0, 0.01, 0.1, 0.2), #
                     gamma = c(0, 0.001, 0.01), #
                     subsample = 0.5,
                     colsample_bytree = c(0.5, 0.8, 1),#
                     rate_drop = c(0, 0.4,0.5), #
```

```
                    skip_drop = c(0, 0.4,0.5), #
                    min_child_weight = c(0,1,2)))
```

```
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
## [14:21:45] WARNING: amalgamation/../src/c_api/c_api.cc:785: ‘ntree_limit‘ is deprecated, use ‘iterat
```

```
## [14:23:04] WARNING: amalgamation/../src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iterati
## [14:23:04] WARNING: amalgamation/../src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iterati
## [14:23:04] WARNING: amalgamation/../src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iterati
## [14:23:04] WARNING: amalgamation/../src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iterati
```

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

```
train_xgb$bestTune #mejor modelo
```

```
##     nrounds max_depth eta gamma subsample colsample_bytree rate_drop skip_drop
## 764      11         6 0.2 0.001       0.5                1         0       0.4
##     min_child_weight
## 764                0
```

```
pred_xgb <- predict(train_xgb, test_data_prep) #predicciones
```

```
## [14:23:07] WARNING: amalgamation/../src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iterati
```

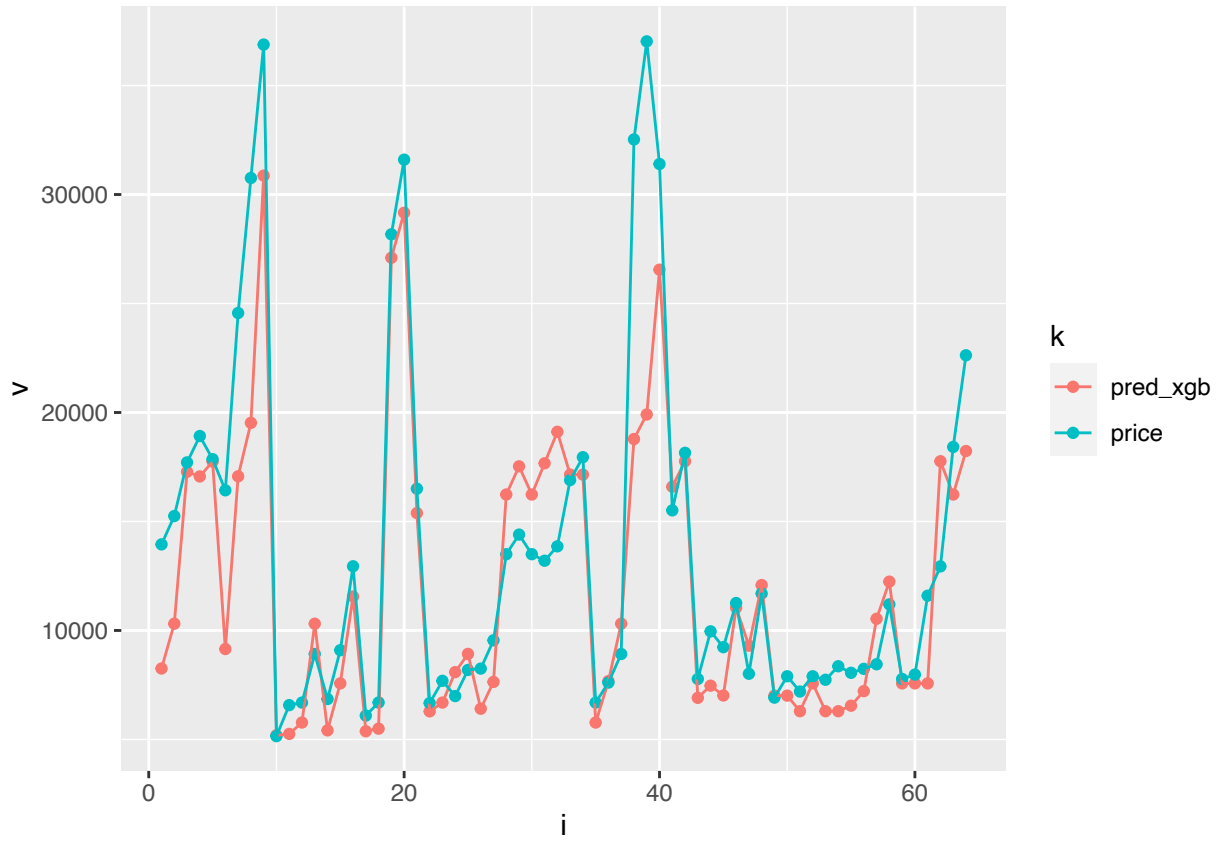
```
RMSE(pred_xgb, test_data_prep$price) #métrica
```

```
## [1] 4030.969
```

```
R2(pred_xgb, test_data_prep$price) #métrica
```

```
## [1] 0.7878767
```

```
#visualización de resultados
data.frame(i = seq(1:nrow(test_data_prep)),pred_xgb,
      price = test_data_prep$price) %>% gather(key = "k",
                                               value = "v",
                                               -i) %>%
  ggplot(aes(i, v, col = k)) +
  geom_point() +
  geom_line()
```



Se observa que el modelo tiene un $rmse = 4379.223$ y $r^2 = 0.7476562$. Por otro lado, las predicciones no tienen una buena aproximación a los valores reales.

- Regresión multilineal:

Se propone el siguiente modelo:

$$price = \beta_0 + \beta_1 * curbweight + \beta_2 * carheight + \beta_3 * stroke + \beta_4 * peakrpm + \beta_5 * enginelocation + \beta_6 * fueltype$$

```
train_lm <- lm(price ~., data = train_data_prep) #ajuste
```

Análisis de coeficientes y coeficiente de determinación:

$H_0 : \beta_0 = \beta_1 = \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = 0$

$$H_1 : \beta_0 \neq 0, \beta_1 \neq 0, \beta_2 \neq 0, \beta_3 \neq 0, \beta_4 \neq 0, \beta_5 \neq 0, \beta_6 \neq 0$$

```
train_lm %>% summary() #resumen
```

```
##
## Call:
## lm(formula = price ~ ., data = train_data_prep)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10119.7  -1951.6   -137.4   1466.5  17645.4
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    13178.61    1205.57  10.931  < 2e-16 ***
## curbweight      7077.13     336.78  21.014  < 2e-16 ***
## carheight       -748.22     346.45  -2.160   0.0326 *
## stroke          -277.81     329.99  -0.842   0.4014
## peakrpm           85.26     361.73   0.236   0.8140
## enginelocation 17244.33    3798.41   4.540 1.24e-05 ***
## fueltype        -162.99    1270.56  -0.128   0.8981
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3710 on 134 degrees of freedom
## Multiple R-squared:  0.7929, Adjusted R-squared:  0.7836
## F-statistic:  85.5 on 6 and 134 DF,  p-value: < 2.2e-16
```

Se rechaza la hipótesis nula.

Análisis de residuos:

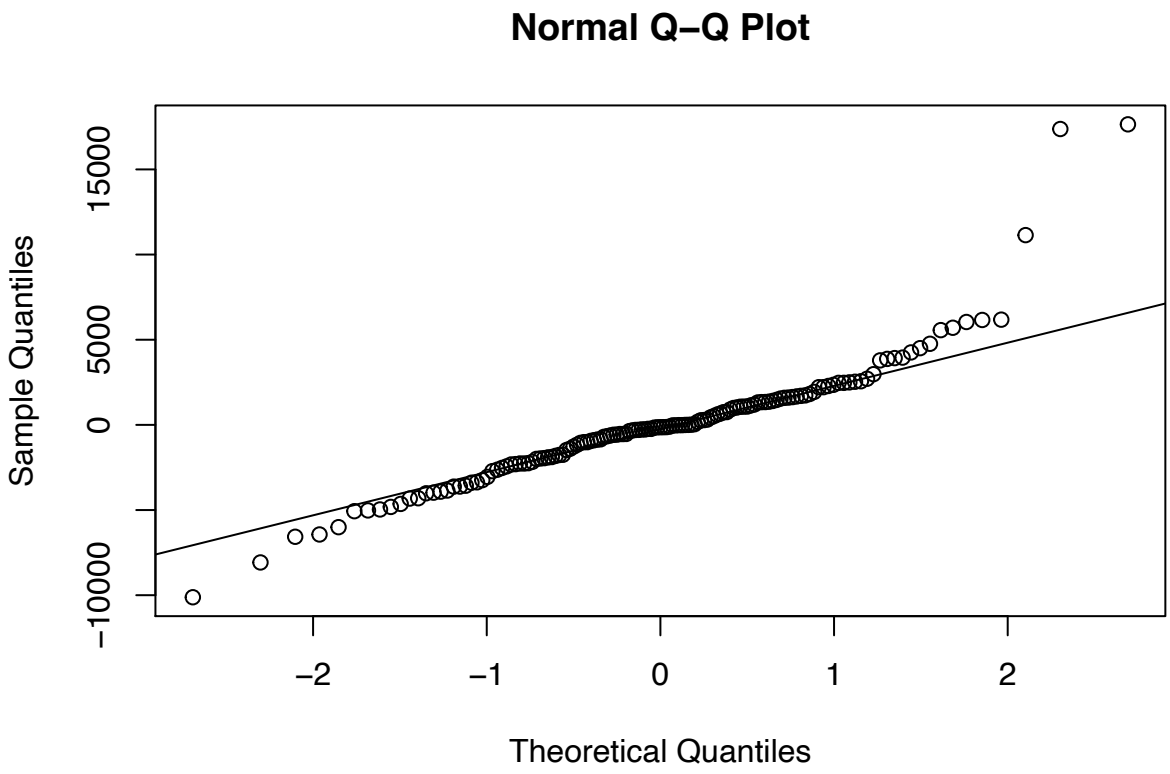
H_0 : los errores provienen de una población normal (con media 0) H_1 : los errores no provienen de una población normal

```
shapiro.test(train_lm$residuals) #test de shapiro
```

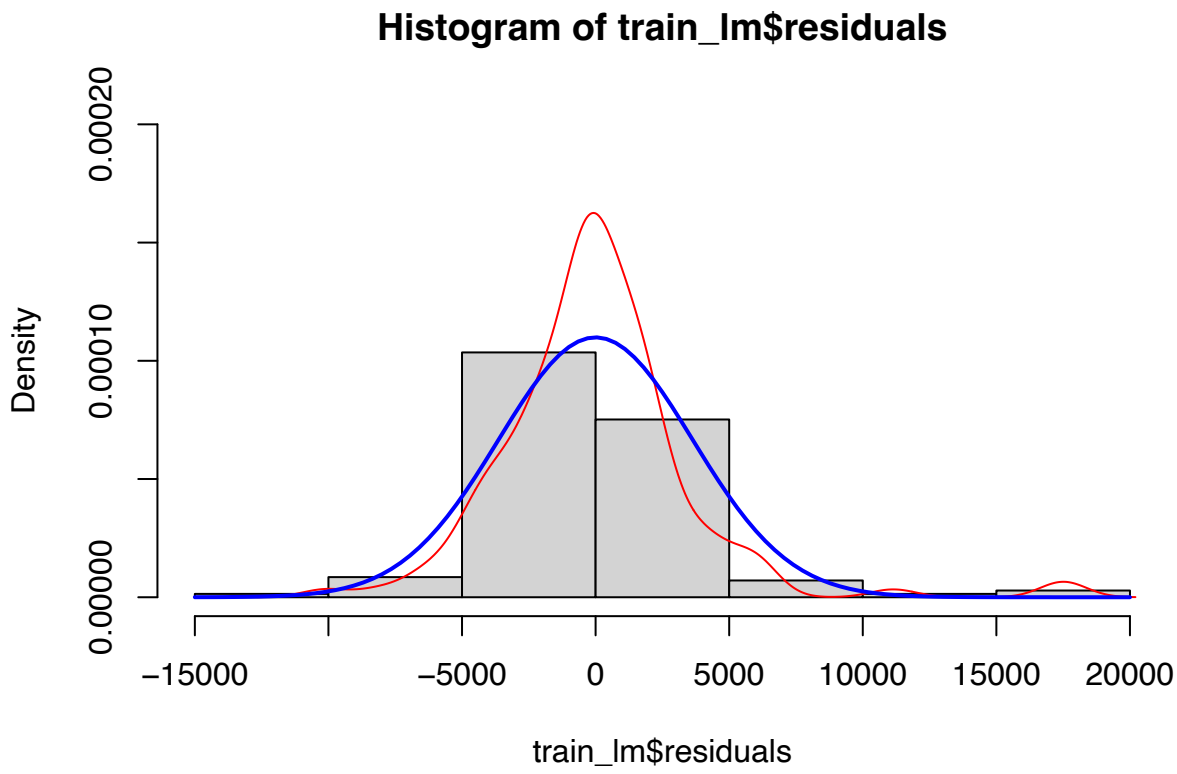
```
##
##  Shapiro-Wilk normality test
##
## data:  train_lm$residuals
## W = 0.88212, p-value = 3.364e-09
```

Se rechaza H_0.

```
qqnorm(train_lm$residuals) #q-q plot
qqline(train_lm$residuals)
```




```
#histograma de frecuencia con gráfico de densidad de distribución normal (azul) y gráfico de densidad d
hist(train_lm$residuals,freq=FALSE, ylim = c(0,2e-04))
lines(density(train_lm$residual),col="red")
curve(dnorm(x,mean(train_lm$residuals),
            sd(train_lm$residuals)),
            from=-15000, to=20000, add=TRUE, col="blue",lwd=2)
```



Prueba de hipótesis para la media de los residuos:

H_0 : los errores tienen media 0 H_1 : los errores no tienen media 0

```
t.test(train_lm$residuals) #prueba de t-student para medias
```

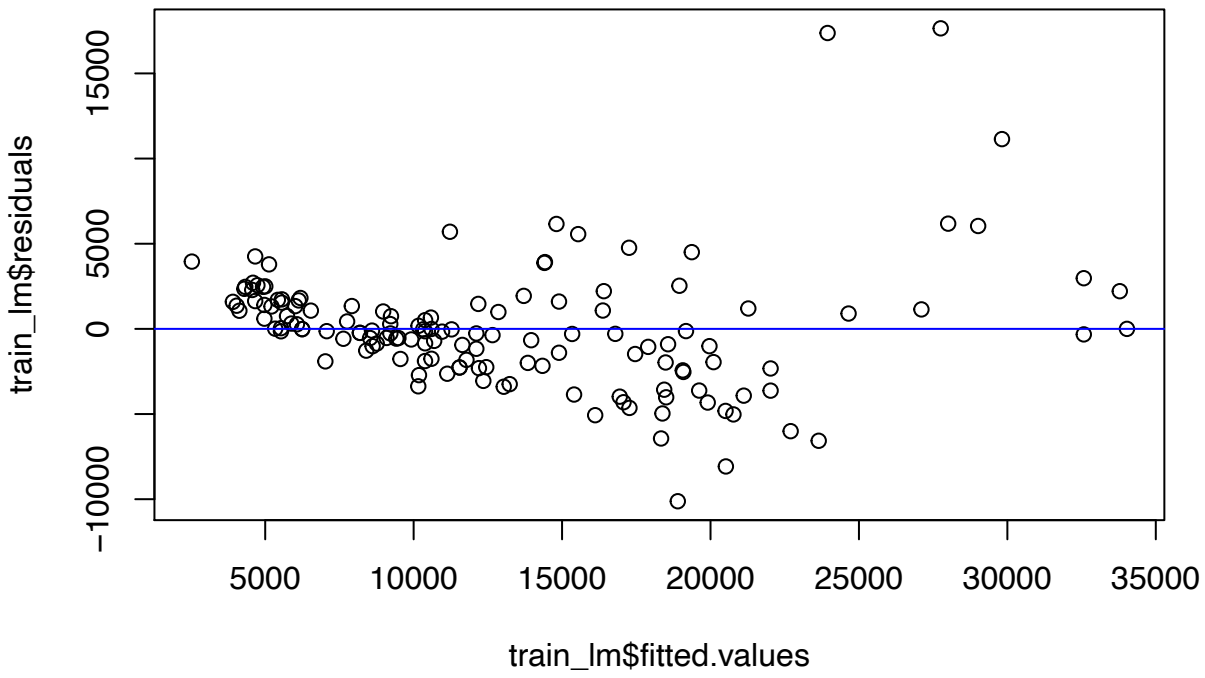
```
##
##  One Sample t-test
##
## data:  train_lm$residuals
## t = -9.8759e-16, df = 140, p-value = 1
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -604.3285  604.3285
## sample estimates:
##     mean of x
## -3.018775e-13
```

Dado el p-value = 1, se acepta la hipótesis nula de que la media sí es 0.

Análisis de simetría y homocedasticidad

En el gráfico se observa simetría y heterocedasticidad dado que los residuos se empiezan a dispersar.

```
plot(train_lm$fitted.values,train_lm$residuals)
abline(h=0, col=c("blue"))
```



Análisis del modelo

Analizando los resultados, el modelo lineal presentó un coeficiente de determinación ajustado adecuado $r^2=0.7836 $. No obstante, al realizar el análisis de residuos se tiene que éstos no se distribuyen de manera normal y hay cierta heterocedasticidad, aunque se tenga que la media tienda a 0. Por lo tanto, el modelo no es adecuado para realizar predicciones.

##8. Conclusión.

Al realizar todo el proceso de selección de variable y de procesamiento de éstas, para entrenar modelos de machine learning, se tiene que el modelo Random Forest Regressor (con el hiperparámetro $mtry = 4$) presenta los mejores resultados, teniendo un $rmse = 3316.399$ y $r^2 = 0.8299176$. Por otro lado, XGboost Regressor no supera las métricas del modelo anterior ($rmse = 4379.223$ y $r^2 = 0.7476562$) y el modelo de regresión multilineal se demostró que no cumple con los requerimiento para realizar predicciones dado que los residuos no siguen una distribución normal (sin importar que $r^2_{a}=0.7836 $).

Se recomienda realizar el proceso nuevamente, seleccionando otras combinaciones de variables (asegurando que no haya correlación entre los predictores) para determinar si hay mejores resultados en la modelación.