Introducción a la ciencia de datos: EDA, Regresión y Clasificación

Jesús Sánchez de Castro

30 Agosto 2018

1 Análisis Exploratorio de Datos

En esta sección se realizan los análisis exploratorios de datos para los conjuntos de regresión y clasificación asignados para este trabajo.

1.1 EDA: Dataset para regresión

Para el problema de regresión ha sido asignado el conjunto de datos AutoMPG6. Este conjunto contiene seis variables, siendo cinco de estas continuas y una de ellas discreta multivariable. El objetivo de esta regresión es predecir el consumo de combustible en millas por galón (mpg) de vehículos que circulan por la ciudad en función al resto de variables. Cabe destacar que este conjunto de datos es una versión modificada en la que se han eliminado dos variables discretas del conjunto original (Cylinders y Origin). Este conjunto de datos se guarda en un dataframe de R y contiene 392 instancias o filas y 6 características o columnas. Las características con las que se trabaja son:

- **Displacement**: Variable de entrada. Se trata del desplazamiento del vehículo estudiado. Variable numérica real.
- **HorsePower**: Variable de entrada. Se trata de la potencia del vehículo. Variable numérica entera.
- **Weight**: Variable de entrada. Es el peso del vehículo en libras. Variable numérica entera.
- **Acceleration**: Variable de entrada. Aceleración del vehículo. Variable numérica real.

- **ModelYear**: Variable de entrada. Año del modelo del vehículo. Variable numérica entera.

- **Mpg**: Variable de salida. Millas recorridas por galón de combustible consumido. Variable numérica entera.

En primer lugar, se realiza un pequeño análisis del conjunto de datos analizando valores como la media, desviación típica, cuartiles y presencia de valores perdidos. Empleando summary() en R, podemos obtener mucha información para comenzar el análisis. En la Figura 1 se presentan los resultados estadísticos de summary().

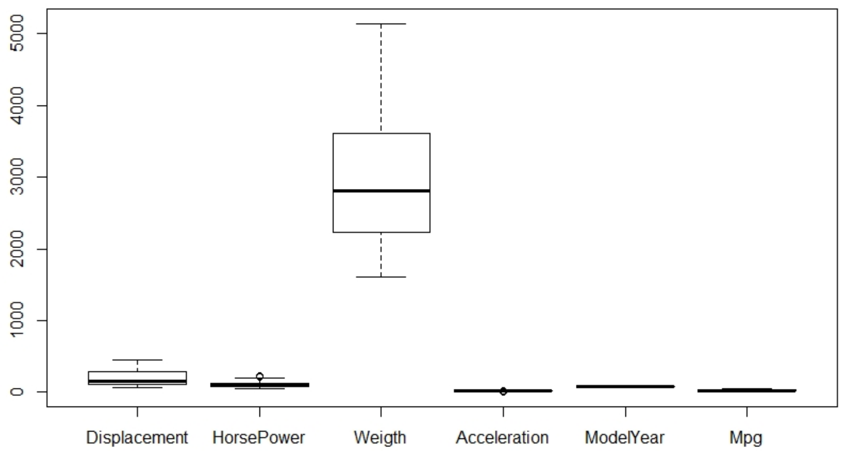
Figure 1: Resultado de summary() del conjunto de datos de regresión.

```
[1] "Summary:"
  Displacement     HorsePower        Weigth      Acceleration     ModelYear         Mpg
 Min.   : 68.0   Min.   : 46.0   Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   : 9.00
 1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:17.00
 Median :151.0   Median : 93.5   Median :2804   Median :15.50   Median :76.00   Median :22.75
 Mean   :194.4   Mean   :104.5   Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :23.45
 3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:29.00
 Max.   :455.0   Max.   :230.0   Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :46.60
```

Lo primero que se puede observar en la Figura 1 es la diferencia que existe en el rango de valores de los diferentes atributos. Puede ser interesante estudiar el uso de normalización de los datos. Además, se puede apreciar diferencias en las distancias entre el 1er y 3er cuartil para diferentes variables. Con el fin de facilitar el análisis de las distribuciones de las variables se incluyen a continuación gráficos boxplot. Debido a la diferencia en el rango de valores, se añaden los boxplots en diferentes gráficos en los que se puedan apreciar el rango de valores en que se mueven las variables y la información que aporta el boxplot en sí.

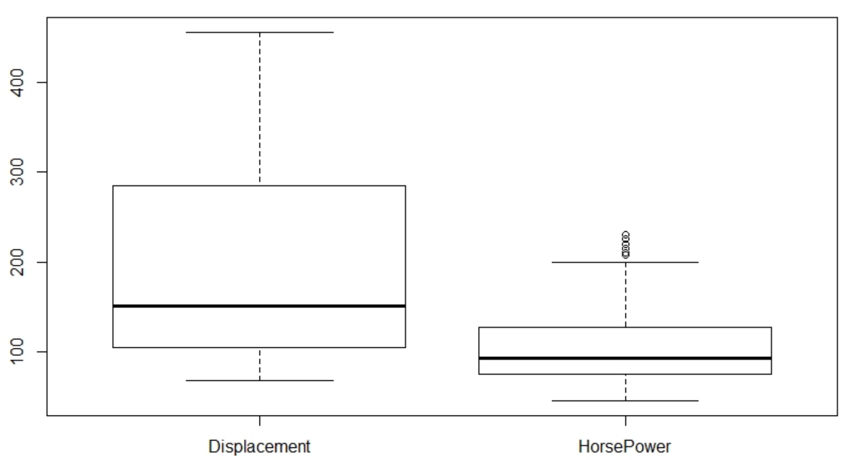
En la Figura 2 se muestran los boxplots de todas las variables del conjunto. Rápidamente se puede observar la diferencia en el rango de valores de las diferentes variables. Las únicas dos que se pueden mostrar en un mismo gráfico son Displacement y HorsePower.

Figure 2: Boxplots de todas las variables del conjunto de datos.



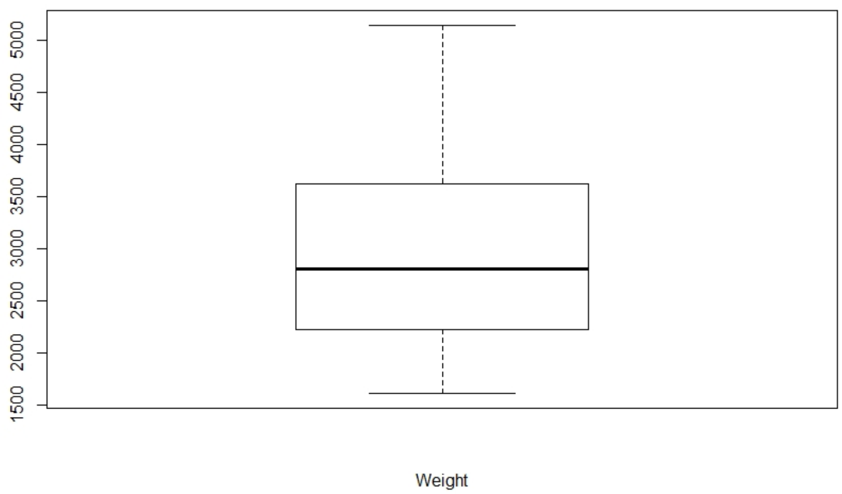
A continuación, se presentan en las Figuras 3, 4, 5 y 6 los boxplots correspondientes a las variables del problema de forma independiente para poder apreciar mejor los detalles individuales.

Figure 3: Boxplots de las variables Displacement y HorsePower.



En la Figura 3 se analizan Displacement y HorsePower. Comenzando por Displacement, se puede observar que el valor medio se encuentra entre 100 y 200, como ya se calculó previamente, la media es 194.4 millas. Además, se observa que la mayoría de los vehículos circulan entre 200 y 300 millas. Por otro lado, HorsePower tiene un rango menor de valores y menor dispersión en los datos que Displacement. Su valor medio es 104.5. Además, se observan una serie de punto en la parte superior del valor máximo, se trata de outliers o valores anómalos.

Figure 4: Boxplots de la variable Weight.



En la Figura 4 se muestra Weight. Se apreciar como la mayoría de los datos se encuentran ligeramente más cercanos al valor mínimo que al máximo y no presenta ningún outlier. Tiene un valor medio de 2978 libras. Por último, en las Figuras 5 y 6 se presentan Acceleration y ModelYear respectivamente.

Figure 5: Boxplots de la variable Acceleration.

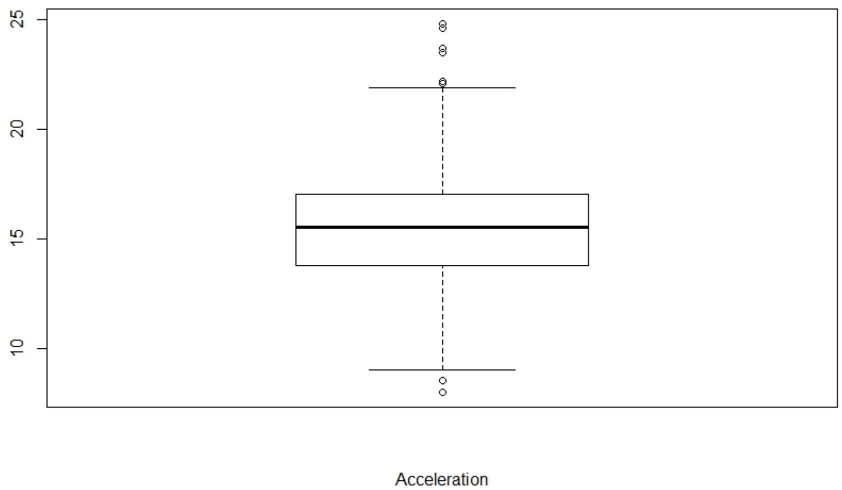
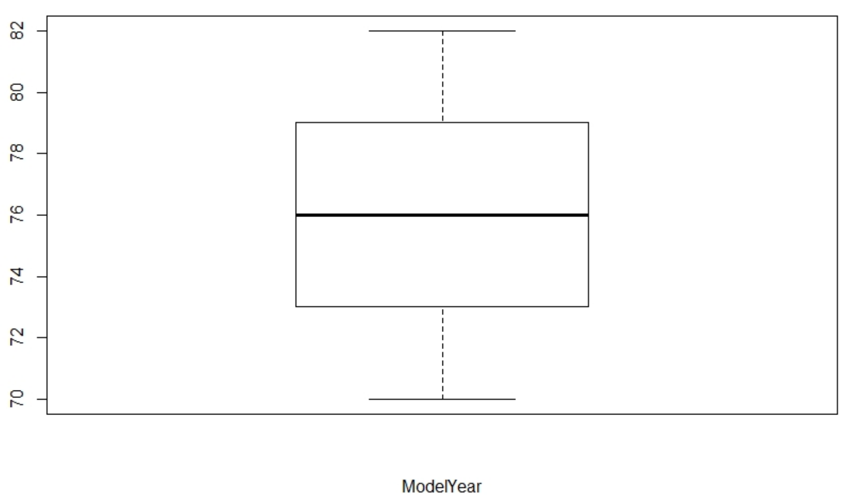
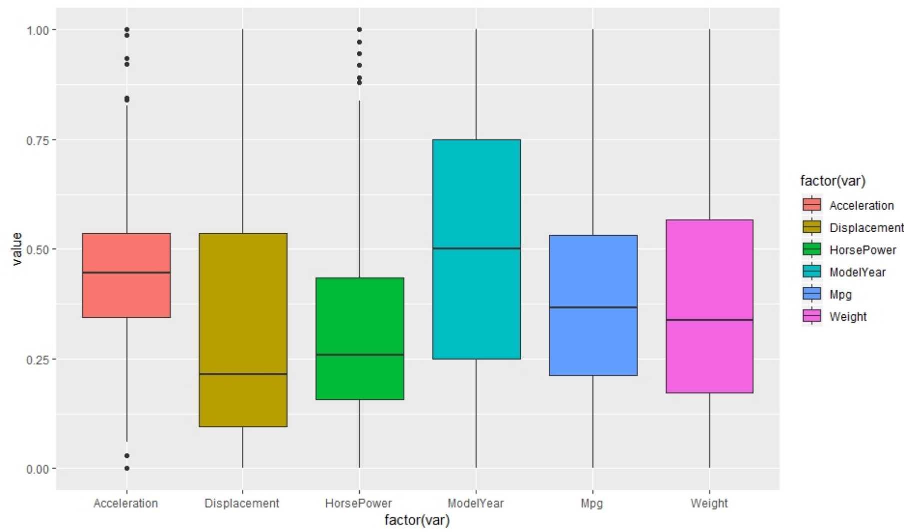


Figure 6: Boxplots de la variable ModelYear.



Por último, tras realizar una normalización max-min para llevar los valores a un rango 0-1, el resultado se puede observar en la Figura 7. En esta figura se puede comenzar a conocer la forma de las distribuciones y mucha de esta información se ve contrastada con histogramas y valores de skewness más adelante en la sección de Regresión.

Figure 7: Boxplots de las variables de regresión normalizadas a rango [0,1].



A continuación, se estudian las correlaciones y la forma de las distribuciones de los datos. Al ser modelos paramétricos, los modelos lineales simples necesitan asumir que existe una relación lineal entre las variables. Realizando un estudio de correlación y realizando gráficas plot() de las variables de entrada frente a la salida, se puede obtener una primera idea de cómo de bien podrá ajustarse la regresión lineal a los datos. Además, un gráfico de matriz de correlación puede ayudar a comprender las relaciones entre variables. A continuación, se presentan las Figuras 7, 8, 9, 10 y 11 asociadas a Displacement, HorsePower, Weight, Accerelation y ModelYear frente a Mpg. Por último, se añade en la Figura 12 una matriz de correlación.

Figure 8: Scatterplot de todas las variables de regresión.

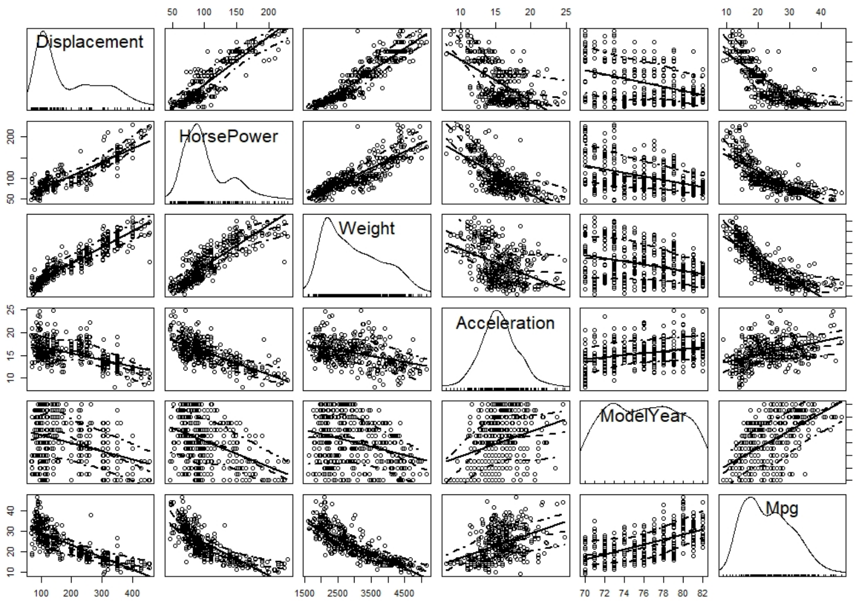
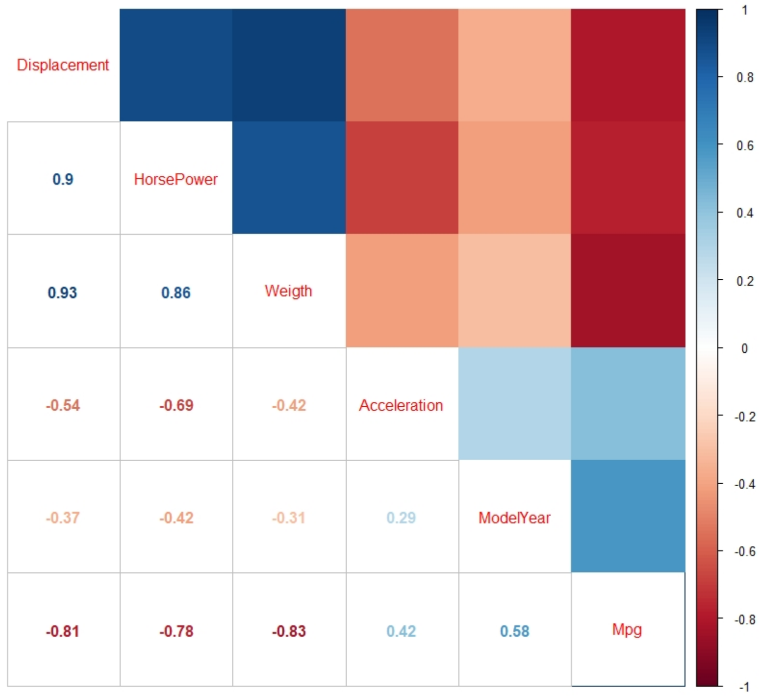


Figure 9: Matriz de correlación de todas las variables de regresión.



Como se puede apreciar en la Figura 8, Displacemente, HorsePower y Weight tienen una tendencia clara y similar. Además se puede ver en la matriz de correlación (Figura 9) una alta correlación negativa. Esto nos indica que pueden ser las variables más interesantes a priori. Por otro lado, para Acceleration y ModelYear se observa una mayor dispersión en los datos y por lo tanto una menor correlación, que en este caso es positiva. Por lo tanto, se parte de la hipótesis de que las tres primeras variables de entrada serán más relevantes en los modelos lineales.

A la hora de realizar interpretaciones sobre las correlaciones existentes, debido a que este conjunto de datos pertenece a un estudio en ciudad, podemos ver como a mayor desplazamiento menor es la cantidad de millas por desplazamiento. Lo mismo ocurre con los caballos del vehículo, en un entorno de ciudad el consumo es siempre alto al conducir a marchas más bajas y los vehículos no pueden llegar a los puntos óptimos de Mpg de velocidades más altas.

Por otro lado, la función scatterplotMatrix() nos permite ver la distribución de los datos en la diagonal, como se puede observar, las tres primeras variables tienen un sesgo hacia la derecha y en menor medida ocurre lo mismo con Mpg. Además de estudiar el sesgo o ”skewness” es interesante ver como de ”picuda” es una distribución con kurtosis. A continuación, se añaden las dos tablas para poder analizar con más detalle estos efectos.

Table 1: Medida Skewness o sesgo para todas las variables.

Displacement	**HorsePower**	**Weight**	**Acceleration**	**ModelYear**	**Mpg**
0.69898128	1.08316116	0.51759535	0.29046997	0.01961288	0.45534138

Table 2: Medida Kurtosis para todas las variables.

Displacement	**HorsePower**	**Weight**	**Acceleration**	**ModelYear**	**Mpg**
2.216308	3.672822	2.185759	3.423320	1.832124	2.475297

Para comenzar a analizar los resultados de skewness, un skewness de 0 no contienen sesgo y se estaría trabajando con unos datos simétricos y de distribución normal. Por otro lado, cuanto más nos alejemos del 0 mayor sesgo habrá en la distribución. Como se puede apreciar en la diagonal de la Figura 7 y en la Tabla 1, casi todas las variables presentan un sesgo hacia la derecha reflejado en las medidas aumentando desde 0 en adelante. El mayor caso de skewness es de 1.083 para HorswPower y el menor skewness para ModelYear con 0.0196. Tanto el gráfico como la tabla reflejan a la perfección este sesgo. En caso de haber habido sesgo hacia la izquierda, la medida skewness habría sido negativa.

En cuanto al análisis de Kurtosis, de nuevo un valor de 0 indica que tenemos una distribución normal. Como se puede comparar de nuevo viendo la Figura 7 y la Tabla 2, la distribuciones de las variables se alejan de una distribución normal en cuanto a Kurtosis, viendose como el valor más alto es el de Acceleration siendo la distribución más "picuda". Por otro lado, es interesante ver como algunas distribuciones son casi bimodales, teniendo dos picos.

El código empleado en esta sección es:

```
library(ggplot2)
library(corrplot)
library(car)
library(dplyr)
library(tidyr)
library(moments)
library(reshape2)
library(caret)
set.seed(77183983)

# Se establece el path y se lee el fichero
datapath = "C://M?ster//IntroCienciaDatos//Datasets Regresion//
    autoMPG6"
data = read.csv(paste0(datapath,"//autoMPG6.dat"), header = FALSE,
    skip = 10)

```

```
# Se cambian los nombres de las variables
colnames(data) <- c("Displacement", "HorsePower", "Weight", "
    Acceleration", "ModelYear", "Mpg")

# Usando str vemos la estructura del set y los tipos de las
    variables
print("Data structure:")
str(data)

# Primer vistazo a los datos
print("Summary:")
print(summary(data))

# Boxplots
boxplot(data)
boxplot(data[,c(1,2)])
boxplot(data[,c(3)], xlab=("Weight"))
boxplot(data[,c(4)], xlab=("Acceleration"))
boxplot(data[,c(5)], xlab=("ModelYear"))

# Estudio de correlacion entre entradas y salida
plot(Displacement~Mpg, data)
plot(HorsePower~Mpg, data)
plot(Weight~Mpg, data)
plot(Acceleration~Mpg, data)
plot(ModelYear~Mpg, data)

# Matriz de correlacion
corrplot.mixed(cor(data), upper="color", lower = "number")

# ScatterPlot Matrix
scatterplotMatrix(~Displacement+HorsePower+Weight+Acceleration+
    ModelYear+Mpg, data=data,
                  col=palette()[1])

# Estudio de skewness y Kurtosis
sk = skewness(data)
kurtosis = kurtosis(data)

# Funcion de normalizacion
normalize <- function(x){
  return((x-min(x))/(max(x)-min(x)))
}

# Normalizacion de una copia del set
norm_data <- data.frame(data)
norm_data["Displacement"] <- as.data.frame(apply(norm_data["
    Displacement"], 2, normalize))
norm_data["HorsePower"] <- as.data.frame(apply(norm_data["
    HorsePower"],2, normalize))
norm_data["Weight"] <- as.data.frame(apply(norm_data["Weight"], 2,
    normalize))
norm_data["Acceleration"] <- as.data.frame(apply(norm_data["
    Acceleration"], 2, normalize))
norm_data["ModelYear"] <- as.data.frame(apply(norm_data["ModelYear"
    ], 2, normalize))
norm_data["Mpg"] <- as.data.frame(apply(norm_data["Mpg"], 2,
```

```
    normalize))

aux <- gather(norm_data, var, value)
ggplot(aux, aes(y=value, x=factor(var)))+geom_boxplot(aes(fill=
    factor(var)))
```

1.2 EDA: Dataset para clasificación

Para el problema de clasificación el conjunto asignado es Vehicle. Se trata de un conjunto de datos con 19 variables y 845 instancias. Las 18 variables de entrada son enteras y las variables de clase tiene 4 valores diferentes (”Bus”, ”Opel”, ”Saab”, ”Van”). Como se muestra en la tabla de detalles de la web de Keel, es un conjunto en el que se incluyen las principales características de un vehículo y no contiene ningún valor perdido, además se ha comprobado en R.

```
> anyNA(data)
[1] FALSE
```

Para tener una idea inicial de las variables, se emplea la función ”summary()”.

Figure 10: Summary de los datos de clasificación.

```
> summary(data)
  Compactness      Circularity    Distance_circularity  Radius_ratio   Praxis_aspect_ratio
 Min.   : 73.00   Min.   :33.00   Min.   : 40.00        Min.   :104.0   Min.   : 47.00
 1st Qu.: 87.00   1st Qu.:40.00   1st Qu.: 70.00        1st Qu.:141.0   1st Qu.: 57.00
 Median : 93.00   Median :44.00   Median : 80.00        Median :167.0   Median : 61.00
 Mean   : 93.68   Mean   :44.86   Mean   : 82.09        Mean   :168.9   Mean   : 61.68
 3rd Qu.:100.00   3rd Qu.:49.00   3rd Qu.: 98.00        3rd Qu.:195.0   3rd Qu.: 65.00
 Max.   :119.00   Max.   :59.00   Max.   :112.00        Max.   :333.0   Max.   :138.00
 Max_length_aspect_ratio Scatter_ratio   Elongatedness   Praxis_rectangular Length_rectangular
 Min.   : 2.000          Min.   :112.0   Min.   :26.00   Min.   :17.00      Min.   :118
 1st Qu.: 7.000          1st Qu.:146.0   1st Qu.:33.00   1st Qu.:19.00      1st Qu.:137
 Median : 8.000          Median :157.0   Median :43.00   Median :20.00      Median :146
 Mean   : 8.566          Mean   :168.8   Mean   :40.93   Mean   :20.58      Mean   :148
 3rd Qu.:10.000          3rd Qu.:198.0   3rd Qu.:46.00   3rd Qu.:23.00      3rd Qu.:159
 Max.   :55.000          Max.   :265.0   Max.   :61.00   Max.   :29.00      Max.   :188
 Major_variance  Minor_variance Gyration_radius Major_skewness   Minor_skewness   Minor_kurtosis
 Min.   :130.0   Min.   : 184   Min.   :109.0   Min.   : 59.00   Min.   : 0.000   Min.   : 0.0
 1st Qu.:167.0   1st Qu.: 318   1st Qu.:149.0   1st Qu.: 67.00   1st Qu.: 2.000   1st Qu.: 5.0
 Median :179.0   Median : 364   Median :173.0   Median : 72.00   Median : 6.000   Median :11.0
 Mean   :188.6   Mean   : 440   Mean   :174.7   Mean   : 72.47   Mean   : 6.378   Mean   :12.6
 3rd Qu.:217.0   3rd Qu.: 587   3rd Qu.:198.0   3rd Qu.: 75.00   3rd Qu.: 9.000   3rd Qu.:19.0
 Max.   :320.0   Max.   :1018   Max.   :268.0   Max.   :135.00   Max.   :22.000   Max.   :41.0
 Major_kurtosis  Hollows_ratio     Class
 Min.   :176.0   Min.   :181.0   bus :218
 1st Qu.:184.0   1st Qu.:190.0   opel:212
 Median :188.0   Median :197.0   saab:217
 Mean   :188.9   Mean   :195.6   van :198
 3rd Qu.:193.0   3rd Qu.:201.0
 Max.   :206.0   Max.   :211.0
```

En la Figura 10, se puede observar que los rangos de los valores de estas variables no son tan dispares como en el caso del conjunto de datos de regresión. Lo primero que llama la atención es el balanceo de clases para los 4 valores:

- Bus: 218 instancias.
- Opel: 212 instancias.
- Saab: 217 instancias.
- Van: 198 instancias.

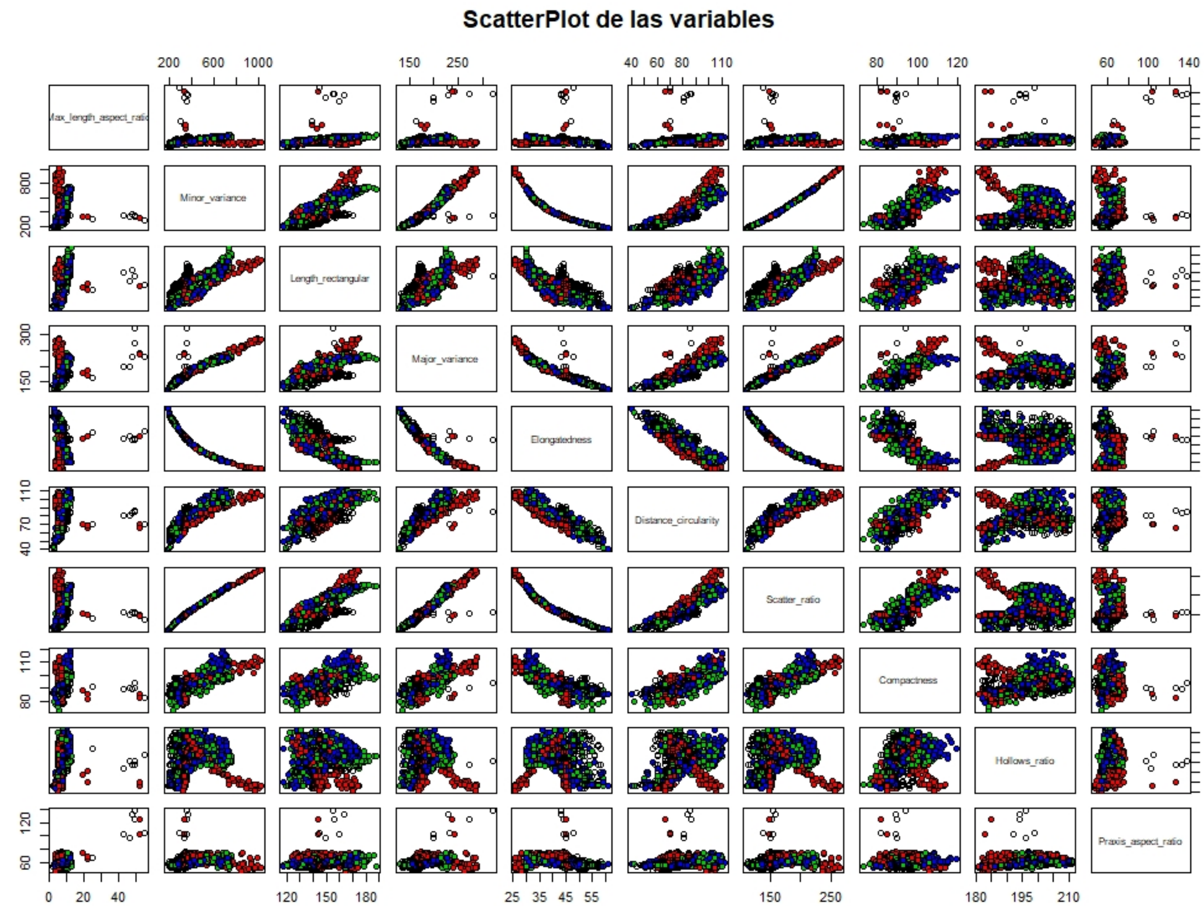
Las cuatro clases tienen un valor muy similar de instancias por lo que esto es un dato muy positivo de cara al rendimiento de la clasificación, sobre todo teniendo en cuenta que es clasificación no binaria. Debido a la gran cantidad de variables es complicado analizar numéricamente estas características mostradas por "summary()". Los gráficos boxplots serán de gran utilidad para hacerlos una idea de la distribución de los datos junto con histogramas.

Para trabajar de forma más cómoda y sin perder información sobre las variables de mayor importancia, se emplea en el conjunto de datos un algoritmo para selección de variables según su importancia. Para ello se emplea el paquete caret que emplea un algoritmo RandomForest y devuelve un ranking de importancia de las variables. De esta forma mejoramos la calidad del modelo en el momento de realizar clasificación y no es necesario realizar un análisis de variables que no son importantes pudiendo centrar los esfuerzos de forma más óptima.

```
> names(data.var)
 [1] "Max_length_aspect_ratio" "Minor_variance"          "Length_
    rectangular"
 [4] "Major_variance"          "Elongatedness"           "Distance_
    circularity"
 [7] "Scatter_ratio"           "Compactness"             "Hollows_
    ratio"
[10] "Praxis_aspect_ratio"
```

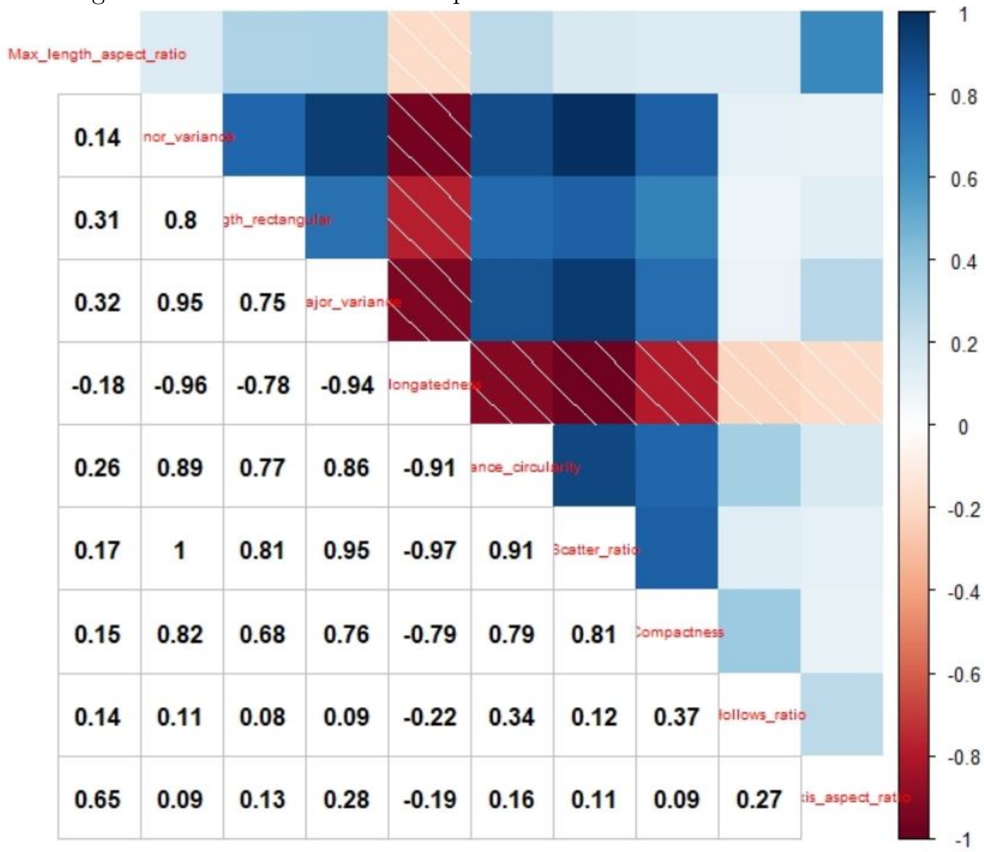
Tras la selección de variables, las que se mantienen son las 10 primeras de las 18 iniciales. Las variables seleccionadas se muestran a continuación en orden de importancia. A continuación, se muestra un scatterPlot de las 10 variables más relevantes del problema en la Figura 11.

Figure 11: ScatterPlot de las 10 variables más relevantes del problema de clasificación.



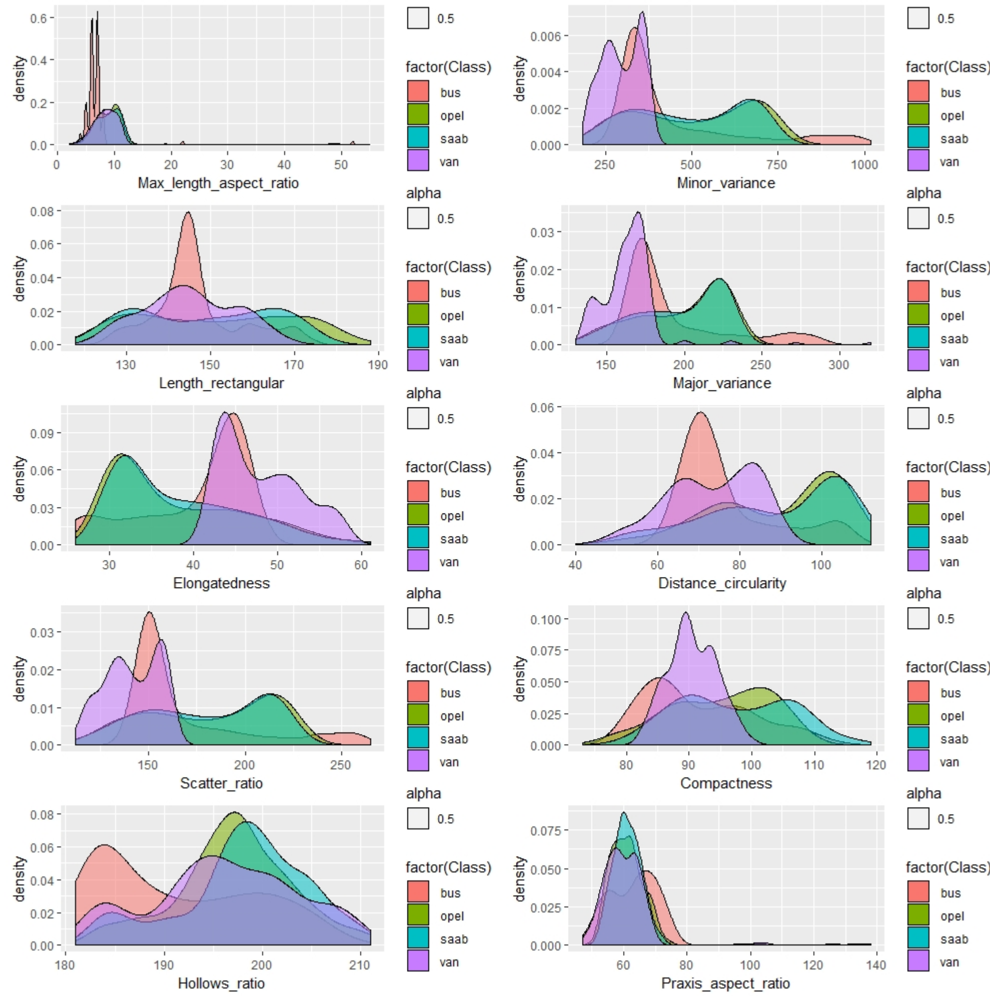
Como se puede observar en la Figura 11, hay algunas variables que tienen una correlación clara tanto positiva como negativa. Por otro lado, otras presentan una dispersión bastante alta. También es interesante observar el solapamiento existente entre las clases y las diferentes distribuciones de los datos en función de la clase a la que pertenecen. A continuación, se una matriz de correlaciones para ver con mayor detalle y contrastar con la figura anterior.

Figure 12: Matriz de correlación para las variables de clasificación.



Como se puede ver en la Figura 12, existe ciertas variables que mantienen una fuerte correlación ya se positiva como negativa. A continuación, en la Figura 13 se muestran las distribuciones de los datos en cuanto al valor de su clase.

Figure 13: Distribución de los datos en función de las variables y valores de clase.



Como se observa en la figura, debido a la existencia de diferencias substanciales en las propiedades en función de que clase tiene la muestra, las distribuciones se alejan de ser normales. Se puede ver como la clase bus tienen unos valores muy altos y se podría decir que seguro tiene valores de kurtosis altos. A lo largo de las diferentes variables se pueden ver sesgos a la derecha y la izquierda, por lo que de nuevo, en cuanto a valores de skewness, los datos se alejan de las distribuciones normales. En las siguientes figuras, se presentan los boxplots de las variables sin y con normalización, con la finalidad de observar las diferencias de rangos entre variables y poder comparar como se distribuyen los datos entre las diferentes variables.

Figure 14: Boxplots de las variables de clase sin normalizar.

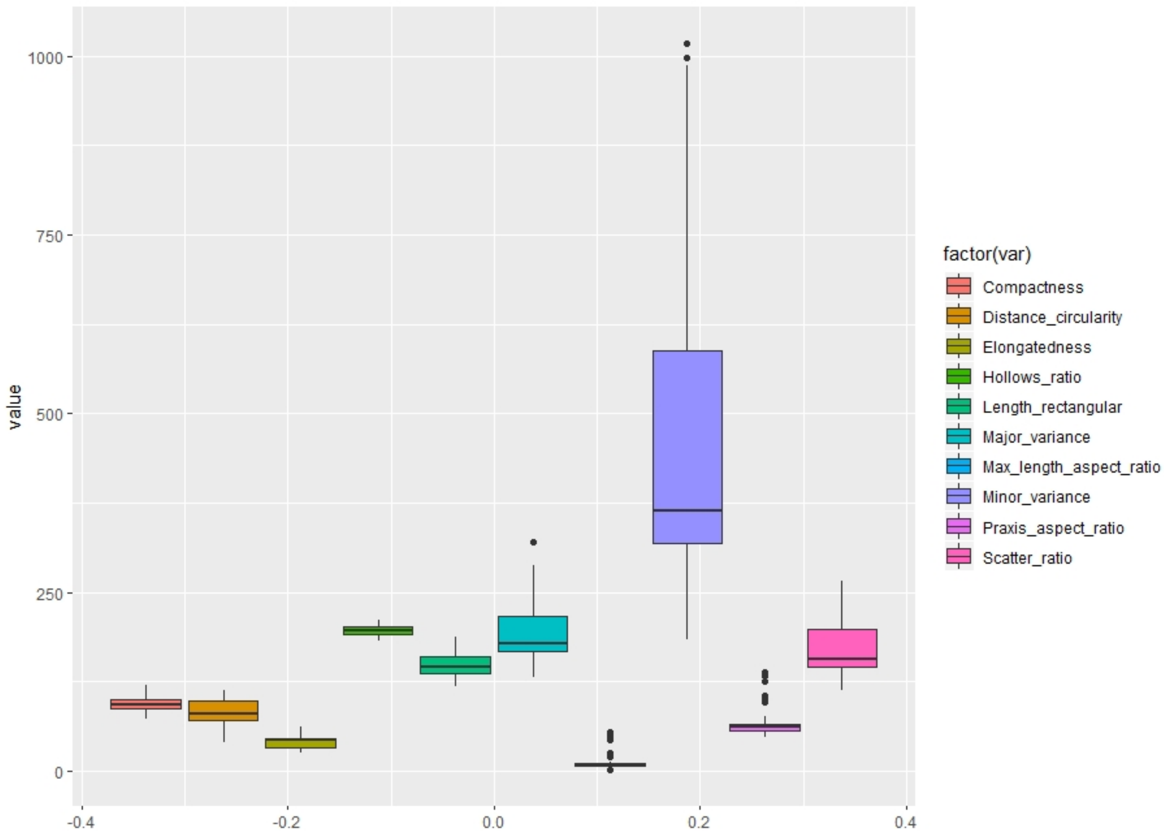
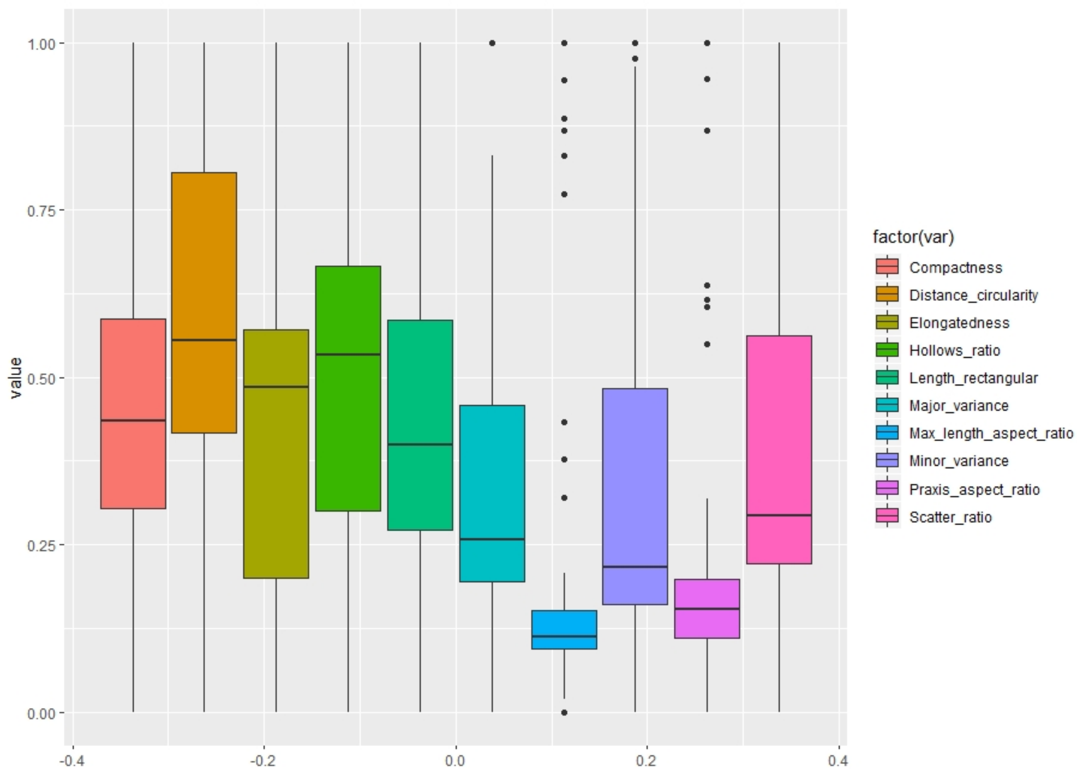


Figure 15: Boxplots de las variables de clase normalizando a rango [0-1]



Como se puede observar en las figuras 14 y 15, la variable Minor_variance tiene un rango mucho mayor al resto de variables. En la versión normalizada de los boxplots ya si se puede apreciar y comparar las distribuciones y presencia de outliers o valores anómalos. Debido a que los boxplots son otra forma de representar distribuciones de datos, se puede ver con claridad los efectos de sesgo hacia los lados y aquellas distribuciones con mayor kurtosis o más picudas como Max_length_aspect_ratio o Praxis_aspect_ratio.

El código empleado para esta sección es:

```
library(ggplot2)
library(corrplot)
library(car)
library(dplyr)
library(tidyr)
library(moments)
library(reshape2)
library(caret)
library(randomForest)
set.seed(77183983)

```

```r
# Se establece el path y se lee el fichero
datapath = "C:/Users/Yus/Google Drive/Master/IntroCienciaDatos/Datasets Clasificacion/vehicle"
data = read.csv(paste0(datapath,"//vehicle.dat"), comment.char = "@")

# Se cambian los nombres de las variables
colnames(data) <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
                    "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
                    "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
                    "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
                    "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")

# As we can see with str(), all the variables are numeric.
print("Data structure:")
str(data)

# Primer vistazo a los datos
print("Summary:")
print(summary(data))

# Eliminamos variables con poco peso en el problema
# seg n RandomForest y su selecci n de variables
rf <- randomForest(formula = Class~., data=data)
ranking <- varImp(rf)
variables <- rownames(ranking)

# Obtenemos los indices de los nombres de las variables ordenadas
# segun relevancia  en el ranking
index = sort(ranking$Overall, decreasing=TRUE, index.return=TRUE)
variables <- variables[index$ix]
variables.seleccionadas <- variables[1:10]

# Nos quedamos con las que nos interesan
data.var <- data[,variables.seleccionadas]
data.var["Class"] <- data$Class

# Gr ficos de apoyo
pairs(data.var[,-11], main = "ScatterPlot de las variables",
      pch = 21, bg = c("red", "green3", "blue")[unclass(data.var$Class)])

corr <- cor(data.var[,-11])
corrplot.mixed(corr, outline = TRUE, tl.cex = 0.6,
               lower.col = "black", lower="number", upper="shade")

# Gr ficos de densidad
p1 <- ggplot(data.var, aes(Max_length_aspect_ratio))+geom_density(aes(fill=factor(Class), alpha=0.5))

p2 <- ggplot(data.var, aes(Minor_variance))+geom_density(aes(fill=factor(Class), alpha=0.5))
```

```r

p3 <- ggplot(data.var, aes(Length_rectangular))+geom_density(aes(fill=factor(Class), alpha=0.5))

p4 <- ggplot(data.var, aes(Major_variance))+geom_density(aes(fill=factor(Class), alpha=0.5))

p5 <- ggplot(data.var, aes(Elongatedness))+geom_density(aes(fill=factor(Class), alpha=0.5))

p6 <- ggplot(data.var, aes(Distance_circularity))+geom_density(aes(fill=factor(Class), alpha=0.5))

p7 <- ggplot(data.var, aes(Scatter_ratio))+geom_density(aes(fill=factor(Class), alpha=0.5))

p8 <- ggplot(data.var, aes(Compactness))+geom_density(aes(fill=factor(Class), alpha=0.5))

p9 <- ggplot(data.var, aes(Hollows_ratio))+geom_density(aes(fill=factor(Class), alpha=0.5))

p10 <- ggplot(data.var, aes(Praxis_aspect_ratio))+geom_density(aes(fill=factor(Class), alpha=0.5))

require(ggpubr)

# Imprimir todas las gr ficas juntas.
ggarrange(p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,ncol=2,nrow =5)

# Funcion de normalizacion
normalize <- function(x){
  return((x-min(x))/(max(x)-min(x)))
}

# Se crea una copia del set y se normalizan las columnas
norm_data <- data.frame(data.var)
norm_data["Max_length_aspect_ratio"] <- as.data.frame(apply(norm_data["Max_length_aspect_ratio"], 2, normalize))
norm_data["Minor_variance"] <- as.data.frame(apply(norm_data["Minor_variance"],2, normalize))
norm_data["Length_rectangular"] <- as.data.frame(apply(norm_data["Length_rectangular"], 2, normalize))
norm_data["Major_variance"] <- as.data.frame(apply(norm_data["Major_variance"], 2, normalize))
norm_data["Elongatedness"] <- as.data.frame(apply(norm_data["Elongatedness"], 2, normalize))
norm_data["Distance_circularity"] <- as.data.frame(apply(norm_data["Distance_circularity"], 2, normalize))
norm_data["Scatter_ratio"] <- as.data.frame(apply(norm_data["Scatter_ratio"], 2, normalize))
norm_data["Compactness"] <- as.data.frame(apply(norm_data["Compactness"], 2, normalize))
norm_data["Hollows_ratio"] <- as.data.frame(apply(norm_data["Hollows_ratio"], 2, normalize))
norm_data["Praxis_aspect_ratio"] <- as.data.frame(apply(norm_data["Praxis_aspect_ratio"], 2, normalize))
```

```
# Boxplots
aux <- gather(data.var[,-c(11)], var, value)
ggplot(aux, aes(y=value))+geom_boxplot(aes(fill=factor(var)))

aux2 <- gather(norm_data[,-c(11)], var, value)
ggplot(aux2, aes(y=value))+geom_boxplot(aes(fill=factor(var)))
```

2 Regresión

En esta sección se emplean algoritmos de regresión para comparar su comportamiento con el conjunto de datos Mpg6. En la subsección 3.1 se emplean modelos de regresión simple con las cinco variables de entrada y se estudia cual es capaz de predecir mejor la variable objetivo mpg. En la subsección 3.2 se emplean regresión lineal múltiple escogiendo las variables de entrada que resulten en el mejor modelo. Por último, en la subsección 3.3 se emplea KNN para regresión y se comparan y analizan los resultados en la subsección 3.4.

2.1 Regresión Lineal Simple

Debido a que el conjunto de datos únicamente tiene cinco variables predictoras, se entrenan 5 modelos lineales simples. Aun así, se parte de la base de que Displacement, HorsePower y Weight deben tener una mayor capacidad predictiva de Mpg. En primer lugar, se emplean todos los datos para ajustar el modelo lineal y posteriormente ver su R^2 ajustado. En la Tabla 3 se pueden ver los resultados.

Table 3: R^2 ajustado de los modelos lineales.

Variable	**Adj.R2**
Displacemt	0.6473274
HorsePower	0.6049379
Weight	0.6918423
Acceleration	0.1771025
ModelYear	0.3353279

Como se puede observar en la Tabla 3, las variables más prometedoras son las tres primeras, lo que coincide con las hipótesis que teníamos tras realizar el EDA. A continuación. Cuanto más cerca a 1 esté R^2 mejor se adapta el modelo a los datos, pues mayor es la capacidad de explicar la variabilidad de los datos.

Una vez analizado que variables son las más interesantes, se realiza el entrenamiento y validación de los cinco modelos con validación cruzada. Se han empleado los folds dados por el profesor. En la Tabla 4 se muestran los resultados de la métrica de error RMSE tanto para el conjunto de entrenamiento como para el de test. Como se puede apreciar, los tres primeros modelos son los que obtienen los mejores resultados. Es lógico que el peso (Weight) sea la variable que mejor resultados de, pues la relación entre el peso de un vehículo y las millas que recorre por galón de combustible es muy lógica. Lo mismo ocurre con el desplazamiento y potencia del vehículo, conociendo estos datos podemos predecir el Mpg con mayor precisión que con el resto.

Table 4: RMSE para entrenamiento y test de todos los modelos lineales simples.

Variables	**Entrenamiento**	**Test**
Displacement	4.562773	4.522197
HorsePower	4.834616	4.851156
Weight	4.275218	4.240137
Acceleration	6.968138	7.007551
ModelYear	6.296417	6.272758

2.2 Regresión múltiple

En esta subsección se buscan modelos de regresión múltiple comenzando por modelos lineales con varias variables y posteriormente modelos con interacciones y no linealidad. Para comenzar el proceso se escoge comenzar con todas las variables y eliminar las menos significativas estadísticamente. En la Figura 10 se muestra el summary del modelo entrenado con todas las variables.

Figure 16: Summary del modelo lineal múltiple todas las variables.

```
Call:
lm(formula = Mpg ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.5211 -2.3920 -0.1036  2.0312 14.2874 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -1.544e+01  4.677e+00  -3.300  0.00106 ** 
Displacement  2.782e-03  5.462e-03   0.509  0.61082    
HorsePower    1.020e-03  1.376e-02   0.074  0.94095    
Weight       -6.874e-03  6.653e-04 -10.333  < 2e-16 ***
Acceleration  9.032e-02  1.019e-01   0.886  0.37599    
ModelYear     7.541e-01  5.261e-02  14.334  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.435 on 386 degrees of freedom
Multiple R-squared:  0.8088,	Adjusted R-squared:  0.8063 
F-statistic: 326.5 on 5 and 386 DF,  p-value: < 2.2e-16
```

En la Fig 11 se puede ver como el modelo a mejorado el Adjusted R-squared a 0.8063 frente a 0.6918 que se obtenía con Weight. Por otro lado, se puede ver como las variables más significativas y con menor p-value son Weight y ModelYear. El siguiente modelo se entrenará con Weight y ModelYear para estudiar el efecto de quitar el resto de variables. Tras este resultado se he probado a añadir algunas de las variables que se han quitado y se ve que no aportan realmente nada al modelo, tal y como se ve en los p-value. Debido a la poca cantidad de variables de entrada el proceso ha sido más rápido de lo esperado. Se escoge como mejor modelo lineal múltiple aquel que emplea Weight y ModelYear.

Figure 17: Summary del modelo lineal múltiple con Weight y ModelYear.

```
Call:
lm(formula = Mpg ~ Weight + ModelYear, data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.8505 -2.3014 -0.1167  2.0367 14.3555 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.435e+01  4.007e+00  -3.581 0.000386 ***
Weight      -6.632e-03  2.146e-04 -30.911  < 2e-16 ***
ModelYear    7.573e-01  4.947e-02  15.308  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.427 on 389 degrees of freedom
Multiple R-squared:  0.8082,	Adjusted R-squared:  0.8072 
F-statistic: 819.5 on 2 and 389 DF,  p-value: < 2.2e-16
```

A continuación, se busca emplear interacciones y no linealidad en los modelos. En primer lugar, se prueba la interacción Weight y ModelYear mejorando el Adjusted R-squared a 0.8326 de 0.8068. En la Figura 12 se puede observar el summary de este modelo. El siguiente paso es añadir interacción y no linealidad se obtiene el mejor modelo con un Adjusted R2 de 0.8548, este resultado se ve en la Figura 13. Por último, eliminando $I(Weight^2)$ con el mayor p-value obtenemos una leve mejora consiguiendo 0.8551, el mejor modelo. El mejor modelo corresponde a la Figura 14.

Figure 18: Summary del modelo lineal múltiple con Interacción.

```
Call:
lm(formula = Mpg ~ (Weight * ModelYear), data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.0397 -1.9956 -0.0983  1.6525 12.9896 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)      -1.105e+02  1.295e+01  -8.531 3.30e-16 ***
Weight            2.755e-02  4.413e-03   6.242 1.14e-09 ***
ModelYear         2.040e+00  1.718e-01  11.876  < 2e-16 ***
Weight:ModelYear -4.579e-04  5.907e-05  -7.752 8.02e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.193 on 388 degrees of freedom
Multiple R-squared:  0.8339,	Adjusted R-squared:  0.8326 
F-statistic: 649.3 on 3 and 388 DF,  p-value: < 2.2e-16
```

Figure 19: Summary del modelo lineal múltiple con Interacción y no linealidad.

```
Call:
lm(formula = Mpg ~ Weight + ModelYear + I(Weight * ModelYear) +
    I(Weight^2) + I(Weight^2 * ModelYear), data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-8.8434 -1.7571 -0.1392  1.2658 12.5988

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)            -8.211e+01  4.824e+01  -1.702  0.08956 .
Weight                  1.990e-02  3.264e-02   0.610  0.54244
ModelYear               1.900e+00  6.415e-01   2.962  0.00324 **
I(Weight * ModelYear)  -5.184e-04  4.360e-04  -1.189  0.23515
I(Weight^2)            -1.360e-06  5.270e-06  -0.258  0.79641
I(Weight^2 * ModelYear)  4.398e-08  7.077e-08   0.621  0.53469
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.974 on 386 degrees of freedom
Multiple R-squared:  0.8566,    Adjusted R-squared:  0.8548
F-statistic: 461.3 on 5 and 386 DF,  p-value: < 2.2e-16
```

Figure 20: Summary del modelo lineal múltiple con Interaccion y no linealidad. Se elimina $I(Weight^2)$.

```
Call:
lm(formula = Mpg ~ Weight + ModelYear + I(Weight * ModelYear) +
    I(Weight^2 * ModelYear), data = data)

Residuals:
   Min     1Q Median     3Q    Max
-8.861 -1.764 -0.119  1.268 12.620

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)            -7.013e+01  1.310e+01  -5.353 1.49e-07 ***
Weight                  1.156e-02  4.586e-03   2.520   0.0122 *
ModelYear               1.740e+00  1.644e-01  10.585  < 2e-16 ***
I(Weight * ModelYear)  -4.067e-04  5.535e-05  -7.349 1.20e-12 ***
I(Weight^2 * ModelYear)  2.573e-08  3.287e-09   7.826 4.85e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.971 on 387 degrees of freedom
Multiple R-squared:  0.8566,    Adjusted R-squared:  0.8551
F-statistic: 577.9 on 4 and 387 DF,  p-value: < 2.2e-16
```

Una vez vistos los Adjusted R2, se incluye a continuación la tabla de resultados de la validación cruzada.

Table 5: Resultados RMSE de los modelos lineales múltiples.

Modelo	**Train**	**Test**
Mpg∼.	3.359348	3.406877
Mpg∼Weight+ModelYear	3.373098	3.351711
Mpg∼Weight*ModelYear	3.146954	3.123947
No lineal e Interacción	2.951446	2.924419

2.3 KNN para Regresión

En esta subsección se emplea el algoritmo no paramétrico KNN. Se trata de un algoritmo que genera las predicciones con un conjunto de vecinos escogido mediante una medida de similitud o distancia. El número de vecinos empleados para la predicción depende del valor k que se escoja.

Para los modelos de KNN se han escogido las mismas fórmulas empleadas en regresión lineal múltiple. Los resultados se muestran en la Tabla 6. El modelo lineal y con interacción es Mpg˜ModelYear +I(Weight*ModelYear)+I(Weightˆ 2 * ModelYear).

Table 6: Resultados de MSE para train y test con diferentes modelos KNN.

Modelo	**Train**	**Test**
Mpg∼.	1.877867	2.755446
Mpg∼Weight+ModelYear	2.175694	2.874861
Mpg∼Weight*ModelYear	2.029122	2.875025
No lineal e interacción	2.094846	2.862043

Como se puede ver en la tabla, el mejor modelo es el que considera todas las variables. Como se menciona en las diapositivas de la asignatura, el mejor modelo con una fórmula no tiene por qué coincidir con el mejor para regresión lineal.

Para llevar a cabo la comparación de resultados, es necesario realizar una normalización como se explica en las diapositivas.

2.4 Comparación de modelos

En esta subsección se emplean los test estadísticos Wilcoxon, Friedman y Holms para comparar los modelos. Para ello se parte de la tabla de soluciones dadas por el profesor que incluyen modelos lineales (LM), KNN y M5. Se sustituyen los resultados obtenidos de LM y KNN para el conjunto de datos asignado y se emplean los test. Los resultados de regresión se muestran a continuación.

X	**out_train_lm**	**out_train_kknn**	**out_train_m5p**
abalone	4.820000e+00	2.220000e+00	4.250000e+00
ANACALT	1.700000e-01	6.300000e-03	5.900000e-03
autoMPG6	**8.710998e+00**	**7.592483e+00**	**6.870000e+00**
autoMPG8	1.079000e+01	3.550000e+00	6.600000e+00
baseball	4.481590e+05	2.020880e+05	3.925890e+05
california	4.826190e+09	1.560869e+09	2.558518e+09
concrete	1.070000e+02	2.870000e+01	3.000000e+01
dee	1.618800e-01	7.611000e-02	1.620100e-01
delta_ail	2.960000e-08	1.400000e-08	2.510000e-08
delta_elv	2.100000e-06	1.050000e-06	2.020000e-06
forestFires	3.945000e+03	2.206000e+03	3.980000e+03
friedman	7.230000e+00	1.420000e+00	4.360000e+00
house	2.061567e+09	5.259870e+08	9.384299e+08
mortgage	1.354300e-02	8.827000e-03	1.101500e-02
stock	5.350000e+00	1.800000e-01	5.900000e-01
treasury	5.520300e-02	1.599800e-02	4.040400e-02
wankara	2.430000e+00	2.740000e+00	1.510000e+00
wizmir	1.565000e+00	2.538000e+00	1.358000e+00

X	**out_test_lm**	**out_test_kknn**	**out_test_m5p**
abalone	4.950000e+00	5.400000e+00	4.680000e+00
ANACALT	1.700000e-01	1.200000e-02	7.000000e-03
autoMPG6	**8.552226e+00**	**3.526384e+00**	**8.240000e+00**
autoMPG8	1.140000e+01	8.110000e+00	8.350000e+00
baseball	5.366760e+05	5.661130e+05	5.464640e+05
california	4.844366e+09	3.845914e+09	3.158145e+09
concrete	1.090000e+02	6.835600e+01	3.800000e+01
dee	1.705200e-01	1.732600e-01	1.699600e-01
delta_ail	2.960000e-08	3.140000e-08	2.720000e-08
delta_elv	2.100000e-06	2.410000e-06	2.050000e-06
forestFires	4.060940e+03	5.841000e+03	4.071040e+03
friedman	7.298700e+00	3.196100e+00	5.349100e+00
house	2.072908e+09	1.425915e+09	1.305419e+09
mortgage	1.484100e-02	3.003600e-02	1.448300e-02
stock	5.510000e+00	4.500000e-01	1.000000e+00
treasury	6.082100e-02	4.743900e-02	8.124800e-02
wankara	2.490000e+00	6.790000e+00	1.650000e+00
wizmir	1.605000e+00	6.060000e+00	1.449000e+00

Como se menciona en las diapositivas de la asignatura, debido a la métrica de error usada, debemos normalizar. Para ello se sigue el procedimiento explicado. Los resultados se presentan a continuación en una tabla.

Table 7: Resultados del test Wilcoxon para LM y KNN.

	Train	**Test**
p-value	0.0004196	0.7337

Como se puede apreciar en la tabla, para test no existen diferencias significativas entre ambos. Existe un (1 - 0.7337)*100 = 26% de confianza de que sean distintos. Ocurre lo contrario para train. A continuación se aplican los test de Friedman y post-hoc Holm. Se adjuntan los resultados en la siguiente tabla.

Table 8: Resultados test de Friedman.

Train p-value	**Test p-value**
2.829e-07	1.07e-07

Como se observa en los resultados de Friedman, ambos p-values son muy bajos, indicando que existe una alta probabilidad de que al menos dos algoritmos sean diferentes entre sí.

Table 9: Resultados test de Holm.

	Train	
	1	2
2	0.0031	-
3	0.0032	0.0139
	Test	
	1	2
2	0.523	-
3	0.081	0.133

Por último, analizando el resultado del test de Holm, vemos como para test 1vs2 no llega a tener una confianza la suficientemente alta, sólamente un 47,7%. Por lo tanto no existen diferencias significativas entre KNN y LM. Por otro lado, vemos como para el caso de M5 frente a LM existe un 91.9% y un 87% para M5 y KNN. Realmente ninguna comparativa alcanza la confianza del 95%. Pero la última comparación tiene altas probabilidades de ser considerados diferentes.

```
library(ggplot2)
library(corrplot)
library(car)
library(dplyr)
```

```
library(tidyr)
library(moments)
library(reshape2)
library(caret)
set.seed(77183983)

# Read the data

datapath = "C://Users//Yus//Google Drive//Master//IntroCienciaDatos//Datasets Regresion//autoMPG6"
data = read.csv(paste0(datapath,"//autoMPG6.dat"), header = FALSE, skip = 10)

# Change the names of the variables
colnames(data) <- c("Displacement", "HorsePower", "Weight", "Acceleration", "ModelYear", "Mpg")

# Normalize max-min function.
normalize <- function(x){
  return((x-min(x))/(max(x)-min(x)))
}

# Copy the data into a new variable an normalize all the columns
norm_data <- data.frame(data)
norm_data["Displacement"] <- as.data.frame(apply(norm_data["Displacement"], 2, normalize))
norm_data["HorsePower"] <- as.data.frame(apply(norm_data["HorsePower"],2, normalize))
norm_data["Weight"] <- as.data.frame(apply(norm_data["Weight"], 2, normalize))
norm_data["Acceleration"] <- as.data.frame(apply(norm_data["Acceleration"], 2, normalize))
norm_data["ModelYear"] <- as.data.frame(apply(norm_data["ModelYear"], 2, normalize))
norm_data["Mpg"] <- as.data.frame(apply(norm_data["Mpg"], 2, normalize))

# Fit a regression model for each variables without normalization, for ordinary least squares it is invariant.
fitDisplacement <- lm(Mpg~Displacement, data=data)
sumDisplacement <- summary(fitDisplacement)

fitHorsePower <- lm(Mpg~HorsePower, data=data)
sumHorsePower <- summary(fitHorsePower)

fitWeight <- lm(Mpg~Weight, data=data)
sumWeight <- summary(fitWeight)

fitAcceleration <- lm(Mpg~Acceleration, data=data)
sumAcceleration <- summary(fitAcceleration)

fitModelYear <- lm(Mpg~ModelYear, data=data)
sumModelYear <- summary(fitModelYear)

lm_fold <- function(iter, formula, tt= "test") {

  # Load the required test or train data fold
```

```r
  file <-paste(paste0(datapath,"//autoMPG6-5-", iter, "tra.dat", sep=""))
  x_tra<-read.csv(file, comment.char="@")
  file <-paste(paste0(datapath,"//autoMPG6-5-", iter, "tst.dat", sep=""))
  x_tst<-read.csv(file, comment.char="@")

  # Add names
  names(x_tst) <- c("Displacement", "HorsePower", "Weight", "Acceleration", "ModelYear", "Mpg")
  names(x_tra) <- c("Displacement", "HorsePower", "Weight", "Acceleration", "ModelYear", "Mpg")

  if (tt== "train") {
    test <-x_tra
  }
  else {
    test <-x_tst
  }

  fitlm=lm(formula,data=x_tra)
  # print(summary(fitlm))
  yprime=predict(fitlm,test)
  # print(test$Mpg)
  # print(yprime)
  # Return the RMSE measure for the given fold
  sqrt(sum(abs(test$Mpg-yprime)^2)/length(yprime))
}

RMSE.Test <- c()
RMSE.Training <- c()

# Train and validate the models for each variable
RMSE.Training <- c(RMSE.Training,mean(sapply(1:5, lm_fold, Mpg~Displacement,"train")))
RMSE.Training <- c(RMSE.Training,mean(sapply(1:5, lm_fold, Mpg~HorsePower,"train")))
RMSE.Training <- c(RMSE.Training,mean(sapply(1:5, lm_fold, Mpg~Weight,"train")))
RMSE.Training <- c(RMSE.Training,mean(sapply(1:5, lm_fold, Mpg~Acceleration,"train")))
RMSE.Training <- c(RMSE.Training,mean(sapply(1:5, lm_fold, Mpg~ModelYear, "train")))
RMSE.Test <- c(RMSE.Test,mean(sapply(1:5, lm_fold, Mpg~Displacement,"test")))
RMSE.Test <- c(RMSE.Test,mean(sapply(1:5, lm_fold, Mpg~HorsePower,"test")))
RMSE.Test <- c(RMSE.Test,mean(sapply(1:5, lm_fold, Mpg~Weight,"test")))
RMSE.Test <- c(RMSE.Test,mean(sapply(1:5, lm_fold, Mpg~Acceleration,"test")))
RMSE.Test <- c(RMSE.Test,mean(sapply(1:5, lm_fold, Mpg~ModelYear,"test")))

# Modelos lineales m?ltiples
multlm <- lm(Mpg~., data=data)
sumMult <- summary(multlm)
```

```
WMlm <- lm(Mpg~Weight+ModelYear, data=data)
sumWMlm <- summary(WMlm)

wminterlm <- lm(Mpg~Weight*ModelYear, data=data)
sumwminterlm <- summary(wminterlm)

# Interacciones y no linealidad
model.quadratic.WM <- lm(Mpg~(Weight*ModelYear), data)
sumQuadWM <- summary(model.quadratic.WM)

inter.nolineal <- lm(Mpg~ ModelYear +I(Weight*ModelYear)+I(Weight^2 * ModelYear), data)
sumInterNoL <- summary(inter.nolineal)

# kfold CV for multiple regression models
RMSE.Test.M <- c()
RMSE.Training.M <- c()

RMSE.Training.M <- c(RMSE.Training.M,mean(sapply(1:5, lm_fold, Mpg~.,"train")))
RMSE.Training.M <- c(RMSE.Training.M,mean(sapply(1:5, lm_fold, Mpg~Weight+ModelYear,"train")))
RMSE.Training.M <- c(RMSE.Training.M,mean(sapply(1:5, lm_fold, Mpg~Weight*ModelYear,"train")))
RMSE.Training.M <- c(RMSE.Training.M,mean(sapply(1:5, lm_fold,
                                                   Mpg~ ModelYear +I(Weight*ModelYear)+I(Weight^2 * ModelYear),
                                                   "train")))

RMSE.Test.M <- c(RMSE.Test.M,mean(sapply(1:5, lm_fold, Mpg~.,"test")))
RMSE.Test.M <- c(RMSE.Test.M,mean(sapply(1:5, lm_fold, Mpg~Weight+ModelYear,"test")))
RMSE.Test.M <- c(RMSE.Test.M,mean(sapply(1:5, lm_fold, Mpg~Weight*ModelYear,"test")))
RMSE.Test.M <- c(RMSE.Test.M,mean(sapply(1:5, lm_fold,
                                           Mpg~ ModelYear +I(Weight*ModelYear)+I(Weight^2 * ModelYear),
                                           "test")))


# KNN for regression
require(kknn)

# A k-fold CV function for KNN is
knn_cvfold <- function(iter, formula, k=7, tt= "test") {

  # Load the required test or train data fold
  file <-paste(paste0(datapath,"//autoMPG6-5-", iter, "tra.dat", sep=""))
  x_tra<-read.csv(file, comment.char="@")
  file <-paste(paste0(datapath,"//autoMPG6-5-", iter, "tst.dat", sep=""))
```

```
  x_tst<-read.csv(file, comment.char="@")

  # Add names
  names(x_tst) <- c("Displacement", "HorsePower", "Weight", "
      Acceleration", "ModelYear", "Mpg")
  names(x_tra) <- c("Displacement", "HorsePower", "Weight", "
      Acceleration", "ModelYear", "Mpg")

  if (tt== "train") {
    test <-x_tra
  }
  else {
    test <-x_tst
  }

  knnfit=kknn(formula,x_tra,test,k=k)
  yprime=knnfit$fitted.values
  # Return the RMSE measure for the given fold
  sqrt(sum(abs(test$Mpg-yprime)^2)/length(yprime))
}

KNN.train <- c()
KNN.test <- c()

# KNN with all the variables
KNN.train <- c(KNN.train, mean(sapply(1:5,knn_cvfold,Mpg~.,7, "
    train")))
KNN.test <- c(KNN.test, mean(sapply(1:5,knn_cvfold,Mpg~., 7, "test"
    )))
# KNN with Weight+ModelYear
KNN.train <- c(KNN.train, mean(sapply(1:5,knn_cvfold,Mpg~Weight+
    ModelYear, 7, "train")))
KNN.test <- c(KNN.test, mean(sapply(1:5,knn_cvfold,Mpg~Weight+
    ModelYear, 7, "test")))
# KNN with interaction
KNN.train <- c(KNN.train, mean(sapply(1:5,knn_cvfold,Mpg~Weight*
    ModelYear, 7, "train")))
KNN.test <- c(KNN.test, mean(sapply(1:5,knn_cvfold,Mpg~Weight*
    ModelYear, 7, "test")))
#KNN with interactions and non-linearity
KNN.train <- c(KNN.train, mean(sapply(1:5,knn_cvfold,
                                      Mpg~ModelYear +I(Weight*
                                          ModelYear)+I(Weight^2 *
                                          ModelYear),
                                      7, "train")))
KNN.test <- c(KNN.test, mean(sapply(1:5,knn_cvfold,
                                    Mpg~ModelYear +I(Weight*
                                        ModelYear)+I(Weight^2 *
                                        ModelYear),
                                    7, "test")))

# Load the regression results for the rest of the datasets
ruta_resultados <- "C://Users//Yus//Google Drive//Master//
    IntroCienciaDatos//"
reg_train_results <- read.csv(paste0(ruta_resultados,"regr_train_
    alumnos.csv"), sep = ";")
reg_test_results <- read.csv(paste0(ruta_resultados,"regr_test_
```

```
    alumnos.csv"), sep = ";")

# Change the results for Mpg6 with ours
reg_test_results$out_test_lm[[3]] <- 2.924419^2
reg_train_results$out_train_lm[[3]] <- 2.95144^2
reg_test_results$out_test_kknn[[3]] <- 1.877867^2
reg_train_results$out_train_kknn[[3]]<- 2.755446^2

# Compare lm(other) to knn(ref) train
dif_train <- (reg_train_results$out_train_lm - reg_train_results$
    out_train_kknn) /
  reg_train_results$out_train_lm

dif_test <- (reg_test_results$out_test_lm - reg_test_results$out_
    test_kknn) /
  reg_test_results$out_test_lm

wilcox_train <- cbind(ifelse (dif_train<0, abs(dif_train)+0.1,
    0+0.1), ifelse (dif_train>0, abs(dif_train)+0.1, 0+0.1))
wilcox_test <- cbind(ifelse (dif_test<0, abs(dif_test)+0.1, 0+0.1),
     ifelse (dif_test>0, abs(dif_test)+0.1, 0+0.1))

colnames(wilcox_train) <- c(colnames(reg_train_results)[2],
    colnames(reg_train_results)[3])
colnames(wilcox_test) <- c(colnames(reg_test_results)[2], colnames(
    reg_test_results)[3])
LM.KNN.train<-wilcox.test( wilcox_train[,1], wilcox_train[,2],
    alternative = "two.sided", paired=TRUE)
LM.KNN.test<-wilcox.test( wilcox_test[,1], wilcox_test[,2],
    alternative = "two.sided", paired=TRUE)

# Test de friedman y holm
friedman.train <- friedman.test(as.matrix(reg_train_results[,-c(1)
    ]))
friedman.test <- friedman.test(as.matrix(reg_test_results[,-c(1)]))

# Holm train
tam <-dim(reg_train_results[,-c(1)])
groups <-rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(reg_train_results[,-c(1)]), groups,
    p.adjust= "holm", paired = TRUE)

# Holm test
tam <-dim(reg_test_results[,-c(1)])
groups <-rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(reg_test_results[,-c(1)]), groups, p
    .adjust= "holm", paired = TRUE)
```

3 Clasificación

En esta sección se emplean KNN, LDA y QDA como algoritmos de clasificación y se comparan los resultados obtenidos.

3.1 KNN

En primer lugar, se busca el mejor valor de k para el algoritmo de vecindad KNN. Para ello se prepara una función de validación cruzada y posteriormente se relanza para los valores de 1 a 17. Basándome en mi experiencia a partir de valores superiores a 11 el algoritmo KNN empieza a perder calidad, puede ser interesante contrastar esta hipótesis con los resultados.

Debido a que en el EDA del conjunto de clasificación se llevó a cabo una selección de características para evitar tener que analizar tantas y que los gráficos fueran difíciles de interpretar, se va a utilizar dicha selección y se va a comparar el rendimiento de los algoritmos con y sin la selección.

El código para la realización de esta parte es:

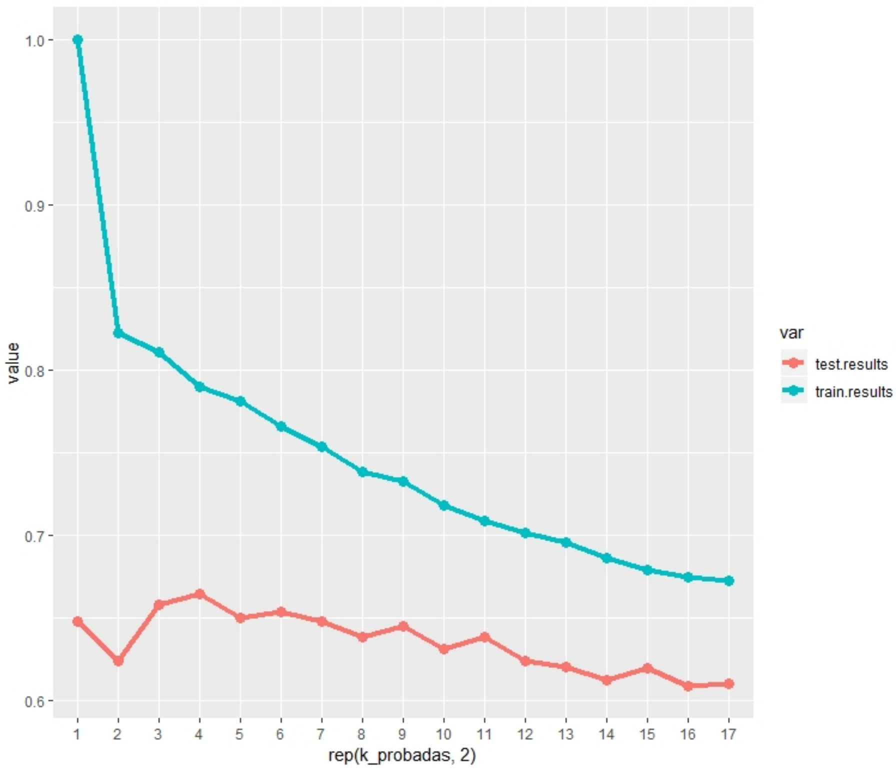
```r
library(ggplot2)
library(corrplot)
library(car)
library(dplyr)
library(tidyr)
library(moments)
library(reshape2)
library(caret)
library(class)
set.seed(77183983)

# Leer los datos
datapath = "C:/Users/Yus/Google Drive/Master/IntroCienciaDatos/Datasets Clasificacion/vehicle"
data = read.csv(paste0(datapath,"//vehicle.dat"), comment.char = "@")

# Cambiar el nombre de las variables
colnames(data) <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
                    "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
                    "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
                    "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
                    "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")

# Se comienza con el modelo KNN, se trata de buscar el mejor valor de k.
# Para ello definimos la funci n de validaci n cruzada primero.

kfold_cv_knn <- function(iter, k, tt= "test") {

  file <-paste0(datapath,"//vehicle-10-", iter, "tra.dat", sep="")
  # print(paste0("Read:",file))
  x_tra<-read.csv(file, comment.char="@", header=F)
  file <-paste0(datapath,"//vehicle-10-", iter, "tst.dat", sep="")
```

```r
  # print(paste0("Read:",file))
  x_tst<-read.csv(file , comment.char="@", header=F)

  # Se a aden los nombres de todas las variables
  names <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
             "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
             "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
             "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
             "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")
  names(x_tst) <- names
  names(x_tra) <- names


  # A adimos las variables que se escogieron en el EDA
  variables <- c("Max_length_aspect_ratio","Minor_variance", "Length_rectangular", "Major_variance",
                 "Elongatedness","Distance_circularity", "Scatter_ratio", "Compactness","Hollows_ratio",
                 "Praxis_rectangular","Class")
  # Filtramos las variables que no queremos
  x_tra <- x_tra[,names(x_tra) %in% variables]
  x_tst <- x_tra[,names(x_tra) %in% variables]

  if (tt == "train"){
    test <- x_tra
  } else if(tt == "test") {
    test <- x_tst
  }

  pred <- knn(train = x_tra[,-ncol(x_tra)], test = test[,-ncol(test)], cl = x_tra$Class, k = k)
  length(pred[pred==test[,ncol(test)]])/length(pred)
}

test_k <- function(k, test = "test"){
  # kfold_cv_knn <- fuunction(iter , k, tt= "test")
  # mean(sapply(1:10, kfold_cv_knn, k=k, tt=test))
  (sapply(1:10, kfold_cv_knn, k=k, tt=test))
}

# Ejecutamos train y test
train.results <- sapply(1:17, test_k, "train")
test.results <- sapply(1:17, test_k, "test")

# Se unen y se preparan los resultados para la gr fica
resultados <- data.frame(train.results, test.results)
k_probadas <- as.factor(1:17)

aux <- gather(resultados, var, value)
# Mostramos los resultados
ggplot(aux, aes(x=rep(k_probadas,2), y=value, group=var, col=var))+
  geom_line(size=1.5) +
```

```
  geom_point(size=3, fill="white")
```

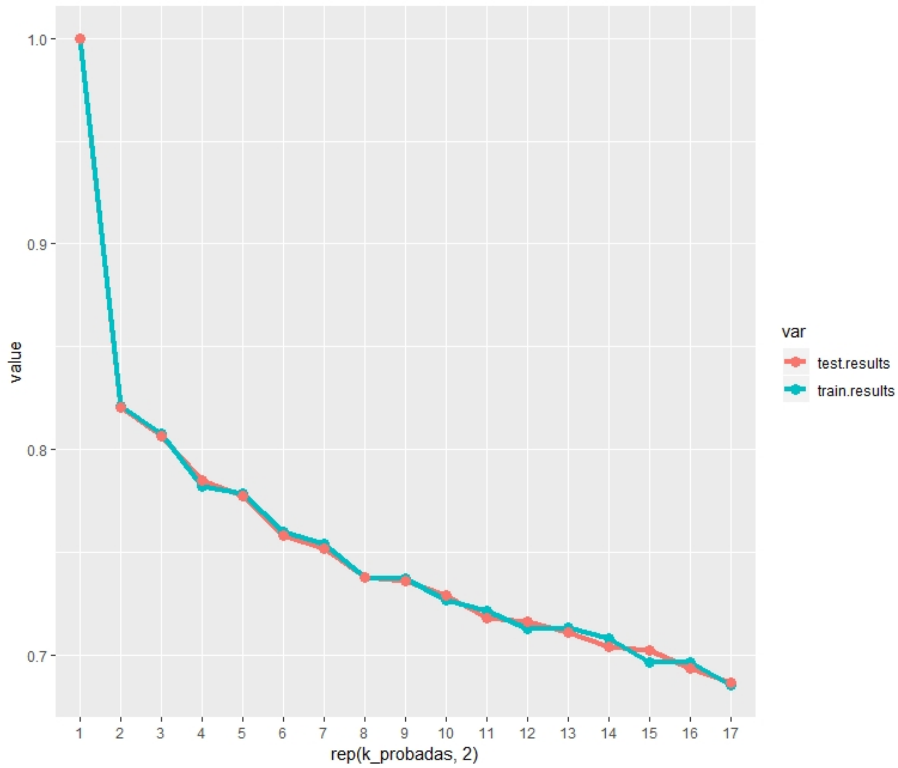
En las Figuras 21 y 22 se muestran los resultados para cada valor de k para train y test. La primera figura corresponde a todas las variables del conjunto y la segunda a la selección de variables hecha en el EDA.

Figure 21: Summary del modelo lineal múltiple con Weight y ModelYear.



En esta primera figura se puede ver como el valor k=1 para entrenamiento da un 100% de precisión y como los valores de train son bastante mejores que los de test. Esto se debe a un sobreajuste del modelo. Para este conjunto de datos, cuanto mayor es el número de vecinos peor es el rendimiento. Por lo tanto, por ahora el mejor valor es k=3 o k=4.

Figure 22: Summary del modelo lineal múltiple con Weight y ModelYear.



En cuanto a la segunda figura, se puede ver como una selección de variables ayuda en el rendimiento del modelo. De nuevo se puede observar como los valores más bajos de k ofrecen los mejores resultados. Por temor a hacer un modelo muy sensible al ruido y no preprocesar los conjuntos en esta asignatura, no eligiría k=1. Por lo tanto, como valor óptimo de k se escoge 3 para ayudar al desempate frente a 2.

3.2 LDA

En esta subsección se emplea Linear Discriminant Analysis para clasificación. Al igual que en Regresión Logística, se determina una función lineal que busca separar los datos por su valor de clase. Debido al funcionamiento de LDA, se ha considerado la eliminación de variables correladas. Tras varios intentos e incluso probando con el subconjunto de variables resultantes de la selección de RandomForest, ninguna alternativa mejora el modelo con todas las variables y un scale() aplicado a estas.

Modelo	Precisión Train	Precisión Test
Todas las vars.	0.7989229	0.781330
Selección vars EDA	0.7341756	0.7341756
Todas las vars. y Scale	0.7989229	0.7989229
-2 vars	0.7892042	0.7892042
-3 vars	0.7926194	0.7926194
-4 vars	0.7872327	0.7872327

Los modelos con -X vars indican que se han eliminado variables altamente correladas.

El código para esta sección es:

```r
library(reshape2)
library(MASS)
set.seed(77183983)

# Leer los datos
datapath = "C:/Users/Yus/Google Drive/Master/IntroCienciaDatos/Datasets Clasificacion/vehicle"
data = read.csv(paste0(datapath,"//vehicle.dat"), comment.char = "@")

# Cambiar el nombre de las variables
colnames(data) <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
                    "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
                    "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
                    "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
                    "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")

kfold_cv_LDA <- function(iter, tt= "test") {

  file <-paste0(datapath,"//vehicle-10-", iter, "tra.dat", sep="")
  # print(paste0("Read:",file))
  x_tra<-read.csv(file, comment.char="@", header=F)
  file <-paste0(datapath,"//vehicle-10-", iter, "tst.dat", sep="")
  # print(paste0("Read:",file))
  x_tst<-read.csv(file, comment.char="@", header=F)

  # Se aniaden los nombres de todas las variables
  names <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
             "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
             "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
             "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
             "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "
```

```
                  Class")
  names(x_tst) <- names
  names(x_tra) <- names


  # Aniadimos las variables que se escogieron en el EDA
  # variables <- c("Max_length_aspect_ratio","Minor_variance", "Length_rectangular", "Major_variance",
  #                "Elongatedness","Distance_circularity", "Scatter_ratio", "Compactness","Hollows_ratio",
  #                "Praxis_rectangular","Class")

  # Variables sin correlaciones
  variables <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
           "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
           "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
           "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
           "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")


  # Filtramos las variables que no queremos
  x_tra <- x_tra[,names(x_tra) %in% variables]
  x_tst <- x_tra[,names(x_tra) %in% variables]
  x_tra[,-ncol(x_tra)] <- scale(x_tra[,-ncol(x_tra)], center=TRUE, scale=TRUE)
  x_tst[,-ncol(x_tra)] <- scale(x_tst[,-ncol(x_tst)], center=TRUE, scale=TRUE)

  # En funci n de si empleamos train o test para evaluar resultados
  if (tt == "train"){
    test <- x_tra
  } else if(tt == "test") {
    test <- x_tst
  }

  # Prediccion de knn
  lda.model <- lda(Class~., data = x_tra)
  pred <- predict(lda.model, test)
  # % de acierto
  length(pred[pred$class==test[,ncol(test)]])/length(pred$class)
}

# Ejecutamos train y test
train.results <- mean(sapply(1:10, kfold_cv_LDA, tt="train"))
test.results <- mean(sapply(1:10, kfold_cv_LDA, tt="test"))

# Se unen y se preparan los resultados para la gr fica
resultados <- data.frame(train.results, test.results)
print(resultados)
```

3.3 QDA

En este caso se aplica Quadratic Discriminant Analysis para clasificación. Existe la posibilidad de que QDA funcione mejor que LDA al haber varianzas distintas entre clases. Se parte de los modelos anteriores como prueba.

Modelos	**Precisión Train**	**Precisión Test**
Todas las vars.	0.9123989	0.8522409
Selección vars EDA	0.7980039	0.7980039
Todas las vars. y Scale	0.9123989	0.9123989

Como se puede observar de nuevo, todas las variables resulta en el mejor modelo. Es interesante ver como en LDA y QDA, la selección de variables de RandomForest no ha hecho más que empeorar el rendimiento. En ambos casos el uso de scale() ha mejorado un poco el rendimiento del test. De nuevo, se ha intentando eliminar algunas variables sin mejora. Parece ser que QDA tiene las suficientes muestras para estimar adecuadamente las varianzas dando un rendimiento bastante bueno de 91% de precisión en test.

El código para esta sección es:

```
library(reshape2)
library(MASS)
set.seed(77183983)

# Leer los datos
datapath = "C:/Users/Yus/Google Drive/Master/IntroCienciaDatos/Datasets Clasificacion/vehicle"
data = read.csv(paste0(datapath,"//vehicle.dat"), comment.char = "@")

# Cambiar el nombre de las variables
colnames(data) <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
                    "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
                    "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
                    "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
                    "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")

kfold_cv_QDA <- function(iter, tt= "test") {

  file <-paste0(datapath,"//vehicle-10-", iter, "tra.dat", sep="")
  # print(paste0("Read:",file))
  x_tra<-read.csv(file, comment.char="@", header=F)
  file <-paste0(datapath,"//vehicle-10-", iter, "tst.dat", sep="")
  # print(paste0("Read:",file))
  x_tst<-read.csv(file, comment.char="@", header=F)
```

```r

# Se aniaden los nombres de todas las variables
names <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
           "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
           "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
           "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
           "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")
names(x_tst) <- names
names(x_tra) <- names


# Aniadimos las variables que se escogieron en el EDA
# variables <- c("Max_length_aspect_ratio","Minor_variance", "Length_rectangular", "Major_variance",
#                  "Elongatedness","Distance_circularity", "Scatter_ratio", "Compactness","Hollows_ratio",
#                  "Praxis_rectangular","Class")

# Variables sin correlaciones
# variables <- c("Compactness","Circularity", "Distance_circularity", "Radius_ratio",
#                  "Praxis_aspect_ratio", "Max_length_aspect_ratio", "Scatter_ratio",
#                  "Elongatedness", "Praxis_rectangular", "Length_rectangular", "Major_variance",
#                  "Minor_variance", "Gyration_radius", "Major_skewness", "Minor_skewness",
#                  "Minor_kurtosis", "Major_kurtosis", "Hollows_ratio", "Class")


# Filtramos las variables que no queremos
# x_tra <- x_tra[,names(x_tra) %in% variables]
# x_tst <- x_tra[,names(x_tra) %in% variables]
x_tra[,-ncol(x_tra)] <- scale(x_tra[,-ncol(x_tra)], center=TRUE, scale=TRUE)
x_tst[,-ncol(x_tra)] <- scale(x_tst[,-ncol(x_tst)], center=TRUE, scale=TRUE)

# En funci n de si empleamos train o test para evaluar resultados
if (tt == "train"){
  test <- x_tra
} else if(tt == "test") {
  test <- x_tst
}

# Prediccion de knn
lda.model <- qda(Class~., data = x_tra)
pred <- predict(lda.model, test)
# % de acierto
length(pred[pred$class==test[,ncol(test)]])/length(pred$class)
```

```
}

# Ejecutamos train y test
train.results <- mean(sapply(1:10, kfold_cv_QDA, tt="train"))
test.results <- mean(sapply(1:10, kfold_cv_QDA, tt="test"))

# Se unen y se preparan los resultados para la gr fica
resultados <- data.frame(train.results, test.results)
print(resultados)
```

3.4 Comparación de algoritmos de Clasificación

Para la comparación de algoritmos de clasificación se sigue el mismo procedimiento que regresión. Se coge el conjunto de soluciones dado para el resto de conjuntos de datos con las soluciones para cada algoritmo. Se sustituyen las soluciones del dataset que me ha sido asignado y se evaluan los test del Wilcoxon y posteriormente se empleará el test de Holm para evaluar los tres algoritmos a la vez. A continuación se presentan los resultados globales con todos los sets y algoritmos.

X	**out_train_knn**	**out_train_lda**	**out_train_qda**
appendicitis	0.8834602	0.8815461	0.8690241
australian	0.7277419	0.8605475	0.8072464
balance	0.9072122	0.8791122	0.9167999
bupa	0.7405521	0.7024224	0.6447628
contraceptive	0.6168944	0.5236485	0.531418
haberman	0.7795116	0.7519934	0.7567115
hayes-roth	0.6475524	0.5604167	0.7361111
heart	0.7342975	0.8576132	0.8777778
iris	0.9791045	0.98	0.9814815
led7digit	0.7636971	0.7635556	0.7680556
mammographic	0.8160856	0.8274465	0.8196843
monk-2	0.9793684	0.7821826	0.930301
newthyroid	0.9158409	0.9183457	0.9700283
pima	0.7791914	0.7792266	0.7633125
tae	0.526346	0.5584858	0.5688072
titanic	0.7892319	0.7760111	0.7732851
vehicle	**0.8209853**	**0.7989229**	**0.9123989**
vowel	0.8378652	0.6457912	0.9701459
wine	0.7745126	1	0.995625
wisconsin	0.9739304	0.9614471	0.9588436

X	out_test_knn	out_test_lda	out_test_qda
appendicitis	0.8966667	0.8690909	0.8109091
australian	0.6838235	0.857971	0.8028986
balance	0.9024546	0.8624101	0.9167905
bupa	0.6865775	0.6837924	0.5991759
contraceptive	0.5448653	0.5091561	0.5173102
haberman	0.7462069	0.748172	0.7512903
hayes-roth	0.5666667	0.55	0.5875
heart	0.6692308	0.8481481	0.8296296
iris	0.9642857	0.98	0.9733333
led7digit	0.7510204	0.742	0.6975
mammographic	0.7977698	0.8241269	0.8194042
monk-2	0.9743632	0.7703433	0.9235535
newthyroid	0.9071429	0.9164502	0.962987
pima	0.7348861	0.770993	0.7412403
tae	0.3838095	0.5245833	0.5425
titanic	0.7850353	0.7760304	0.7733032
vehicle	**0.820463**	**0.7989229**	**0.9123989**
vowel	0.6428571	0.6030303	0.9191919
wine	0.6959559	0.9944444	0.9888889
wisconsin	0.9735023	0.9592185	0.9519476

En primer lugar se analizan los resultados del test de Wilconxon.

Table 10: Resultados del test Wilconxon por parejas.

Comparación	Train	Test
LDAvsKNN	0.3683	0.9563
QDAvsKNN	0.3118	0.1769
QDAvsLDA	0.1769	0.8124

Como se puede ver en los resultados, no existe ningun test cuyo p-value indique que los algoritmos son significativamente diferentes. Existen algunos casos donde la probabilidad es más alta, como el test de QDA y KNN.

Table 11: Resultados test de Holm.

	Train	
	1	2
2	0.62	-
3	0.62	0.53
	Test	
	1	2
2	1	-
3	0.53	1

Por último, como podemos comprobar, ninguna comparación tiene una probabilidad alta como para decir que existen diferencias entre los algoritmos. Los p-value se alejan mucho del 95% de confianza. Hay incluso casos extremos de p-value = 1.

El código empleado para los test en este apartado de clasifición es:

```
# Load the regression results for the rest of the datasets
ruta_resultados <- "C://Users//Yus//Google Drive//Master//IntroCienciaDatos//"
clasif_train_results <- read.csv(paste0(ruta_resultados,"clasif_train_alumnos.csv"), sep = ";")
clasif_test_results <- read.csv(paste0(ruta_resultados,"clasif_test_alumnos.csv"), sep = ";")

# Change the results for Mpg6 with ours
clasif_train_results$out_train_knn[17] <- 0.8209853
clasif_train_results$out_train_lda[17] <- 0.7989229
clasif_train_results$out_train_qda[17] <- 0.9123989
clasif_test_results$out_test_knn[17]<- 0.8204630
clasif_test_results$out_test_lda[17] <- 0.7989229
clasif_test_results$out_test_qda[17] <- 0.9123989

# Compare lm(other) to knn(ref) train
LDAvsKNNtrain <- wilcox.test(clasif_train_results$out_train_knn, clasif_train_results$out_train_lda, alternative="two.sided", paired=T)
LDAvsKNNtest <- wilcox.test(clasif_test_results$out_test_knn, clasif_test_results$out_test_lda, alternative="two.sided", paired=T)

QDAvsKNNtrain <- wilcox.test(clasif_train_results$out_train_knn, clasif_train_results$out_train_qda, alternative="two.sided", paired=T)
QDAvsKNNtest <- wilcox.test(clasif_test_results$out_test_knn, clasif_test_results$out_test_qda, alternative="two.sided", paired=T)

QDAvsLDAtrain <- wilcox.test(clasif_train_results$out_train_lda, clasif_train_results$out_train_qda, alternative="two.sided", paired=T)
QDAvsLDAtest <- wilcox.test(clasif_test_results$out_test_lda,
```

```
    clasif_test_results$out_test_qda, alternative="two.sided",
    paired=T)

clasif_train_results <- clasif_train_results[,-1]
clasif_test_results <- clasif_test_results[,-1]
tam <- dim(clasif_train_results)
groups <- rep(1:tam[2],each=tam[1])
pairwise.wilcox.test(x=as.matrix(clasif_train_results),groups,p.
    adjust="holm",paired=T)
tam <- dim(clasif_test_results)
groups <- rep(1:tam[2],each=tam[1])
pairwise.wilcox.test(x=as.matrix(clasif_test_results),groups,p.
    adjust="holm",paired=T)
```