Introducción a la Ciencia de Datos

José Alberto

10 de diciembre de 2018

Análisis Exploratorio de los Datos

Los datasets que se nos han asignado son heart para hacer el estudio de clasificación y friedman para el de regresión. Lo que vamos a hacer en esta sección será un estudio preliminar sobre las variables que tenemos en cada caso. Vamos a ver cuántas variables son, de qué tipo, qué relación puede existir entre ellas y qué si su distribución se asemeja a alguna distribución que conozcamos.

EDA - Clasificación (heart)

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## Attaching package: 'GGally'

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

## corrplot 0.84 loaded

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:kknn':
##
##     contr.dummy

##
## Attaching package: 'MASS'

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

Para el problema de clasificación tenemos el datset heart con información de 270 pacientes a los que se les realizaron una serie de pruebas para obtener información sobre su sexo, edad o presión sanguínea entre otros. En este caso trataremos clasificar a los pacientes en función de si tienen una enfermedad cardíaca o no.

Primero haremos un estudio básico sobre las dimensiones del dataset.

```
dim(heart)
```

```
## [1] 270  14
```

```
heart[!complete.cases(heart),]
```

```
##  [1] X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 Y
## <0 rows> (or 0-length row.names)
```

Como vemos el dataset heart tiene 270 observaciones de 14 variables. Además vemos que no tenemos ningún missing value en todo el dataset, por lo que no tendremos que hacer consideraciones especiales sobre qué hacer con los valores NA (o missing value).

En particular las varibles $X1, ..., X13$ son:

1. age: la edad del paciente.
2. sex: el sexo del paciente (0 hombre, 1 mujer).
3. chest pain type: tipo de dolor de pecho (toma valores enteros del 1 al 4).
4. resting blood pressure: la presión sanguínea en reposo (toma valores reales entre 94 y 200).
5. serum cholestoral in mg/dl: colesterol sérico del paciente en miligramos por decilitros (toma valores reales entre 126 y 564).
6. fasting blood sugar $>$ 120 mg/dl: toma valor 1 si el azúcar en sangre en ayunas es mayor a 120 miligramos por decilitro y 0 de lo contrario.
7. resting electrocardiographic results: valores del electrocardiograma en reposo (toma valores enteros del 0 al 2).
8. maximum heart rate achieved: máximas pulsaciones alcanzadas (toma valores reales entre 71 y 202).
9. exercise induced angina: toma 1 si el ejercicio le ha causado una angina al paciente y 0 de lo contrario.
10. oldpeak: depresión del segmento ST.
11. slope: inclinación máxima del semento ST.
12. major vessels: número de vasos mayores coloreados por una fluoroscopia (toma valores entre 0 y 3).
13. thal: si el paciente padece talasemia (3 no padece, 6 crónica, 7 reversible).

La variable a predecir Y determina si el paciente tiene una enfermedad cardíaca o no tomando los valores 2 y 1 respectivamente.

Aunque tal y como nos han dado el dataset todas las variables son numéricas (pues las variables nominales han sido transformadas), atendiendo a su naturaleza original las podemos separar en:

- Reales: $X4, X5, X8, X10$.
- Enteras: $X1, X12$.
- Ordinales (nominales con una relación de orden): $X11$.
- Binarias (nominales con dos valores): $X2, X6, X9$.
- Nominales: $X3, X7, X13$.

Dicho esto nosotros vamos a considerar las variables $X1, X4, X5, X8, X10$ como si fueran continuas y el resto como categóricas. Hacemos esto porque la variable $X12$, pese a ser numérica y su valor tener un significado numérico real, solo toma 4 valores. Por tanto a la hora de representarla y demás será más útil hacerlo con las mismas herramientas que usaremos con el resto de variables categóricas. Por otro lado la variable edad $X1$ es entera y toma valores discretos, pero su rango de valores es tan amplio que la mejor forma de representarla es hacerlo como si fuera continua. Esto es un criterio que hemos tomado nosotros y que podría cambiar a lo largo del estudio si los resultados no fuesen satisfactorios. Otra forma de actuar sería coger la edad y partirla en intervalos para tratarla como un factor ordenado, hay múltiples criteriors.

```
summary(heart)
```

```
##        X1              X2              X3              X4
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0
##  Median :55.00   Median :1.0000   Median :3.000   Median :130.0
##  Mean   :54.43   Mean   :0.6778   Mean   :3.174   Mean   :131.3
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
```

```
##        X5              X6               X7              X8
##  Min.   :126.0   Min.   :0.0000   Min.   :0.000   Min.   : 71.0
##  1st Qu.:213.0   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:133.0
##  Median :245.0   Median :0.0000   Median :2.000   Median :153.5
##  Mean   :249.7   Mean   :0.1481   Mean   :1.022   Mean   :149.7
##  3rd Qu.:280.0   3rd Qu.:0.0000   3rd Qu.:2.000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.000   Max.   :202.0
##        X9              X10             X11             X12
##  Min.   :0.0000   Min.   : 0.0   Min.   :1.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.: 0.0   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median : 4.0   Median :2.000   Median :0.0000
##  Mean   :0.3296   Mean   : 8.9   Mean   :1.585   Mean   :0.6704
##  3rd Qu.:1.0000   3rd Qu.:15.0   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :62.0   Max.   :3.000   Max.   :3.0000
##       X13              Y
##  Min.   :3.000   Min.   :1.000
##  1st Qu.:3.000   1st Qu.:1.000
##  Median :3.000   Median :1.000
##  Mean   :4.696   Mean   :1.444
##  3rd Qu.:7.000   3rd Qu.:2.000
##  Max.   :7.000   Max.   :2.000
```

El summary nos da información sobre todo sobre las variables reales. En el resto de variables al ser discretas (incluso nominales en algunos casos) podemos inferir de los percentiles o de la media un poco de información. Por ejemplo la media del sexo es mayor que 0.5, por tanto hay más hombres que mujeres. Sin embargo, esto aporta muy poco conocimiento sobre la distribución de estas variables así que las convertiremos en factores y posteriormnete trataremos de obtener mayor información por otras vías.

```
heart[,c(2,3,6,7,9,11,12,13,14)]=apply(heart[,c(2,3,6,7,9,11,12,13,14)],MARGIN=2,as.factor)
```

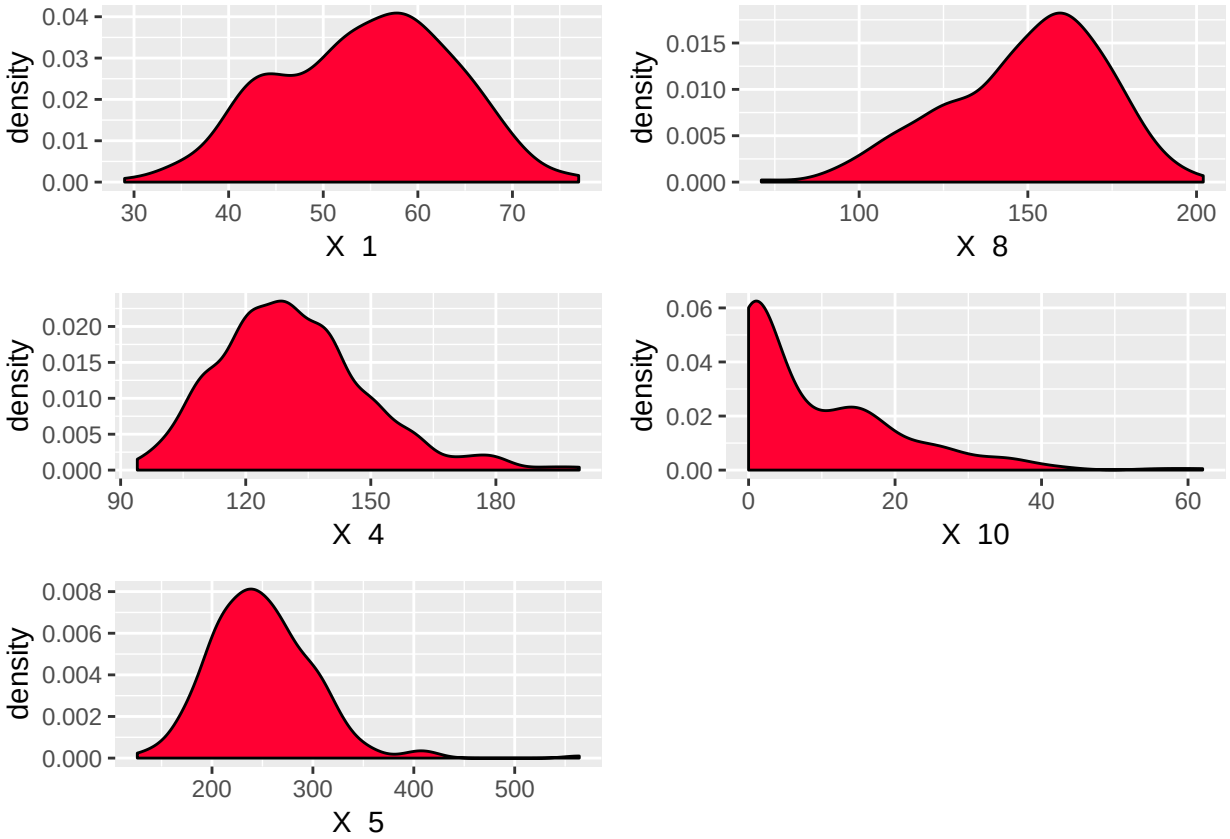
Para el caso de las variables reales, queremos conocer también su desviación estándar. Que la encontraremos en la siguiente tabla.

```
tabla<-data.frame( Standard.Dev=apply(as.matrix(heart[,c(1,4,5,8,10)]),MARGIN=2,FUN=sd))
knitr::kable(tabla)
```

	Standard.Dev
X1	9.109067
X4	17.861608
X5	51.686237
X8	23.165717
X10	11.003936

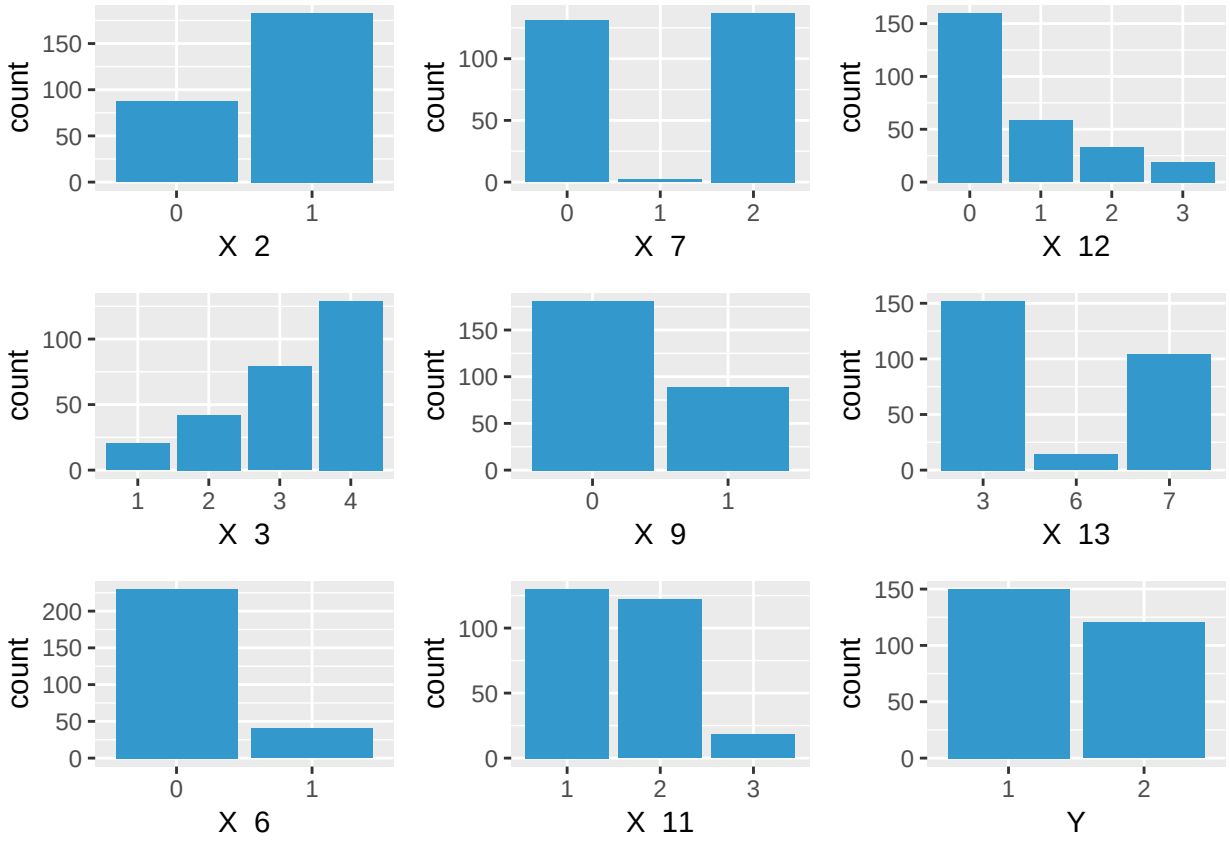
Como vemos las variables con mayor desviación estándar son $X5$ y $X8$, lo que quiere decir que en nuestra muestra donde más variabilidad hay en los pacientes es el nivel de colesterol sérico de los pacientes y en las pulsaciones máximas alcanzadas. Mientras que en la edad es donde vemos que existe menor variabilidad y que la edad de los pacientes no se desvía tanto de la media que estaba en torno a los 54 años.

A continuación visualizaremos las funciones de densidad que podrían seguir las variables reales que tenemos según nuestros datos.



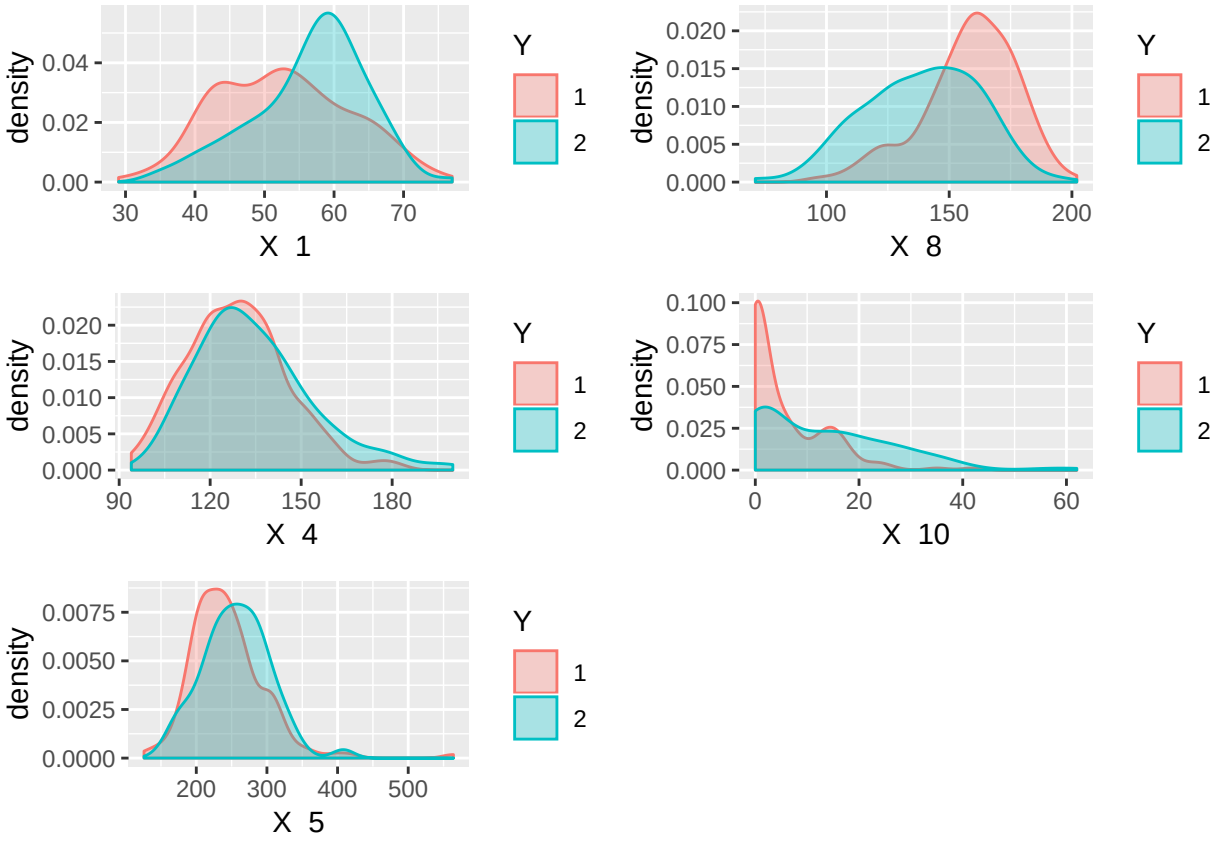
Como podemos observar salvo la variable $X10$ que sigue una distribución claramente sesgada hacia la izquierda, el resto están más o menos centradas. No parece que sigan distribuciones normales dado el tamaño de las colas. Aún así lo comprobaremos más tarde con un test de Shapiro ya que cuando apliquemos LDA la hipótesis de normalidad es importante y si obtenemos buenos o malos resultados querremos darles explicación.

A continuación haremos el procedimiento análogo para las variables discretas. En este caso lo que haremos será un diagrama de barras para ver cómo se distribuyen las clases dentro de nuestro dataset.



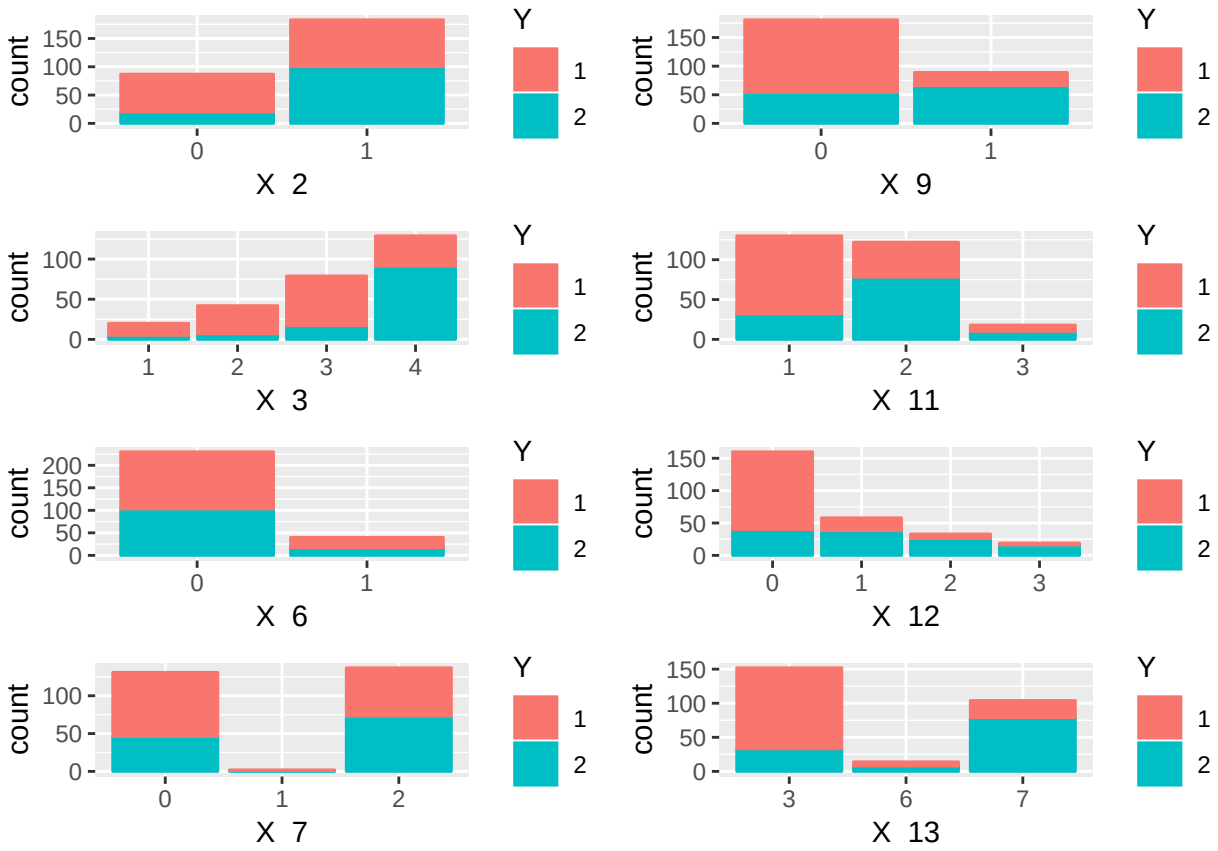
Como anunciábamos vemos que hay más hombres que mujeres en el estudio del dataset. También resulta interesante ver que de la variable $X7$ para la categoría 1 no tenemos prácticamente ejemplos, tan solo dos. Algo parecido ocurre con la categoría 6 dentro de la variable $X13$ y con la categoría 3 de la variable $X11$. Luego hay algunas variables como la $X3$ o $X12$ en las que la distribución entre clases sigue un orden ascendente o descendente. En el caso de la variable $X12$ tiene sentido porque es una variable que toma valores discretos pero sigue siendo numérica, con lo cual hay un orden entre estos valores. La variable $X3$ indica el tipo de dolor de pecho del paciente, así que su distribución ascendente puede indicar simplemente que el criterio a la hora de asignarle un valor a cada tipo fuera ir del menos común al más común. Adicionalmente hemos visualizado la varible objetivo para ver cómo se distribuían sus dos clases y afortunadamente vemos que ambas están más o menos balanceadas. Esto es importante, porque cuando tenemos un dataset en los que proporcionalmente hay muy pocos ejemplos de una de las clases tenemos que aplicar técnicas para solucionar este problema.

Una vez conocido mejor cómo se distribuyen nuestras variables por si solas, vamos a hacer el mismo proceso pero esta vez teniendo en cuenta el valor de la variable objetivo Y. Primero vamos a graficar de nuevo las distribuciones de las variables distinguiendo por su etiqueta.



Como vemos en la mayoría de casos hay una diferencia considerable entre las distribuciones que siguen las variables en función del valor que toma Y. Por ejemplo en las variables $X8$ y $X1$, ocurre que el grueso de cada una de las distribuciones está distanciado uno del otro. En la variable $X10$ lo que sucede es que para $Y = 1$ la distribución tiene una curtosis mucho mayor. En el caso de la variable $X4$ ambas distribuciones son muy parecidas, y en el de la variable $X5$ no son tan parecidas las distribuciones, pero también se solapan mucho. Si posteriormente tuviéramos que eliminar algunas de las variables, serían una de estas dos. Parece por tanto que la presión sanguínea en reposo y el nivel de colesterol sérico en sangre no da mucha información sobre la presencia de una afección cardíaca, mientras que la edad, el máximo de pulsaciones y la depresión del segmento ST sí que la dan.

Hacemos lo mismo con las variables discretas e interpretamos posteriormente los gráficos.



Por lo que podemos observar las variables $X6$,$X7$ y $X2$ presentan distribuciones similares en la presencia o no de afección cardíaca con respecto a las distintos valores que toman. Por tanto los valores que muestra el electrocardiograma en reposo y el nivel de azúcar en sangre en reposo no parece que nos vayan a servir para distinguir si el paciente padece o no una afección cardíaca. El sexo del paciente tampoco parece influir demasiado en el diagnóstico, sí es cierto que si es mujer parece que el riesgo de padecer la afección es algo menor, pero si es hombre las probabilidades de padecer o no la afección no varían demasiado. Por otro lado las variables $X3, X12$ y $X13$ sí parecen presentar diferencias notables en la distribución de Y en función del valor que toman. Estas variables indicaban el tipo de dolor de pecho, si se padecía talasemia y el número de vasos mayores coloreados por la fluoroscopia. Más tarde comprobaremos todo esto con medidas más fiables que nuestra primera impresión al inspeccionar el gráfico.

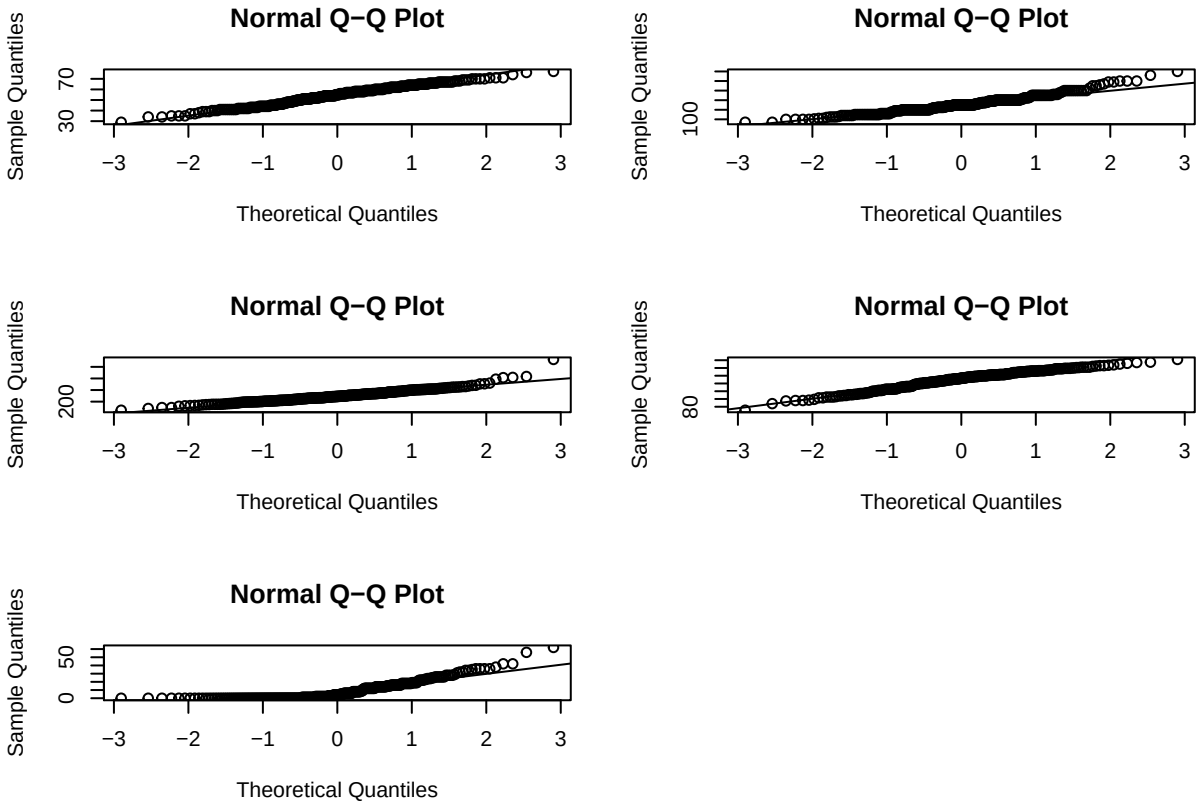
Una vez observado los gráficos, vamos a proceder a hacer algunos tests y construir algunas tablas cruzadas sobre las variables para comprobar con medidas estadísticas lo que los gráficos parecen decirnos.

En primer lugar procedemos con las variables reales, dijimos que las distribuciones no parecían seguir una normal y vamos a comprobarlo mostrando sus qqplots y los resultados del test Shapiro.

```
par(mfrow=c(3,2))
shapiro<-c()
qqnorm(heart$X1)
qqline(heart$X1)
shapiro<-c(shapiro,shapiro.test(heart$X1)$p.value)
qqnorm(heart$X4)
qqline(heart$X4)
shapiro<-c(shapiro,shapiro.test(heart$X4)$p.value)
qqnorm(heart$X5)
qqline(heart$X5)
shapiro<-c(shapiro,shapiro.test(heart$X5)$p.value)
```

```r
qqnorm(heart$X8)
qqline(heart$X8)
shapiro<-c(shapiro,shapiro.test(heart$X8)$p.value)
qqnorm(heart$X10)
qqline(heart$X10)
shapiro<-c(shapiro,shapiro.test(heart$X10)$p.value)
P.value<-shapiro
tabla<-data.frame(Varible=c("X1","X4","X5","X8","X10"),P.value)
knitr::kable(tabla)
```

Varible	P.value
X1	0.0276523
X4	0.0000037
X5	0.0000000
X8	0.0001450
X10	0.0000000



Como vemos es los qqplots las variables no se adaptan a la forma que debía tener una distribución normal. Los resultados de los test lo confirman, con p-valores tan bajos hay evidencia suficiente como para rechazar la hipótesis nula que es que se distribuyen según una normal. Por tanto rechazamos esta idea.

A continuación vamos a hacer un ggpairs para ver si existe correlación entre las variables reales o no.

```r
ggpairs(heart[,c(1,4,5,8,10)],aes(color=heart$Y))
```

Como vemos las variables reales entre sí no están nada correladas, siendo que el mayor coeficiente de correlación de Pearson en valor absoluto no llega a 0.3. Esto por una parte es malo porque además en los scatterplots generados vemos que ninguna combinación generada llega a separar bien los casos de pacientes con enfermedad cardíaca y sin ella. Por otra parte quiere decir que las variables que tenemos nos dan información distinta y no tenemos variables redundantes, lo cual es bueno.

Con las variables categóricas vamos a hacer primero unas tablas para ver en qué proporción se distribuyen las variables de nuestros datos.

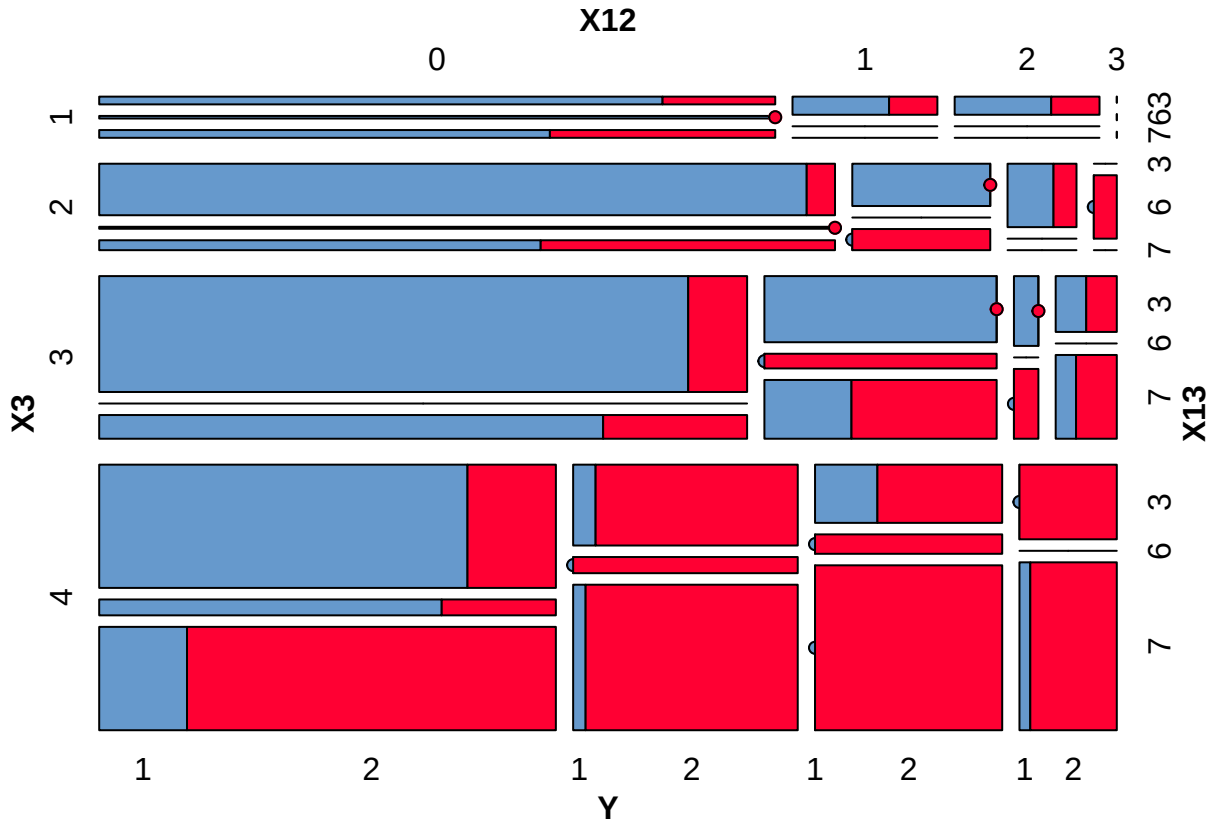
Nuestro siguiente paso es mostrar una tabla con los resultados de hacer un test chi cuadrado con las variables categóricas y nuestra variable objetivo para saber si podemos descartar que alguna de ellas esté correlada. Además mostraremos la V de Cramer que es una medida que nos dirá cuánto están de correladas las varialbes. Es el análogo del coeficiente de Pearson para variables categóricas.

Variables	cramer	Pvalue
X2	0.2977208	0.0000010
X3	0.5040143	0.0000000
X6	0.0163188	0.7885871
X7	0.1823656	0.0112237
X9	0.4193027	0.0000000
X11	0.3866781	0.0000000
X12	0.4825207	0.0000000
X13	0.5255309	0.0000000

Como podemos observar la variable $X6$ relacionada con el nivel de azúcar en sangre en ayunas no parece estar relacionada con la presencia de una enfermedad cardíaca en el paciente. El resto sin embargo presentan

p-valores muy bajos, lo cual es indicativo de que deberíamos rechazar la hipótesis de que no estén relacionadas las variables. La variable $X7$ no tiene un p valor tan alto, sin embargo en función de dónde quisiéramos poner el nivel de confianza podríamos también descartarla. Con respecto a los valores de la V de Cramer, las variables que parecen tener más información son $X13$,$X3$ y $X12$, que indicaban si el paciente padecía talasemia, el tipo de dolor de pecho y el número de vasos mayores coloreados por la fluoroscopia. Todo esto verifica las impresiones que teníamos cuando observábamos los primeros gráficos.

Para finalizar nuestro análisis exploratorio de dataset heart, vamos a mostrar un mosaico que compare las tres variables que más información parecen dar, con la variable Y



Como podemos observar, cuando la variable $X13$ toma el valor 6 o 7 (lo cual quiere decir que se padece talasemia) y la variable $X12$ toma el valor 4, si el número de vasos mayores coloreados por la fluoroscopia es mayor que 0, el riesgo de padecer una enfermedad cardíaca es muy grande. Por otro lado, si no ha habido vasos mayores coloreados ($X12 = 0$), y el tipo de dolor de pecho ($X3$) no es 4, entonces el riesgo de padecer una enfermedad cardíaca considerablemente bajo, aunque siempre que se padezca talasemia reversible ($X13 = 7$) el riesgo aumenta un poco. Lo que podemos sacar en claro es que conociendo el el valor que toman estas tres variables en un paciente, podemos dar un diagnóstico que es relativamente bueno teniendo en cuenta el número de variables. Esto sucede porque si observamos el mosaico, la mayoría de clases cruzadas están muy desbalanceadas en favor de una de las etiquetas de Y.

EDA - Regresión (friedman)

Para el problema de regresión tenemos el dataset friedman. Por lo que podemos encontrar este dataset recibe el nombre de la persona que lo propuso y se usa como 'benchmark' para comparar métodos o simplemente para comprobar si éstos funcionan adecuadamente. El dataset tiene 5 variables reales que tienen el nombre Input1,. . . ,Input5 a las que llamaremos $X1, ..., X5$ respectivamente y la variable objetivo Output a la que llamaremos Y, siendo consistentes con la notación que hemos utilizado en el anterior caso.

```
friedman <- read.csv("friedman.dat", comment.char="@", header=FALSE)
n <-length(names(friedman)) -1
names(friedman)[1:n] <-paste ("X", 1:n, sep="")
names(friedman)[n+1] <-"Y"
```

```
dim(friedman)
```

```
## [1] 1200    6
```

```
friedman[!complete.cases(friedman),]
```

```
## [1] X1 X2 X3 X4 X5 Y
## <0 rows> (or 0-length row.names)
```

Como vemos el dataset tiene 6 variables (5 variables con las que trabajamos y la variable objetivo) y consta de 1200 casos. Además no tiene ningún missing value.

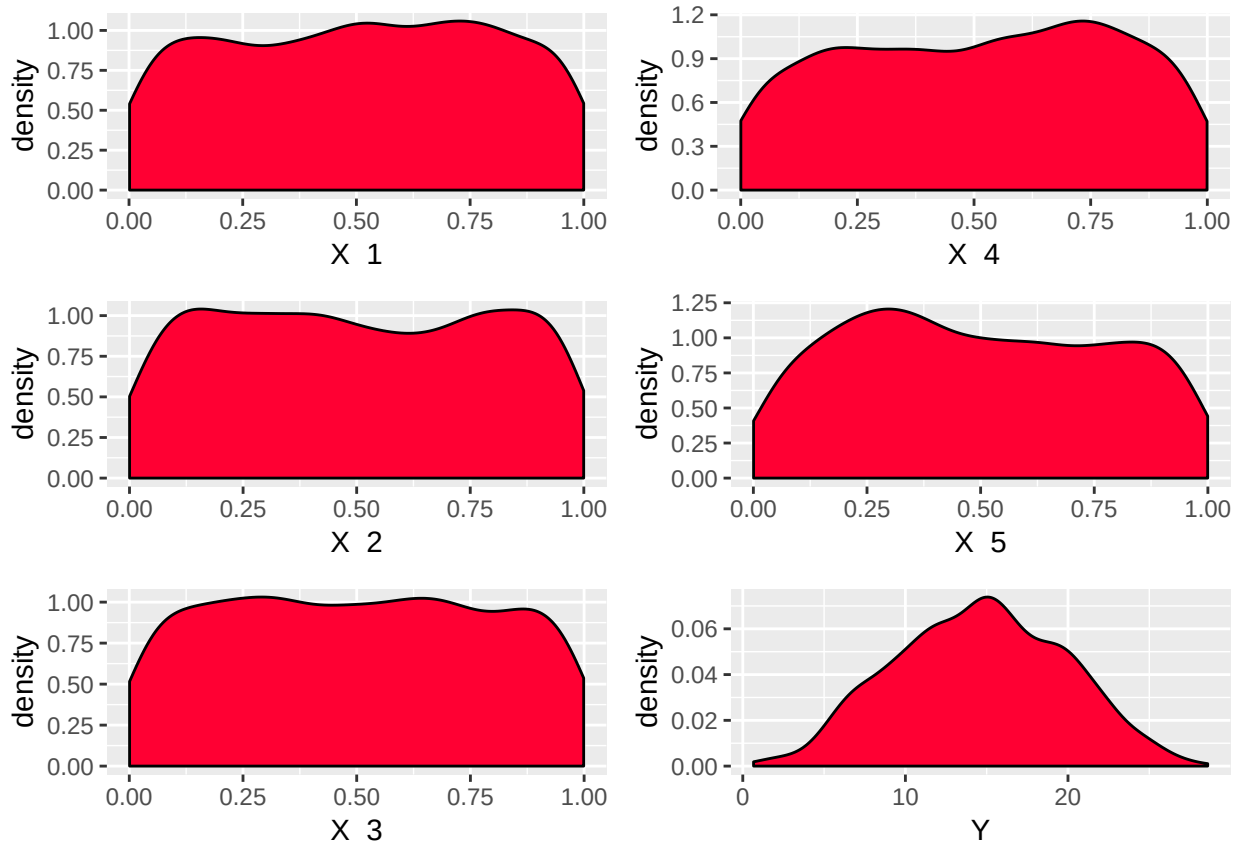
A continuación vamos a ver el summary del dataset para ver entre otras cosas la media de cada variable así como sus cuartiles.

```
summary(friedman)
```

```
##        X1                X2                X3
##  Min.   :0.001212   Min.   :0.0001603   Min.   :0.0006546
##  1st Qu.:0.249184   1st Qu.:0.2423287   1st Qu.:0.2485096
##  Median :0.519293   Median :0.4932687   Median :0.4993111
##  Mean   :0.506193   Mean   :0.4999592   Mean   :0.4995141
##  3rd Qu.:0.751131   3rd Qu.:0.7655960   3rd Qu.:0.7441912
##  Max.   :0.999719   Max.   :0.9996775   Max.   :0.9990619
##        X4                X5                Y
##  Min.   :0.0002123   Min.   :0.0004299   Min.   : 0.664
##  1st Qu.:0.2703118   1st Qu.:0.2578755   1st Qu.:10.859
##  Median :0.5328840   Median :0.4753492   Median :14.654
##  Mean   :0.5122272   Mean   :0.4928214   Mean   :14.567
##  3rd Qu.:0.7566648   3rd Qu.:0.7385440   3rd Qu.:18.494
##  Max.   :0.9994802   Max.   :0.9995394   Max.   :28.590
```

Como vemos todas las varibles de entrada se mueven en el intervalo $[0, 1]$ y además parece que siguen una distribución uniforme ya que media y mediana prácticamente coinciden (lo cual indica que tienen distribuciones simétricas), pero además los cuartiles coinciden prácticamente con la proporción de la población que dejan atrás. Por otro lado la variable Y sí parece tener un comportamiento simétrico respecto de la media, ya que media y mediana se acercan mucho, sin embargo la distribución que sigue en este caso no está tan clara.

Veamos a continuación los gráficos de las funciones de densidad que tienen nuestros datos y comprobemos si podrían confirmar o descartar por completo nuestras sospechas inferidas puramente del summary de los datos.



Como vemos en las gráficasm parece que efectivamente las variables de entrada podrían seguir una distribución uniforme en el intervalo $[0, 1]$. Por otro lado la variable de salida Y parece que sigue una distribución simétrica como decíamos, pero no una uniforme sino más bien una normal o una t de Student.

Vamos a ver a continuación la desviación típica de las variables.

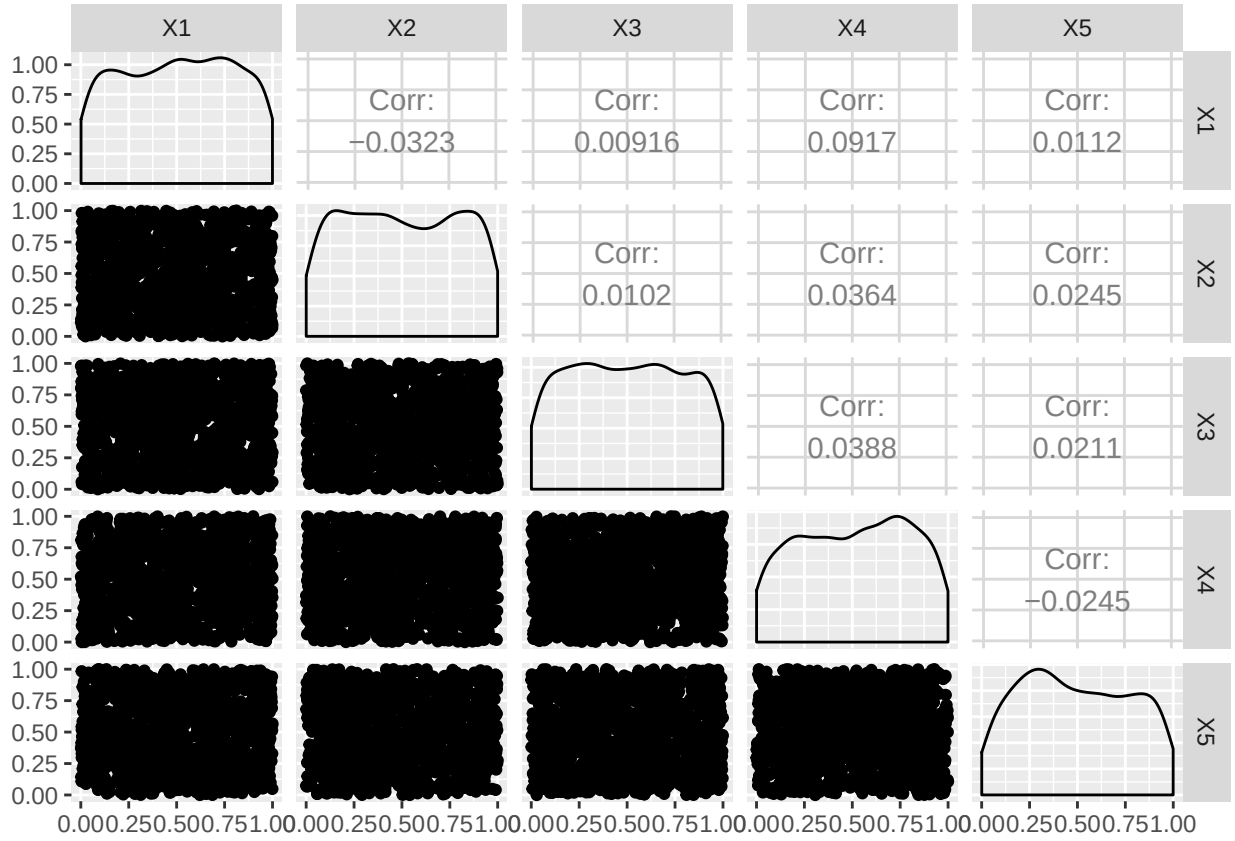
```
standard_dev<-c()
standard_dev<-c(standard_dev,sd(friedman$X1))
standard_dev<-c(standard_dev,sd(friedman$X2))
standard_dev<-c(standard_dev,sd(friedman$X3))
standard_dev<-c(standard_dev,sd(friedman$X4))
standard_dev<-c(standard_dev,sd(friedman$X5))
standard_dev<-c(standard_dev,sd(friedman$Y))
desviaciones<-data.frame(names(friedman),standard_dev)
colnames(desviaciones)<-c("Variable","Desviación Estándar")
knitr::kable(desviaciones)
```

Variable	Desviación Estándar
X1	0.2910749
X2	0.2938897
X3	0.2900604
X4	0.2842922
X5	0.2795909
Y	5.1844877

Todas ellas tienen una desviación típica entorno al 0.28-0.29, mientras que la variable objetivo Y tiene una

desviación típica de 5.184.

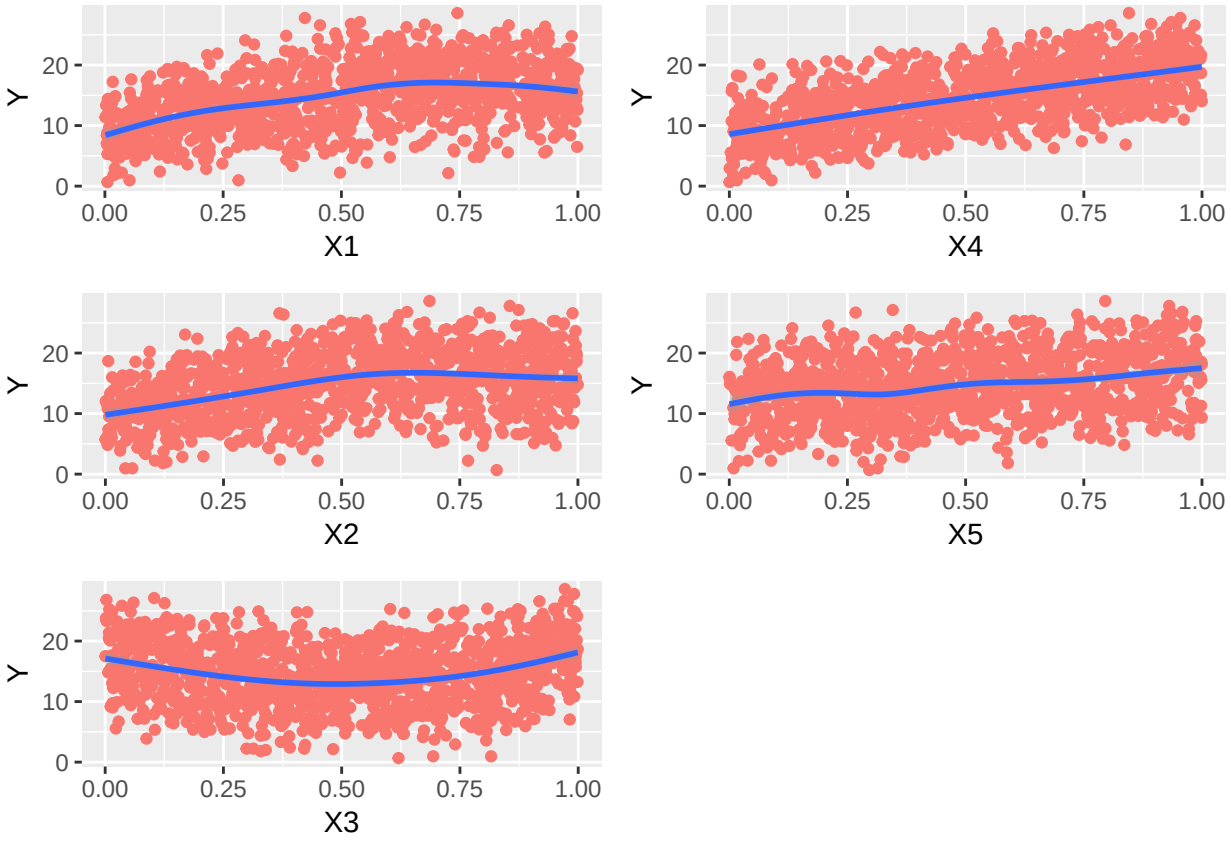
```
ggpairs(data=friedman[,-6])
```



A primera vista parece que el "ggpairs" no nos da ninguna información, pero el hecho es que nos dice que las variables $X1, ..., X5$ no guardan relación entre sí. Basta ver que en los gráficos que enfrentan las variables, éstas parecen estar distribuidas uniformemente sobre la cuadrícula, lo cual indicaría que son independientes. Le medida que puede cuantificar lo que vemos en la gráfica son los coeficientes de Pearson que efectivamente están siempre muy próximos a 0.

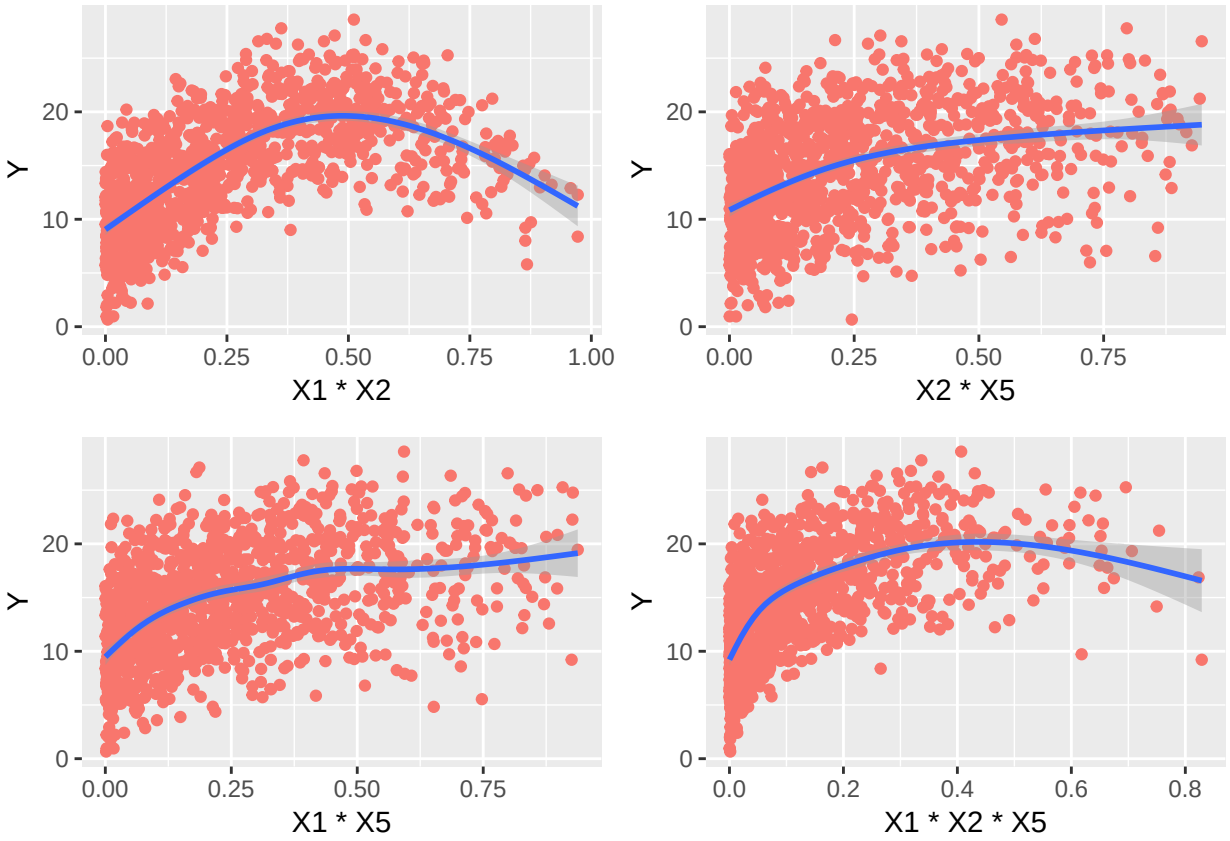
A continuación lo que queremos ver es qué relación puede haber entre las variables $X1, ..., X5$ y la variable objetivo, para ello vamos a hacer un scatter plot de los datos y acompañarlos de una curva generada con un modelo aditivo generalizado.

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



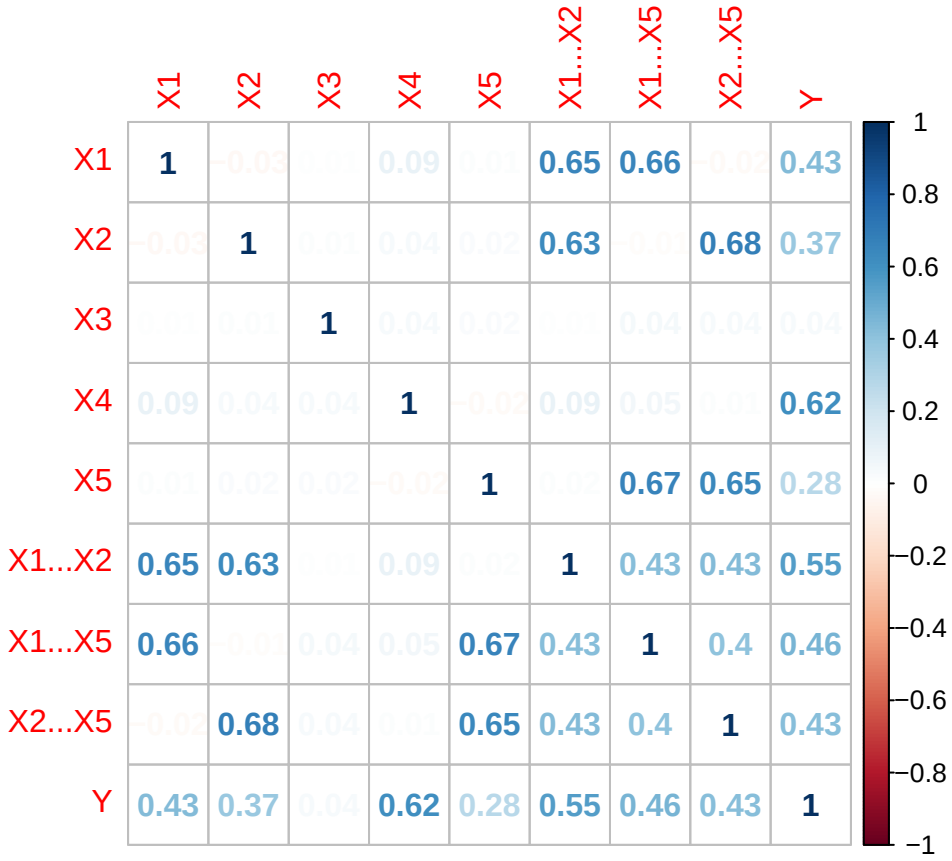
La idea de usar la curva generada por un GAM es sencillamente el hecho de que la nube de puntos es tal que adivinar las tendencias que pueden tener las variables con respecto a la variable objetivo es muy difícil. Como podemos observar, con respecto a $X4$ podemos adivinar un crecimiento lineal bastante claro. De la variable $X5$ se puede decir lo mismo salvo porque parece que existen algunas pequeñas oscilaciones pese a que el comportamiento general sea de crecimiento. En la variable $X3$ podría haber una relación cuadrática, ya que se puede adivinar una parábola con una curvatura muy leve. Con respecto a las variables $X1$ y $X2$ parece que la relación que existe es algo más que lineal, ya que no hay una tendencia general de crecimiento o decrecimiento en la gráfica. Podría ser una relación cuadrática, pero no es tan clara como en el anterior caso.

Ahora vamos a probar cómo se realaciona la variable Y con algunas interacciones entre las variables $X1, X2$ y $X5$ ya que son las variables que más dudas nos generan. Quizás alguna de estas interacciones presentan comportamientos muy evidentes que luego podamos utilizar cuando lleguemos a la parte de regresión.



En principio no parecen darnos mucha información salvo quizás el producto $X1 * X2$, que podría tener con Y una relación cuadrática o sinusoidal. Quizás $X1 * X5$ y $X2 * X5$ podrían tener una relación logarítmica con Y por el tipo de curva generada. Cuando lleguemos a la parte de regresión será el momento de comprobar si lo que comentamos tiene algún sentido o no.

Antes terminar el análisis exploratorio de los datos vamos a comprobar si éstas variables nuevas que hemos enfrentado gráficamente con Y tienen en efecto relación con la variable Y al menos midiendo esta relación con el coeficiente de Pearson.



En el plot de correlación lo que podemos observar es que la variable $X4$ es la que parece influenciar más a la variable Y y que de las variables que hemos construido la que más información da parece ser la variable $X1 * X2$. Adicionalmente por los datos obtenidos, si tuviéramos que deshacernos de alguna de las variables probablemente nos desharíamos de la variable $X3$ y de la variable $X5$ ya que son las que menos se relacionan con Y. Por último como es evidente las variables $X1, X2$ y $X5$ tienen coeficientes de correlación más o menos altos con sus variables aisladas, lo cual es lógico.

Antes de terminar nuestro análisis exploratorio de los datos del dataset friedman, nos gustaría mostrar los resultados de aplicar el test de Kolmogorov-Smirnov a las variables $X1, ..., X5$ para comprobar si su distribución es uniforme como sospechábamos cuando graficamos las funciones de densidad de la muestra. Este test compara muestras del mismo tamaño de dos distribuciones y tiene como hipótesis nula que ambas muestras han sido extraídas de una variable aleatoria con la misma distribución.

```
Pvalue<-c()
set.seed(123456)
muestra<-runif(1200,0,1)
Pvalue<-c(Pvalue,ks.test(friedman$X1, muestra)$p.value)
Pvalue<-c(Pvalue,ks.test(friedman$X2, muestra)$p.value)
Pvalue<-c(Pvalue,ks.test(friedman$X3, muestra)$p.value)
Pvalue<-c(Pvalue,ks.test(friedman$X4, muestra)$p.value)
Pvalue<-c(Pvalue,ks.test(friedman$X5, muestra)$p.value)
kolmogorov<-data.frame(Variable=c("X1","X2","X3","X4","X5"),Pvalue)
knitr::kable(kolmogorov)
```

Variable	Pvalue
X1	0.9408423
X2	0.8995694

Variable	Pvalue
X3	0.9876983
X4	0.6527114
X5	0.3412249

Como podemos observar, los p valores son considerablemente altos. Lo suficiente como para no descartar la hipótesis nula que en este caso es que las variables $X1, ..., X5$ siguen una distribución uniforme en el intervalo $[0, 1]$. Por supuesto la semilla que hemos usado para obtener siempre los mismos resultados influye en el test. Podemos pues plantearnos repetir el test con otras semillas, observaremos que prácticamente siempre los resultados nos invitan a no rechazar la hipótesis nula. Por tanto podemos asumir que las variables siguen esta distribución.

Clasificación

Nuestro conjunto a estudiar en el caso de clasificación es el dataset "heart", que consta de 13 variables $X1, ..., X13$ (8 de ellas que hemos considerado categóricas y 5 continuas) de entrada y una variable Y de salida que toma los valores 1 y 2 en función de si un paciente no padece o padece una enfermedad cardíaca respectivamente. En la parte de EDA podemos encontrar una explicación más detallada del significado de las varibles y de por qué hemos las hemos considerado continuas o discretas. En esta sección nos concentraremos más en los métodos de clasificación para no repetir muchas de las cosas de las que ya hemos hablado.

Algoritmo K-nn

Vamos a empezar aplicando el algoritmo k-nn a nuestro dataset, tratando primero de buscar un k óptimo en condiciones generales y luego buscando modificaciones al modelo inicial que puedieran mejorar el accuracy.

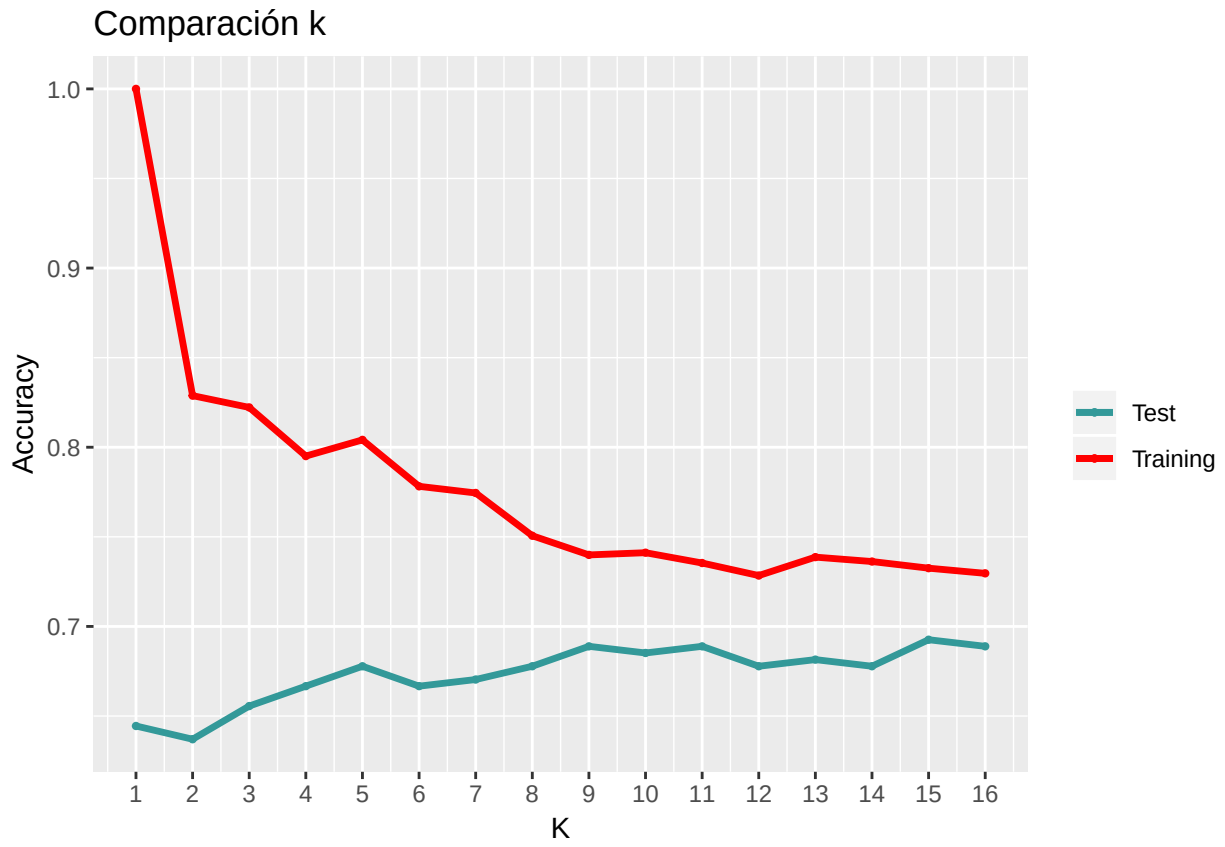
Primero nos vamos a preocupar por buscar un k óptimo utilizando una 10-fold cross validation. Es decir partimos el conjunto en 10 y cada vez vamos a seleccionar una de esos 10 subconjuntos como conjunto test y el resto como training. Vamos a hacer la media de aciertos en cada uno de los 10 casos y vamos a comparar estas medias.

```
nombre<-'heart'
run_knn_fold_k_clasif<-function(i, x, tt= "test",kchoice) {
file <-paste(x, "-10-", i, "tra.dat", sep="")
x_tra<-read.csv(file, comment.char="@",header=FALSE)
file <-paste(x, "-10-", i, "tst.dat", sep="")
x_tst<-read.csv(file, comment.char="@",header=FALSE)
In <-length(names(x_tra)) -1
names(x_tra)[In+1] <-"Y"
names(x_tst)[In+1] <-"Y"
if (tt== "train") {
test <-x_tra
}
else {
test <-x_tst
}
x_tra$Y<-as.factor(x_tra$Y)
test$Y<-as.factor(test$Y)
yprime=knn(x_tra[,1:In], k=kchoice, test[,1:In], x_tra$Y)
sum(yprime==test$Y)/length(yprime)
}

k_test_clasif<-function(k){
```

```
  mean(sapply(1:10, run_knn_fold_k_clasif, nombre, "test", k))
}
k_tra_clasif<-function(k){
  mean(sapply(1:10, run_knn_fold_k_clasif, nombre, "train", k))
}
```

Ahora utilizamos esta última función definida junto con la función "sapply" para poder vectorizar el cálculo de cada promedio de acierto para cada k.



Parece que la elección de k que nos da más acierto es $k = 14$. En lo sucesivo compararemos algunas modificaciones del modelo original para algunos valores de k y $k = 14$ será uno de ellos. La siguiente tabla nos resume de forma menos visual pero más precisa la información que acabamos de graficar.

K	accuracy_tra	accuracy_tst
1	1.0000000	0.6444444
2	0.8288066	0.6370370
3	0.8222222	0.6555556
4	0.7950617	0.6666667
5	0.8041152	0.6777778
6	0.7781893	0.6666667
7	0.7744856	0.6703704
8	0.7506173	0.6777778
9	0.7399177	0.6888889
10	0.7411523	0.6851852
11	0.7353909	0.6888889
12	0.7283951	0.6777778

K	accuracy_tra	accuracy_tst
13	0.7386831	0.6814815
14	0.7362140	0.6777778
15	0.7325103	0.6925926
16	0.7296296	0.6888889

Si no tuviéramos las particiones ya hechas también podríamos haber utilizado la librería "caret" que nos permite hacer una 10-fold cross validation de forma automática. En el siguiente chunk de código contrastamos los resultados obtenidos con los de las particiones generadas por caret.

```
set.seed(123)
control <- trainControl(method = "repeatedcv", repeats = 5)
knn_caret <- train(x = heart[,1:13], y = heart$Y, method = 'knn', trControl = control,
                   tuneGrid = data.frame(k=1:16))
knn_caret
```

```
## k-Nearest Neighbors
##
## 270 samples
##  13 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 243, 243, 243, 243, 243, 243, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.6444444  0.2802733
##    2  0.6540741  0.2965262
##    3  0.6748148  0.3354185
##    4  0.6651852  0.3143931
##    5  0.6881481  0.3584719
##    6  0.6777778  0.3378849
##    7  0.6829630  0.3460664
##    8  0.6814815  0.3436589
##    9  0.6866667  0.3548103
##   10  0.6822222  0.3470834
##   11  0.6822222  0.3476518
##   12  0.6733333  0.3282987
##   13  0.6940741  0.3723987
##   14  0.6829630  0.3472187
##   15  0.6955556  0.3740799
##   16  0.7029630  0.3867304
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 16.
```

En este caso nos da $k = 16$, pero en general vemos que para valores altos de k se obtienen mejores resultados, cosa que ocurría también en nuestra partición original.

Ahora vamos a hacer un par de modificaciones al método y a comparar resultados. Las modificaciones van a ser las siguientes:

1. Primero el modelo original usando todas las variables.

2. Posteriormente eliminamos las variables que menos información nos daban según el análisis exploratorio de datos inicial (en este caso $X1, X6, X7$).
3. Hacemos un modelo utilizando únicamente las varibles que mejor información nos daban según el análisis exploratorio de datos (en este caso $X3, X5, X10, X12, X13$).
4. El mismo modelo que el anterior pero reescalando las variables numéricas.

Modelo	Accuracy.k5	Accuracy.k14	Accuracy.k16
Original	0.6777778	0.6777778	0.6888889
Eliminando	0.6444444	0.6629630	0.6703704
Mejores	0.6925926	0.7111111	0.6777778
Mejores escalado	0.8111111	0.7851852	0.8037037

Los resultados mostrados en la tabla son optimistas con respecto a la precisión inicial obtenida. Ya que vemos que eliminando algunas varibles no perdemos demasiada información, pero cuando seguimos eliminando y nos quedamos con las variables que más nos aportan descubrimos que incluso podemos mejorar la puntuación inicial (en el caso $k = 5$). Además con estas pocas variables utilizando simplemente un reescalado mejoramos considerablemente la precisión en todos los casos. Si siguiéramos probando algunas otras combinaciones o interacciones con herramientas tan sencillas como un reescalado parece que se pueden conseguir resultados bastante aceptables (al menos teniendo en cuenta desde dónde partíamos).

Con respecto a la elección de k parece que aunque $k = 16$ no daba malos resultados inicialmente, cuando hacemos modificaciones al modelo original, esta elección de k no responde tan bien a esos cambios.

Algoritmo LDA

Siguiendo nuestro estudio de los métodos de clasificación con nuestro dataset heart, vamos a utilizar ahora el método LDA (Linear Discriminant Analysis). Este método pretende partir el espacio en regiones delimitadas por hiperplanos de manera que dependiendo de la región en la que caigan nuestros inputs, se dictamine si el paciente padece o no una enfermedad cardíaca.

Este método funciona mejor cuando las variables siguen una distribución normal, lo cual no es nuestro caso, por tanto los resultados en un principio no deberían ser demasiado buenos.

El código utilizado para hacer la 10-fold cross validation es el siguiente.

```
n<-length(names(heart))-1
run_lda_fold<-function(i,formula,x, tt= "test") {
file <-paste(x, "-10-", i, "tra.dat", sep="")
x_tra<-read.csv(file, comment.char="@",header=FALSE)
file <-paste(x, "-10-", i, "tst.dat", sep="")
x_tst<-read.csv(file, comment.char="@",header=FALSE)
names(x_tra)[1:n] <-paste ("X", 1:n, sep="")
names(x_tra)[n+1] <-"Y"
names(x_tst)[1:n] <-paste ("X", 1:n, sep="")
names(x_tst)[n+1] <-"Y"
if (tt== "train") {
test <-x_tra
}
else {
test <-x_tst
}
x_tra$Y <- as.factor(x_tra$Y)
test$Y <- as.factor(test$Y)
modelo=lda(formula,data=x_tra)
yprime=predict(modelo,test)
```

```
sum(yprime$class==test$Y)/length(yprime$class)
}
```

En este caso tenemos la ventaja de que podemos utilizar una única función con el parámetro fórmula para indicar qué variable vamos a utilizar. Las variables que vamos a utilizar son:

1. Modelo con todas las variables.
2. Modelo eliminando las variables $X1, X6, X7$.
3. Modelo utilizando solo las variables $X3, X5, X10, X12, X13$.
4. Modelo 1 con escalado.
5. Modelo 3 con escalado.

Los resultados han sido los siguientes:

Modelo	Accuracy_train	Accuracy_test
1	0.8576132	0.8481481
2	0.8559671	0.8444444
3	0.8279835	0.8333333
4	0.8576132	0.8407407
5	0.8279835	0.8259259

Los mejores resultados los ha dado el modelo original que usaba todas las variables y a medida que hemos eliminando variables hemos perdido algo de precisión. Sin embargo como podemos observar, el modelo 3 que utiliza únicamente 5 variables da resultados decentes frente al primer modelo de todos que utiliza bastantes más variables (concretamente 13). Para este problema no es importante reducir el número de variables ya que el dataset es pequeño (de $270x13$), pero en otras ocasiones puede compensar reducir el número de variables y perder algo de precisión para reducir considerablemente el coste computacional. Otra cosa que podemos apreciar que el escalado ha empeorado los resultados, hay que tenerlo en cuenta. En K-nn escalar los datos es crucial si queremos dar la misma importancia a todos las variables, en LDA lo que puede haber pasado es que al reescalar hemos "juntado" los datos espacialmente. Como LDA lo que precisamente hace es dividir el espacio en regiones, este tipo de estrategias pueden conseguir mezclar los datos y empeorar los resultados lejos de mejorarlos.

Para finalizar decir que podríamos haber añadido variables al cuadrado o interacciones entre ellas, pero eso es lo que hace el siguiente método por lo que lo hemos obviado. Sí que hemos probado a usar logaritmos y exponenciales en algunas variables para ver qué ocurría, pero no ha dado resultado. Si una de las variables al aplicarle una de estas transfomraciones se acerca más a una distribución normal, entonces quizás sustituir esta variable por su transformada nos de mejores resultados en LDA.

Algoritmo QDA

Vamos a terminar nuestro estudio de los métodos de clasificación con el algortimo QDA (Quadratic Discriminant Analysis). Este algoritmo sigue la filosofía del anterior de dividir el espacio en regiones. En este caso las fronteras de estas regiones no son hiperplanos, sino lo que serían hipercuádricas. Esto quiere decir que los discriminantes que determinan si un punto cae en una región un otra son polinomios de segundo grado en las variables que queramos utilizar.

El código utilizado para la 10-fold cross validation es el siguiente.

```
run_qda_fold<-function(i,formula,x, tt= "test") {
file <-paste(x, "-10-", i, "tra.dat", sep="")
x_tra<-read.csv(file, comment.char="@",header=FALSE)
file <-paste(x, "-10-", i, "tst.dat", sep="")
x_tst<-read.csv(file, comment.char="@",header=FALSE)
names(x_tra)[1:n] <-paste ("X", 1:n, sep="")
```

```
names(x_tra)[n+1] <-"Y"
names(x_tst)[1:n] <-paste ("X", 1:n, sep="")
names(x_tst)[n+1] <-"Y"
if (tt== "train") {
test <-x_tra
}
else {
test <-x_tst
}
x_tra$Y <- as.factor(x_tra$Y)
test$Y <- as.factor(test$Y)
modelo=qda(formula,data=x_tra)
yprime=predict(modelo,test)
sum(yprime$class==test$Y)/length(yprime$class)
}
```

Los modelos utilizados han sido los siguientes:

1. Utilizando todas las variables.
2. Eliminando $X1, X6, X7$.
3. Utilizando $X3, X5, X10, X12, X13$.
4. Utilizando todas las variables y $X5^2, X10^2$.
5. Añadiendo al modelo 4 la interacción $X5 * X10$.

Los resultados son los siguientes:

Modelo	Accuracy_train
1	0.8777778
2	0.8650206
3	0.8230453
4	0.8777778
5	0.8769547

Los resultados son interesantes. Lo que podemos ver es que si bien cuando hemos eliminado algunas variables en el mo

Comparación de Algoritmos

Para finalizar con la parte de clasificación vamos a hacer la comparación de algoritmos con los ficheros de prado "clasif_test_alumnos.csv" y "clasif_train_alumnos.csv". Nosotros vamos a comparar cómo funcionan los algoritmos con los conjuntos de test y de training. El rendimiento sobre los conjuntos test es el más interesante, sin embargo también puede darnos información comparar los resultados con los conjuntos de training para ver si hay overfitting.

Primero leemos la tabla de resultados en los conjuntos test y training que hemos descargado de prado y mostramos los resultados en test y training respectivamente.

```
resultados <- read.csv("clasif_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]
resultados <- read.csv("clasif_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]
knitr:: kable(tablatst)
```

	out_test_knn	out_test_lda	out_test_qda
appendicitis	0.8966667	0.8690909	0.8109091
australian	0.6838235	0.8579710	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	0.7481720	0.7512903
hayes-roth	0.5666667	0.5500000	0.5875000
heart	0.6692308	0.8481481	0.8296296
iris	0.9642857	0.9800000	0.9733333
led7digit	0.7510204	0.7420000	0.6975000
mammographic	0.7977698	0.8241269	0.8194042
monk-2	0.9743632	0.7703433	0.9235535
newthyroid	0.9071429	0.9164502	0.9629870
pima	0.7348861	0.7709930	0.7412403
tae	0.3838095	0.5245833	0.5425000
titanic	0.7850353	0.7760304	0.7733032
vehicle	0.6291452	0.7813305	0.8522409
vowel	0.6428571	0.6030303	0.9191919
wine	0.6959559	0.9944444	0.9888889
wisconsin	0.9735023	0.9592185	0.9519476

```
knitr::kable(tablatra)
```

	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	0.5314180
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	0.9800000	0.9814815
led7digit	0.7636971	0.7635556	0.7680556
mammographic	0.8160856	0.8274465	0.8196843
monk-2	0.9793684	0.7821826	0.9303010
newthyroid	0.9158409	0.9183457	0.9700283
pima	0.7791914	0.7792266	0.7633125
tae	0.5263460	0.5584858	0.5688072
titanic	0.7892319	0.7760111	0.7732851
vehicle	0.7213300	0.7989229	0.9123989
vowel	0.8378652	0.6457912	0.9701459
wine	0.7745126	1.0000000	0.9956250
wisconsin	0.9739304	0.9614471	0.9588436

Comparamos los algoritmos utilizando wilcoxon. Primero los comparamos dos a dos utilizando wilcoxon. Vamos a observar los p valores de los tests para comprobar si existen diferencias signficativas.

```
P.value<-c()
LDAvsQDAtst<-wilcox.test(tablatst[,2], tablatst[,3], alternative = "two.sided", paired=TRUE)
```

```
P.value<-c(P.value,LDAvsQDAtst$p.value)
LDAvsQDAtra<-wilcox.test(tablatra[,2], tablatra[,3], alternative = "two.sided", paired=TRUE)
P.value<-c(P.value,LDAvsQDAtra$p.value)
LDAvsKNNtst<-wilcox.test(tablatst[,2], tablatst[,1], alternative = "two.sided", paired=TRUE)
P.value<-c(P.value,LDAvsKNNtst$p.value)
LDAvsKNNtra<-wilcox.test(tablatra[,2], tablatra[,1], alternative = "two.sided", paired=TRUE)
P.value<-c(P.value,LDAvsKNNtra$p.value)
QDAvsKNNtst<-wilcox.test(tablatst[,3], tablatst[,1], alternative = "two.sided", paired=TRUE)
P.value<-c(P.value,QDAvsKNNtst$p.value)
QDAvsKNNtra<-wilcox.test(tablatra[,3], tablatra[,1], alternative = "two.sided", paired=TRUE)
P.value<-c(P.value,QDAvsKNNtra$p.value)
Algoritmos<-c("LDAvsQDA.test","LDAvsQDA.tra","LDAvsKNN.test","LDAvsKNN.tra",
              "QDAvsKNN.test","QDAvsKNN.tra")
tabla<-data.frame(Algoritmos,P.value)
knitr::kable(tabla)
```

Algoritmos	P.value
LDAvsQDA.test	0.8408222
LDAvsQDA.tra	0.1768532
LDAvsKNN.test	0.5958195
LDAvsKNN.tra	0.6476555
QDAvsKNN.test	0.1768532
QDAvsKNN.tra	0.2942524

Los resultados de los test dan p valores suficientemente altos como para rechazar la hipótesis de que los algoritmos tienen el mismo rendimiento. Donde parece que hay más discrepancia es en el caso de LDAvsQDA y QDAvsKNN sobre los conjuntos test donde según el p valor, la probabilidad de que los algoritmos ofrezcan un rendimiento significativamente distinto es de algo más de 0.8. Pero sigue siendo un valor bajo. Resulta interesante también comprobar que sobre los conjuntos de training no hay diferencias importantes según estos resultados, por lo que parece que en principio ningún modelo sobreajusta. Al menos no más que el resto.

Ahora aplicamos el test de friedman para ver si hay diferencias entre los tres algoritmos en conjunto.

```
friedman.test(as.matrix(tablatst))
```

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatst)
## Friedman chi-squared = 0.7, df = 2, p-value = 0.7047
```

```
friedman.test(as.matrix(tablatra))
```

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatra)
## Friedman chi-squared = 1.3, df = 2, p-value = 0.522
```

En vista de los resultados obtenidos con el test de Friedman, no se puede concluir que haya diferencias entre los distintos algoritmos, ni en training ni en test. Aún así vamos a hacer un estudio post-hoc Holm comparando nuevamente los métodos dos a dos.

```
tam_tst <- dim(tablatst)
groups_tst <- rep(1:tam_tst[2], each=tam_tst[1])
```

```
pairwise.wilcox.test(as.matrix(tablatst), groups_tst, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatst) and groups_tst
##
##   1    2
## 2 1.00 -
## 3 0.53 1.00
##
## P value adjustment method: holm
```

```
tam_tra <- dim(tablatra)
groups_tra <- rep(1:tam_tra[2], each=tam_tra[1])
pairwise.wilcox.test(as.matrix(tablatra), groups_tra, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatra) and groups_tra
##
##   1    2
## 2 0.65 -
## 3 0.59 0.53
##
## P value adjustment method: holm
```

El estudio post-hoc Holm sigue la misma línea que los test anteriores que comprobaban que no existían diferencias significativas entre los métodos. De hecho esta vez vemos con en algunos de los casos al comparar los resultados en test el p valor es 1, lo que nos indica que deberíamos aceptar la hipótesis de que los algoritmos tiene el mismo rendimiento.

Regresión

Ahora trabajaremos con el dataset friedman de la parte de regresión, aplicando algunos algoritmos vistos en clase e interpretando los resultados. Además al final haremos una comparación general de tres algoritmos de regresión utilizando los datos de los resultados que éstos han dado sobre distintos datasets.

Algoritmo LM

```
friedman <- read.csv("friedman.dat", comment.char="@", header=FALSE)
n <-length(names(friedman)) -1
names(friedman)[1:n] <-paste ("X", 1:n, sep="")
names(friedman)[n+1] <-"Y"
```

```
run_lm_fold<-function(i,formula,x, tt= "test") {
file <-paste(x, "-5-", i, "tra.dat", sep="")
x_tra<-read.csv(file, comment.char="@",header=FALSE)
file <-paste(x, "-5-", i, "tst.dat", sep="")
x_tst<-read.csv(file, comment.char="@",header=FALSE)
names(x_tra)[1:n] <-paste ("X", 1:n, sep="")
names(x_tra)[n+1] <-"Y"
names(x_tst)[1:n] <-paste ("X", 1:n, sep="")
```

```
names(x_tst)[n+1] <-"Y"
if (tt== "train") {
test <-x_tra
}
else {
test <-x_tst
}
fitMulti=lm(formula,x_tra)
yprime=predict(fitMulti,test)
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
```

```
model_x1<-lm(Y~X1,data=friedman)
model_x2<-lm(Y~X2,data=friedman)
model_x3<-lm(Y~X3,data=friedman)
model_x4<-lm(Y~X4,data=friedman)
model_x5<-lm(Y~X5,data=friedman)
radjusted<-c()
radjusted<-c(radjusted,summary(model_x1)$adj.r.squared)
radjusted<-c(radjusted,summary(model_x2)$adj.r.squared)
radjusted<-c(radjusted,summary(model_x3)$adj.r.squared)
radjusted<-c(radjusted,summary(model_x4)$adj.r.squared)
radjusted<-c(radjusted,summary(model_x5)$adj.r.squared)
pvalue<-rep("< 2e-16",5)
pvalue[3]<-"  0.218"
medidas<-data.frame(variable=c("X1","X2","X3","X4","X5"),pvalue,radjusted)
colnames(medidas)<-c("Variable","P-value","Adjusted R squared")
knitr::kable(medidas)
```

Variable	P-value	Adjusted R squared
X1	< 2e-16	0.1872342
X2	< 2e-16	0.1372045
X3	0.218	0.0004351
X4	< 2e-16	0.3786643
X5	< 2e-16	0.0752652

```
MSE_lm_test<-c()
MSE_lm_test<-c(MSE_lm_test,mean(sapply(1:5,run_lm_fold, Y~X1,"friedman","test")))
MSE_lm_test<-c(MSE_lm_test,mean(sapply(1:5,run_lm_fold, Y~X2,"friedman","test")))
MSE_lm_test<-c(MSE_lm_test,mean(sapply(1:5,run_lm_fold, Y~X3,"friedman","test")))
MSE_lm_test<-c(MSE_lm_test,mean(sapply(1:5,run_lm_fold, Y~X4,"friedman","test")))
MSE_lm_test<-c(MSE_lm_test,mean(sapply(1:5,run_lm_fold, Y~X5,"friedman","test")))
MSE_lm_test<-c(MSE_lm_test,mean(sapply(1:5,run_lm_fold, Y~.,"friedman","test")))
MSE_lm_train<-c()
MSE_lm_train<-c(MSE_lm_train,mean(sapply(1:5,run_lm_fold, Y~X1,"friedman","train")))
MSE_lm_train<-c(MSE_lm_train,mean(sapply(1:5,run_lm_fold, Y~X2,"friedman","train")))
MSE_lm_train<-c(MSE_lm_train,mean(sapply(1:5,run_lm_fold, Y~X3,"friedman","train")))
MSE_lm_train<-c(MSE_lm_train,mean(sapply(1:5,run_lm_fold, Y~X4,"friedman","train")))
MSE_lm_train<-c(MSE_lm_train,mean(sapply(1:5,run_lm_fold, Y~X5,"friedman","train")))
MSE_lm_train<-c(MSE_lm_train,mean(sapply(1:5,run_lm_fold, Y~.,"friedman","train")))
errores<-data.frame(c("X1","X2","X3","X4","X5","X1,...,X5"),
                    MSE_lm_train,MSE_lm_test)
```

```r
colnames(errores)<-c("Variables","MSE Train","MSE Test")
```

Como podemos ver utilizar las variables aisladas como regresores no da resultados demasiado buenos. Si observamos la tabla, el mayor valor del R cuadrado ajustado es el del modelo que usa como única variable regresora la variable $X4$ y el valor alcanzado no es muy bueno. Por otro lado podemos ver que la variable $X3$ no parece aportar demasiada información en un principio, ya que su R cuadrado ajustado es el menor de todas las variables con diferencia y además su p-value es muy alto con respetcto al resto. De hecho ese valor nos dice que en torno a un 0.8 de probabilidad de que la variable $X3$ no explique nada la variable Y. Si como se nos sugiere tuviéramos que elegir una sola de las variables para hacer el modelo, elegiríamos sin lugar a dudas la variable $X4$, y luego añadiríamos las variable $X1$ y $X2$. Otro procedimiento para montar el modelo sería considerar el modelo que utiliza todas las variables e ir descartando las que menos información nos aporten.

En la siguiente tabla mostramos los resultados que ofrecen los modelos anteriores en cuanto al error cuadrático medio en los conjuntos de training y de test utilizando las particiones que nos han proporcionado de antemano.

```r
knitr::kable(errores)
```

Variables	MSE Train	MSE Test
X1	21.796889	21.926374
X2	23.146665	23.203584
X3	26.805874	26.971073
X4	16.664110	16.753433
X5	24.797543	24.965771
X1,. . . ,X5	7.223837	7.306857

Como vemos en cuanto a los modelos de regresión simple vemos que efectivamente funcionan mucho mejor aquellos que utilizan variables con un R cuadrado ajustado más alto. Por otro lado observamos que en todos los modelos (el de regresión múltiple incluido) no hay demasiada discrepancia entre los errores hallados en training y en test. Esto quiere decir que los modelos son fiables en el sentido de que no hacen overfitting. Digamos que no son lo suficientemente complejos como para ajusatarse "demasiado bien" a los conjuntos de training. Como decíamos dada la complejidad de los modelos esto era de esperar, pero en adelante quizás alguno de los modelos que probemos es significativamente mejor para los conjuntos de training, y esto es un problema. Finalmente comentar que el modelo de regersión múltiple mejora significativamente todos los modelos de regresión simple como podemos observar, y que además tampoco pierde interpretabilidad porque sigue siendo un modelo linar, con lo cual los coeficientes no dejan de ser la relación de lo que se gana y pierde en la variable Y cuando se aumenta en una unidad la variable correspondiente. No podemos especificar mucho más sobre el problema ya que el dataset es una benchmark que no proviene de un problema real como comentamos en la parte de EDA.

Ahora pasaremos a probar los resultados de utilizar el método de regresión cuando utilizamos distintas variables como regresores. Las enumeraremos todas y posteriormente explicaremos cuál ha sido el razonamiento para ir pasando de una a otra.

1. Utilizamos como regresores todas las variables y todas sus posibles interacciones sobre ellas todas elevadas a su primera potencia.
2. Como la anterior estrategia no da p-valores buenos probamos con todas las variables y éstas al cuadrado.
3. Mantenemos el cuadrado de las 3 primeras variables que tienen buenos p valores y añadimos interacciones entre las 2 últimas que tienen p-valores malos y las que tienen buenos p-valores, sin mezclarlas.
4. Utilizamos todas las variables, el cuadrado de las 3 primeras y el seno de la última. Nuestro objetivo ahora es intentar ver si hay alguna transformación de la última variable que de más información.
5. Como utilizar el seno no ha dado buenos resultados, cambiamos el seno por la variable $X5$ elevada a su tercera, cuarta y quinta potencia.
6. El anterior modelo ha dado muy malos resultados así que usamos esa información para abandonar la

idea de hacer más transformaciones a esta variable. Ahora hacemos el mismo procedimiento con la cuarta variable. Este modelo es el análogo al modelo 4 pero con el seno de $X4$.

7. Este modelo es el análogo al modelo 5 con las potencias de $X4$.
8. Con los anteriores modelos hemos descartado utilizar modelos que utilicen las variables $X4$ y $X5$ de forma no lineal.
9. Utilizamos los cuadrados de las 3 primeras variables y las potencias tercera, cuarta y quinta de la variable $X1$, para ver si obtenemos más información.
10. Como no da resultados demasiado buenos probamos de nuevo con el cuadrado de las 3 primeras variables y el producto de $X1$ y $X2$.

Llegados aquí los avances han sido pocos, los R cuadrado justados rondan siempre el 0.9 sin llegar a alcanzarlo nunca. Cuando investigamos sobre el dataset encontramos que la variable Y del dataset friedman se sacó con la siguiente fórmula:

$$Y = 10\sin(\pi X1X2) + 20(X3 - 0.5)^2 + 10X4 + 5X5 + e,$$

donde e es un ruido gaussiano aleatorio que sigue una normal $N(0, 1)$. Así que nuestros siguientes modelos usan esta idea para ver cómo podríamos habernos acercado a los resultados que da un modelo que use directamente esta información sin conocerlo de antemano.

Cabe destacar que efectivamente las variables $X4$ y $X5$ no necesitan transformaciones para ser usadas en el modelo, con lo cual nuestro procedimiento para descartarlas aunque no fuera el mejor dio buenos resultados.

11. Como conocemos el desarrollo en serie de potencias del seno utilizamos el producto de las variables $X1$ y $X2$ en las potencias primera y tercera.
12. Hacemos lo mismo añadiendo la potencia quinta.
13. Probamos el modelo utilizando la información sobre cómo se ha calculado la variable Y.

A continuación dejamos un chunk del código de las fórmulas para facilitar la comprensión de lo que hemos tratado de explicar con palabras.

```
formula_1<-Y~.+X1*X2*X3*X4*X5
formula_2<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+I(X4^2)+I(X5^2)
formula_3<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+I(X4*X5)+I(X1*X2)+I(X1*X3)+I(X2*X3)
formula_4<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+sin(X5)
formula_5<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+I(X5^3)+I(X5^4)+I(X5^5)
formula_6<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+sin(X4)
formula_7<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+I(X4^3)+I(X4^4)+I(X4^5)
formula_8<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+I(X1^3)+I(X2^3)+I(X3^3)
formula_9<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+I(X1^3)+I(X1^4)+I(X5^5)
formula_10<-Y~.+I(X1^2)+I(X2^2)+I(X3^2)+I(X1*X2)
formula_11<-Y~.+I(X3^2)+I(X1*X2)+I((X1*X2)^3)
formula_12<-Y~.+I(X3^2)+I(X1*X2)+I((X1*X2)^3)+I((X1*X2)^5)
formula_13<-Y~X3+X4+X5+I(X3^2)+I(sin(pi*X1*X2))
```

Modelo	Adj R squared	MSE Train	MSE Test
1	0.7333728	6.902392	7.689300
2	0.8919430	2.869646	2.951615
3	0.8918162	2.864357	2.986020
4	0.8919381	2.872869	2.948185
5	0.8917691	2.871224	2.960010
6	0.8917874	2.877267	2.948549
7	0.8916832	2.873904	2.958509
8	0.8918099	2.871717	2.944046
9	0.8922399	2.858989	2.944657

Modelo	Adj R squared	MSE Train	MSE Test
10	0.8917536	2.876496	2.964811
11	0.9559447	1.172962	1.196416
12	0.9595456	1.076183	1.097551
13	0.9595813	1.079848	1.091163

Como vemos el planteamiento inicial de considerar todas las variables con todas sus posibles intearcciones entre ellas da muy malos resultados. Además de utilizar una gran cantidad de regresores dada la gran cantidad de combinaciones que se pueden hacer, esto complica el modelo mucho, ya que interpretar los coeficientes cuando metemos tantas interacciones entre las variables resulta muy complicado, incluso si estuviéramos en un contexto de un problema real donde estas interacciones pueden tener cierta lógica. Por tanto el hecho de que resultados tan pobres incluso comparándolo con el regresor múltiple que usaba todas las variables, hace que no tengamos que preocuparnos por sacrificar interpretabilidad por resultados.

El modelo 2 nos da resultados bastante mejores y no complica demasiado el modelo con respecto al regresor múltiple que usaba todas las variables. Luego probamos modelos con poco éxito con respecto a los anteriores pero que nos sirven para descartar el utilizar las variables $X4$ y $X5$ transformadas o elevadas a potencias superiores. Del mismo modo tratamos de hacer lo mismo con las variables $X1$ y $X2$ sin demasiado éxito. Llegados a este punto utilizamos la información que tenemos y nuestros conocimientos matemáticos, para comprobar si podríamos haber llegado a resultaods similares a los que obtiene el modelo 13 utilizando interacciones menos complejas entre las variables. Efectivamente basta ver los modelos 11 y 12 que pese a que no son muy complejos dan justo con la clave para mejorar significativamente los resultados.

La filosofía durante todo el proceso de creación de modelos utilizando distintos regresores ha sido conseguir el modelo más sencillo que diera mejores resultados, por eso en lugar de simplemente ir añadiendo variables a los sucesivos modelos las hemos sustituido por otras. Lo que hemos ido haciendo es sustituirlas en función de los p valores. Lo que queríamos era no perder interpretabilidad. Hay que decir que pese a que al final hemos utilizado información que en principio no tendríamos en un problema real, el modelo 11 es una prueba de que probando interacciones más o menos sencillas habríamos llegado eventualmente a conseguir un modelo que diera resultados como los últimos. Por otro lado observemos que si miramos el summary del modelo 13,

```
##
## Call:
## lm(formula = formula_13, data = friedman)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3811 -0.6731 -0.0324  0.6994  3.2800
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             5.12098    0.12328   41.54   <2e-16 ***
## X3                    -20.29982    0.41305  -49.15   <2e-16 ***
## X4                     10.18325    0.10649   95.63   <2e-16 ***
## X5                      4.97621    0.10779   46.16   <2e-16 ***
## I(X3^2)                20.22487    0.39845   50.76   <2e-16 ***
## I(sin(pi * X1 * X2))    9.86019    0.08936  110.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.042 on 1194 degrees of freedom
## Multiple R-squared:  0.9597, Adjusted R-squared:  0.9596
## F-statistic:  5694 on 5 and 1194 DF,  p-value: < 2.2e-16
```

observamos que los coeficientes que acompañan a las variables son muy parecidos a los de la fórmula que tenemos. Lo que no nos da información nueva en sí, pero verifica la que ya tenemos que tampoco está mal.

Finalmente observando la tabla que compara los modelos, podemos ver que ninguno parece hacer overfitting, ya que los errores en test y en training son muy similares. Quizás el primero de todos por ser el mas ambicioso sobreajusta algo más los datos de training, pero el resto no. Esto quiere decir que nuestra estrategia de conseguir un modelo sencillo que tuviera interpretabilidad también ha dado como resultado que los modelos no sobreajustasen los datos de entrenamiento, dada la (baja) complejidad de los modelos utilizados.

Algoritmo KNN

Nuestra tarea ahora será aplicar el algoritmo knn a nuestro dataset friedman tal y como hicimos en la clase de práticas. Vamos a utilizar las funciones de la parte de laboratorio y a usar knn con todas las variables y luego con algunas transformaciones utilizando lo que ya sabemos del dataset.

Vamos a usar el paquete "kknn" visto en clase, con la elección del kernell "optimal" y por defecto el k será 7, salvo que se especifique lo contrario.

```
nombre<-"friedman"
run_knn_fold<-function(i, x, tt= "test") {
file <-paste(x, "-5-", i, "tra.dat", sep="")
x_tra<-read.csv(file, comment.char="@",header=FALSE)
file <-paste(x, "-5-", i, "tst.dat", sep="")
x_tst<-read.csv(file, comment.char="@",header=FALSE)
In <-length(names(x_tra)) -1
names(x_tra)[1:In] <-paste ("X", 1:In, sep="")
names(x_tra)[In+1] <-"Y"
names(x_tst)[1:In] <-paste ("X", 1:In, sep="")
names(x_tst)[In+1] <-"Y"
if (tt== "train") {
test <-x_tra
}
else {
test <-x_tst
}
fitMulti=kknn(Y~.,x_tra,test)
yprime=fitMulti$fitted.values
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
```

```
knnMSEtrain<-mean(sapply(1:5,run_knn_fold,nombre,"train"))
knnMSEtest<-mean(sapply(1:5,run_knn_fold,nombre,"test"))
knnMSEtest
```

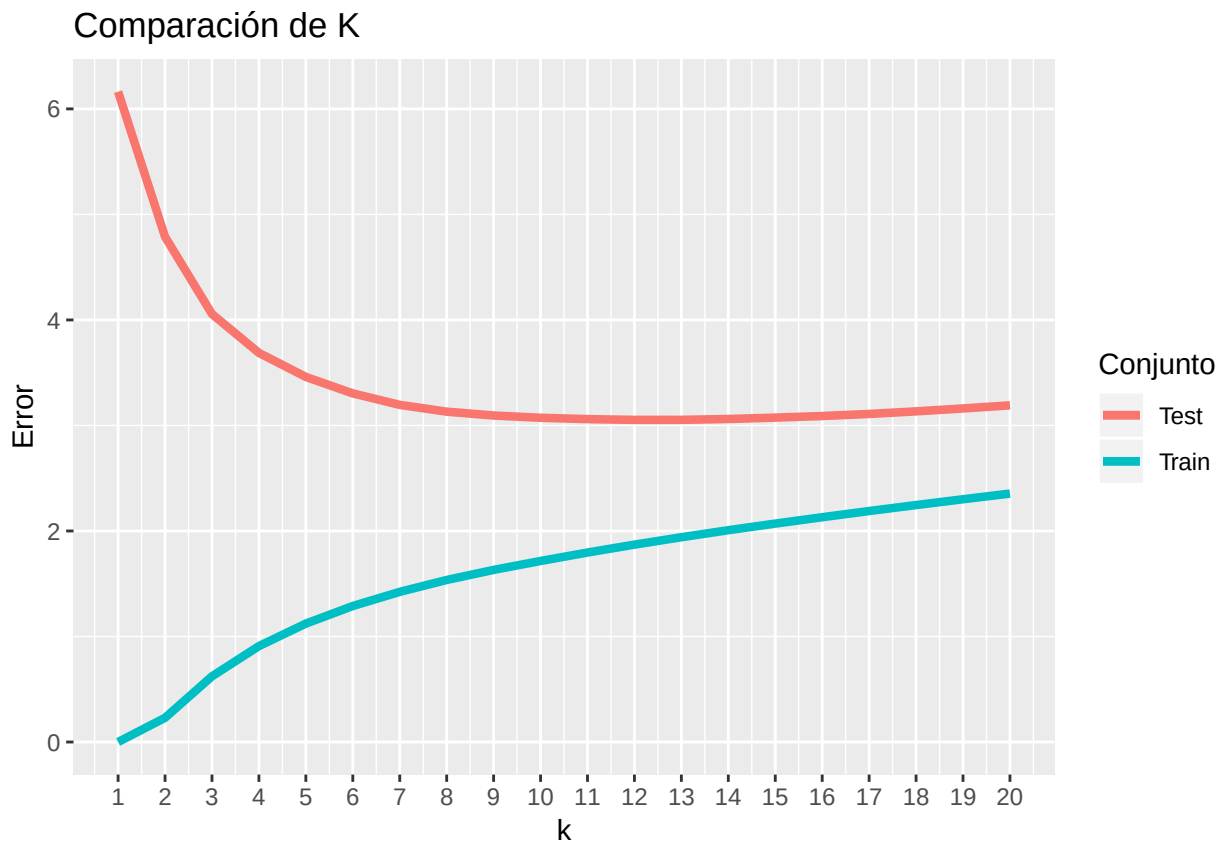
```
## [1] 3.195399
```

```
knnMSEtrain
```

```
## [1] 1.423071
```

Hay bastante diferencia en el rendimiento del método en train y en test en vista de que el error en test es más del doble que en training. Esto sucede mucho cuando k toma valores bajos y aunque estamos considerando $k = 7$ que es un valor aceptable, el modelo parece que ha sobreajustado los datos.

Pese a que se sale de los objetivos iniciales, vamos a ilustrar esto mismo con un gráfico comparando el rendimiento en training y en test para distintos valores de k.



En el gráfico podemos observar como para valores más pequeños de k, la discrepancia entre los resultados de test y de training es mayor.

Ahora vamos a continuar con nuestro estudio utilizando knn con distintas variables para el mismo $k = 7$. Como habitualmente, terminaremos mostrando una tabla comparando los resultados para luego interpretarlos.

Los modelos que vamos a probar son los siguientes:

1. Haciendo uso de todas las variables
2. Añadiendo $X1^2, X2^2, X3^2$.
3. Añadiendo $X1 * X2$.
4. Añadiendo $X1 * X2^3$ y $X1 * X2^5$.
5. Utilizando todas las varibles con $X3^2$ y $sin(\pi X1 * X2)$.

Estos modelos son los más interesantes de entre todos los que hemos probado anteriormente y queremos comprobar si los modelos que mejores resultados han dado en LM son los que mejores resultados darán ahora.

```
knnMSEtest<-c()
knnMSEtrain<-c()
knnMSEtrain<-c(knnMSEtrain,mean(sapply(1:5,run_knn_fold_formula,nombre,"train",Y~.)))
knnMSEtest<-c(knnMSEtest,mean(sapply(1:5,run_knn_fold_formula,nombre,"test",Y~.)))
knnMSEtrain<-c(knnMSEtrain,mean(sapply(1:5,run_knn_fold_formula,nombre,"train",
                                       Y~.+I(X1^2)+I(X2^2)+I(X3)^2)))
knnMSEtest<-c(knnMSEtest,mean(sapply(1:5,run_knn_fold_formula,nombre,"test",
                                     Y~.+I(X1^2)+I(X2^2)+I(X3)^2)))
knnMSEtrain<-c(knnMSEtrain,mean(sapply(1:5,run_knn_fold_formula,nombre,"train",
                                       Y~.+I(X1^2)+I(X2^2)+I(X3)^2+X1*X2)))
knnMSEtest<-c(knnMSEtest,mean(sapply(1:5,run_knn_fold_formula,nombre,"test",
                                     Y~.+I(X1^2)+I(X2^2)+I(X3)^2+X1*X2)))
```

```
knnMSEtrain<-c(knnMSEtrain,mean(sapply(1:5,run_knn_fold_formula,nombre,"train",
                                Y~.+I(X1^2)+I(X2^2)+I(X3)^2+X1*X2+
                                  I((X1*X2)^3)+I((X1*X2)^5))))
knnMSEtest<-c(knnMSEtest,mean(sapply(1:5,run_knn_fold_formula,nombre,"test",
                                   Y~.+I(X1^2)+I(X2^2)+I(X3)^2+X1*X2+
                                     I((X1*X2)^3)+I((X1*X2)^5))))
knnMSEtrain<-c(knnMSEtrain,mean(sapply(1:5,run_knn_fold_formula,nombre,"train",
                                     Y~.+I(X3^2)+sin(pi*X1*X2))))
knnMSEtest<-c(knnMSEtest,mean(sapply(1:5,run_knn_fold_formula,nombre,"test",
                                   Y~.+I(X3^2)+sin(pi*X1*X2))))
Modelos<-1:5
tabla<-data.frame(Modelos,knnMSEtrain,knnMSEtest)
knitr::kable(tabla)
```

Modelos	knnMSEtrain	knnMSEtest
1	1.423071	3.195399
2	1.393787	3.440392
3	1.216601	3.013324
4	1.241782	3.151435
5	1.107405	2.574884

Por los resultados de la tabla podemos concluir que hay cierto sobreajuste cuando utilizamos el algoritmo knn. Cosa que no ocurría cuando probábamos LM con las mismas variables regresoras. Lo que sí sorprende es que en el caso de knn, el modelo 3 da mejores resultados que el 4, cuando en el caso de LM era al contrario y la diferencia era significativa. Además vemos que si bien cuando utilizábamos únicamente todas las variables como regresores KNN era mejor que LM con diferencia, en cuanto metemos interacciones LM mejora considerablemente mientras que KNN no. Esto puede deberse a la elección de k o incluso del kernell, ya que no hemos hecho consideraciones especiales para el modelo. En la siguiente sección hacemos una comparación entre ambos algoritmos en un ambiente más general.

Comparación de Algoritmos

Para finalizar con la parte de regresión vamos a hacer la comparación de algoritmos con los ficheros de prado "regr_test_alumnos.csv" y "regr_train_alumnos.csv". Nosotros vamos a comparar cómo funcionan los algoritmos con los conjuntos de test y de training. Evidentemente es más relevante cómo funcionan de bien los algoritmos en test, pero comprobar su funcionamiento en los conjuntos de training también nos puede dar información sobre si alguno de los algoritmos tiende a sobreajustar más que otros los conjuntos de entrenamiento.

Primero leemos la tabla de resultados en los conjuntos test y training que hemos descargado de prado.

```
resultados <- read.csv("regr_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]
resultados <- read.csv("regr_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]
knitr:: kable(tablatst)
```

	out_test_lm	out_test_kknn	out_test_m5p
abalone	4.950000e+00	5.400000e+00	4.680000e+00

	out_test_lm	out_test_kknn	out_test_m5p
ANACALT	1.700000e-01	1.200000e-02	7.000000e-03
autoMPG6	1.162000e+01	7.740000e+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	0.000000e+00	0.000000e+00	0.000000e+00
delta_elv	2.100000e-06	2.400000e-06	2.000000e-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

```
knitr::kable(tablatra)
```

	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	1.129000e+01	3.530000e+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	0.000000e+00	0.000000e+00	0.000000e+00
delta_elv	2.100000e-06	1.000000e-06	2.000000e-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

Primero comparamos lm con knn utilizando el test de wilcoxon y normalizando los errores, ya estamos haciendo regresión. Utilizamos la normalización sugerida en los apuntes.

```
difs <- (tablatst[,1] - tablatst[,2]) / tablatst[,1]
wilc_1_2_tst <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
difs <- (tablatra[,1] - tablatra[,2]) / tablatra[,1]
wilc_1_2_tra <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
```

Aplicamos el test

```
LMvsKNNtst <- wilcox.test(wilc_1_2_tst[,1], wilc_1_2_tst[,2], alternative = "two.sided", paired=TRUE)
LMvsKNNtst$statistic
```

```
##  V
## 78
```

```
LMvsKNNtst <- wilcox.test(wilc_1_2_tst[,2], wilc_1_2_tst[,1], alternative = "two.sided", paired=TRUE)
LMvsKNNtst$statistic
```

```
##  V
## 93
```

```
LMvsKNNtst$p.value
```

```
## [1] 0.7660294
```

```
LMvsKNNtrain <- wilcox.test(wilc_1_2_tra[,1], wilc_1_2_tra[,2], alternative = "two.sided", paired=TRUE)
LMvsKNNtrain$statistic
```

```
##  V
## 10
```

```
LMvsKNNtrain <- wilcox.test(wilc_1_2_tra[,2], wilc_1_2_tra[,1], alternative = "two.sided", paired=TRUE)
LMvsKNNtrain$statistic
```

```
##   V
## 161
```

```
LMvsKNNtrain$p.value
```

```
## [1] 0.000328064
```

Los primeros resultados sobre los conjuntos test, nos dan un R+ de 78 y un R- de 93, pero sin embargo, el p-value nos sale muy alto, con aproximadamente un 0.23 de probabilidad se rechazaría la hipótesis de que los algoritmos dan los mismos resultados. Sobre los conjuntos de entrenamiento la cosa cambia con un R+ d 10 y un R- de 161, además el p-value es muy bajo, con más de 0.999 de probabilidad se rechazaría la hipótesis de que los algoritmos dan los mismos resultados. Por tanto vemos que el método knn funciona mucho mejor que el lm sobre los conjuntos de entrenamiento.

Ahora aplicamos el test de friedman para ver si entre LM KNN y M5' hay algún modelo que sea mejor o peor que el resto.

```
friedman.test(as.matrix(tablatst))
```

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatst)
## Friedman chi-squared = 8.4444, df = 2, p-value = 0.01467
```

```
friedman.test(as.matrix(tablatra))
```

```
##
##  Friedman rank sum test
##
## data:  as.matrix(tablatra)
## Friedman chi-squared = 20.333, df = 2, p-value = 3.843e-05
```

Como vemos al utilizar el test de friedman sobre los conjuntos de test, el p valor es de 0.01467. Lo cual es lo suficientemente bajo como para rechazar la hipótesis de que los tres algoritmos dan los mismo resultados. Por tanto luego haremos un post-hoc Holm para ver las diferencias entre los algoritmos. Por otro lado los

resultados sobre los conjuntos de entrenamiento son contundentes, en este caso el p valor es significativamente bajo, lo cual quiere decir que también sobre los conjuntos de training los métodos dan resultados distintos. Procedemos a hacer el estudio posterior para ver qué métodos difieren. tam <- dim(tablatst) groups <- rep(1:tam[2], each=tam[1]) pairwise.wilcox.test(as.matrix(tablatst), groups, p.adjust = "holm", paired = TRUE)

```
tam_tst <- dim(tablatst)
groups_tst <- rep(1:tam_tst[2], each=tam_tst[1])
pairwise.wilcox.test(as.matrix(tablatst), groups_tst, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatst) and groups_tst
##
##   1     2
## 2 0.580 -
## 3 0.081 0.108
##
## P value adjustment method: holm
```

```
tam_tra <- dim(tablatra)
groups_tra <- rep(1:tam_tra[2], each=tam_tra[1])
pairwise.wilcox.test(as.matrix(tablatra), groups_tra, p.adjust = "holm", paired = TRUE)
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  as.matrix(tablatra) and groups_tra
##
##   1      2
## 2 0.0031 -
## 3 0.0032 0.0032
##
## P value adjustment method: holm
```

Como podemos ver en la primera tabla, sobre los conjuntost test solo hay diferencias significativas en favor de M5'. Pero no parece haber diferencias significativas entre los algoritmos knn y lm. Por otro lado en los conjuntos de entrenamiento parece que todos los modelos son diferentes cuando los comparamos dos a dos, dado que en este caso todos los p valores son muy bajos. Si observamos las tablas que mostrábamos al principio del estudio parece que el que mejores resultados da en conjuntos de training es knn seguido de M5' y dejando en último lugar a lm.