

# Proyecto final de Introducción a Ciencia de Datos

*Laura del Pino Díaz*

*15/12/2016 - 08/01/2017*

## Índice

<b>Introducción</b>	<b>3</b>
<b>Análisis de datos</b>	<b>3</b>
Australian (Australian Credit Approval) . . . . .	3
Estudio de las variables numéricas . . . . .	5
Estudio del número de valores perdidos . . . . .	5
Estadísticos principales . . . . .	5
Test de normalidad . . . . .	6
Información gráfica . . . . .	14
Estudio de las correlaciones . . . . .	19
Estudio de las variables categóricas. . . . .	21
Estudio de los valores perdidos . . . . .	21
Estudio de los estadísticos principales . . . . .	21
Conclusiones del estudio de Australian . . . . .	27
Wizmir (Weather of Izmir) . . . . .	28
Hipótesis previas . . . . .	28
Variables numéricas. . . . .	29
Estudio de los valores perdidos . . . . .	29
Test de normalidad . . . . .	29
Estudio de los principales estadísticos . . . . .	29
Estudio de correlación . . . . .	31
Conclusión . . . . .	32
<b>Regresión</b>	<b>33</b>
Problema a tratar . . . . .	33
Regresores elegidos . . . . .	33
Modelo de regresión simple . . . . .	33
Modelo de regresión lineal múltiple . . . . .	35
Modelos de regresión lineal múltiple con interacciones. . . . .	36
Modelos no lineales. . . . .	39
Modelo Knn con Cross-validation de 5-particiones . . . . .	42
Comparación entre algoritmos (KNN y LM) . . . . .	43
Comparación de algoritmos . . . . .	44
Wilconxon . . . . .	44
Friedman . . . . .	45
Holms . . . . .	46
Conclusión a la regresión . . . . .	46
<b>Clasificación</b>	<b>47</b>
Preparación de los datos para clasificación. . . . .	47
Los k vecinos más cercanos. . . . .	48
Linear Discriminat Analysis (LDA) . . . . .	49
Quadratic Discriminant Analysis (QDA) . . . . .	50
Comparación de los tres algoritmos . . . . .	51
Conclusión al apartado de clasificación. . . . .	52

**Conclusión** **53**

**Bibliografía** **54**

# Introducción

En este proyecto vamos a realizar un análisis de dos bases de datos: la base de datos de la aprobación de créditos en australia ( *australian credit approval* ) y el tiempo atmosférico de la ciudad de Izmir(Turquía) ( *wizmir* ). A partir de este análisis de los datos se realizará un estudio de los modelos de clasificación con la base de datos de la aprobación de los créditos para determinar si se le pueden conceder o no el crédito. Mientras que con la base de datos del tiempo se elaborarán distintos modelos de regresión con el objetivo de predecir la temperatura media.

## Análisis de datos

En este apartado estudiaremos la estructura de los datos contenidos en las bases de datos *Australian Credit Approval*(abreviado *australian*) para el problema de clasificación y *Weather of Izmir*(abreviado *wizmir*) para el problema de regresión.

### Australian (Australian Credit Approval)

La base de datos *australian credit approval* tiene 15 atributos de los cuales actúan como predictores los primeros 14.

Los atributos de esta base de datos en particular no tienen un nombre descriptivo que te permita conocer que es lo que representan los datos por razones de confidencialidad, tal y como se detalla en la página de UCI. Lo que si conocemos es el número de observaciones, 690, y los diferentes tipos de variables que componen la base de datos y el intervalo o valores que puede tomar cada variable y se enlistan a continuación.

- A1 nominal {0, 1}
- A2 real [16.0,8025.0]
- A3 real [0.0,26335.0]
- A4 nominal {1, 2, 3}
- A5 entero [1,14]
- A6 entero [1,9]
- A7 real [0.0,14415.0]
- A8 nominal {0, 1}
- A9 nominal {0, 1}
- A10 entero [0,67]
- A11 nominal {0, 1}
- A12 nominal {1, 2, 3}
- A13 entero [0,2000]
- A14 entero [1,100001]
- Class nominal {0,1}

Dado la nula descriptividad de los nombres no podemos realizar hipótesis previas sobre la base de datos. Por lo que procedemos a realizar un estudio de los principales estadísticos de cada variables. Este estudio lo dividiremos en dos partes: una parte dedicada a las variables numéricas y otra parte dedicada a las variables categóricas.

```
australian <- read.csv("./AustralianClassification/australian/australian.dat",
  comment.char = "@", header = FALSE)
names(australian) <- c("A1", "A2", "A3", "A4", "A5", "A6", "A7",
  "A8", "A9", "A10", "A11", "A12", "A13", "A14", "A15")

numerical_australian <- australian[, c(-1, -4, -8, -9, -11, -12,
  -15)]
```

```
categorical_australian <- australian[, c(1, 4, 8, 9, 11, 12)]  
output_australian <- australian[, 15]
```

## Estudio de las variables numéricas

Son varios los parámetros que nos permiten conocer con más detalle la base de datos aunque no se tenga un nombre descriptivo para cada una de ellas, ejemplo de esto es el número de valores perdidos, la media, mediana, moda y cuartiles. Así mismo se hace necesario comprobar algunos de los supuestos que realizan los algoritmos LDA y QDA que emplearemos en este problema de clasificación, como son los test de normalidad del conjunto de datos de entrada y la igualdad de las varianzas.

### Estudio del número de valores perdidos

En la página web de la base de datos de *Australian Credit Approval* se indica que la base de datos tiene valores perdidos, en esta sección vamos a comprobar qué variables tienen esos valores perdidos.

```
numerical_australian_na <- apply(is.na(numerical_australian),  
  2, sum)  
numerical_australian_na
```

```
A2  A3  A5  A6  A7  A10 A13 A14  
0   0   0   0   0   0   0   0
```

Puesto que en las variables numéricas no hay valores perdidos, podemos seguir el estudio con las variables en este estado sin necesidad de imputar valores.

### Estadísticos principales

```
numerical_stats <- summary(numerical_australian)  
numerical_std <- apply(numerical_australian, 2, sd)  
numerical_stats <- rbind(numerical_stats, numerical_std)  
numerical_stats
```

```
          A2           A3  
"Min.    : 16   " "Min.    : 0   "  
"1st Qu.:1942  " "1st Qu.: 15  "  
"Median  :2629  " "Median  : 125 "  
"Mean    :2697  " "Mean    : 1187 "  
"3rd Qu.:3525  " "3rd Qu.: 665 "  
"Max.    :8025  " "Max.    :26335 "  
numerical_std "1554.55973203261" "3069.11004226953"  
          A5           A6  
"Min.    : 1.000  " "Min.    :1.000  "  
"1st Qu.: 4.000  " "1st Qu.:4.000  "  
"Median  : 8.000  " "Median  :4.000  "  
"Mean    : 7.372  " "Mean    :4.693  "  
"3rd Qu.:10.000  " "3rd Qu.:5.000  "  
"Max.    :14.000  " "Max.    :9.000  "  
numerical_std "3.68326478743128" "1.9923160695339"  
          A7           A10  
"Min.    : 0.0   " "Min.    : 0.0  "  
"1st Qu.: 5.0   " "1st Qu.: 0.0  "  
"Median  : 35.0  " "Median  : 0.0  "  
"Mean    : 453.4  " "Mean    : 2.4  "  
"3rd Qu.: 219.8  " "3rd Qu.: 3.0  "  
"Max.    :14415.0 " "Max.    :67.0  "  
numerical_std "1387.90032404432" "4.862940034227"  
          A13          A14
```

```

"Min.    :  0   " "Min.    : 1.0   "
"1st Qu.: 80   " "1st Qu.: 1.0   "
"Median : 160  " "Median : 6.0   "
"Mean   : 184  " "Mean   : 1018.4 "
"3rd Qu.: 272  " "3rd Qu.: 396.5 "
"Max.    :2000  " "Max.    :100001.0 "
numerical_std "172.159273536299" "5210.10259830269"

```

Como podemos ver las variables con mayor varianza son la variable A2,A3,A7 y A14 debido a que los rangos que puede tomar dicha variable son mayores.

Si queremos comparar las varianzas de cada grupo de la salida, pensando comprobar el supuesto de que las covarianzas de los grupos son iguales para el algoritmo LDA podemos usar un test de Levene dentro del paquete *car*. En este caso, lo que hemos hecho es una función que comprueba el valor del p-value del test de Levene de cada variable y nos devuelve si la hipótesis nula de que las varianzas son iguales para las categorías de la salida es aceptada o no.

```

require(car)
leveneTestForNumerical = function(column) {
  l = leveneTest(numerical_australian[, column], output_australian)
  ifelse(l$`Pr(>F)` > 0.05, "Nula aceptada", "Nula rechazada")
}
results_Levene = sapply(1:(dim(numerical_australian)[2]), leveneTestForNumerical)
results_Levene[1, ]

[1] "Nula rechazada" "Nula aceptada"  "Nula aceptada"
[4] "Nula rechazada" "Nula rechazada" "Nula rechazada"
[7] "Nula aceptada"  "Nula rechazada"

```

La hipótesis de igualdad de varianzas solo se acepta para las variables A3, A5 y A10, por lo que un modelo LDA con estas variables puede dar un buen resultado.

## Test de normalidad

Establezcamos como hipótesis nula que todas las variables numéricas pertenecen a una misma distribución. Para comprobarlo realizaremos el test no paramétrico de Kruskal Wallis.

```
kruskal.test(numerical_australian)
```

```

Kruskal-Wallis rank sum test

data: numerical_australian
Kruskal-Wallis chi-squared = 2713.3, df = 7, p-value
< 2.2e-16

```

Puesto que el p-value es menor que 0.05 rechazamos la hipótesis de que todas las variables siguen la misma distribución. Necesitamos encontrar aquellas variables que sí que sigan una distribución normal, para ello someteremos a todas al test de Shapiro-Wilk. Este test mantiene como hipótesis nula que la muestra que tiene como entrada viene de una distribución normal, en caso de que el p-value obtenido sea menor que 0.05 se rechaza la hipótesis nula.

```

numerical_australian_shapiro <- apply(numerical_australian, 2,
                                         shapiro.test)
numerical_australian_shapiro

```

\$A2

```
Shapiro-Wilk normality test

data: newX[, i]
W = 0.96297, p-value = 3.452e-12
```

\$A3

```
Shapiro-Wilk normality test

data: newX[, i]
W = 0.42625, p-value < 2.2e-16
```

\$A5

```
Shapiro-Wilk normality test

data: newX[, i]
W = 0.95306, p-value = 5.121e-14
```

\$A6

```
Shapiro-Wilk normality test

data: newX[, i]
W = 0.78023, p-value < 2.2e-16
```

\$A7

```
Shapiro-Wilk normality test

data: newX[, i]
W = 0.34139, p-value < 2.2e-16
```

\$A10

```
Shapiro-Wilk normality test

data: newX[, i]
W = 0.53306, p-value < 2.2e-16
```

\$A13

```
Shapiro-Wilk normality test

data: newX[, i]
W = 0.82069, p-value < 2.2e-16
```

\$A14

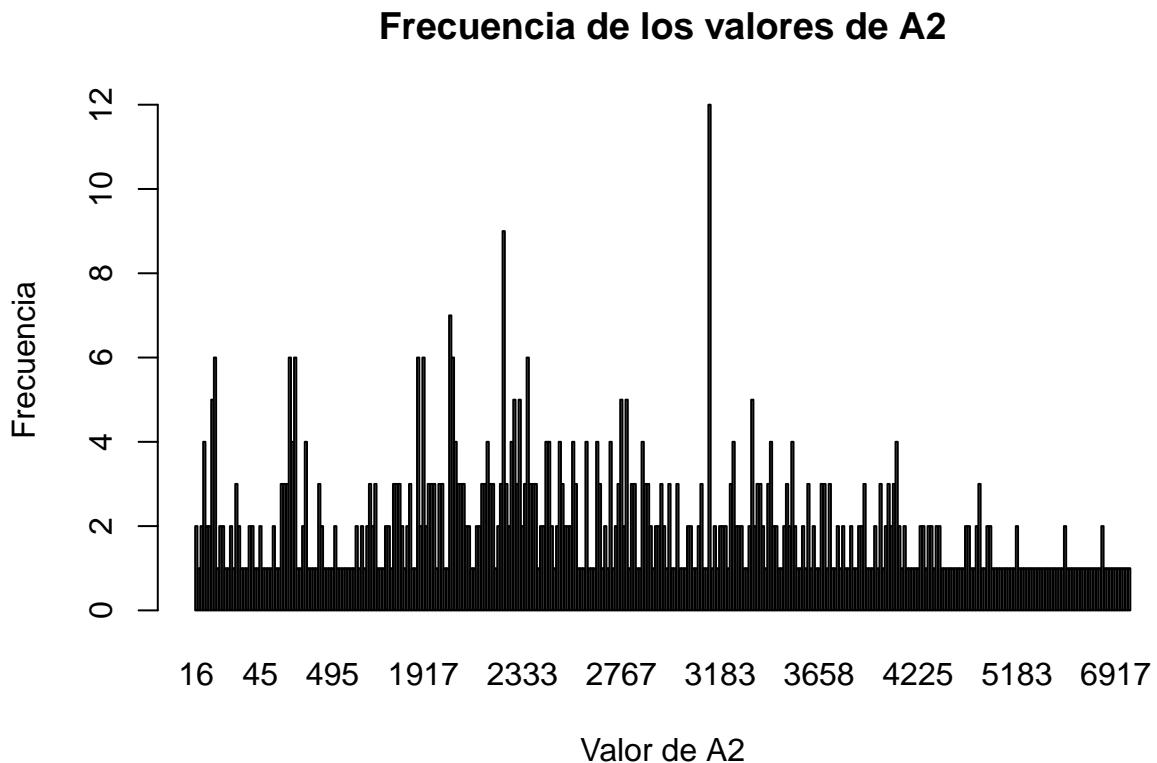
Shapiro-Wilk normality test

```
data: newX[, i]
W = 0.16985, p-value < 2.2e-16
```

Dado que todos los p-value de todas las variables es menor que 0.05 deducimos que ninguna de las variables sigue una distribución normal por lo que no se espera que ni el algoritmo LDA ni el algoritmo QDA funcionen bien para la clasificación.

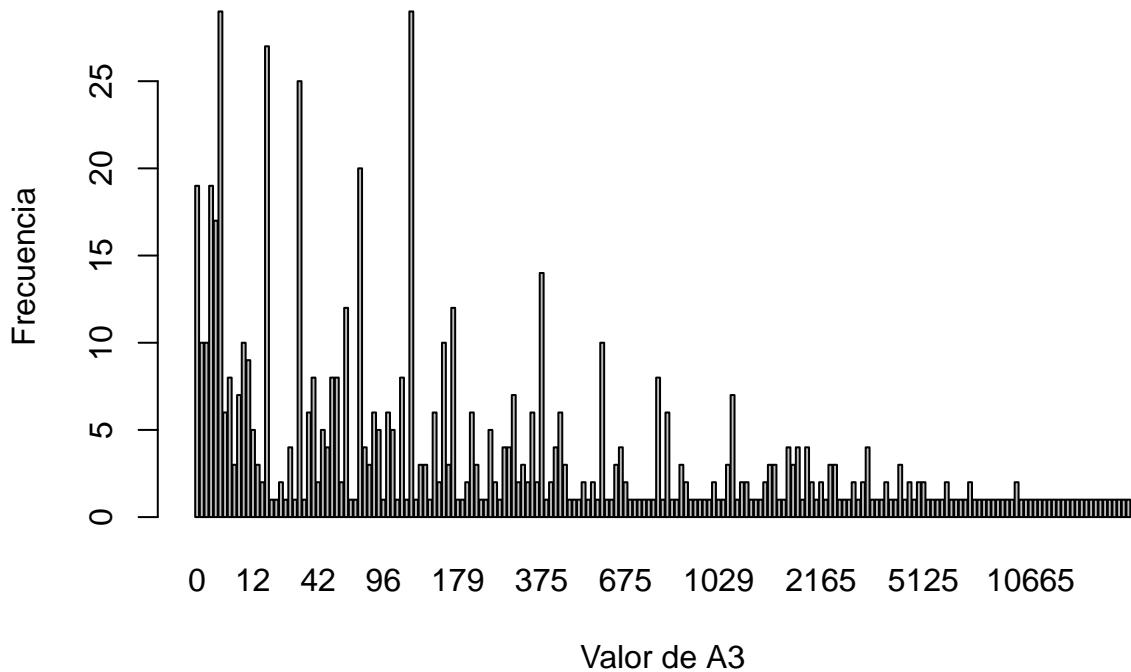
A continuación se muestran los gráficos de barras de todas las variables numéricas:

```
barplot(table(numerical_australian[, 1]), main = "Frecuencia de los valores de A2",
       xlab = "Valor de A2", ylab = "Frecuencia")
```



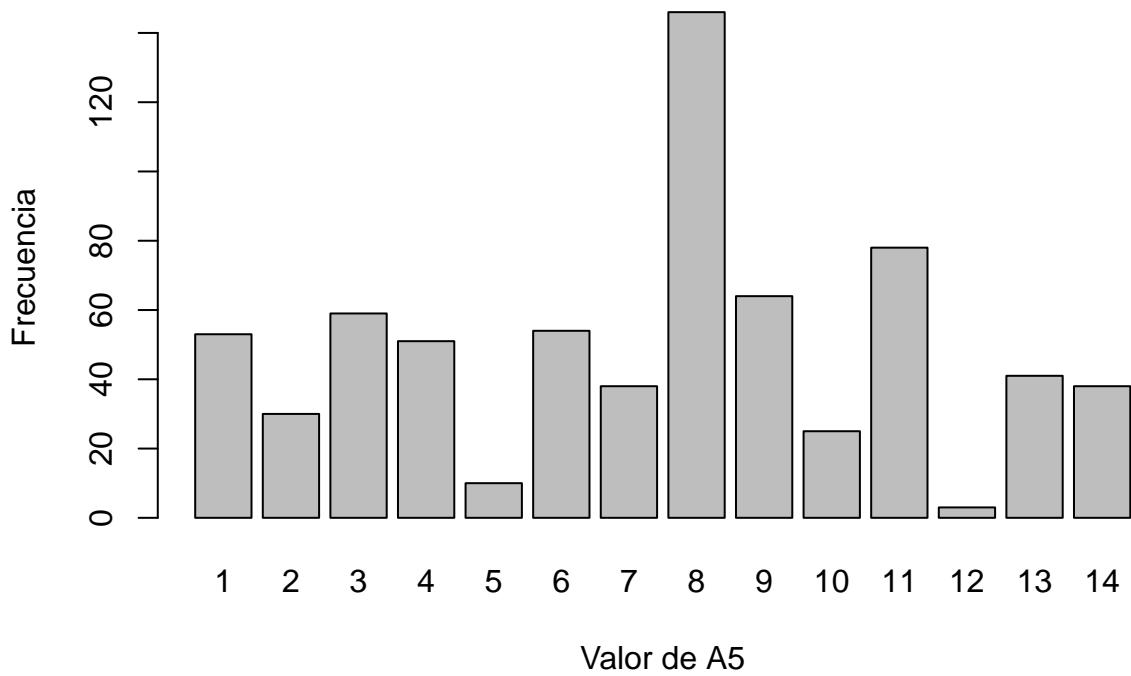
```
barplot(table(numerical_australian[, 2]), main = "Frecuencia de los valores de A3",
       xlab = "Valor de A3", ylab = "Frecuencia")
```

### Frecuencia de los valores de A3



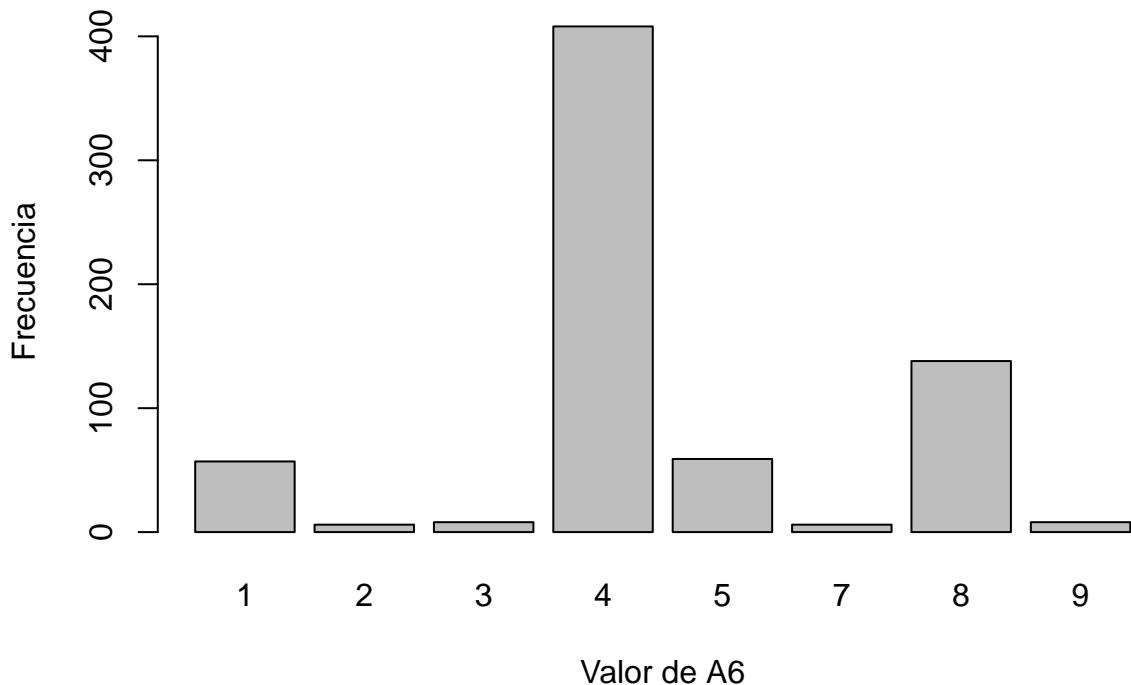
```
barplot(table(numerical_australian[, 3]), main = "Frecuencia de los valores de A5",
       xlab = "Valor de A5", ylab = "Frecuencia")
```

### Frecuencia de los valores de A5



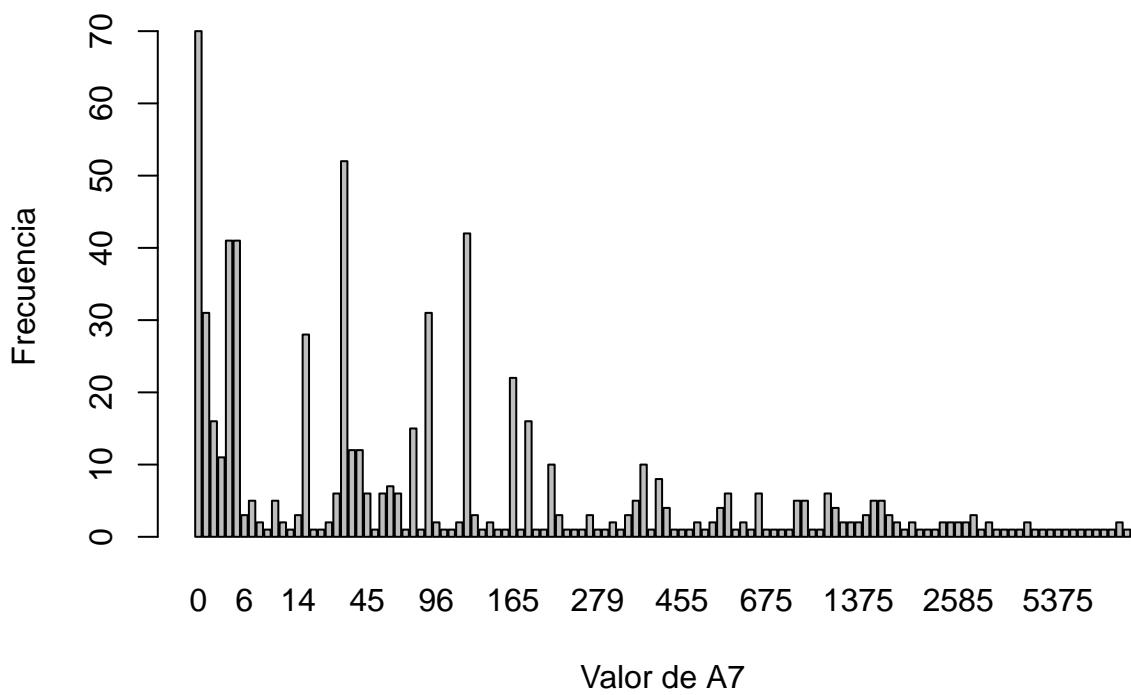
```
barplot(table(numerical_australian[, 4]), main = "Frecuencia de los valores de A6",
       xlab = "Valor de A6", ylab = "Frecuencia")
```

### Frecuencia de los valores de A6



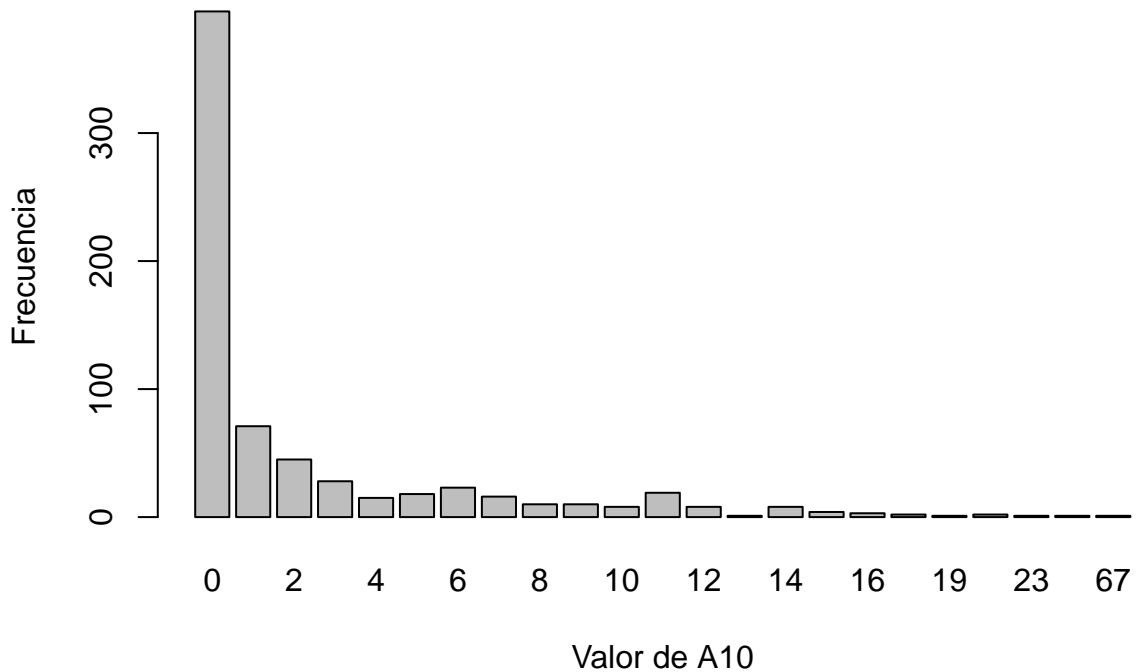
```
barplot(table(numerical_australian[, 5]), main = "Frecuencia de los valores de A7",
       xlab = "Valor de A7", ylab = "Frecuencia")
```

### Frecuencia de los valores de A7



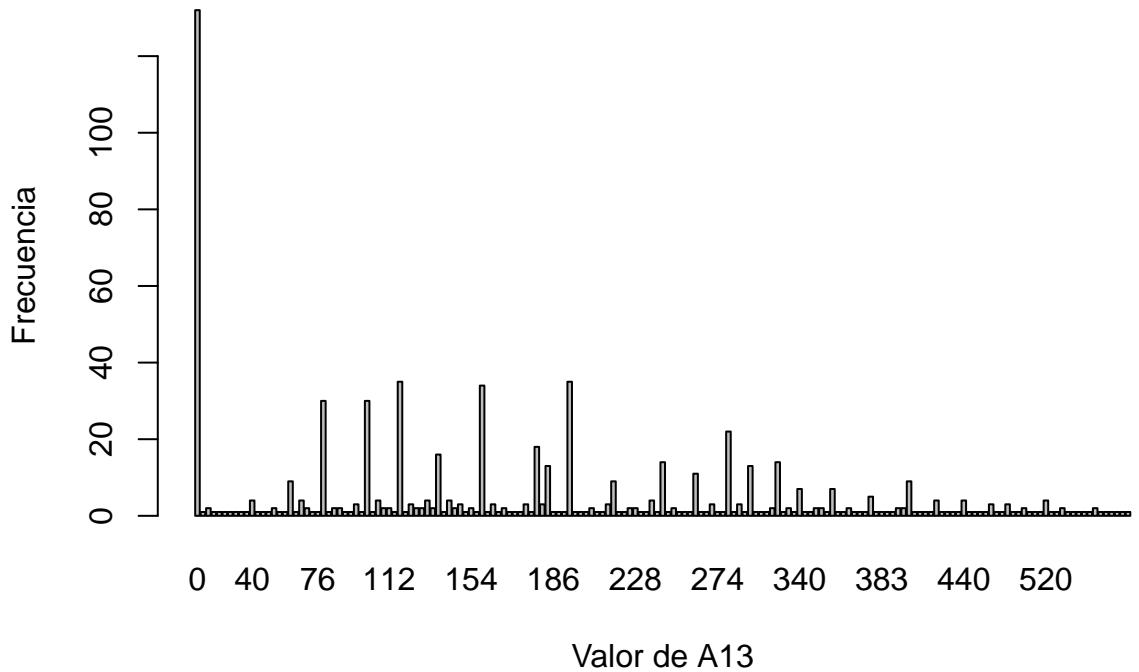
```
barplot(table(numerical_australian[, 6]), main = "Frecuencia de los valores de A10",
       xlab = "Valor de A10", ylab = "Frecuencia")
```

## Frecuencia de los valores de A10



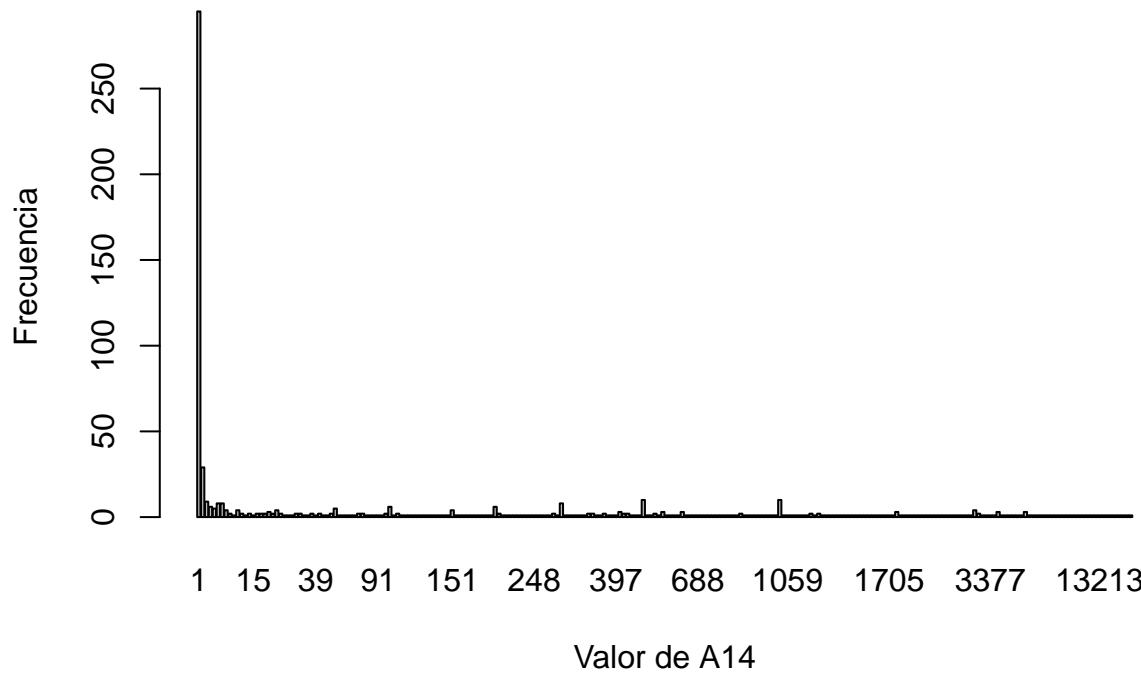
```
barplot(table(numerical_australian[, 7]), main = "Frecuencia de los valores de A13",
       xlab = "Valor de A13", ylab = "Frecuencia")
```

## Frecuencia de los valores de A13



```
barplot(table(numerical_australian[, 8]), main = "Frecuencia de los valores de A14",
       xlab = "Valor de A14", ylab = "Frecuencia")
```

## Frecuencia de los valores de A14



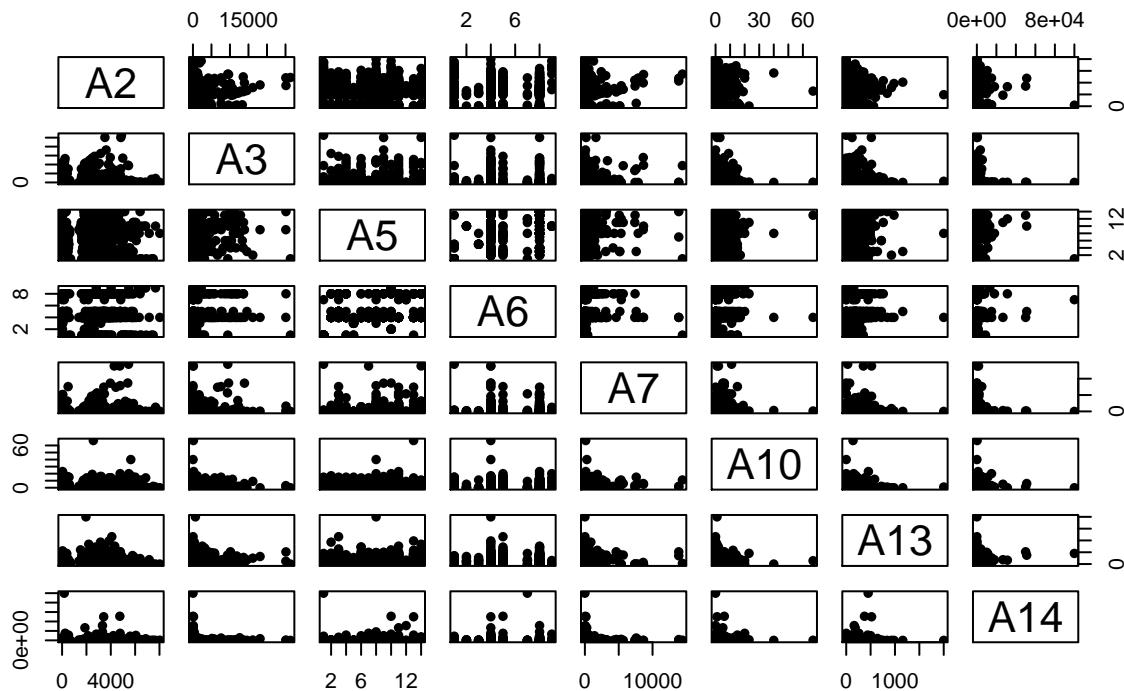
Como ya habíamos visto de forma numérica, por las gráficas no se puede decir que ninguna de las variables numéricas sea una distribución normal.

## Información gráfica

Vamos a mostrar los valores que toman cada una de las variables y a compararlas entre ellas con un scatter plot.

```
pairs(numerical_australian, main = "Comparación de las variables numéricas con la salida",  
      pch = 16)
```

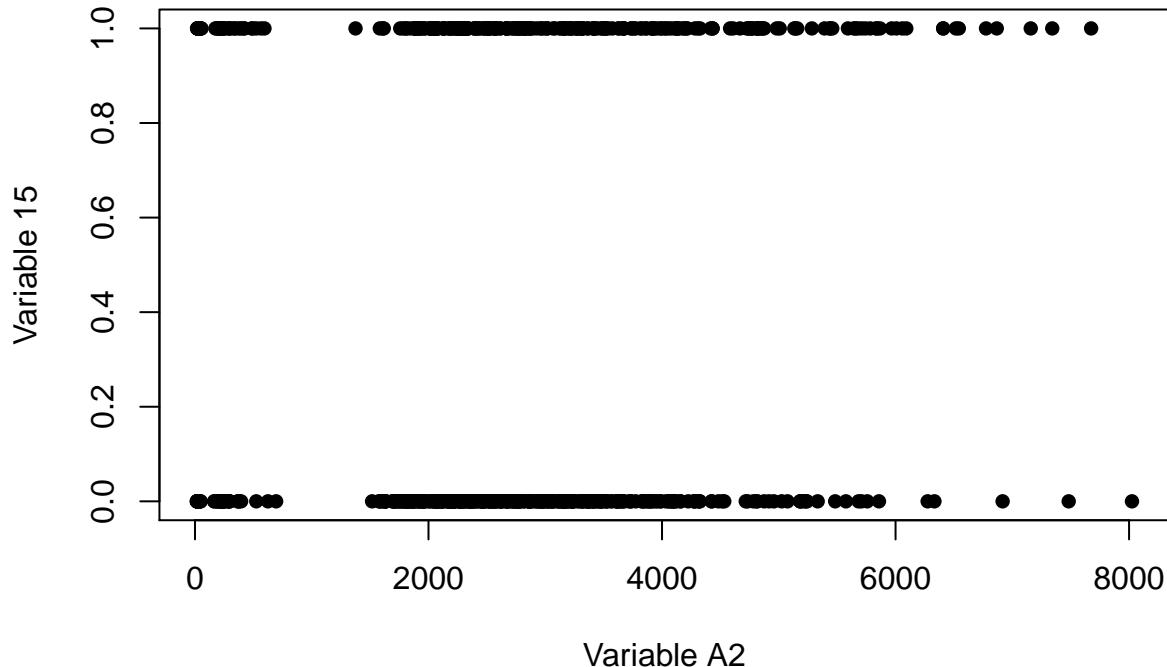
### Comparación de las variables numéricas con la salida



Puesto que es una base de datos para clasificación con dos clases tiene sentido que en todas ellas aparezcan las dos columnas. Pero verlas así en pequeño no nos permite deducir si esa variable aporta mucho o poco a la salida, por lo que vamos a realizar unos plots para analizar mejor los datos.

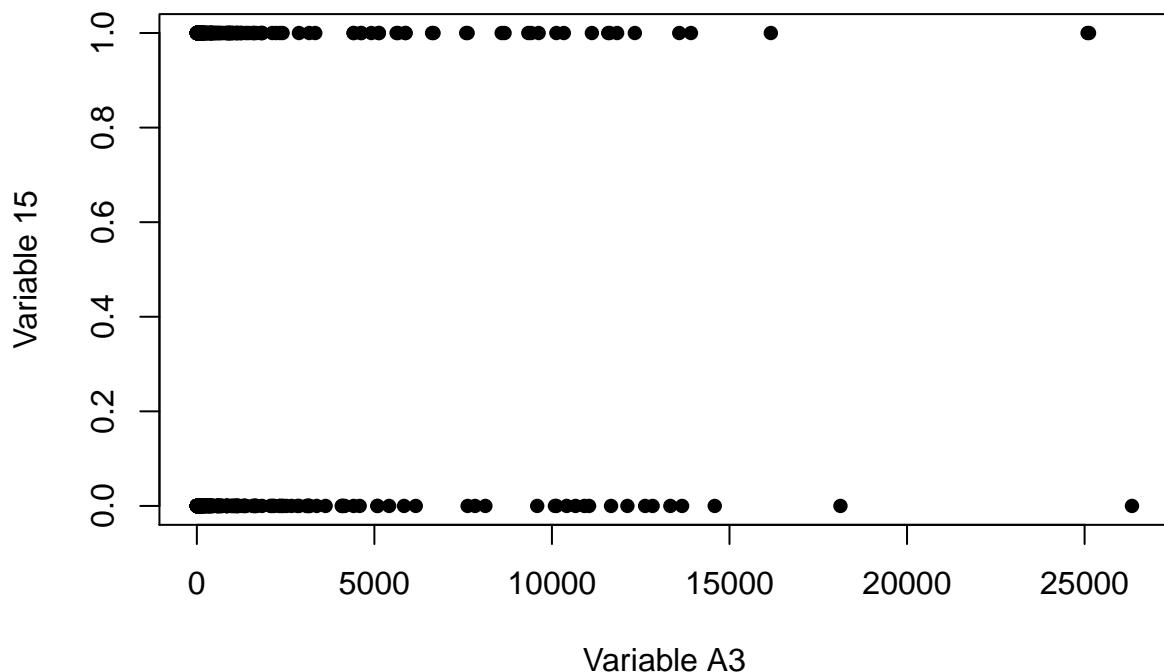
```
plot(australian[, 2], australian[, 15], main = "Comparación A2 con la salida",
      pch = 16, xlab = "Variable A2", ylab = "Variable 15")
```

Comparación A2 con la salida



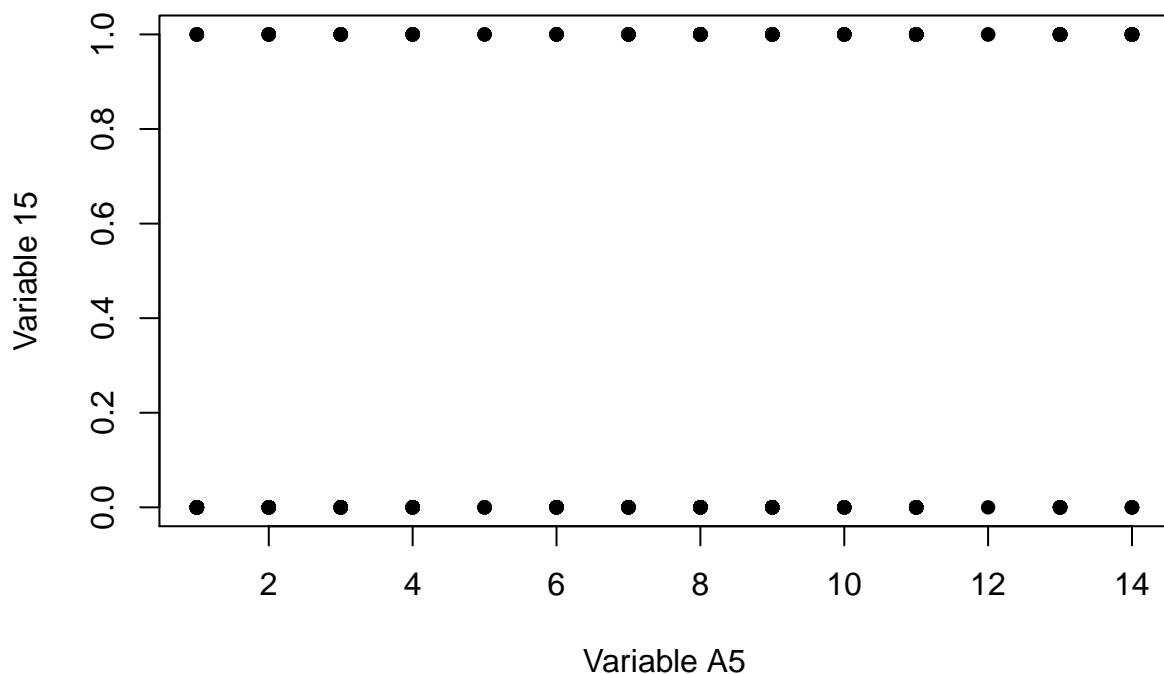
```
plot(australian[, 3], australian[, 15], main = "Comparación A3 con la salida",
      pch = 16, xlab = "Variable A3", ylab = "Variable 15")
```

### Comparación A3 con la salida



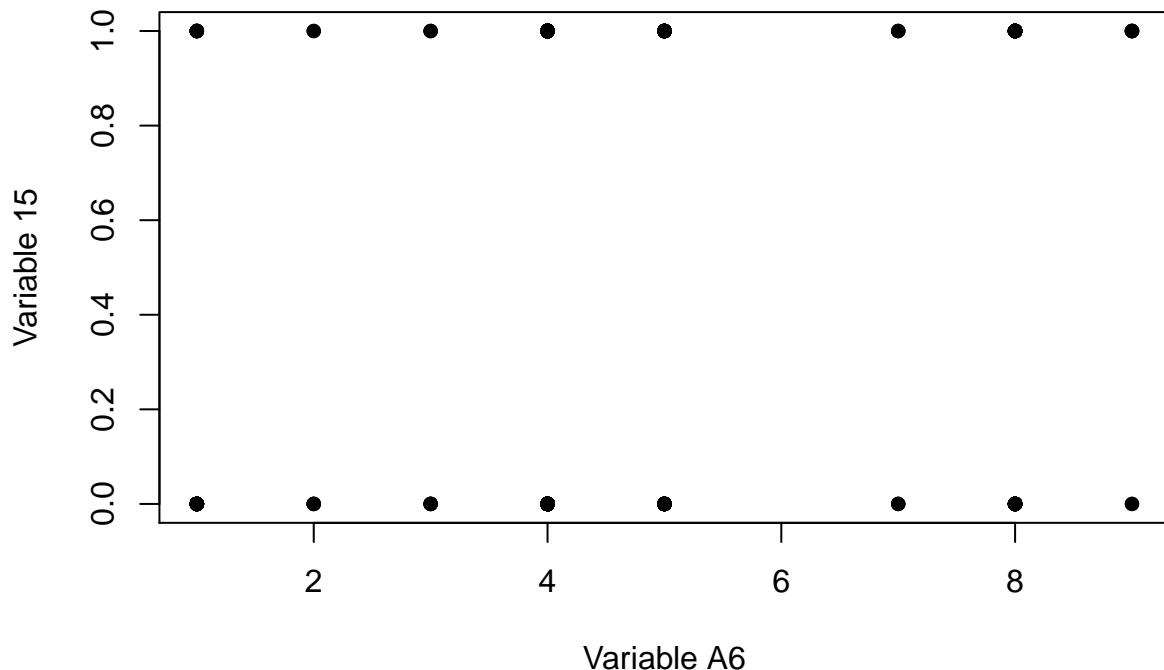
```
plot(australian[, 5], australian[, 15], main = "Comparación A5 con la salida",
      pch = 16, xlab = "Variable A5", ylab = "Variable 15")
```

### Comparación A5 con la salida



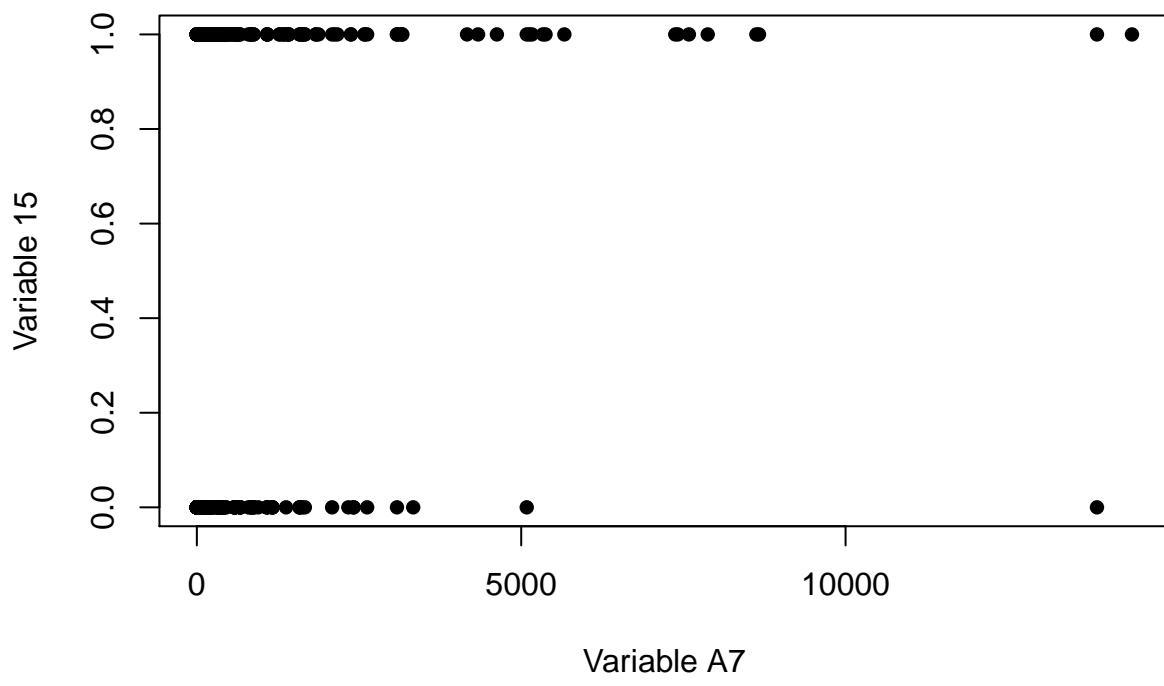
```
plot(australian[, 6], australian[, 15], main = "Comparación A6 con la salida",
      pch = 16, xlab = "Variable A6", ylab = "Variable 15")
```

## Comparación A6 con la salida



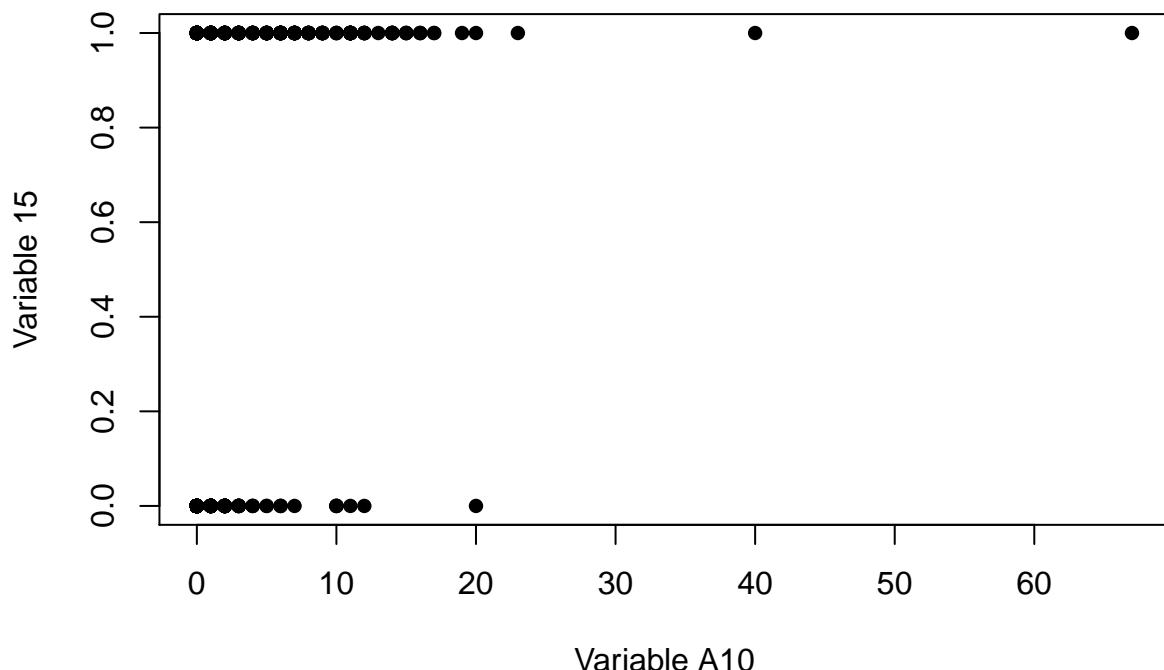
```
plot(australian[, 7], australian[, 15], main = "Comparación A7 con la salida",
      pch = 16, xlab = "Variable A7", ylab = "Variable 15")
```

## Comparación A7 con la salida



```
plot(australian[, 10], australian[, 15], main = "Comparación A10 con la salida",
      pch = 16, xlab = "Variable A10", ylab = "Variable 15")
```

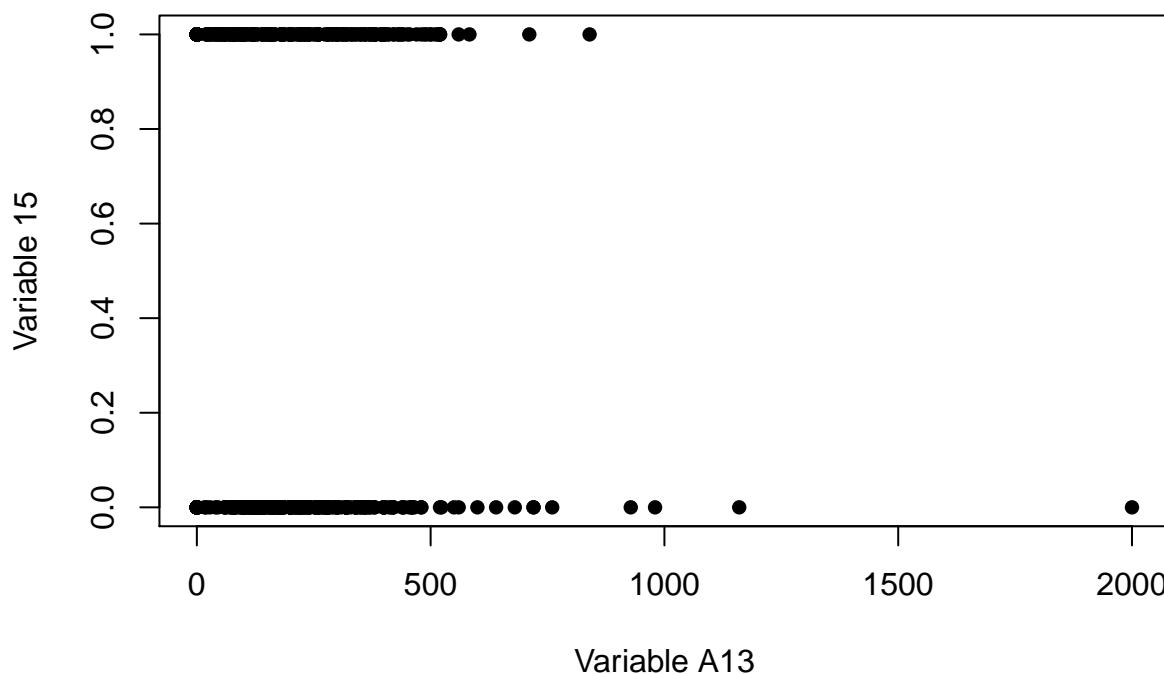
## Comparación A10 con la salida



Variable A10

```
plot(australian[, 13], australian[, 15], main = "Comparación A13 con la salida",
      pch = 16, xlab = "Variable A13", ylab = "Variable 15")
```

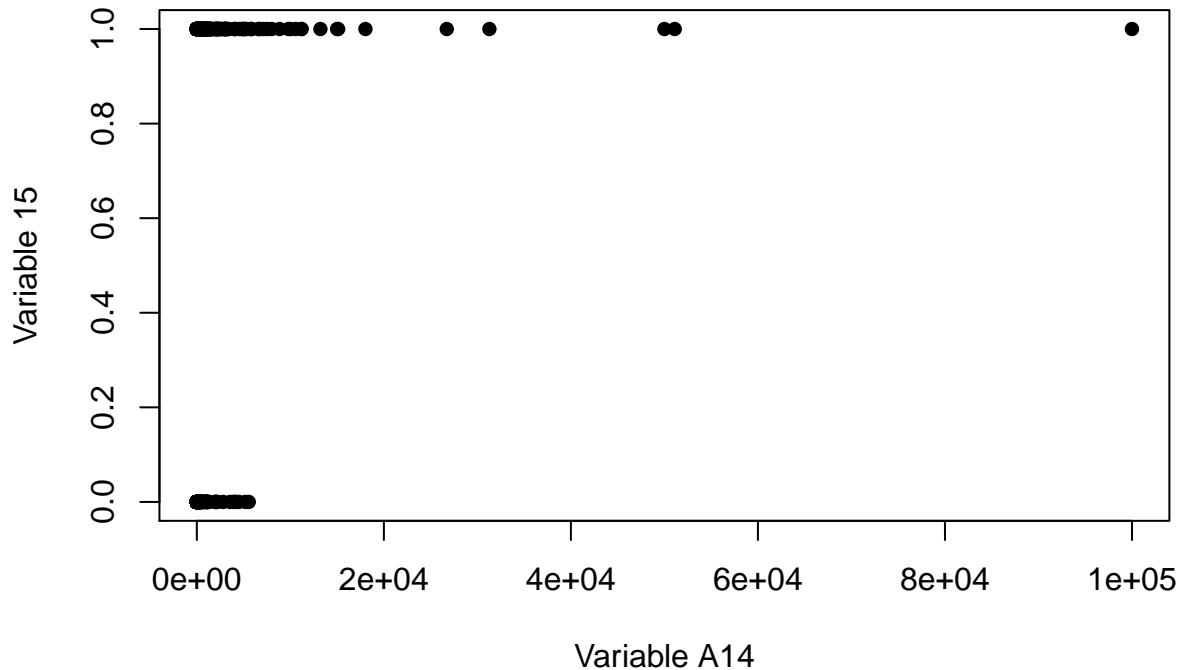
## Comparación A13 con la salida



Variable A13

```
plot(australian[, 14], australian[, 15], main = "Comparación A15 con la salida",
      pch = 16, xlab = "Variable A14", ylab = "Variable 15")
```

## Comparación A15 con la salida



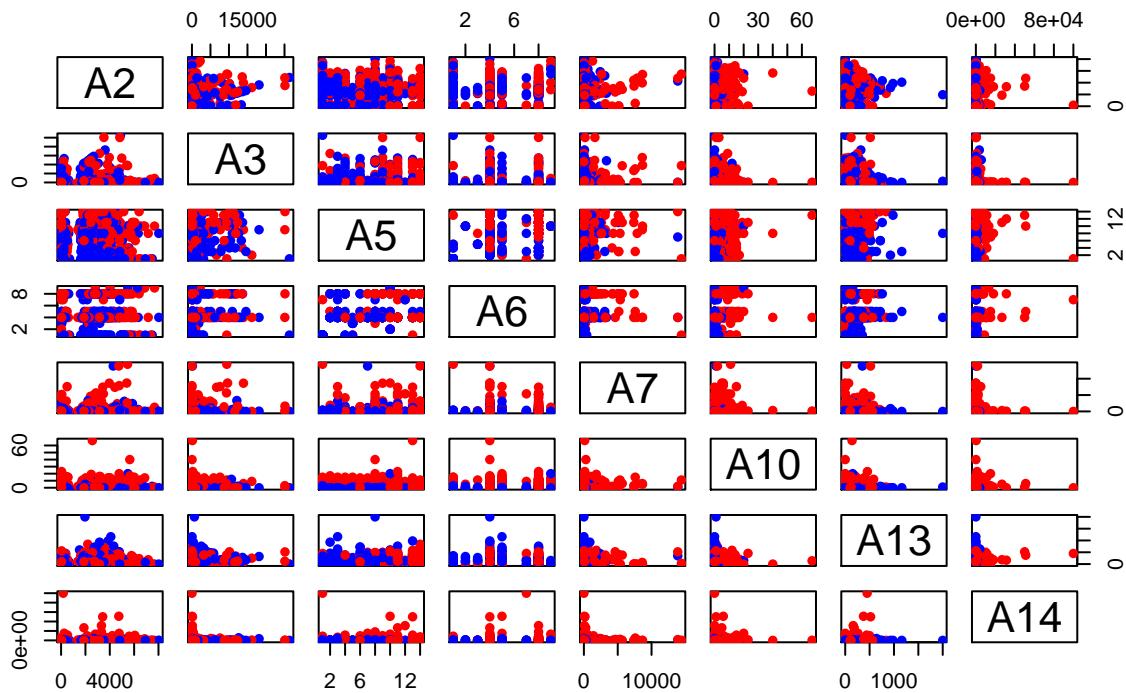
No podemos decir que ninguna de las variables numéricas sea realmente discriminante para la salida. Sin embargo, no descartaría la variable 14 para que intervenga un modelo de clasificación, porque para valores altos solamente da salida positiva para la clase 1, lo que puede servir para clasificar en estos casos.

### Estudio de las correlaciones

Volveremos a representar las variables ahora dejando la variable de salida fuera

```
pairs(numerical_australian, main = "Comparación de las variables numéricas entre ellas",
      col = ifelse(australian[, 15] == 1, "red", "blue"), pch = 16)
```

## Comparación de las variables numéricas entre ellas



Aparentemente no hay relación entre las variables, lo que parece curioso es que cualquiera de las variables que interacciona con la variable 6 forma una nube de puntos que recuerda a un histograma.

Vamos a comprobar de forma numérica que no existe esta correlación, para ello calcular la correlación entre todos los pares de variables

```
cor(numerical_australian, method = "pearson")
```

	A2	A3	A5	A6
A2	1.000000000	0.014978547	-0.05862121	-0.001506815
A3	0.014978547	1.000000000	0.05232107	0.065151425
A5	-0.058621212	0.052321072	1.000000000	0.402283761
A6	-0.001506815	0.065151425	0.40228376	1.000000000
A7	0.073636240	0.149224719	0.09780098	0.077021503
A10	0.117647696	-0.009713481	0.15016586	0.098840728
A13	-0.025026165	-0.014903065	0.08813968	0.070661763
A14	0.002460758	-0.035580640	0.03073527	0.064840849
	A7	A10	A13	A14
A2	0.073636240	0.117647696	-0.025026165	0.002460758
A3	0.149224719	-0.009713481	-0.014903065	-0.035580640
A5	0.097800977	0.150165862	0.088139683	0.030735269
A6	0.077021503	0.098840728	0.070661763	0.064840849
A7	1.000000000	0.104332170	0.005644437	-0.018071625
A10	0.104332170	1.000000000	-0.119808064	0.063692439
A13	0.005644437	-0.119808064	1.000000000	0.065608872
A14	-0.018071625	0.063692439	0.065608872	1.000000000

Como ya pensábamos nos existe correlación significativa entre ningún par de variables, pero llama la atención la correlación del 0.4 entre la variable A5 y la variable A6.

## Estudio de las variables categóricas.

Las variables categóricas merecen un estudio propio puesto que estadísticos como la media o la desviación típica no tienen un valor interesante ya que no tiene sentido si tenemos las clases 1,2,3 y que la clase media sea 2,2. Por ello los estadísticos que vamos a usar en esta sección son los cuartiles y el valor más frecuente o moda. Pero antes que nada vamos a estudiar si el conjunto de los datos categóricos contienen valores pedidos.

## Estudio de los valores perdidos

```
categorical_australian_na <- apply(is.na(cbind(categorical_australian,
  output_australian)), 2, sum)
categorical_australian_na
```

A1	A4	A8
0	0	0
A9	A11	A12
0	0	0
output_australian		
0		

Las variables categóricas no tienen valores perdidos, dado que en la página web de esta base de datos en UCI indica que si que tiene valores perdido, asumimos que la copia de la base de datos, obtenida de Prado, que tenemos se le han imputado los valores perdidos antes de proporcionárnosla a los alumnos.

## Estudio de los estadísticos principales

Como mencionamos anteriormente los estadísticos que nos interesan en estas variables categóricas son: el mínimo, el máximo, los cuartiles y la moda. Por suerte para nosotros todos estos valores los podemos obtener con un solo comando excepto la moda que la obtendremos aparte.

```
categorical_stats <- summary(australian[, c(1, 4, 8, 9, 11, 12)])
categorical_stats <- categorical_stats[-4, ]
```

Para obtener el valor más frecuente o moda del conjunto haremos uso de la siguiente función:

```
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

Realizando la moda sobre cada una de las variables categóricas tenemos:

```
categorical_stats <- rbind(categorical_stats, apply(australian[, 
  c(1, 4, 8, 9, 11, 12)], 2, Mode))
categorical_stats
```

A1	A4	A8
"Min. :0.0000 "	"Min. :1.000 "	"Min. :0.0000 "
"1st Qu.:0.0000 "	"1st Qu.:2.000 "	"1st Qu.:0.0000 "
"Median :1.0000 "	"Median :2.000 "	"Median :1.0000 "
"3rd Qu.:1.0000 "	"3rd Qu.:2.000 "	"3rd Qu.:1.0000 "
"Max. :1.0000 "	"Max. :3.000 "	"Max. :1.0000 "
"1"	"2"	"1"
A9	A11	A12
"Min. :0.0000 "	"Min. :0.000 "	"Min. :1.000 "
"1st Qu.:0.0000 "	"1st Qu.:0.000 "	"1st Qu.:2.000 "
"Median :0.0000 "	"Median :0.000 "	"Median :2.000 "

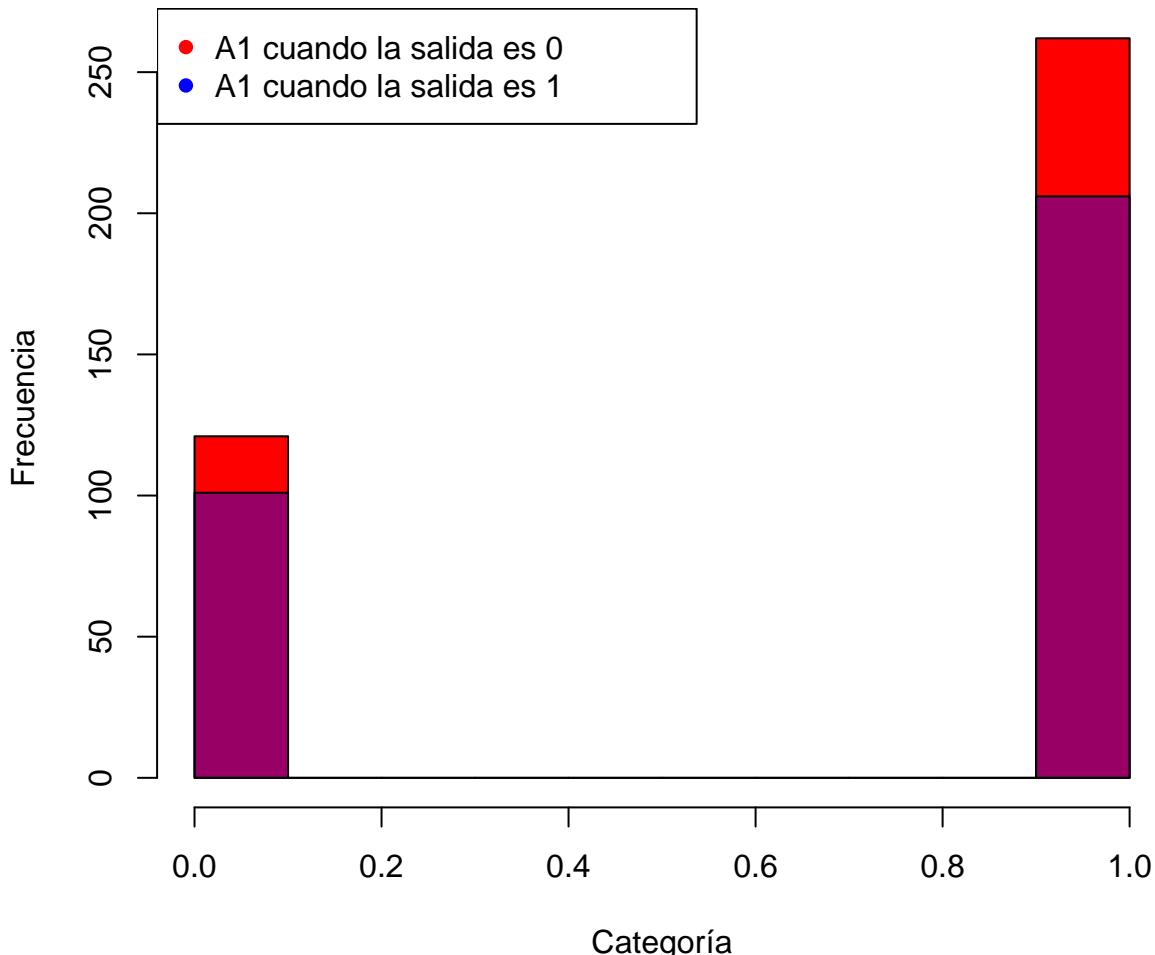
```
"3rd Qu.:1.0000  " "3rd Qu.:1.000  " "3rd Qu.:2.000  "
"Max.    :1.0000  " "Max.    :1.000  " "Max.    :3.000  "
"0"          "0"          "2"
```

Las variables 4 y 12 son las únicas variables que tienen más de dos categorías por lo que cobra más importancia los cuartiles. De hecho en estas variables son las que la categoría 2 aparece tanto en el primer cuartil como en el 3 lo que implica que al menos el 50% de las observaciones son de esta categoría, convirtiéndose así en el valor más frecuente, es decir en la moda. En el resto de casos la moda podría haber sido sustituida perfectamente por la mediana, puesto que solo tienen dos categorías aquella que obtenga el puesto que deje la mitad de las observaciones ordenadas por debajo suya es el más frecuente.

La información anterior nos ilustra como se distribuye cada una de las variables pero no como se relacionan con la salida, para ello dibujaremos gráficos en donde comparamos la frecuencia de cada valor con respecto al valor que toma la salida.

```
hist(australian[which(australian[, 15] == 0), 1], col = rgb(1,
  0, 0, 1), main = "Frecuencia de la variable A1", ylab = "Frecuencia",
  xlab = "Categoría")
hist(australian[which(australian[, 15] == 1), 1], col = rgb(0,
  0, 1, 0.4), add = TRUE)
legend("topleft", legend = c("A1 cuando la salida es 0", "A1 cuando la salida es 1"),
  text.width = 0.5, col = c("red", "blue"), pch = 16)
```

### Frecuencia de la variable A1

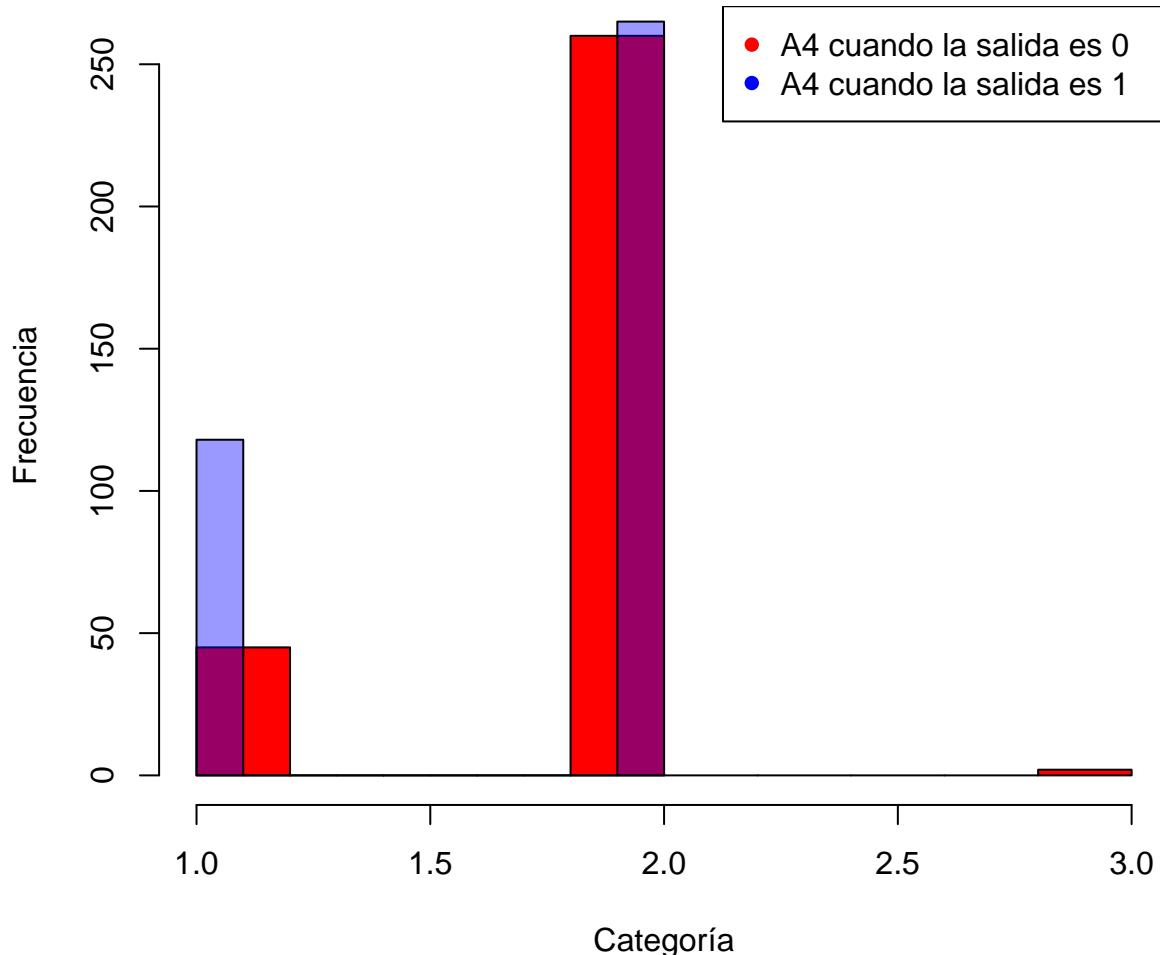


```

hist(australian[which(australian[, 15] == 1), 4], col = rgb(1,
  0, 0, 1), main = "Frecuencia de la variable A4", ylab = "Frecuencia",
  xlab = "Categoría")
hist(australian[which(australian[, 15] == 0), 4], col = rgb(0,
  0, 1, 0.4), add = T)
legend("topright", legend = c("A4 cuando la salida es 0", "A4 cuando la salida es 1"),
  text.width = 0.8, col = c("red", "blue"), pch = 16)

```

## Frecuencia de la variable A4

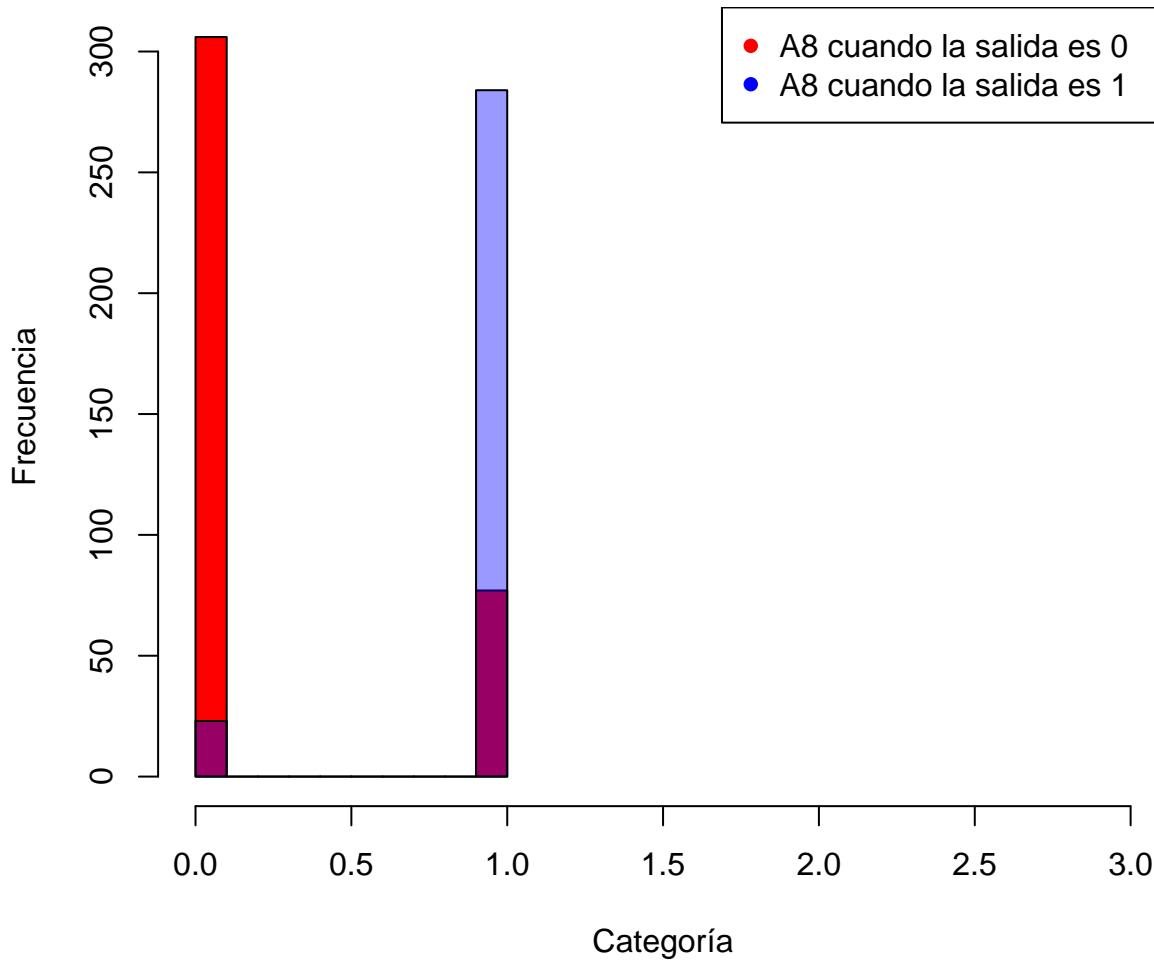


```

hist(australian[which(australian[, 15] == 0), 8], col = rgb(1,
  0, 0, 1), main = "Frecuencia de la variable A8", ylab = "Frecuencia",
  xlab = "Categoría", xlim = c(0, 3))
hist(australian[which(australian[, 15] == 1), 8], col = rgb(0,
  0, 1, 0.4), add = T)
legend("topright", legend = c("A8 cuando la salida es 0", "A8 cuando la salida es 1"),
  text.width = 1.2, col = c("red", "blue"), pch = 16)

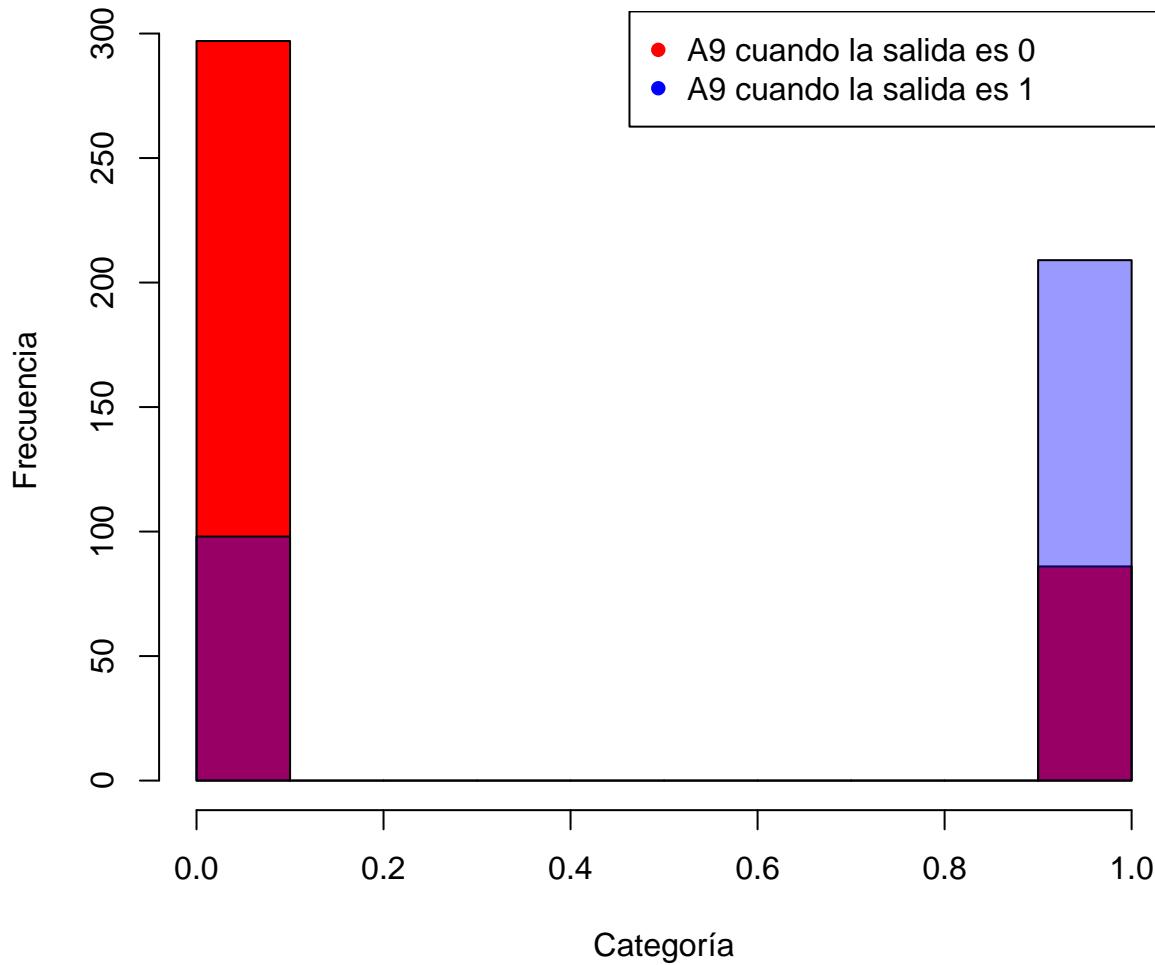
```

## Frecuencia de la variable A8



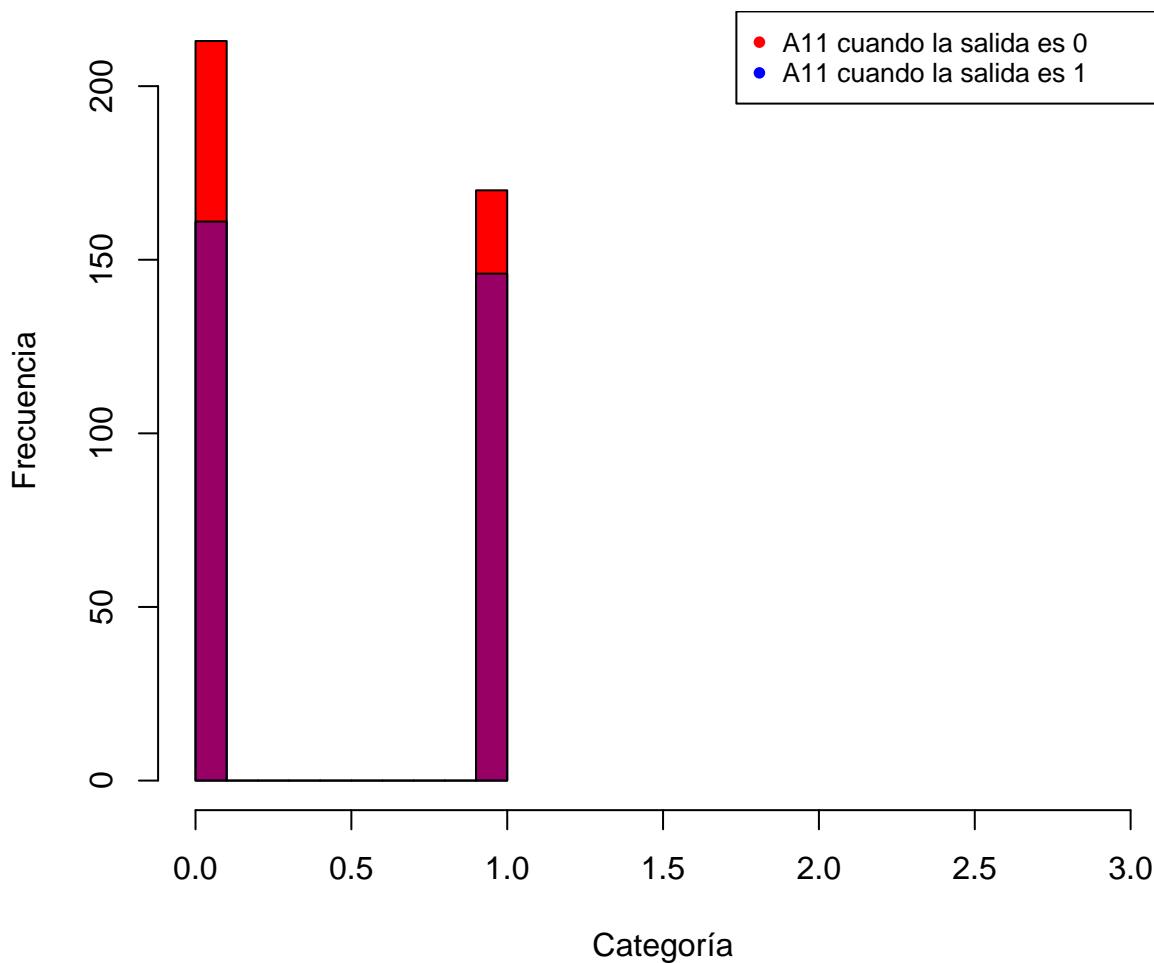
```
hist(australian[which(australian[, 15] == 0), 9], col = rgb(1,
0, 0, 1), main = "Frecuencia de la variable A9", ylab = "Frecuencia",
xlab = "Categoría")
hist(australian[which(australian[, 15] == 1), 9], col = rgb(0,
0, 1, 0.4), add = T)
legend("topright", legend = c("A9 cuando la salida es 0", "A9 cuando la salida es 1"),
text.width = 0.5, col = c("red", "blue"), pch = 16)
```

## Frecuencia de la variable A9



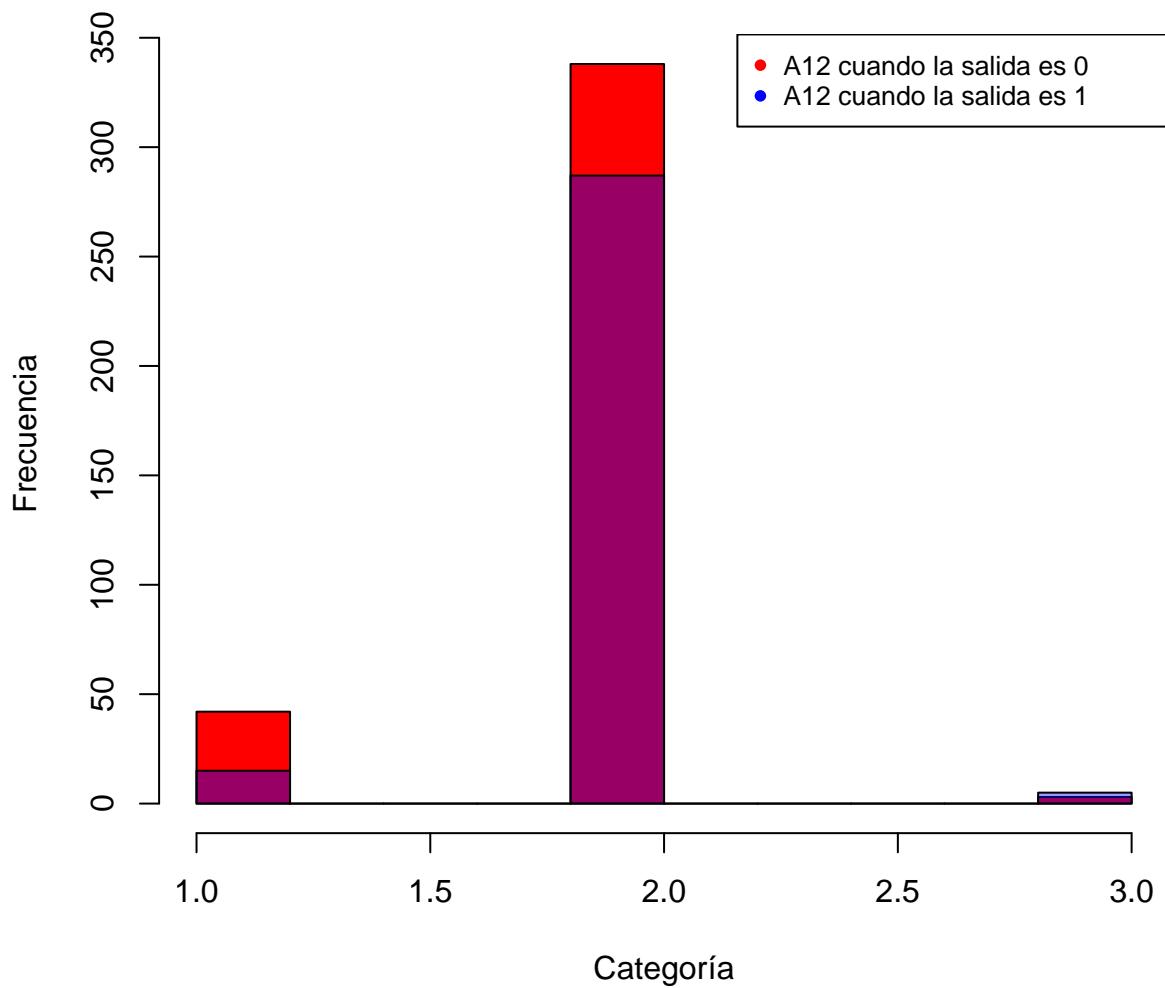
```
hist(australian[which(australian[, 15] == 0), 11], col = rgb(1,
  0, 0, 1), main = "Frecuencia de la variable A11", ylab = "Frecuencia",
  xlab = "Categoría", xlim = c(0, 3))
hist(australian[which(australian[, 15] == 1), 11], col = rgb(0,
  0, 1, 0.4), add = T)
legend("topright", legend = c("A11 cuando la salida es 0", "A11 cuando la salida es 1"),
  text.width = 1.2, col = c("red", "blue"), pch = 16, cex = 0.8)
```

## Frecuencia de la variable A11



```
hist(australian[which(australian[, 15] == 0), 12], col = rgb(1,
  0, 0, 1), main = "Frecuencia de la variable A12", ylab = "Frecuencia",
  xlab = "Categoría")
hist(australian[which(australian[, 15] == 1), 12], col = rgb(0,
  0, 1, 0.4), add = T)
legend("topright", legend = c("A12 cuando la salida es 0", "A12 cuando la salida es 1"),
  text.width = 0.8, col = c("red", "blue"), pch = 16, cex = 0.8)
```

## Frecuencia de la variable A12



Viendo las gráficas anteriores, vemos que la gran mayoría de las variables no discriminan bien la salida, pero las variables A8 y A9 si que discriminan bien la mayoría de los casos.

### Conclusiones del estudio de Australian

Tras este estudio podemos concluir que:

- \* Pueden actuar como buenos discriminantes las variables categóricas A8 y A9 así como la variable A14.
- \* A primera vista no esperaría buenos resultados de ningún modelo, excepto aquellos que generados por el algoritmo de los K vecinos más cercanos (Knn) que al ajustarse mejor a los datos puede ser más flexible.
- \* Los modelos generados mediante la técnica de LDA que cuenten con las variables A3,A5 y A10 pueden dar buenos resultados, debido a que cumplen uno de los dos requisitos que se piden: las varianzas de los conjuntos de la salida son iguales.

## Wizmir (Weather of Izmir)

La base de datos de *Weather of Izmir* contiene datos referentes a distintas variables relacionadas con el tiempo atmosférico, a saber:

- Temperatura máxima (Max\_temperature) real[36.7,105.0]
- Temperatura mínima (Min\_temperature) real[15.8,78.6]
- Rocío (Dewpoint) real[13.6,64.4]
- Precipitación (Precipitation) real[0.0,7.6]
- Presión a nivel del mar (Sea\_level\_pressure) real[29.26,30.48]
- Presión normal (Standard\_pressure) real[2.3,10.1]
- Visibilidad (Visibility) real[0.92,29.1]
- Velocidad del viento (Wind\_speed) real[4.72,68.8]
- Velocidad máxima del viento (Max\_wind\_speed) real[16.11,55.24]
- Temperatura media (Mean\_temperature) real[29.4,89.9]

El objetivo con esta base de datos es calcular la temperatura media a partir de las demás variables de datos.

Al contrario que el conjunto de datos anterior, los nombres de las variables son descriptivos lo que nos permite formular hipótesis antes de visualizar los datos.

## Hipótesis previas

Las hipótesis previas nos permiten una primera aproximación del modelo de datos, nos permite crear los primeros modelos a partir de los cuales iterar para obtener un mejor resultado.

1. La temperatura media es un modelo lineal en el que intervienen la temperatura mínima y la temperatura máxima. Posiblemente  $0.5 \times \text{temperatura mínima} + 0.5 \times \text{temperatura máxima}$ , por la propia definición de media.
2. Por la ley física que relaciona la temperatura y la presión, a mayor presión mayor temperatura. No se espera que esta ley se cumpla por completo ya que está estipulada para gases de volumen constante, por esto tanto la presión como la presión a nivel del mar tienen que ver en cierta medida con la temperatura.
3. La velocidad del viento y la velocidad máxima del mismo, no intervienen o lo hacen en una medida despreciable, puesto que son factores que intervienen más en la sensación térmica que en la propia temperatura.
4. El rocío y las precipitaciones, tienen que ver más como consecuencia de la temperatura que como factor generador de la temperatura, no se descarta su intervención pero se espera que sea mínima.
5. Sobre la visibilidad, no sabemos qué comportamiento tendrá puesto que en ocasiones hay poca visibilidad a causa de bancos de niebla que se generan por las bajas temperaturas, pero en ciertas zonas del planeta puede ser por polvo en suspensión traído por el viento, que genera que haya mayor temperatuda, por ello como actuará esta variable en el modelo es todo un misterio.

Ahora con estas hipótesis previas, procedemos a estudiar las variables, que en este caso son todas numéricas, con respecto de la salida.

## Variables numéricas.

En primer lugar vamos a cargar el dataset.

```
wizmir <- read.csv("./WizmirRegression/wizmir/wizmir.dat", header = FALSE,  
  comment.char = "@")  
names(wizmir) <- c("Max_temperature", "Min_temperature", "Dewpoint",  
  "Precipitation", "Sea_level_pressure", "Standar _pressure",  
  "Visibility", "Wind_speed", "Wind_max_speed", "Mean_temperature")
```

Tras ello vamos a obtener el número de valores perdidos que tenemos en el conjunto de datos.

### Estudio de los valores perdidos

```
wizmir_na <- apply(is.na(wizmir), 2, sum)  
wizmir_na
```

Max_temperature	Min_temperature	Dewpoint
0	0	0
Precipitation	Sea_level_pressure	Standar _pressure
0	0	0
Visibility	Wind_speed	Wind_max_speed
0	0	0
Mean_temperature		
0		

Esta base de datos no tiene valores perdidos, lo que nos facilita el trabajo.

### Test de normalidad

En este caso, el problema no es un problema de clasificación por lo que los algoritmos no hacen suposiciones de los datos de entrada como el caso anterior que necesitaba que los datos estuvieran normalmente distribuidos ni que las varianzas de los conjuntos de salida fueran iguales, por ello no vamos a realizar un estudio de la normalidad de los datos ni de igualdad de las varianzas.

### Estudio de los principales estadísticos

La obtención de los principales valores estadísticos la obtenemos, como hemos visto en los casos anteriores, mediante la función *summary*.

```
summary(wizmir)
```

Max_temperature	Min_temperature	Dewpoint
Min. : 36.70	Min. :15.80	Min. :13.60
1st Qu.: 59.00	1st Qu.:40.10	1st Qu.:41.30
Median : 70.70	Median :50.00	Median :48.20
Mean : 72.22	Mean :50.74	Mean :46.62
3rd Qu.: 87.10	3rd Qu.:62.20	3rd Qu.:53.60
Max. :105.00	Max. :78.60	Max. :64.40
Precipitation	Sea_level_pressure	Standar _pressure
Min. : 0.00000	Min. :29.26	Min. : 2.300
1st Qu.: 0.00000	1st Qu.:29.85	1st Qu.: 7.100
Median : 0.00000	Median :29.95	Median : 7.300
Mean : 0.09257	Mean :29.97	Mean : 7.197
3rd Qu.: 0.00000	3rd Qu.:30.08	3rd Qu.: 7.600
Max. : 7.60000	Max. :30.48	Max. :10.100

```

Visibility      Wind_speed      Wind_max_speed
Min.    : 0.92   Min.    : 4.72   Min.    :16.11
1st Qu.: 6.56   1st Qu.:16.10   1st Qu.:34.28
Median  :10.50   Median  :19.81   Median  :34.28
Mean    :11.16   Mean    :19.81   Mean    :34.28
3rd Qu.:15.40   3rd Qu.:23.00   3rd Qu.:34.28
Max.    :29.10   Max.    :68.80   Max.    :55.24
Mean_temperature
Min.    :29.40
1st Qu.:49.60
Median  :60.00
Mean    :61.51
3rd Qu.:75.20
Max.    :89.90

```

Exceptuando las variables de precipitación (Precipitation) y la presión normal (Standar\_pressure) podemos decir que todas las variables se mueven en el mismo rango de 20-100 aproximadamente por lo que si se generase un modelo que no tuviese dichas variables podríamos evitar el paso intermedio de normalizar las variables para igualar el rango.

Nos queda por conocer la desviación estándar de los datos:

```
wizmir_std <- apply(wizmir[, ], 2, sd)
wizmir_std
```

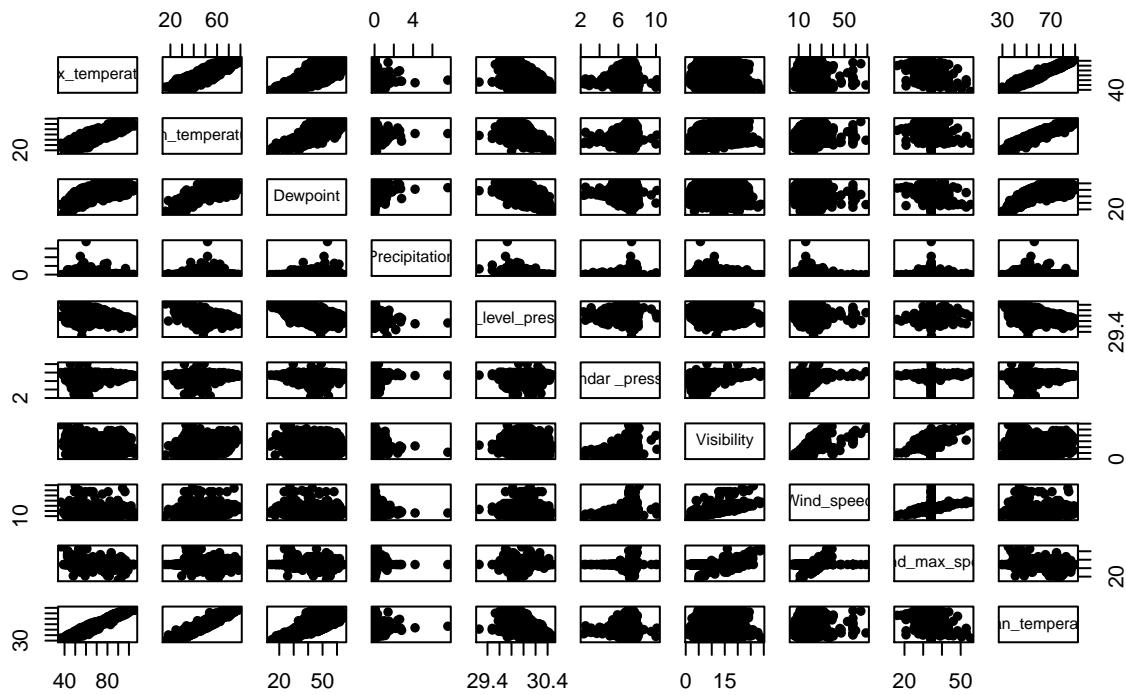
Max_temperature	Min_temperature	Dewpoint
15.9267131	13.2260798	9.3445446
Precipitation	Sea_level_pressure	Standar _pressure
0.3528008	0.1676890	0.6849532
Visibility	Wind_speed	Wind_max_speed
5.4066647	7.1352505	2.4418256
Mean_temperature		
14.3762319		

Viendo las desviaciones típicas de la temperatura mínima y máxima tener un valor tan cercano al de la temperatura media, que es el valor que tenemos que obtener, hace pensar que podrían compartir distribución.

Si representamos el valor de todas las variables con respecto de la salida obtenemos los siguientes gráficos

```
plot(wizmir, main = "Relación de las variables entre ellas",
     pch = 16)
```

## Relación de las variables entre ellas



Primero, nos vamos a concentrar en la última fila de la ilustración anterior, en ella están reflejadas todas las variables de entrada en el eje horizontal y la variable de salida en el eje vertical. En esta fila podemos ver que las variables de temperatura máxima y mínima tienen una relación lineal con la temperatura media, como era de esperar por la hipótesis 1, pero además esta tendencia lineal también la tiene la variable de rocío que puede ser debido a que el rocío es dependiente de la temperatura mínima, como se ve en la relación entre estas dos variables (y en la relación de la variable rocío con la temperatura máxima también) pero es un detalle a tener en cuenta a la hora de elaborar un modelo.

Otras variables que podrían intervenir en el modelo de una forma, no tan claramente lineal o incluso de orden superior, son la presión a nivel del mar y la velocidad máxima del viento, puesto que dentro de la nube de puntos se puede dibujar una recta decreciente e incluso una curva.

El resto de variables la nube de puntos tiene tal dispersión que podrían no intervenir porque no se ve una función definida que encaje con los datos.

### Estudio de correlación

Una forma de comprobar que la variable *Dewpoint* o punto de rocío está relacionada con la temperatura mínima y/o máxima está en la tabla de correlaciones:

```
cor(wizmir[, -10], method = "pearson")
```

	Max_temperature	Min_temperature
Max_temperature	1.0000000	0.89611403
Min_temperature	0.89611403	1.0000000
Dewpoint	0.74937745	0.78634048
Precipitation	-0.19713825	-0.07922557
Sea_level_pressure	-0.51077796	-0.62301824
Standar_press	0.09967300	0.17983060
Visibility	0.11244011	0.35256802
Wind_speed	0.08825536	0.25255985

Wind_max_speed	-0.14522657	-0.08434084
Dewpoint Precipitation		
Max_temperature	0.74937745	-0.19713825
Min_temperature	0.78634048	-0.07922557
Dewpoint	1.00000000	0.02005389
Precipitation	0.02005389	1.00000000
Sea_level_pressure	-0.58739019	-0.15185635
Standar_pressure	0.04351714	0.06492428
Visibility	-0.05894845	-0.04562776
Wind_speed	-0.05782537	-0.01743911
Wind_max_speed	-0.12906064	0.02320726
Sea_level_pressure Standar_pressure		
Max_temperature	-0.51077796	0.09967300
Min_temperature	-0.62301824	0.17983060
Dewpoint	-0.58739019	0.04351714
Precipitation	-0.15185635	0.06492428
Sea_level_pressure	1.00000000	-0.18339796
Standar_pressure	-0.18339796	1.00000000
Visibility	-0.20288466	0.30903837
Wind_speed	-0.19647381	0.26081990
Wind_max_speed	0.04049549	0.03857268
Visibility Wind_speed Wind_max_speed		
Max_temperature	0.11244011	0.08825536
Min_temperature	0.35256802	0.25255985
Dewpoint	-0.05894845	-0.05782537
Precipitation	-0.04562776	-0.01743911
Sea_level_pressure	-0.20288466	-0.19647381
Standar_pressure	0.30903837	0.26081990
Visibility	1.00000000	0.76428686
Wind_speed	0.76428686	1.00000000
Wind_max_speed	0.23699238	0.21889408
		1.00000000

Las correlaciones que podemos ver del cuadrante anterior son:

- La temperatura mínima y máxima están correlacionadas entre ellas.
- El punto de rocío está relacionado tanto con las temperaturas máxima y mínima y, en menor medida, y de forma inversa, con la presión a nivel del mar.
- Tanto la temperatura máxima como mínima están relacionadas de forma inversa con la presión atmosférica a nivel del mar.
- La visibilidad y la presión estándar tienen una ligera correlación.
- La visibilidad y la velocidad del viento están fuertemente correlacionadas.

## Conclusión

Sobre este dataset se podría decir:

- El mejor modelo para predecir la temperatura media es un modelo de regresión lineal donde intervengan las variables de temperatura máxima y mínima.
- Habría que estudiar si añadir las variables de rocío y presión a nivel del mar supone alguna mejora.

# Regresión

La regresión es el proceso estadístico por el que se estiman la relación entre una o varias variables independientes o predictoras y la variable dependiente que se quiere predecir, dado un conjunto de datos de entrada. El objetivo de la regresión es obtener un modelo que permita predecir o estimar el valor que tendrá un nuevo dato a su salida, siendo este dato distinto de todos los anteriores que se usaron para crear el modelo.

Existen varios algoritmos para realizar regresión, pero en este trabajo solamente utilizaremos la regresión lineal simple, la regresión lineal múltiple y el algoritmo de los k- vecinos más cercanos.

## Problema a tratar

Obtener un modelo de regresión que nos permita predecir la temperatura media de la ciudad de Izmir(Turquía) usando las técnicas de regresión lineal, los k-vecinos más cercanos en conjunto con la validación cruzada.

## Regresores elegidos

En el apartado de análisis de datos anterior pudimos ver que los mejores regresores para este problema son las variables:

- Temperatura máxima
- Temperatura mínima

Pero vamos a realizar una prueba de regresión lineal simple con cinco regresores así que añadiremos a la lista:

- Punto de rocío
- Presión a nivel del mar
- Visibilidad

## Modelo de regresión simple

Con el modelo de regresión simple comprobamos que los datos se pueden explicar de la forma de una línea recta usando solamente una variable de predicción. La bondad de este modelo se puede medir por dos coeficientes:  $R^2$  y el RMSE. El primer parámetro nos lo realizar la operación *summary* sobre el modelo obtenido, pero el RMSE es preferible calcularlo a mano por ello definimos la siguiente función *rmse*.

```
rmse <- function(original, algorithm_output) {  
  sqrt(sum((original - algorithm_output)^2)/length(algorithm_output))  
}
```

Los modelos los crearemos haciendo uso de todos los datos del conjunto de datos

```
lmModel_maxTemp = lm(wizmir$Mean_temperature ~ wizmir$Max_temperature)  
lmModel_minTemp = lm(wizmir$Mean_temperature ~ wizmir$Min_temperature)  
lmModel_dewPoint = lm(wizmir$Mean_temperature ~ wizmir$Dewpoint)  
lmModel_seaPressure = lm(wizmir$Mean_temperature ~ wizmir$Sea_level_pressure)  
lmModel_visibility = lm(wizmir$Mean_temperature ~ wizmir$Visibility)
```

Ahora obtendremos los dos parámetros de comparación:

```
summary_maxTemp = summary(lmModel_maxTemp)  
rSquared_maxTemp = summary_maxTemp$r.squared  
adjusted_r_maxTemp = summary_maxTemp$adj.r.squared  
rmse_maxTemp = rmse(wizmir$Mean_temperature, lmModel_maxTemp$fitted.values)
```

```

summary_minTemp = summary(lmModel_minTemp)
rSquared_minTemp = summary_minTemp$r.squared
adjusted_r_minTemp = summary_minTemp$adj.r.squared
rmse_minTemp = rmse(wizmir$Mean_temperature, lmModel_minTemp$fitted.values)

summary_dewPoint = summary(lmModel_dewPoint)
rSquared_dewPoint = summary_dewPoint$r.squared
adjusted_r_dewPoint = summary_dewPoint$adj.r.squared
rmse_dewPoint = rmse(wizmir$Mean_temperature, lmModel_dewPoint$fitted.values)

summary_seaPress = summary(lmModel_seaPressure)
rSquared_seaPress = summary_seaPress$r.squared
adjusted_r_seaPress = summary_seaPress$adj.r.squared
rmse_seaPress = rmse(wizmir$Mean_temperature, lmModel_seaPressure$fitted.values)

summary_visibility = summary(lmModel_visibility)
rSquared_visibility = summary_visibility$r.squared
adjusted_r_visibility = summary_visibility$adj.r.squared
rmse_visibility = rmse(wizmir$Mean_temperature, lmModel_visibility$fitted.values)

rSquared = c(rSquared_maxTemp, rSquared_minTemp, rSquared_dewPoint,
            rSquared_seaPress, rSquared_visibility)
adjRSquared = c(adjusted_r_maxTemp, adjusted_r_minTemp, adjusted_r_dewPoint,
               adjusted_r_seaPress, adjusted_r_visibility)
rmse_lm_results = c(rmse_maxTemp, rmse_minTemp, rmse_dewPoint,
                     rmse_seaPress, rmse_visibility)

linealSimpleModels_comparisonDataFrame = data.frame(rSquared,
                                                    adjRSquared, rmse_lm_results)
row.names(linealSimpleModels_comparisonDataFrame) <- c("MaxTemp",
                                                       "MinTemp", "DewPoint", "SeaPressure", "Visibility")
linealSimpleModels_comparisonDataFrame

      rSquared adjRSquared rmse_lm_results
MaxTemp    0.95764878   0.95761975     2.957531
MinTemp    0.91902245   0.91896695     4.089580
DewPoint    0.61549043   0.61522688     8.911483
SeaPressure 0.33981129   0.33935879    11.676977
Visibility   0.05107395   0.05042356    13.999502

```

Si miramos la columna de la R<sup>2</sup> ( *rSquared* ) de esta tabla podemos ver que la temperatura máxima es el mejor regresor del conjunto puesto con un modelo lineal simple explica el 95 % de los casos, seguido de cerca por la temperatura mínima que explica el 91 % de los casos. El resto de regresores no son tan buenos.

La columna de la R<sup>2</sup> ajustada en este caso no tiene mucho sentido, puesto que es el parámetro de la R<sup>2</sup> ajustada al número de variables del modelo que en este caso es 1, pero se calcula para después poder comparar con el modelo de regresión múltiple.

El RMSE crece a medida que el modelo empeora su predicción de la salida, siendo de prácticamente 3 para el caso de la temperatura máxima y de prácticamente 14 para el caso de la visibilidad que solamente explica el 5 % de los casos.

## Modelo de regresión lineal múltiple

En la regresión lineal múltiple, se emplea un modelo matemático en el que intervienen varias variables a la vez, creando un hiperplano.

En esta sección crearemos modelos de regresión lineal múltiple con la esperanza de mejorar el modelo lineal simple anterior.

El primero que desarrollaremos intervienen únicamente las variables temperatura máxima y temperatura mínima, tal y como se propuso en la sección de hipótesis previas.

```
lmmModel_maxMinTemp = lm(wizmir$Mean_temperature ~ wizmir$Max_temperature +
  wizmir$Min_temperature)
rmse_lmm_maxMin = rmse(wizmir$Mean_temperature, lmmModel_maxMinTemp$fitted.values)
summary_lmm_maxMin = summary(lmmModel_maxMinTemp)
c(summary_lmm_maxMin$r.squared, summary_lmm_maxMin$adj.r.squared,
  rmse_lmm_maxMin)
```

```
[1] 0.9915545 0.9915430 1.3207111
```

```
summary_lmm_maxMin
```

Call:

```
lm(formula = wizmir$Mean_temperature ~ wizmir$Max_temperature +
  wizmir$Min_temperature)
```

Residuals:

Min	1Q	Median	3Q	Max
-8.3783	-0.7617	-0.0479	0.7314	7.3927

Coefficients:

	Estimate	Std. Error	t value
(Intercept)	-0.934295	0.161645	-5.78
wizmir\$Max_temperature	0.547739	0.004895	111.90
wizmir\$Min_temperature	0.450962	0.005894	76.51
	Pr(> t )		
(Intercept)	9.13e-09	***	
wizmir\$Max_temperature	< 2e-16	***	
wizmir\$Min_temperature	< 2e-16	***	

---

Signif. codes:

```
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 1.322 on 1458 degrees of freedom

Multiple R-squared: 0.9916, Adjusted R-squared: 0.9915

F-statistic: 8.559e+04 on 2 and 1458 DF, p-value: < 2.2e-16

```
comparisonDF = rbind(linealSimpleModels_comparisonDataFrame,
  c(summary_lmm_maxMin$r.squared, summary_lmm_maxMin$adj.r.squared,
    rmse_lmm_maxMin))
```

Viendo los valores de  $R^2$  y  $R^2$  ajustado de 0.99 así como la reducción a menos de la mitad del RMSE, creo que será muy difícil encontrar un modelo capaz de mejorar este. Aún así, siguiendo el enunciado de la actividad elaboramos un modelo de regresión lineal múltiple en donde intervienen de forma conjunta los regresores de los modelos de regresión lineal.

```

lmmModel_simpleSubSet = lm(wizmir$Mean_temperature ~ wizmir$Max_temperature +
  wizmir$Min_temperature + wizmir$Dewpoint + wizmir$Sea_level_pressure +
  wizmir$Visibility)
summary_lmm_simpleSubset = summary(lmmModel_simpleSubSet)
rmse_lmm_simpleSubset = rmse(wizmir$Mean_temperature, lmmModel_simpleSubSet$fitted.values)
comparisonDF = rbind(comparisonDF, c(summary_lmm_simpleSubset$r.squared,
  summary_lmm_simpleSubset$adj.r.squared, rmse_lmm_simpleSubset))
comparisonDF

```

	rSquared	adjRSquared	rmse_lm_results
MaxTemp	0.95764878	0.95761975	2.957531
MinTemp	0.91902245	0.91896695	4.089580
DewPoint	0.61549043	0.61522688	8.911483
SeaPressure	0.33981129	0.33935879	11.676977
Visibility	0.05107395	0.05042356	13.999502
6	0.99155455	0.99154296	1.320711
7	0.99234370	0.99231739	1.257494

La introducción de las tres variables del punto de rocío(modelo notado con el número 7), la presión al nivel del mar y la visibilidad solamente mejora el modelo en una décima del parámetro de R^2 ajustado con respecto al modelo que solamente tiene la temperatura máxima y mínima, por lo que ante la decisión de elegir este modelo u el otro preferiría el anterior por simplicidad.

### Modelos de regresión lineal múltiple con interacciones.

Ante los resultados del análisis de correlación tenemos que la temperatura mínima y máxima están muy correladas, por lo que vamos a crear un modelo de regresión lineal múltiple con una iteración entre ellas.

```

lmmInnt = lm(wizmir$Mean_temperature ~ wizmir$Max_temperature *
  wizmir$Min_temperature)
summary_lmm_Innt = summary(lmmInnt)
rmse_lmm_Innt = rmse(wizmir$Mean_temperature, lmmInnt$fitted.values)
comparisonDF = rbind(comparisonDF, c(summary_lmm_Innt$r.squared,
  summary_lmm_Innt$adj.r.squared, rmse_lmm_Innt))
summary_lmm_Innt

```

Call:

```
lm(formula = wizmir$Mean_temperature ~ wizmir$Max_temperature *
  wizmir$Min_temperature)
```

Residuals:

Min	1Q	Median	3Q	Max
-8.3212	-0.7603	-0.0658	0.7496	7.2755

Coefficients:

	Estimate
(Intercept)	0.1869122
wizmir\$Max_temperature	0.5315974
wizmir\$Min_temperature	0.4277485
wizmir\$Max_temperature:wizmir\$Min_temperature	0.0003172
	Std. Error
(Intercept)	0.7085095
wizmir\$Max_temperature	0.0110708

```

wizmir$Min_temperature          0.0154499
wizmir$Max_temperature:wizmir$Min_temperature 0.0001952
t value
(Intercept)                   0.264
wizmir$Max_temperature        48.018
wizmir$Min_temperature         27.686
wizmir$Max_temperature:wizmir$Min_temperature 1.625
Pr(>|t|)
(Intercept)                   0.792
wizmir$Max_temperature        <2e-16 ***
wizmir$Min_temperature         <2e-16 ***
wizmir$Max_temperature:wizmir$Min_temperature 0.104
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Residual standard error: 1.321 on 1457 degrees of freedom  
Multiple R-squared: 0.9916, Adjusted R-squared: 0.9916  
F-statistic: 5.712e+04 on 3 and 1457 DF, p-value: < 2.2e-16

Si observamos la tabla de las variables que intervienen en el modelo, podemos ver como la interacción entre las dos variables de temperatura no es significativo para el modelo.

Además que los coeficientes de las dos temperaturas se acercan a la estimación de la hipótesis previa de que cada una de ellas intervienen en un 50%, cada una por la definición de media, pero esto ya sucedía desde el modelo de regresión lineal múltiple.

El siguiente modelo se tiene en cuenta que las variables de temperatura están correladas con el punto de rocío, y añade este último al modelo.

```

lmmInnt2 = lm(wizmir$Mean_temperature ~ wizmir$Max_temperature *
               wizmir$Min_temperature * wizmir$Dewpoint)
summary_lmm_Innt2 = summary(lmmInnt2)
rmse_lmm_Innt2 = rmse(wizmir$Mean_temperature, lmmInnt2$fitted.values)
comparisonDF = rbind(comparisonDF, c(summary_lmm_Innt2$r.squared,
                                         summary_lmm_Innt2$adj.r.squared, rmse_lmm_Innt2))
summary_lmm_Innt2

```

Call:  
lm(formula = wizmir\$Mean\_temperature ~ wizmir\$Max\_temperature \*  
wizmir\$Min\_temperature \* wizmir\$Dewpoint)

Residuals:

Min	1Q	Median	3Q	Max
-8.3764	-0.7600	-0.0192	0.7078	6.7562

Coefficients:

	Estimate
(Intercept)	1.014e+01
wizmir\$Max_temperature	2.336e-01
wizmir\$Min_temperature	2.482e-01
wizmir\$Dewpoint	-2.895e-03
wizmir\$Max_temperature:wizmir\$Min_temperature	5.287e-03
wizmir\$Max_temperature:wizmir\$Dewpoint	3.802e-03
wizmir\$Min_temperature:wizmir\$Dewpoint	-2.791e-04

```
wizmir$Max_temperature:wizmir$Min_temperature:wizmir$Dewpoint -5.667e-05
                                         Std. Error
(Intercept)                               3.049e+00
wizmir$Max_temperature                     5.523e-02
wizmir$Min_temperature                     7.484e-02
wizmir$Dewpoint                           6.649e-02
wizmir$Max_temperature:wizmir$Min_temperature 1.063e-03
wizmir$Max_temperature:wizmir$Dewpoint    1.120e-03
wizmir$Min_temperature:wizmir$Dewpoint    1.526e-03
wizmir$Max_temperature:wizmir$Min_temperature:wizmir$Dewpoint 2.065e-05
                                         t value
(Intercept)                               3.327
wizmir$Max_temperature                     4.229
wizmir$Min_temperature                     3.316
wizmir$Dewpoint                           -0.044
wizmir$Max_temperature:wizmir$Min_temperature 4.973
wizmir$Max_temperature:wizmir$Dewpoint    3.395
wizmir$Min_temperature:wizmir$Dewpoint    -0.183
wizmir$Max_temperature:wizmir$Min_temperature:wizmir$Dewpoint -2.745
                                         Pr(>|t|)
(Intercept)                               0.000899
wizmir$Max_temperature                     2.50e-05
wizmir$Min_temperature                     0.000936
wizmir$Dewpoint                           0.965279
wizmir$Max_temperature:wizmir$Min_temperature 7.38e-07
wizmir$Max_temperature:wizmir$Dewpoint    0.000704
wizmir$Min_temperature:wizmir$Dewpoint    0.854870
wizmir$Max_temperature:wizmir$Min_temperature:wizmir$Dewpoint 0.006128

                                         ***
wizmir$Max_temperature                     ***
wizmir$Min_temperature                     ***
wizmir$Dewpoint                           ***
wizmir$Max_temperature:wizmir$Min_temperature  ***
wizmir$Max_temperature:wizmir$Dewpoint    ***
wizmir$Min_temperature:wizmir$Dewpoint    ***
wizmir$Max_temperature:wizmir$Min_temperature:wizmir$Dewpoint **

---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 1.287 on 1453 degrees of freedom  
Multiple R-squared: 0.992, Adjusted R-squared: 0.992  
F-statistic: 2.581e+04 on 7 and 1453 DF, p-value: < 2.2e-16

Como se muestra en el resumen, la variable de punto de rocío, no es significante en el modelo al igual que su interacción con la variable punto de rocío. Por su lado, la interacción entre las tres variables es menos significante que el resto de variables.

Viendo estos modelos con interacción, la afirmación anterior de que será difícil encontrar un modelo que mejore el modelo lineal múltiple en el que intervienen las dos temperaturas se mantiene cierta.

## Modelos no lineales.

Si volvemos amirar las representaciones de la temepratura media con respecto al punto de rocío y a la presión a nivel del mar podemos dudar de que el modelo que sigue es un modelo lineal o un modelo de orden dos o modelo cuadrático. Por ello vamos a probar modelos en los que se le añaden estas dos variables al modelo lineal múltiple de las dos variables de temperatura, para ello vamos a hacer uso de la función *poly* al que le indicamos la variable y el orden al que queremos el polinomio y esta función nos añadirá al modelo el término especificado y todos aquellos que tienen orden inferior.

```
lmmNL = lm(wizmir$Mean_temperature ~ wizmir$Max_temperature +
            wizmir$Min_temperature + poly(wizmir$Dewpoint, 2))
summary_lmm_NL = summary(lmmNL)
rmse_lmm_NL = rmse(wizmir$Mean_temperature, lmmNL$fitted.values)
comparisonDF = rbind(comparisonDF, c(summary_lmm_NL$r.squared,
                                         summary_lmm_NL$adj.r.squared, rmse_lmm_NL))
summary_lmm_NL
```

Call:

```
lm(formula = wizmir$Mean_temperature ~ wizmir$Max_temperature +
    wizmir$Min_temperature + poly(wizmir$Dewpoint, 2))
```

Residuals:

Min	1Q	Median	3Q	Max
-8.4415	-0.7586	-0.0240	0.7233	7.3641

Coefficients:

	Estimate	Std. Error	t value
(Intercept)	-0.553902	0.246844	-2.244
wizmir\$Max_temperature	0.545785	0.004945	110.374
wizmir\$Min_temperature	0.446246	0.006407	69.652
poly(wizmir\$Dewpoint, 2)1	4.721006	2.173611	2.172
poly(wizmir\$Dewpoint, 2)2	-3.200953	1.329550	-2.408

Pr(>|t|)

(Intercept)	0.0250 *
wizmir\$Max_temperature	<2e-16 ***
wizmir\$Min_temperature	<2e-16 ***
poly(wizmir\$Dewpoint, 2)1	0.0300 *
poly(wizmir\$Dewpoint, 2)2	0.0162 *

---

Signif. codes:

0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.318 on 1456 degrees of freedom  
Multiple R-squared: 0.9916, Adjusted R-squared: 0.9916  
F-statistic: 4.308e+04 on 4 and 1456 DF, p-value: < 2.2e-16

Este modelo no supone ninguna diferencia con respecto a la modelo lineal múltiple con las dos temperaturas sin el punto de rocío. Por ello vamos a probar con un modelo de grado dos con la temperatura a nivel del mar.

```
lmmNL2 = lm(wizmir$Mean_temperature ~ wizmir$Max_temperature +
             wizmir$Min_temperature + poly(wizmir$Sea_level_pressure,
             2))
summary_lmm_NL2 = summary(lmmNL2)
rmse_lmm_NL2 = rmse(wizmir$Mean_temperature, lmmNL2$fitted.values)
comparisonDF = rbind(comparisonDF, c(summary_lmm_NL2$r.squared,
                                         summary_lmm_NL2$adj.r.squared, rmse_lmm_NL2))
summary_lmm_NL2
```

Call:

```
lm(formula = wizmir$Mean_temperature ~ wizmir$Max_temperature +
    wizmir$Min_temperature + poly(wizmir$Sea_level_pressure,
    2))
```

Residuals:

Min	1Q	Median	3Q	Max
-8.4715	-0.7277	-0.0562	0.7109	6.9177

Coefficients:

	Estimate	Std. Error
(Intercept)	-0.082838	0.185368
wizmir\$Max_temperature	0.551783	0.004830
wizmir\$Min_temperature	0.428425	0.006379
poly(wizmir\$Sea_level_pressure, 2)1	-13.807703	1.670228
poly(wizmir\$Sea_level_pressure, 2)2	-4.810706	1.308014
	t value	Pr(> t )
(Intercept)	-0.447	0.655023
wizmir\$Max_temperature	114.239	< 2e-16 ***
wizmir\$Min_temperature	67.161	< 2e-16 ***
poly(wizmir\$Sea_level_pressure, 2)1	-8.267	3.06e-16 ***
poly(wizmir\$Sea_level_pressure, 2)2	-3.678	0.000244 ***

---

Signif. codes:

0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.289 on 1456 degrees of freedom  
Multiple R-squared: 0.992, Adjusted R-squared: 0.992  
F-statistic: 4.502e+04 on 4 and 1456 DF, p-value: < 2.2e-16

En este modelo todas las variables son significantes pero no supone ningún tipo de mejoría relevante si miramos el parámetro de la R<sup>2</sup> ajustada con respecto a los modelos anteriores.

```
row.names(comparisonDF) <- c("MaxTemp", "MinTemp", "DewPoint",
  "SeaPressure", "Visibility", "ML_MinMax", "ML_Subset", "Innt_MinXMax",
  "Innt_XDewPoint", "NL_DewPoint", "NL_SeaPress")
comparisonDF
```

	rSquared	adjRSquared	rmse_lm_results
MaxTemp	0.95764878	0.95761975	2.957531
MinTemp	0.91902245	0.91896695	4.089580
DewPoint	0.61549043	0.61522688	8.911483
SeaPressure	0.33981129	0.33935879	11.676977
Visibility	0.05107395	0.05042356	13.999502
ML_MinMax	0.99155455	0.99154296	1.320711
ML_Subset	0.99234370	0.99231739	1.257494
Innt_MinXMax	0.99156983	0.99155248	1.319515
Innt_XDewPoint	0.99202127	0.99198283	1.283699
NL_DewPoint	0.99162203	0.99159901	1.315424
NL_SeaPress	0.99197959	0.99195756	1.287048

Como se puede comprobar de la tabla anterior, el modelo lineal de la temperatura máxima es lo suficientemente bueno por si solo aunque tiene un 5 % de fallo que corrige casi por completo al añadir al modelo la temperatura mínima. Siendo este modelo el más simple que representa el 99 % de los casos.

## Modelo Knn con Cross-validation de 5-particiones

Otra técnica para elaborar un modelo de regresión es el modelo de los K vecinos más cercanos( *K nearest neighbours* o *knn*), en esta técnica se utiliza la información de los k vecinos más cercanos para dar la salida a los datos de entrada. Este modelo tiende a ajustarse a los datos cuanto mayor sea el valor de k.

Vamos a combinar esta técnica con la validación cruzada. La validación cruzada consiste en dividir los datos de entrada en n particiones, en este caso serán 5, aplicar el algoritmo seleccionando una de las particiones como test y el resto como entrenamiento para la elaboración del modelo. Una vez tenga el modelo obtendrá la bondad con la partición de test y guardará esta bondad para volver a crear el modelo a partir de una nueva selección de particiones y una nueva partición de test, y así hasta que todas las particiones hayan sido la partición de test. Entonces se hará una media de todas las bondades calculadas y se hará la media.

Para la creación de un modelo del tipo Knn necesitamos de las librerías *MASS* y *kknn* de R por lo que las cargamos ahora con los comando *require*

```
require("MASS")
require("kknn")
```

Creamos la función que nos devolverá el RMSE del modelo KNN que le indiquemos.

```
nombre <- "./WizmirRegression/wizmir/wizmir"
run_knn_fold <- function(i, x, tt = "test") {
  file <- paste(x, "-5-", i, "tra.dat", sep = "")
  x_tra <- read.csv(file, comment.char = "@")
  file <- paste(x, "-5-", i, "tst.dat", sep = "")
  x_tst <- read.csv(file, comment.char = "@")
  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
}
```

Ahora es el momento de obtener el RMSE de la fase de entrenamiento y de la fase de testeo.

```
knnMSEtrain <- mean(sapply(1:5, run_knn_fold, nombre, "train"))
knnMSEtest <- mean(sapply(1:5, run_knn_fold, nombre, "test"))
knnMSEtrain
[1] 2.5383
knnMSEtest
[1] 6.060107
```

La diferencia entre los errores es de 4, por lo que se deduce que el modelo knn se ajusta demasiado a los datos en la fase de entrenamiento.

Si miramos estos errores con los obtenidos por los distintos modelos lineales anteriormente obtenidos

```
comparisonDF$rmse_lm_results
```

```
[1] 2.957531 4.089580 8.911483 11.676977 13.999502  
[6] 1.320711 1.257494 1.319515 1.283699 1.315424  
[11] 1.287048
```

Vemos que el error de 2.53 podría aproximarse con el modelo de regresión lineal simple obtenido solamente con la variable temperatura máxima que tiene un 0.95 de valor en  $R^2$ , mientras que el error de 6.06 cae entre el error del modelo lineal simple de la temperatura mínima que tiene un valor de 0.91 de  $R^2$  y el modelo lineal simple del punto de rocío que tiene 0.61 como valor para el  $R^2$ . Mientras que tiene mayor error que cualquiera de los modelos de regresión lineal multiple que ninguno llega al 1.4 de error y todos tienen un 0.99 de  $R^2$ .

## Comparación entre algoritmos (KNN y LM)

No se pueden comparar los dos algoritmos con los modelos que hemos creado hasta este punto puesto que no tienen ni las mismas variables en el modelo ni los mismos ejemplos en la fase de entrenamiento y testeo.

Por ello vamos a desarrollar un nuevo modelo lineal múltiple que al igual que el modelo obtenido por knn tenga todas las variables para la construcción del mismo y los mismos ejemplos para cada fase, o lo que es lo mismo que intervenga la validación cruzada. Para ello modificamos la función anterior para que el modelo que se genere sea un modelo lineal y nos siga devolviendo el MSE del mismo.

```
run_lm_fold <- function(i, x, tt = "test") {  
  file_n <- paste(x, "-5-", i, "tra.dat", sep = "")  
  x_tra <- read.csv(file_n, comment.char = "@", header = FALSE)  
  file <- paste(x, "-5-", i, "tst.dat", sep = "")  
  x_tst <- read.csv(file, comment.char = "@")  
  ln <- length(names(x_tra)) - 1  
  names(x_tra)[1:ln] <- paste("X", 1:ln, sep = "")  
  names(x_tra)[ln + 1] <- "Y"  
  names(x_tst)[1:ln] <- paste("X", 1:ln, sep = "")  
  names(x_tst)[ln + 1] <- "Y"  
  if (tt == "train") {  
    test <- x_tra  
  } else {  
    test <- x_tst  
  }  
  fitMulti = lm(Y ~ ., x_tra)  
  yprime = predict(fitMulti, test)  
  sum(abs(test$Y - yprime)^2)/length(yprime) ##MSE  
}  
lmMSEtrain <- mean(sapply(1:5, run_lm_fold, nombre, "train"))  
lmMSEtest <- mean(sapply(1:5, run_lm_fold, nombre, "test"))
```

```
lmMSEtrain
```

```
[1] 1.563401
```

```
lmMSEtest
```

```
[1] 1.604942
```

Con estos nuevos valores de error si que podemos comparar los dos algoritmos a nivel de error cometido. Podemos ver claramente que el error de entrenamiento es menor para el modelo de regresión lineal tanto para el entrenamiento como para el testeo que el obtenido por el modelo de knn, y en ambos casos con valores parecidos a los que obtuvimos para los modelos sin la valoración cruzada.

## Comparación de algoritmos

Para hacer un test más fiable y cuantitativo que compare los algoritmos podemos realizar los test de Wilconxon,Friedman y Holm.

### Wilconxon

El test de Wilconxon es un test no paramétrico con el que se comprueba que dos conjuntos de muestras pertenecen a la misma distribución estadística. La hipótesis nula de este test es que ambas muestras pertenecen a una misma distribución continua y simétrica respecto a una mediana que es igual a 0.

Esta distribución se construye con la diferencia entre las dos muestras, las muestras son los errores cometidos en cada una de las particiones de la validación cruzada, tanto para la fase de entrenamiento como para la de test.

```
collection_mse_train_lm = sapply(1:5, run_lm_fold, nombre, "train")
collection_mse_test_lm = sapply(1:5, run_lm_fold, nombre, "test")
collection_mse_train_knn = sapply(1:5, run_knn_fold, nombre,
                                 "train")
collection_mse_test_knn = sapply(1:5, run_knn_fold, nombre, "test")

collection_mse_lm = c(collection_mse_train_lm, collection_mse_test_lm)
collection_mse_knn = c(collection_mse_train_knn, collection_mse_test_knn)

differences = collection_mse_lm - collection_mse_knn
wilconxon_diff = cbind(ifelse(differences < 0, abs(differences) +
                                0.1, 0.1), ifelse(differences > 0, abs(differences) + 0.1,
                                0.1))

w1 = wilcox.test(collection_mse_lm, collection_mse_knn, alternative = "two.sided",
                  paired = TRUE)
w2 = wilcox.test(collection_mse_knn, collection_mse_lm, alternative = "two.sided",
                  paired = TRUE)

Rplus = w1$statistic
Rminus = w2$statistic
p_value = w1$p.value

c(Rplus, Rminus, p_value)
```

V	V
0.000000000	55.000000000
0.001953125	

Puesto que el p-value es menor que 0.05, rechazamos la hipótesis nula de que ambos errores siguen la misma distribución, y, por tanto, de que los algoritmos sean igual de buenos.

## Friedman

El test de Friedman nos permite realizar la misma prueba que en el caso anterior pero comparando tres muestras distintas. En este caso el algoritmo que vamos a añadir a la comparación es el modelo lineal de cinco regresores. Para poder compararlo usaremos también la validación cruzada sobre él.

```
run_lmm_fold <- function(i, x, tt = "test") {
  file_n <- paste(x, "-5-", i, "tra.dat", sep = "")
  x_tra <- read.csv(file_n, comment.char = "@", header = FALSE)
  file <- paste(x, "-5-", i, "tst.dat", sep = "")
  x_tst <- read.csv(file, comment.char = "@")
  ln <- length(names(x_tra)) - 1
  names(x_tra)[1:ln] <- paste("X", 1:ln, sep = "")
  names(x_tra)[ln + 1] <- "Y"
  names(x_tst)[1:ln] <- paste("X", 1:ln, sep = "")
  names(x_tst)[ln + 1] <- "Y"
  if (tt == "train") {
    test <- x_tra
  } else {
    test <- x_tst
  }
  attach(x_tra)
  fitMulti = lm(Y ~ X1 + X2 + X3 + X5 + X7, x_tra)
  yprime = predict(fitMulti, test)
  sum(abs(test$Y - yprime)^2)/length(yprime) ##MSE
}

collection_mse_train_lmm = sapply(1:5, run_lmm_fold, nombre,
  "train")
collection_mse_test_lmm = sapply(1:5, run_lmm_fold, nombre, "test")
collection_mse_lmm = c(collection_mse_train_lmm, collection_mse_test_lmm)

data = as.matrix(cbind(collection_mse_lm, collection_mse_knn,
  collection_mse_lmm))
friedman.test(data)
```

Friedman rank sum test

```
data: data
Friedman chi-squared = 16.8, df = 2, p-value =
0.0002249
```

Dado que el valor del p-value es menor que 0.05, podemos decir que existe una diferencia significativa entre los tres conjuntos.

## Holms

Ahora que sabemos que existen diferencias entre los algoritmos vamos a realizar el test de Holms para ver entre que pares de algoritmos existen las diferencias.

```
groups <- rep(1:dim(data)[2], each = dim(data)[1])
pairwise.wilcox.test(data, groups, p.adjust = "holm", paired = TRUE)
```

```
Pairwise comparisons using Wilcoxon signed rank test

data: data and groups

 1     2
2 0.0059 -
3 0.0273 0.0059

P value adjustment method: holm
```

Existen diferencias significativas entre cualquier par de algoritmos al 95 % de confianza.

## Conclusión a la regresión

Una vez vistos todos los algoritmos y sus posibilidades, si tuviera que elegir un modelo entre todos los generados escogería el modelo de regresión lineal múltiple en el que intervienen únicamente la temperatura mínima y la máxima, por la simplicidad del modelo y porque su valor de R^2 es del 0.99 y la explicación del mismo es muy similar a la definición de la media aritmética entre la temperatura máxima y mínima.

## Clasificación

El problema de clasificación que se intentará resolver en esta sección es el de la aprobación de crédito en Australia. El inconveniente que tiene este conjunto de datos es que ninguna de las variables que se proporcionan tiene un nombre descriptivo solamente contamos con el rango de valores de cada una de ellas, como ya se vió en la sección de análisis de datos.

Para realizar la clasificación se probarán tres algoritmos: los K-vecinos más cercanos, LDA y QDA, todos ellos implementados en el paquete *caret*.

```
require("caret")
```

En este apartado recuperamos el dataset que habíamos desarrollado para la fase de análisis de datos:

```
australian <- read.csv("./AustralianClassification/australian/australian.dat",
  comment.char = "@", header = FALSE)
names(australian) <- c("A1", "A2", "A3", "A4", "A5", "A6", "A7",
  "A8", "A9", "A10", "A11", "A12", "A13", "A14", "A15")
```

### Preparación de los datos para clasificación.

En nuestro dataset las variables A1, A4, A8, A9, A11, A12 son variables categóricas, al igual que la variable de salida, la variable 15. Por defecto estas variables se leen como números, al tener estas variables en su gran mayoría solamente dos clases anotadas como 0 y 1 podemos usar este contraste en nuestro beneficio a la hora de hacer el modelo para clasificación, pero en aquellas variables que tienen tres clases, como es el caso de las variables A4 y A12, tenemos que utilizar otro método para darle contraste a cada clase. En este caso utilizaremos la técnica de codificación “dummy”.

La codificación dummy transforma cada nivel en un número binario compuesto por un único ‘1’ en una posición diferente para cada nivel. Para conseguir dicha codificación añadimos el parámetro contrast a cada una de las variables con tres categorías, para poder ejecutar el contraste primero convertimos la variable en factor.

```
australian[, 4] = as.factor(australian[, 4])
australian[, 12] = as.factor(australian[, 12])
australian[, 15] = as.factor(australian[, 15]) #Variable de salida
contrasts(australian$A4) = contr.treatment(3)
contrasts(australian$A12) = contr.treatment(3)
output_australian = australian[, 15]
```

La generación de los modelos de clasificación usando cada algoritmo la haremos dividiendo en conjunto de datos en dos partes: entrenamiento y testeo. En este caso será una división del 60% para entrenamiento y 40% para testeo. De esta forma los primeros 414 registros de la base de datos serán de entrenamiento y los restantes de testeo.

```
training_australian = australian[1:414, -15]
training_labels_australian = australian[1:414, 15]

test_australian = australian[414:690, -15]
test_labels_australian = australian[414:690, 15]
```

## Los k vecinos más cercanos.

La técnica de los K-vecinos más cercanos (KNN de su traducción al inglés) es una técnica considerada *lazy algorithm* o algoritmo gandul, por el hecho de que no realiza ningún cálculo especial durante la fase de entrenamiento sino que se dedica a almacenar los datos y en la fase de predicción realiza todo el trabajo.

El procedimiento que realiza la técnica de los k-vecinos más cercanos es el cálculo de la distancia del punto a clasificar con todos los puntos existentes en la base de datos hasta el momento, seleccionando los k más cercanos y decidiendo según sus datos que valor se le da a la etiqueta de clasificación.

Los datos tienen que ser normalizados antes de utilizarlos en el algoritmo ya que muchos de ellos tienen un rango muy amplio y otros más reducido. Para ello utilizaremos el parámetro de la función de knn “scale” que nos permitirá de forma sencilla normalizar los datos y ejecutar el algoritmo en una única línea de código. Otra precaución que hay que tomar con los datos de las etiquetas es que tienen que ser tratados como factores para que caret sea capaz de determinar que el algoritmo es para clasificación y no para regresión, pero de eso ya nos hemos encargado en el apartado anterior de preparación de datos.

```
set.seed(2)
knnModel = train(x = training_australian, y = training_labels_australian,
  method = "knn", preProc = c("center", "scale"), metric = "Accuracy",
  tuneGrid = data.frame(.k = 1:15))
knnModel

k-Nearest Neighbors

414 samples
14 predictor
2 classes: '0', '1'

Pre-processing: centered (12), scaled (12), ignore (2)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 414, 414, 414, 414, 414, 414, ...
Resampling results across tuning parameters:

  k    Accuracy   Kappa
  1   0.8079840  0.6109850
  2   0.8022531  0.5992927
  3   0.8126384  0.6197211
  4   0.8230354  0.6404459
  5   0.8268375  0.6478383
  6   0.8318226  0.6579076
  7   0.8370862  0.6687103
  8   0.8404404  0.6748467
  9   0.8417346  0.6776757
 10  0.8394032  0.6733541
 11  0.8427034  0.6799526
 12  0.8450974  0.6844451
 13  0.8462137  0.6864462
 14  0.8472950  0.6888850
 15  0.8477826  0.6899063

Accuracy was used to select the optimal model using
the largest value.
The final value used for the model was k = 15.
```

Hay que puntualizar que la generación del modelo con caret, pese al esfuerzo, ha ignorado las variables factor que teníamos con sus correspondientes contrastes.

En la generación del modelo Knn hemos determinado que busque el mejor número de vecinos, k, para el modelo entre 1 y 15 y ha dicho que el mejor modelo lo ha obtenido con 15 vecinos más cercanos. Como vemos el primer modelo ya da una presición del 81% y el mejor modelo tiene una presición del 85%, esta pequeña mejora a pesar de haber aumentado el número de vecinos que intervienen en el valor, causa sospechas de que se está sobreajustando a los datos y que nos va a dar problemas a la hora de generalizar. Para comprobarlo vamos a realizar una clasificación con este modelo y vamos a obtener su presición:

```
knnPredict = predict(knnModel, test_australian)
cm_knn = confusionMatrix(knnPredict, test_labels_australian)
(cm_knn$table[1, 1] + cm_knn$table[2, 2])/sum(cm_knn$table)
```

```
[1] 0.8231047
```

La presición obtenida no varía significativamente con respecto de la obtenida en la fase de entrenamiento, por lo que podemos dar el modelo por bueno.

## Linear Discriminat Analysis (LDA)

La técnica de la discriminación linear separa los datos asumiendo que cada clase pertenece a una distribución de probabilidad distinta, con la misma covarianza. Para cada una de las variables de entrada sigue una distribución normal, hecho que se probó en la fase de análisis que no ocurría con ninguna de las variables y por ello no se espera que ni este ni el algoritmo QDA funcionen bien.

Usando el mismo conjunto de entrenamiento que antes vamos a elaborar un nuevo modelo basado en el algoritmo LDA:

```
ldaModel = train(x = training_australian[, c(-4, -12)], y = training_labels_australian,
  method = "lda", preProc = c("center", "scale"), metric = "Accuracy")
ldaModel
```

```
Linear Discriminant Analysis
```

```
414 samples
12 predictor
 2 classes: '0', '1'
```

```
Pre-processing: centered (12), scaled (12)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 414, 414, 414, 414, 414, 414, ...
Resampling results:
```

```
  Accuracy   Kappa
0.8729697 0.7437517
```

```
ldaPrediction = predict(ldaModel, test_australian[, c(-4, -12)])
cm_lda = confusionMatrix(ldaPrediction, test_labels_australian)
(cm_lda$table[1, 1] + cm_lda$table[2, 2])/sum(cm_lda$table)
```

```
[1] 0.8447653
```

La eliminación de la variable A4 y A12 al igual que en el caso anterior se debe a que caret no trabaja con factores, ni maneja los contrastes de los mismos.

Hay que puntualizar, que tanto LDA como QDA generan los mismos resultados tanto para los datos normalizados como para los datos sin normalizar.

Tras construir el modelo se utilizó éste para predecir los valores de train que teníamos separados desde antes y realizamos una tabla de confusión de la que extraemos los verdaderos positivos y los verdaderos negativos para sumarlos y compararlos con el total del conjunto, de esta forma tenemos la medida de presición de la fase de entrenamiento y de la fase de testeo.

Dicho error no varía significativamente de una fase a otra, ni tampoco con el algoritmo de los K vecinos más cercanos, por lo que el algoritmo se da por bueno.

## Quadratic Discriminant Analysis (QDA)

Como se comentó antes, LDA asume que las varianzas de las distribuciones de salida son la misma pero QDA no. QDA estima una varianza/covarianza para cada una de las clases de salida. En la fase de análisis de datos se demostró que los conjuntos de salida tienen la misma varianza por lo que no se espera que este método sea especialmente bueno.

```
qdaModel = train(x = training_australian[, c(-4, -12)], y = training_labels_australian,
  method = "qda", preProc = c("center", "scale"), metric = "Accuracy")
qdaModel
```

Quadratic Discriminant Analysis

```
414 samples
12 predictor
2 classes: '0', '1'
```

```
Pre-processing: centered (12), scaled (12)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 414, 414, 414, 414, 414, 414, ...
Resampling results:
```

```
Accuracy      Kappa
0.8121076   0.6089828
```

```
qdaPrediction = predict(qdaModel, test_australian[, c(-4, -12)])
```

```
cm_qda = confusionMatrix(qdaPrediction, test_labels_australian)
(cm_qda$table[1, 1] + cm_qda$table[2, 2])/sum(cm_qda$table)
```

```
[1] 0.7906137
```

El método QDA se ha presentado como el método con el peor resultado de presición de los tres métodos explorados, con un 81% de presición para la etapa de entrenamiento y con un 79% en la fase de test. Esto será debido a que las varianzas de los conjuntos de salida son muy similares y al presuponer QDA que son distintas entra en error.

## Comparación de los tres algoritmos

A la hora de comparar los tres algoritmos vamos a cambiar el método de elaboración de los modelos para introducirles la validación cruzada. De esta forma podremos comparar sabiendo que los tres métodos han probado todas las combinaciones de particiones para generar el modelo en la fase de entrenamiento.

La validación cruzada la realizaremos con la función que hemos usado en el apartado de regresión y le haremos unos ajustes para que nos devuelva la presición del método que deseamos, también ajustando el número de particiones a 10.

```
nombre <- "australian"
runFold <- function(i, x, tt = "test", method) {
  file_n <- paste(x, "-10-", i, "tra.dat", sep = "")
  file_n
  x_tra <- read.csv(file_n, comment.char = "@", header = FALSE)
  file <- paste(x, "-10-", i, "tst.dat", sep = "")
  x_tst <- read.csv(file, comment.char = "@")
  ln <- length(names(x_tra)) - 1
  names(x_tra)[1:ln] <- paste("X", 1:ln, sep = "")
  names(x_tra)[ln + 1] <- "Y"
  names(x_tst)[1:ln] <- paste("X", 1:ln, sep = "")
  names(x_tst)[ln + 1] <- "Y"

  x_tra[, ln + 1] = as.factor(x_tra[, ln + 1])
  x_tst[, ln + 1] = as.factor(x_tst[, ln + 1])
  if (tt == "train") {
    test <- x_tra
  } else {
    test <- x_tst
  }
  attach(x_tra)
  if (method == "knn") {
    model = train(x = x_tra[, c(-4, -12, -15)], y = as.factor(Y),
                  method = "knn", preProc = c("center", "scale"))
  } else if (method == "lda") {
    model = train(x = x_tra[, c(-4, -12, -15)], y = as.factor(Y),
                  method = "lda", preProc = c("center", "scale"))
  } else if (method == "qda") {
    model = train(x = x_tra[, c(-4, -12, -15)], y = as.factor(Y),
                  method = "qda", preProc = c("center", "scale"))
  }
  prediction = predict(model, test[, c(-4, -12, -15)])
  cm = confusionMatrix(prediction, test[, 15])
  (cm$table[1, 1] + cm$table[2, 2])/sum(cm$table)
}

knnPredict = sapply(1:10, runFold, nombre, "train", "knn")
knnPredict = c(knnPredict, sapply(1:10, runFold, nombre, "test",
                                  "knn"))
ldaPrediction = sapply(1:10, runFold, nombre, "train", "lda")
ldaPrediction = c(ldaPrediction, sapply(1:10, runFold, nombre,
                                         "test", "lda"))
qdaPrediction = sapply(1:10, runFold, nombre, "train", "qda")
qdaPrediction = c(qdaPrediction, sapply(1:10, runFold, nombre,
```

```
"test", "qda"))
```

En este punto tenemos en *knnPredict*, *ldaPredict* y *qdaPredict* los valores de las presiciones para cada una de las particiones de la base de datos. Con estos nuevos datos podemos realizar la comparación de los algoritmos. Dado que el número de algoritmos que vamos a comparar es tres, vamos a realizar únicamente las pruebas de Friedman y Holms para saber si existe alguna diferencia.

```
classificationPredictions = as.matrix(cbind(knnPredict, ldaPrediction,
                                             qdaPrediction))
friedman.test(classificationPredictions)
```

```
Friedman rank sum test
```

```
data: classificationPredictions
Friedman chi-squared = 25.74, df = 2, p-value =
2.574e-06
```

Dado que el p-values es menor que 0.05 podemos determinar que existen diferencias entre los algoritmos, es decir, al menos uno de ellos es diferente de los demás. Para averiguarlo realizamos el test de Holms.

```
groups2 <- rep(1:dim(classificationPredictions)[2], each = dim(classificationPredictions)[1])
pairwise.wilcox.test(classificationPredictions, groups2, p.adjust = "holm",
                     paired = TRUE)
```

```
Pairwise comparisons using Wilcoxon signed rank test
```

```
data: classificationPredictions and groups2
1      2
2 0.12690 -
3 0.00036 0.00036
```

```
P value adjustment method: holm
```

Podemos ver en el cuadrante anterior que para el test de Holm el modelo KNN y el modelo de LDA son iguales con un p-value de más del 0.05, mientras que en el caso de QDA con KNN y QDA con LDA no hay un p-value lo suficientemente significativo para decir que son iguales.

## Conclusión al apartado de clasificación.

Ante los modelos de los tres algoritmos no podemos decir que exista diferencia entre el modelo de KNN y LDA a nivel de presición de la clasificación, pero como ya se comentó en su momento, el método KNN es un algoritmo que no construye el modelo en la fase de entrenamiento, sino que en la fase de test realiza todos los cálculos lo que implica que es lento para dar una respuesta ante una entrada concreta. Por esto me inclino a decir que para este problema concreto el mejor modelo es el de LDA.

## Conclusión

En este trabajo he aplicado los conocimientos obtenidos en la asignatura, como pueden ser:

- Análisis de datos usando el mínimo, máximo, los cuartiles, la mediana, la media y la moda de los mismos.
- La elaboración de scatter plots para comparar variables.
- La construcción de gráficos de barras para mostrar la frecuencia.
- La realización de test de normalidad de los datos.
- La realización de test de igualdad de varianzas.
- El planteamiento de hipótesis previas sobre el problema para corroborarlo con los datos.
- La elaboración de modelos de regresión lineal simple, múltiple, con interacciones entre variables y con no linealidades.
- La elaboración de modelos basados en la técnica de los k-vecinos más cercanos tanto para modelos predictivos como para clasificación.
- La elaboración de modelos de discriminación lineal y cuadrática para clasificación de datos.
- La comparación de algoritmos mediante los test de Wilcoxon, Friedman y Holms.
- La utilización de la validación cruzada para evitar el sobreajuste de los modelos a los datos de entrenamiento.

## Bibliografía

1. UCI Machine Learning Repository.
2. Validación cruzada - Wikipedia.
3. Perform a Shapiro-Wilk Normality Test - Stackoverflow
4. Linear and Quadratic Discriminant Analysis - SciKit Learn
5. How to use Levene Test - StackExchange
6. Test de Levene para la igualdad de varianzas - El blog de los erreros
7. Igualdad de varianzas en R - Aqueronte
8. The caret package
9. Preprocess function - R Documentation
10. Contrast Coding Systems for categorical variables