Analisis Breast Cancer

Montse Figueiro

26 de julio de 2016

Summary

Nos da un resumen de como estan medidas las diferentes variables, mostrando:

- Mínimo **min()**
- Primer cuartil
- Mediana
- Tercer cuartil
- Máximo **max()**

La diferencia entre el mínimo y el máximo es el rango. "range()" and "diff()" te permitiría analizar esa diferencia.

Los cuartiles dividen el dataset en 4 partes, cada uno con el mismo número de valores.

La diferencia entre cuartil 1 y cuartil 3 es Rango intercuartil (IQR), se puede calcular con la función **IQR()**

- **quantile()** nos devuelve los cinco números de summary.
- **quantile(usedcars$price,probs=c(0.01,0.99))** podemos sacar los quantiles que queramos, aquí por ejemplo en el 1% y en el 99%.
- **quantile(usedcars$price,seq(from=0,to=1,by=0.20))** aquí saco el 0%,20%.....

Interpretación

Esto nos ayuda a ver la dispersión de los datos. El dataset nos dice que el mínimo es 3800 y el máximo 21992, la diferencia entre el mínimo y el Q1 es 7000, la diferencia entre el máximo y el Q3 son 7000 tambien. En cambio la diferencia entre Q1 y Q3 que es el 50% del medio son 2000. Los datos están más estrechamente agrupados en el centro, esto es típico en una distribución normal.

Esto explica porque la media es mucho más grande que la mediana, porque la media es sensible a los valores extremos.

BOXPLOTS - visualizando los datos

El boxplot muestra los cinco números de summary usando lineas horizontales. La caja muestra Q1, mediana y Q3, leyendo desde abajo hacia arriba. La mediana es la linea en negrita. El mínimo y el máximo está representado por rayas discontinuas. Solo permite extender las rayas discontinuas hasta 1.5 veces el IQR, debajo de Q1 o encima de Q3.

Por ejemplo: Q1 es 26 Q3 es 56

IQR es 30 (diferencia entre Q3 y Q1)

Q3 + 1.5 * 30 = 101 (no puede llegar hasta 120 que es el máximo) Q1 - 1.5 * 30 = -9 (llegará hasta el mínimo que es 2)

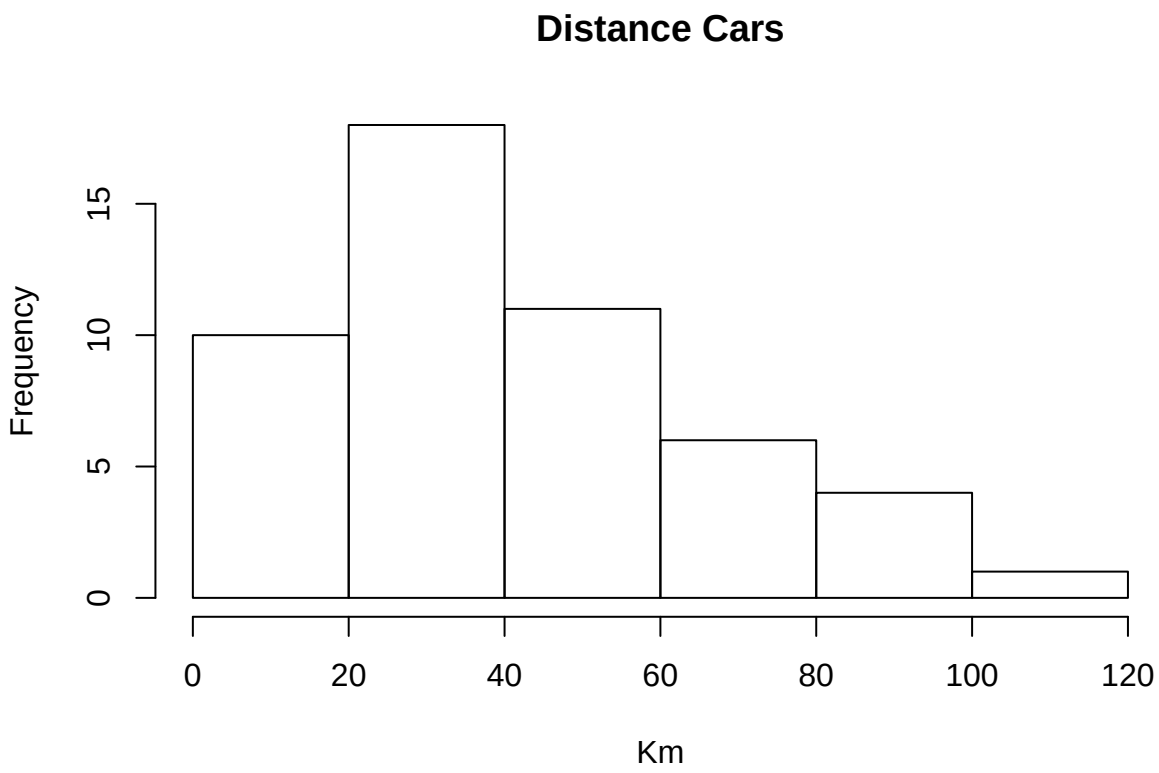
boxplot(cars$dist="Dist Cars",ylab="Km")

```
summary(cars$dist)
boxplot(cars$dist,main="Dist Cars",ylab="Km")
```

HISTOGRAMAS

Divide los valores de las variables en **bins** columnas, un boxplot requiere cada una de las 4 porciones contenga el mismo número de valores. En contraste el histograma usa cualquier número de barras de identico ancho y permite contener dentro diferente número de valores.

```
hist(cars$dist,main="Distance Cars",xlab="Km")
```



Las barras indican la frecuencia de los valores, es este ejemplo vemos que lo más frecuente es que el coche para frenar recorra una distancia de entre 20 y 40 metros.

El histograma tiene una oblicuidad hacia la derecha.

Distribución NORMAL

Varianza y Desviación Standar

varianza se define como la media de la suma de las diferencias entre cada valor y la media al cuadrado.

Var= (sum((x-Media)^2))/n Std= sqrt(Var) R utiliza n-1 (sample variance), excepto en dataset muy pequeños la diferencia es mínima.

```
var(cars$dist)
```

```
## [1] 664.0608
```

```r
sd(cars$dist)
```

```
## [1] 25.76938
```

Cuando la varianza es muy grande esto indica que los datos se difunden muy ampliamente alrededor de la media. La desviación standar te indica para cada valor cuanto difiere de la media.

Variables Categóricas

Cuando importamos los datos como astringsAsFactors = FALSE, R deja las variables como character en lugar de convertirlas en factores. Podemos considerar poner los años como categórica, aunque está como entero cada año es una categoría.

Este tipo de datos se examina usando tablas en lugar de estadísticas.

One-way table : presenta una sola variable categórica.

```r
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Nos hace una lista de las diferentes categorías y cuantos valores hay dentro de cada una.

R puede calcular la tabla de proporciones:

```r
species_table <- table(iris$Species)
prop.table(species_table)
```

```
##
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
```

```r
species_pct <- prop.table(species_table) *100
round(species_pct,digits=1)
```

```
##
##     setosa versicolor  virginica
##       33.3       33.3       33.3
```

Así podemos ver para las diferentes categorías cual es la más frecuente.

Analizando la Moda

En estadística la moda es el valor que ocurre con más frecuencia. Esta se utiliza frecuentemente en datos categóricos, ya que no se puede usar la media o la mediana.

Una variable puede tener una moda o más * unimodal * bimodal * multimodal

Es mejor relacionar la moda con otras categorias. Hay alguna categoría que domina a otra? Si los colores de coche son plata y negro, considero que estamos hablando de coches de lujo? o coches económicos, cuales se venden con menos opciones de color? En un histograma la moda sería la barra más alta.

RELACIÓN ENTRE VARIABLES

SCATTERPLOT

Es un diagrama que representa la relación entre dos variables mediante circulos, una variable se representa en el eje "x" y otra variables en el eje "y".

```
plot(x=cars$dist,y=cars$speed,main="Scatterplot of Speed and Distance",ylab="Distance",xlab="Speed")
```

Vemos una relación entre la velocidad y la Distancia recorrida. Tienen una relación positiva, el patrón de circulos es ascendente. Podría ser una relación negativa si la linea fuera descendiente. Y no tendrían nada que ver si la linea fuera plana.

La **Correlación** mide la relación entre dos variables.

Tabulación Cruzada

Mide la relación entre dos variables. Como los valores de una variable varian en función de otra variable. El formato es una tabla donde las filas son los niveles de una variable mientras las columna son los niveles de la otra

CrossTable()

library(gmodels) CrossTable(x=train_sample$Cat1, y = train_sample$Cat2)

Clasificación usando Nearest Neighbors

Mide la similitud de dos muestras midiendo distancias. Se ha utilizado por ejemplo:

- Reconocimiento facial en imagenes y videos
- Predecir cuando una persona va a disfrutar una película que le ha sido recomendada (Netflix)
- Identificar patrones en datos genéticos al detectar una específica proteína.

Por lo general este método de clasificación es bueno cuando la relación entre las variables y las clases objetico es alto, complicado y difícil de entender. Si no hay una clara distinción entre grupos el algorítmo es largo y no es bueno identificando los límites.

The KNN algorithm

Fortalezas	Debilidades
Simple y efectivo	No produce un modelo, lo que lo limita para encontrar nuevos conocimientos en las relaciones entre las variables
No hace suposiciones sobre la distribucion de los datos subyacente	Fase de clasificación lenta
Fase de entrenamiento rápida	Requiere mucha memoria Variables nominales y missing data requieren procesamiento adicional

kNN empieza con un training dataset con observaciones classificado en diferentes categorias, y etiquetadas por una variable nominal. Tenemos también un test dataset sin etiquetar que tiene el mismo número de variables que el training data. Para cada observación en el test dataset kNN identifica **k** observaciones en training dataset que son "vecinos", **k** es un entero especificado con anterioridad. A la observación del test datset se le asigna la clase de la mayoria de los k vecinos.

```
ingredient <- c("apple","bacon","banana","carrot","celery","cheese")
sweetness <- c(10,1,10,7,3,1)
crunchiness <- c(9,4,1,10,10,1)
foodtype <- c("fruit","protein","fruit","vegetable","vegetable","protein")
df <- data.frame(ingredient,sweetness,crunchiness,foodtype)
df
```

```
##   ingredient sweetness crunchiness  foodtype
## 1      apple        10           9     fruit
## 2      bacon         1           4   protein
## 3     banana        10           1     fruit
## 4     carrot         7          10 vegetable
## 5     celery         3          10 vegetable
## 6     cheese         1           1   protein
```

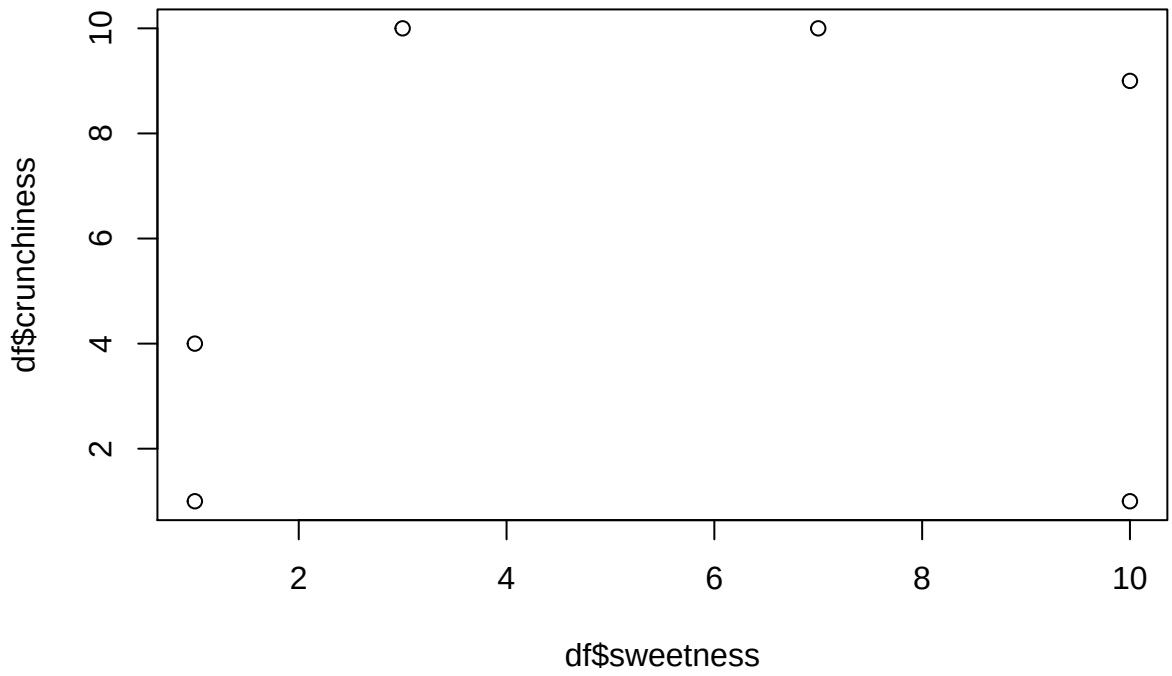
```
str(df)
```

```
## 'data.frame':    6 obs. of  4 variables:
##  $ ingredient : Factor w/ 6 levels "apple","bacon",..: 1 2 3 4 5 6
##  $ sweetness  : num  10 1 10 7 3 1
##  $ crunchiness: num  9 4 1 10 10 1
##  $ foodtype   : Factor w/ 3 levels "fruit","protein",..: 1 2 1 3 3 2
```

En este ejemplo se han clasificado las comidas según unas variables, en 3 clases, fruta, carne o verdura.

En este ejemplo solo tenemos 2 variables con lo que podemos representar los datos con un scatterplot:

```
plot(x=df$sweetness,y=df$crunchiness)
```



En este ejemplo hay muy pocas observaciones y no se pueden ver agrupaciones.

Calculo Distancia

El algoritmo kNN usa la distancia Euclidean. La distancia entre dos observaciones **p** y **q** es la raíz cuadrada de la suma de las distancias al cuadrado. p1 es el valor de la variable 1 para p q1 es el valor de la variable 1 para q

$dist(p,q) = \sqrt{\sum_{i=1}^{n}((p1-q1)^2 + (p2-q2)^2.....)}$

La formula compara los valores para cada variable. Por ejemplo entre el tomate y la banana o entre el tomate y el queso...

Cuando calculemos todas las distancias del tomate al resto de ingredientes, veremos con que ingrediente tiene menos distancia, se llama clasificación 1NN porque k=1, por ejemplo, si el más cercano es la naranja, clasificará al tomate como "fruit". Si usamos kNN con k=3, lleva a cabo una votación entre los 3 vecinos más cercanos, los tres más cercanos son naranja, uva y cacahuete, 2 de 3 son frutas, el tomate lo clasificará como "fruit".

Elección de k

Determinar un número correcto de vecinos determinará como de bueno será el modelo para generalizar a datos futuros. Escoger una k muy grande reduce el impacto de la varianza causada por el ruido, pero corre el riesgo de ignorar pequeños patrones.

Ejemplo: Tomar una k igual al nº de observaciones, cada observación estaría representada en el voto final.

Tomar una k=1 implica que permita ruido y outliers, que influencian la clasificación de los ejemplos. Algunas de las observaciones pueden estar mal clasificadas.

El mejor valor para **k** es alguno entre estos dos.

Valores pequeños permiten más complejas decisiones, que mas cuidadosamente encaja en el training data.

Normalmente k está entre 3 y 10. Una práctica común es la raíz cuadrada del número de ejemplos del training dataset. En el ejemplo anterior teníamos 15 ingredientes, 3.87.

Una alternativa es usar diferentes **k** en diferentes test datasets y elegir uno que nos dé la mejor clasificación. si los datos tienen mucho ruido y son muy grande el número de observaciones, la elección de la **k** puede ser menos importante.

Preparando datos para usar kNN

Las variables se suelen normalizar antes de usar kNN. Esto es así porque el cálculo de las distancias depende de las unidades con las que esté medida la variable. Esto no sería un problema en el caso anterior porque las medidas de las comidas iban de 1 a 10. Pero si por ejemplo una variable estuviera medida en valores de 1 a 1000000, el impacto de las otras variables se vería disminuido.

Necesitamos escalar las variables de manera que cada una de ellas contribuya relativamente igual dentro de la fórmula. En el caso anterior querremos que la última variable esté entre 1 y 10.

- Min-Max normalización

Transforma todas las medidas para que estén entre el rango 0-1.

$Xnew = \frac{X - min(X)}{max(X) - min(X)}$

- Z-score Standarización

Es el valor menos la media dividido por la desviación standar.

$Xnew = \frac{X - \mu}{\sigma}$

Esta fórmula está basada en propiedades de la distribución normal. Reescala cada valor de cada variable según cuanta desviacion standar tenga por encima o por debajo de la media. El resultado se llama z-score.

La distancia Euclidean no está definida para variables nominales, antes hay que pasarlas a numéricas. Una solución típica es utilizar dummy coding, donde 1 indica una categoría y 0 la otra.

Nueva variable llamada “male”, tendrá los siguientes valores:

- 1 male
- 0 otherwise

Una variable nominal con n categorias puede ser codificada con dummy, por ejemplo si tenemos 3 categorias de tiempo (caliente, templado y frío):

- Caliente = 1 es caliente y 0 el resto
- Templado = 1 es templado y 0 el resto

Si es 0 sabemos que es frío, no necesitamos una tercera variable, tendremos (n-1) variables dummy. La distancia entre variables dummy es siempre 0 o 1.

Otra alternativa sería numerar las categorias, 1,2,3 y después normalizar. Quedando 0, 0.5 y 1, esto solo puede ser usado si sabes que la diferencia entre las categorías es proporcional. Si por ejemplo estamos midiendo, pobre, clase media y ricos, la diferencia entre pobre y clase media es más o menos que la diferencia entre clase media y ricos, entonces es mejor utilizar dummy.

kNN es un algoritmo perezoso?

Un aprendizaje perezoso es un aprendizaje de memoria, no construye un modelo, es una modelo de aprendizaje no paramétrico, no se aprenden parámetros sobre los datos. Limita nuestra capacidad de entender como está clasificando los datos. Aunque puede ser bastante potente.

Diagnosticando Cancer de pecho con kNN

El exámen de rutina de cáncer de mama permite ser diagnosticado y tratado antes de que se noten los síntomas. El proceso consiste en detectar masas anormales. Si una masa es detectada se extrae una muestra y se analiza (biopsia), para determinar si es maligna o benigna.

Los datos aportados por esas biopsias son los que vamos a analizar para investigar con machine learning.

Base de Datos

vamos a utilizar el Breast Cancer Wisconsin Diagnostic dataset de la siguiente página:

"BreastCancerDataset"

```
DataCancer <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsi
write.csv(DataCancer,file="DataCancer.csv")
datacancer <- read.csv("DataCancer.csv")
head(datacancer)
```

```
##   X       V1 V2    V3    V4     V5     V6      V7      V8     V9     V10
## 1 1   842302  M 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001 0.14710
## 2 2   842517  M 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869 0.07017
## 3 3 84300903  M 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974 0.12790
## 4 4 84348301  M 11.42 20.38  77.58  386.1 0.14250 0.28390 0.2414 0.10520
## 5 5 84358402  M 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980 0.10430
## 6 6   843786  M 12.45 15.70  82.57  477.1 0.12780 0.17000 0.1578 0.08089
##      V11     V12    V13    V14   V15    V16      V17     V18     V19
## 1 0.2419 0.07871 1.0950 0.9053 8.589 153.40 0.006399 0.04904 0.05373
## 2 0.1812 0.05667 0.5435 0.7339 3.398  74.08 0.005225 0.01308 0.01860
## 3 0.2069 0.05999 0.7456 0.7869 4.585  94.03 0.006150 0.04006 0.03832
## 4 0.2597 0.09744 0.4956 1.1560 3.445  27.23 0.009110 0.07458 0.05661
## 5 0.1809 0.05883 0.7572 0.7813 5.438  94.44 0.011490 0.02461 0.05688
## 6 0.2087 0.07613 0.3345 0.8902 2.217  27.19 0.007510 0.03345 0.03672
##       V20      V21       V22   V23   V24    V25    V26    V27    V28    V29
## 1 0.01587 0.03003 0.006193 25.38 17.33 184.60 2019.0 0.1622 0.6656 0.7119
## 2 0.01340 0.01389 0.003532 24.99 23.41 158.80 1956.0 0.1238 0.1866 0.2416
## 3 0.02058 0.02250 0.004571 23.57 25.53 152.50 1709.0 0.1444 0.4245 0.4504
## 4 0.01867 0.05963 0.009208 14.91 26.50  98.87  567.7 0.2098 0.8663 0.6869
## 5 0.01885 0.01756 0.005115 22.54 16.67 152.20 1575.0 0.1374 0.2050 0.4000
## 6 0.01137 0.02165 0.005082 15.47 23.75 103.40  741.6 0.1791 0.5249 0.5355
##      V30    V31     V32
## 1 0.2654 0.4601 0.11890
## 2 0.1860 0.2750 0.08902
## 3 0.2430 0.3613 0.08758
## 4 0.2575 0.6638 0.17300
## 5 0.1625 0.2364 0.07678
## 6 0.1741 0.3985 0.12440
```

```r
datacancer$X <- NULL
str(datacancer)
```

```
## 'data.frame':    569 obs. of  32 variables:
##  $ V1 : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001 ...
##  $ V2 : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
##  $ V3 : num  18 20.6 19.7 11.4 20.3 ...
##  $ V4 : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ V5 : num  122.8 132.9 130 77.6 135.1 ...
##  $ V6 : num  1001 1326 1203 386 1297 ...
##  $ V7 : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ V8 : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ V9 : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ V10: num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ V11: num  0.242 0.181 0.207 0.26 0.181 ...
##  $ V12: num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ V13: num  1.095 0.543 0.746 0.496 0.757 ...
##  $ V14: num  0.905 0.734 0.787 1.156 0.781 ...
##  $ V15: num  8.59 3.4 4.58 3.44 5.44 ...
##  $ V16: num  153.4 74.1 94 27.2 94.4 ...
##  $ V17: num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ V18: num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ V19: num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ V20: num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ V21: num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ V22: num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ V23: num  25.4 25 23.6 14.9 22.5 ...
##  $ V24: num  17.3 23.4 25.5 26.5 16.7 ...
##  $ V25: num  184.6 158.8 152.5 98.9 152.2 ...
##  $ V26: num  2019 1956 1709 568 1575 ...
##  $ V27: num  0.162 0.124 0.144 0.21 0.137 ...
##  $ V28: num  0.666 0.187 0.424 0.866 0.205 ...
##  $ V29: num  0.712 0.242 0.45 0.687 0.4 ...
##  $ V30: num  0.265 0.186 0.243 0.258 0.163 ...
##  $ V31: num  0.46 0.275 0.361 0.664 0.236 ...
##  $ V32: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

Tenemos 569 observaciones con 32 variables.

```r
colnames(datacancer)[1] <- "id"
colnames(datacancer)[c(2,3,4,5,6)] <-c("diagnosis","radius_mean","texture_mean","perimeter_mean","area_m
```

La primera variable es un identificador de cada paciente, no tiene que haber registros duplicados.

```r
anyDuplicated(datacancer$id)
```

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

```r
length(unique(datacancer$id))
```

```
## [1] 569
```

No nos aporta información para el modelo y hay que excluirla. Suelen ser excluidas siempre las variables ID

```
datacancer <- datacancer[-1]
```

Diagnosis indica si la masa es maligna o benigna

```
table(datacancer$diagnosis)
```

```
##
##   B   M
## 357 212
```

```
datacancer$diagnosis <- factor(datacancer$diagnosis,levels=c("B","M"), labels=c("Benign","Malignant"))
head(datacancer)
```

```
##   diagnosis radius_mean texture_mean perimeter_mean area_mean      V7
## 1 Malignant       17.99        10.38         122.80    1001.0 0.11840
## 2 Malignant       20.57        17.77         132.90    1326.0 0.08474
## 3 Malignant       19.69        21.25         130.00    1203.0 0.10960
## 4 Malignant       11.42        20.38          77.58     386.1 0.14250
## 5 Malignant       20.29        14.34         135.10    1297.0 0.10030
## 6 Malignant       12.45        15.70          82.57     477.1 0.12780
##        V8     V9     V10    V11     V12    V13    V14   V15    V16
## 1 0.27760 0.3001 0.14710 0.2419 0.07871 1.0950 0.9053 8.589 153.40
## 2 0.07864 0.0869 0.07017 0.1812 0.05667 0.5435 0.7339 3.398  74.08
## 3 0.15990 0.1974 0.12790 0.2069 0.05999 0.7456 0.7869 4.585  94.03
## 4 0.28390 0.2414 0.10520 0.2597 0.09744 0.4956 1.1560 3.445  27.23
## 5 0.13280 0.1980 0.10430 0.1809 0.05883 0.7572 0.7813 5.438  94.44
## 6 0.17000 0.1578 0.08089 0.2087 0.07613 0.3345 0.8902 2.217  27.19
##        V17     V18     V19     V20     V21      V22   V23   V24    V25
## 1 0.006399 0.04904 0.05373 0.01587 0.03003 0.006193 25.38 17.33 184.60
## 2 0.005225 0.01308 0.01860 0.01340 0.01389 0.003532 24.99 23.41 158.80
## 3 0.006150 0.04006 0.03832 0.02058 0.02250 0.004571 23.57 25.53 152.50
## 4 0.009110 0.07458 0.05661 0.01867 0.05963 0.009208 14.91 26.50  98.87
## 5 0.011490 0.02461 0.05688 0.01885 0.01756 0.005115 22.54 16.67 152.20
## 6 0.007510 0.03345 0.03672 0.01137 0.02165 0.005082 15.47 23.75 103.40
##      V26    V27    V28    V29    V30    V31     V32
## 1 2019.0 0.1622 0.6656 0.7119 0.2654 0.4601 0.11890
## 2 1956.0 0.1238 0.1866 0.2416 0.1860 0.2750 0.08902
## 3 1709.0 0.1444 0.4245 0.4504 0.2430 0.3613 0.08758
## 4  567.7 0.2098 0.8663 0.6869 0.2575 0.6638 0.17300
## 5 1575.0 0.1374 0.2050 0.4000 0.1625 0.2364 0.07678
## 6  741.6 0.1791 0.5249 0.5355 0.1741 0.3985 0.12440
```

357 casos son benignos y 212 malignos Queremos saber el porcentaje de casos

```
round(prop.table(table(datacancer$diagnosis)) * 100, digits=1)
```

Las otras 30 variables son numéricas

```r
summary(datacancer)
```

```
##      diagnosis    radius_mean      texture_mean   perimeter_mean
##  Benign   :357   Min.   : 6.981   Min.   : 9.71   Min.   : 43.79
##  Malignant:212   1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17
##                  Median :13.370   Median :18.84   Median : 86.24
##                  Mean   :14.127   Mean   :19.29   Mean   : 91.97
##                  3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10
##                  Max.   :28.110   Max.   :39.28   Max.   :188.50
##    area_mean            V7                V8                V9
##  Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000
##  1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956
##  Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154
##  Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880
##  3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070
##  Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680
##       V10               V11               V12               V13
##  Min.   :0.00000   Min.   :0.1060   Min.   :0.04996   Min.   :0.1115
##  1st Qu.:0.02031   1st Qu.:0.1619   1st Qu.:0.05770   1st Qu.:0.2324
##  Median :0.03350   Median :0.1792   Median :0.06154   Median :0.3242
##  Mean   :0.04892   Mean   :0.1812   Mean   :0.06280   Mean   :0.4052
##  3rd Qu.:0.07400   3rd Qu.:0.1957   3rd Qu.:0.06612   3rd Qu.:0.4789
##  Max.   :0.20120   Max.   :0.3040   Max.   :0.09744   Max.   :2.8730
##       V14               V15               V16               V17
##  Min.   :0.3602   Min.   : 0.757   Min.   :  6.802   Min.   :0.001713
##  1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169
##  Median :1.1080   Median : 2.287   Median : 24.530   Median :0.006380
##  Mean   :1.2169   Mean   : 2.866   Mean   : 40.337   Mean   :0.007041
##  3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146
##  Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130
##       V18                V19               V20
##  Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
##  Median :0.020450   Median :0.02589   Median :0.010930
##  Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
##  3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
##  Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
##       V21                V22               V23              V24
##  Min.   :0.007882   Min.   :0.0008948   Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480   1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870   Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949   Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580   3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400   Max.   :36.04   Max.   :49.54
##       V25               V26               V27               V28
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
##  Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##       V29               V30               V31               V32
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.1565   Min.   :0.05504
```

```
##  1st Qu.:0.1145   1st Qu.:0.06493   1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993   Median :0.2822   Median :0.08004
##  Mean   :0.2722   Mean   :0.11461   Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140   3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100   Max.   :0.6638   Max.   :0.20750
```

```
colnames(datacancer)[6] <- "smoothness_mean"
```

Nos centramos en 3 variables, radius_mean, area_mean y smoothness_mean, el rango de la primera va desde 6.9 a 28.11, la segunda 143.5 a 2501 y la última de 0.052 a 0.1634, el impacto de la segunda variable va a ser mas que las otras al calcular las distancias.

Vamos a aplicar la normalización.

Normalización Datos

Vamos a crear la función "normalize":

```
normalize <- function(x){return ((x-min(x))/(max(x)-min(x)))}
```

Prueba (el resultado es el mismo para los dos vectores):

```
normalize(c(1,2,3,4,5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
normalize(c(100,200,300,400,500))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

Aplicamos la función a las 30 variables, podemos usar **lapply()** para aplicar la función a cada variable y convertir la lista devuelta por lapply en un dataframe con **as.data.frame()**

```
datoscancer_norm <- as.data.frame(lapply(datacancer[2:31],normalize))
summary(datoscancer_norm$radius_mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2233  0.3024  0.3382  0.4164  1.0000
```

Creando Training y Test Datasets

Cogeremos las primeras 469 filas para el training dataset y las otras 100 para el test dataset. Podemos hacerlo de esta manera si los datos no han sido ordenados y son aleatorios.

```
dc_train <- datoscancer_norm[1:469,]
dc_test <- datoscancer_norm[470:569,]
```

Cuando construimos los dataset excluimos la columna diagnosis. Esta es la columna diagnosis para train y test:

```
datos_train_labels <- datacancer[1:469,1]
datos_test_labels <- datacancer[470:569,1]
```

Utilizaremos esta columna más adelante para evaluar nuestra clasificación.

Training un modelo en nuestros datos

La función knn() está en el paquete "class", para cada observación la función identificará el k-vecino más cercano, usando la distancia Euclidean, donde "k" es un número especificado.

```
library(class)
```

Sintaxis de la función knn():

p <- knn(train,test,class,k) * train es una data frame con los datos numericos (training dataset) * test es un data frame con los datos numéricos (test dataset) * class es un vector de factores con la clasificación de cada fila en el train dataset * k es un entero que indica el numero de vecinos.

```
datacancer_test_pred <-  knn(train=dc_train,test=dc_test,cl=datos_train_labels,k=3)
```

Nos devuelve un vector con la prediccion para cada fila del test dataset (factores benignos o malignos) ##Evaluación del modelo

El siguiente paso es evaluar como de bien ha ajustado las predicciones en el vector "datacancer_test_pred"

```
library(gmodels)
```

```
## Warning: package 'gmodels' was built under R version 3.3.1
```

Podemos cruzar el **datos_test_labels** con el **datacancer_test_pred** para ver si coinciden los dos vectores.

```
CrossTable(x=datos_test_labels,y=datacancer_test_pred, prop.chisq = FALSE )
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##                   | datacancer_test_pred
## datos_test_labels |    Benign | Malignant | Row Total |
## ------------------|-----------|-----------|-----------|
##            Benign |        72 |         5 |        77 |
```

```
##                   |     0.935 |     0.065 |     0.770 |
##                   |     0.986 |     0.185 |           |
##                   |     0.720 |     0.050 |           |
## ------------------|-----------|-----------|-----------|
##         Malignant |         1 |        22 |        23 |
##                   |     0.043 |     0.957 |     0.230 |
##                   |     0.014 |     0.815 |           |
##                   |     0.010 |     0.220 |           |
## ------------------|-----------|-----------|-----------|
##      Column Total |        73 |        27 |       100 |
##                   |     0.730 |     0.270 |           |
## ------------------|-----------|-----------|-----------|
##
##
```

El la celda superior izquierda están los TN, 72 valores de 100, indican los casos que eran Benignos y el knn algoritmo los identifica correctamente. La celda inferior derecha son los casos Malignos que el algoritmo clasifica correctamente.TP.

Los otros 6 casos son los que no ha clasificado correctamente.Hay un caso el el nivel inferior izquierdo que son los Falsos Negativos, el algoritmo los clasificó como Benignos y en realidad fueron Positivos. Los otros 5 casos son Falsos Positivos, los clasificó como Malignos pero realmente son Benignos.

Un 1% son falsos Negativos, son los verdaderamente peligrosos. Podemos probar otra iteracion del modelo para ver si podemos reducir el número de valores incorrectos.

Improving Model performance

vamos a intentar dos variaciones: * Un método alternativo para escalar las variables numéricas * Probaremos diferentes valores de k

Z-score estandarización

Con tumores malignos podemos ver algunos "outliers" valores extremos, estos tienen que tener más peso en el cálculo de las distancias. Las función scale() utiliza la estandarización z-score.

```
data_z <- as.data.frame(scale(datacancer[-1]))

summary(data_z$area_mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.4530 -0.6666 -0.2949  0.0000  0.3632  5.2460
```

La media de una variable estandarizada z-score tiene que ser 0. Una variable mayor que 3 y menor que -3 indica valores extremos raros.

Como hicimos antes hay que dividir los datos en 2 partes:

```
train <- data_z[1:469,]
test <- data_z[470:569,]
train_labels <- datacancer[1:469,1]
test_labels <- datacancer[470:569,1]
test_pred <- knn(train=train,test=test,cl=train_labels,k=21)
CrossTable(x=test_labels,y=test_pred,prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  100
##
##
##              | test_pred
##  test_labels |    Benign | Malignant | Row Total |
## -------------|-----------|-----------|-----------|
##       Benign |        77 |         0 |        77 |
##              |     1.000 |     0.000 |     0.770 |
##              |     0.975 |     0.000 |           |
##              |     0.770 |     0.000 |           |
## -------------|-----------|-----------|-----------|
##    Malignant |         2 |        21 |        23 |
##              |     0.087 |     0.913 |     0.230 |
##              |     0.025 |     1.000 |           |
##              |     0.020 |     0.210 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        79 |        21 |       100 |
##              |     0.790 |     0.210 |           |
## -------------|-----------|-----------|-----------|
##
##
```

En este caso hemos clasificado el 98% de los casos correctamente, con respecto al 94% anterior.

Alternativos valores de K