Trabajo 3: AJUSTE DE MODELOS LINEALES

Nuria Rodríguez Barroso, Juan Luis Suárez Díaz.

26 de mayo de 2017

Clasificación.

Comprensión del problema a resolver.

Para el problema de clasificación hemos elegido la base de datos de *South African Heart Disease*, que almacena una muestra recapituladora de hombres en alto riesgo de cardiopatía en la región de Western Cape, en Sudáfrica. Hay dos casos de CHD (0 o 1). Algunos de los hombres que tienen CHD positivo, se han sometido a un tratamiento de reducción de la presión sanguínea, siendo los datos que aparecen en la muestra posteriores a estos tratamientos. Estos datos datan de 1983.

Nuestra base de datos consta de 462 muestras donde cada una de ellas cuenta con 10 atributos, uno de ellos la variable de respuesta. Los diferentes atributos a tratar son:

- **sbp:** presión arterial sistólica. Toma valores entre 101 y 218, siendo la media 138.3.
- **tobacco:** tabaco acumulativo (en kg). Toma como valor mínimo 0 y máximo 31.2. En este caso la media es de 3.6356.
- **ldl:** lipoproteína de baja densidad (colesterol). Toma valores entre 0.98 y 15.330, siendo la media 4.74.
- **adiposidad:** adiposi Tomando valores entre 6.74 y 42.49, siendo la media de los valores 25.41.
- **famhist:** historial familiar de cardiopatías. Toma como valores {Present, Absent}, habiendo del primer tipo 270 y del segundo 192.
- **typea:** Personalidad Tipo-A (mide el grado de estrés en el día a día). Toma valores entre 13 y 78, siendo la media 53.1.
- **obesity:** Obesidad. Toma valores entre 14.7 y 46.58, siendo la media 26.04.
- **alcohol:** Actual consumición de alcohol. Toma valores entre 0 y 147.19, encontrándose el valor medio en 17.04.
- **age:** Edad de los hombres al comienzo de las pruebas. Se encuentra entre 15 y 64 años, siendo la edad media 42.82.
- **chd:** Variable de respuesta, indica si se tiene o no alguna cardiopatía. Toma como valores {0,1}, siendo la media 0.3463.

```
##       sbp           tobacco             ldl           adiposity
##  Min.   :101.0   Min.   : 0.0000   Min.   : 0.980   Min.   : 6.74
##  1st Qu.:124.0   1st Qu.: 0.0525   1st Qu.: 3.283   1st Qu.:19.77
##  Median :134.0   Median : 2.0000   Median : 4.340   Median :26.11
##  Mean   :138.3   Mean   : 3.6356   Mean   : 4.740   Mean   :25.41
##  3rd Qu.:148.0   3rd Qu.: 5.5000   3rd Qu.: 5.790   3rd Qu.:31.23
##  Max.   :218.0   Max.   :31.2000   Max.   :15.330   Max.   :42.49
##     famhist        typea          obesity         alcohol
##  Absent :270   Min.   :13.0   Min.   :14.70   Min.   :  0.00
##  Present:192   1st Qu.:47.0   1st Qu.:22.98   1st Qu.:  0.51
##                Median :53.0   Median :25.80   Median :  7.51
##                Mean   :53.1   Mean   :26.04   Mean   : 17.04
##                3rd Qu.:60.0   3rd Qu.:28.50   3rd Qu.: 23.89
##                Max.   :78.0   Max.   :46.58   Max.   :147.19
##       age              chd
##  Min.   :15.00   Min.   :0.0000
##  1st Qu.:31.00   1st Qu.:0.0000
```

```
##  Median :45.00   Median :0.0000
##  Mean   :42.82   Mean   :0.3463
##  3rd Qu.:55.00   3rd Qu.:1.0000
##  Max.   :64.00   Max.   :1.0000
```

Preprocesado de datos.

Como podemos observar en el breve estudio de los diferentes atributos que vamos a trabajar, cada uno toma valores en una franja muy diferente, lo que hace primar unos atributos sobre otros en los métodos basados en distancias. Además, algunos atributos presentan una gran asimetría, lo cual también es conveniente evitar. Por estos motivos y otros más que estudiaremos a continuación, tenemos que preprocesar los datos.

Modificación de los atributos cualitativos.

Tenemos que convertir los atributos cualitativos en atributos numéricos para que las funciones que usemos más adelante puedan trabajar con ellos. En nuestro problema concreto, solo contamos con un atributo cualitativo: *famhist*, que toma los valores {Present, Absent}. Convertimos este atributo en el atributo *present_famhist*, que tomará el valor 1, cuando *famhist* tomaba el valor Present y 0 en el otro caso. Así, el atributo *present_famhist* tomará valores en {0,1}.

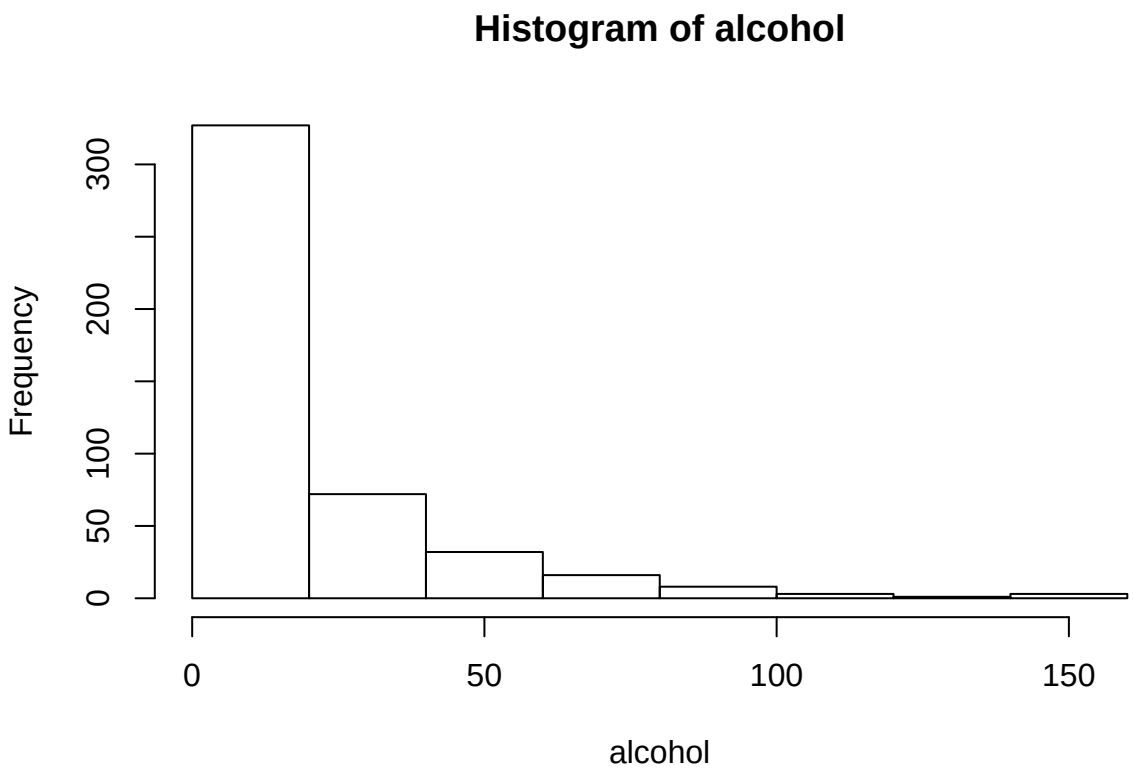
Para el resto de pasos, utilizaremos una función llamada *preProcess()*, la cual terminará con el preprocesado de los datos. Esta función realizará los siguientes cambios:

Tratamiento de la asimetría con BoxCox.

Como ya hemos comentado anteriormente, hay varios atributos que presentan una alta asimetría, lo cual podría hacer que los métodos de predicción que apliquemos a continuación obtengan resultados peores. Para solucionar esto, utilizaremos un método llamado BoxCox. Este método se basa en la transformación potencial según un valor (λ) para aumentar la correlación entre las variables. Para elegir la mejor potencia (mejor λ), se busca entre los λ que proporcionen un menor error residual. Aunque en la práctica, esto se realizará de manera automátcica con la función *preProcess()* vamos a ver cómo funciona en el caso de un atributo. Para que se vea mejor el funcionamiento, vamos a elegir el atributo que presente una mayor asimetría. Para ello, ordenamos los atributos en función de su asimetría:

```
##         alcohol         tobacco             ldl             sbp
##       2.2977031       2.0657278       1.3045896       1.1729355
##         obesity             chd             age           typea
##       0.8993498       0.6438924       0.3792590       0.3441914
## present_famhist       adiposity
##       0.3414684       0.2132541
```

Si dibujamos el histograma correspondiente al atributo con mayor asimetría, *alcohol* corroboramos que los datos se encuentran muy concentrados en los primeros valores que este atributo toma.



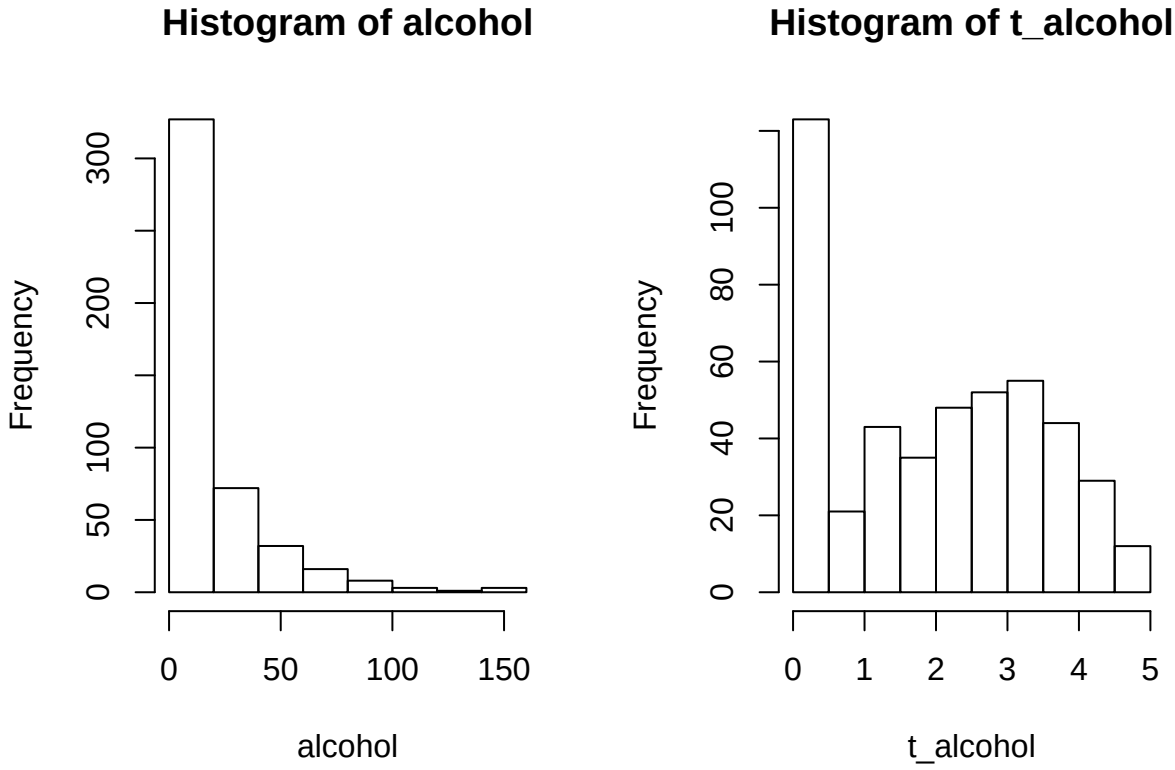
Vamos a aplicar ahora el método *BoxCox* y veremos cómo mejora la simetría del atributo.

```
## Box-Cox Transformation
##
## 462 data points used to estimate Lambda
##
## Input data summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.51    7.51   17.04   23.89  147.20
##
## Lambda could not be estimated; no transformation is applied
```

Como observamos, no se encuentra un λ válido para realizar la transformación, y esto se debe a que entre los valores que toma el atributo se encuentra el 0, punto en el cual no está definida la función logaritmo. Para solucionar esto, realizamos una translación de los datos de la forma: datos = min(datos) + 1 + datos, para así conseguir que el mínimo de los datos se desplace a 1. Tras realizar esta transformación obtenemos:

```
## [1] "La asimetría del alcohol transformado es:"
```

```
## [1] 0.02949713
```



Para aplicar esta transformación a todos los atributos con el λ correspondiente, pasaremos como parámetro al método *preProcess* que realice el método *BoxCox* y todo esto se hará de forma automática.

Eliminación de atributos con PCA.

El algoritmo PCA (Principal Components Analysis) es un filtro no supervisado que es de gran utilidad cuando disponemos de una base de datos con un gran número de atributos, entre los que algunos pueden ser redundantes o irrelevantes. Como nuestra base de datos solo consta de 10 atributos, no es necesaria la aplicación de este método.

Centrar y escalar.

También es conveniente centrar y escalar las variables para que no prioricen unas sobre otras, dado que puede ser útil en algunos métodos que utilizaremos más adelante. Al escalar una variable lo que se está haciendo es dividir cada dato entre la desviación típica del conjunto de datos, transformándo así el conjunto de datos en un conjunto de varianza 1. En cuanto a centrar el conjunto de datos, lo que se hace es restar a cada dato la media del conjunto, transformándolo así en un conjunto de media 0. Al escalar y centrar a la vez, estamos aplicando la transformación $X \leftarrow (X - \mu)/\sigma$, normalizando así el conjunto a un conjunto de media 0 y varianza 1. Esto nos permite tener los atributos normalizados, y además manteniendo las mismas distribuciones entre los distintos atributos, como veremos más adelante en la regresión. El método *preProcess* se encargará de centrar las variables en 0 y de escalar las variables para tener varianza unitaria. Para ello, bastaría con pasar como argumento *scale* y *center* cuando llamemos al método *preProcess*.

Llamada al método PreProcess.

Una vez entendidas las modificaciones que vamos a realizar a los datos, utilizaremos el método preProcess que se encarga de realizar todas estas modificaciones sobre el conjunto de datos pasado como argumento. Como ya hemos comentado, no vamos a aplicar el método *PCA*, luego la llamada quedaría de la siguiente forma:

```
ObjetoTrans = preProcess(sahd[,names(sahd)!="chd"],method = c("BoxCox","center","scale"))
sahdTrans <- predict(ObjetoTrans,sahd)
```

Los conjuntos de validación, training y test usados.

A continuación pasaremos a explorar los distintos modelos sobre los que resolver el problema de clasificación para nuestro conjunto de datos. El procedimiento de validación que usaremos consistirá en tomar múltiples veces distintos conjuntos de entrenamiento sobre nuestro conjunto de datos (una vez transformados), con los que aprenderemos el modelo. Usaremos el resto del conjunto como test para evaluar cómo de bien predice el modelo aprendido con nuevos datos. Las proporciones utilizadas serán del 70 % de los datos para train, y el 30 % para test.

Selección de clases de funciones a usar

Los modelos que vamos a intentar ajustar son los proporcionados por la función `glm` (Generalized Linear Models) de `R`. Las familias que vamos a considerar para el ajuste son (!!!BUSCAR QUE SON !!!):

- **Binomial**, con link **logit**. Regresión logística.
- **Binomial**, con link **probit**. Regresión logística. (?)
- **Binomial**, con link **cauchit**. Regresión logística. (?)
- **Gaussiana**, con link **identity**. Regresión lineal (?)
- **Gaussiana**, con link **log**.
- **Poisson**. Regresión de Poisson.
- **Quasi**.
- **Quasibinomial.**
- **Quasipoisson.**

Una vez definidos los modelos y las funciones a usar, procedemos al ajuste de los distintos modelos y al análisis de sus errores:

```
##             Ein      Eout
##  [1,] 0.2558824 0.2727338
##  [2,] 0.2574923 0.2727338
##  [3,] 0.2498452 0.2751079
##  [4,] 0.2566563 0.2733813
##  [5,] 0.2490093 0.2676259
##  [6,] 0.2506192 0.2665468
##  [7,] 0.2566563 0.2733813
##  [8,] 0.2558824 0.2727338
##  [9,] 0.2506192 0.2665468
```

Obtenemos que el modelo que mejores resultados proporciona es el de Poisson. Hay que destacar que los resultados de Poisson coinciden con los de quasipoisson, pero elegimos el de Poisson por ser más conocido.

Regularización

A continuación nos planteamos la necesidad de regularización, sobre el mejor modelo que hemos obtenido. Para ello utilizamos la regularización lasso (least absolute shrinkage and selection operator). Lasso es un método que lleva a cabo la regularización a la misma vez que realiza selección de características.

El objetivo se basa en reducir el error de predicción, para ello, se ocupa de reducir la función:

$$R(\beta) = \sum_{i=1}^{n}(y_i - x_i\omega)^2 + \lambda \sum_{i=1}^{p} |\omega_i|$$

donde n es el número de muestras y p el número de atributos y w el vector de pesos solución. Así, obtiene diferentes valores de λ. Entre estos valores devueltos, podemos considerar dos de ellos:

- *lambda.min*, que nos devuelve el valor del menor λ obtenido.
- *lambda.1se*, (1se significa one-stand-error) ??????????

Podemos contemplar cualquiera de estos valores de λ para regularizar nuestro modelo.

Para contestar a la pregunta de si era necesario aplicar regularización a nuestra base de datos, realizamos 100 experimentos en los que, para diferentes subconjuntos de datos de nuestra muestra calculamos el error al regularizar con ambos valores de λ y al no regularizar. Debemos quedarnos con el modelo que menos error presente.

```
## [1] 0.2655396
## [1] 0.2780576
## [1] 0.2641007
```

Como podemos observar, el error medio obtenido es menor con el modelo sin regularizar, obteniendo así una respuesta negativa a la pregunta. Por tanto, seguiremos con el modelo sin regularización. <- JUANLU -> LA TIA COMENTÓ AQUI ALGO DE LAS VARIANZAS CHICAS??

A modo de ampliación, comentar que el método de regularización lasso, cuando devuelve valores de λ nulos significa que está despreciando estos atributos para la predicción. Si imprimimos los coeficientes correspondientes al valor de *lambda.1se* observamos que los atriburos que no selecciona para la predicción son: sbp, adiposity, typea, obesity y alcohol.

```
## 10 x 1 sparse Matrix of class "dgCMatrix"
##                              1
## (Intercept)     -1.21372567
## sbp              .
## tobacco          0.07874346
## ldl              0.13039387
## adiposity        .
## present_famhist  0.17546991
## typea            .
## obesity          .
## alcohol          .
## age              0.33868560
```

A continuación, en la selección del número de atributos a utilizar, veremos que dichos atributos son, en efecto, algunos de los que participan en menos combinaciones "óptimas", por lo tanto, serán algunos de los menos relevantes.

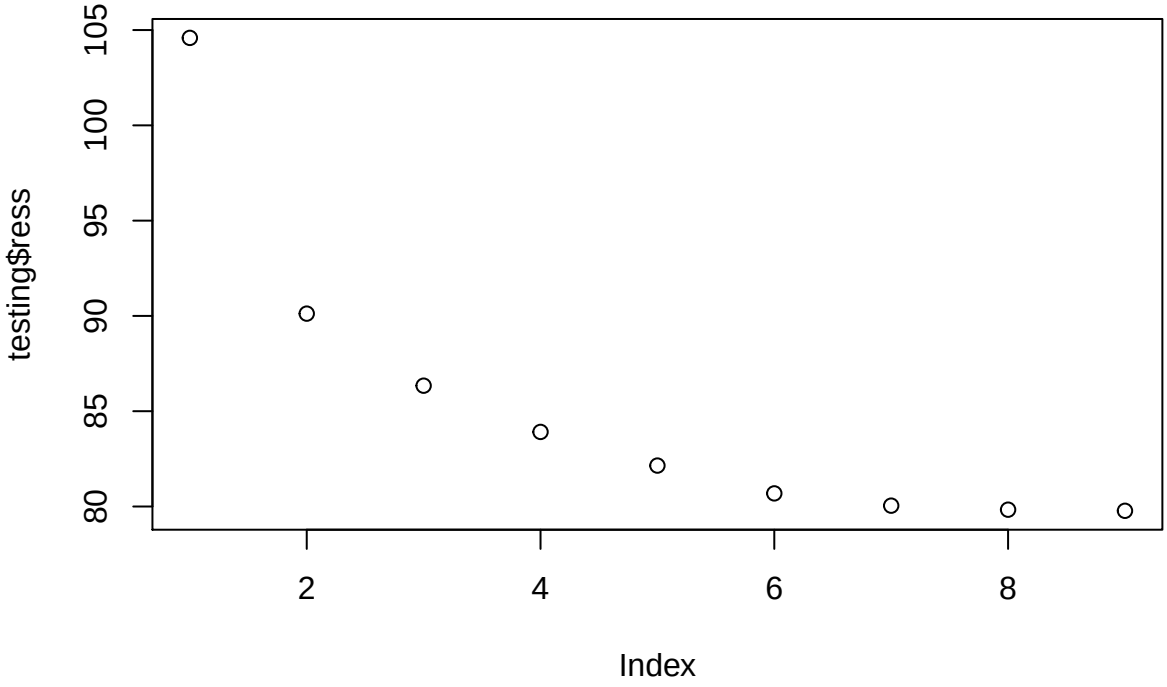
Optimización del número de atributos para el modelo seleccionado

Para optimizar el número de atributos para el modelo seleccionado, vamos a hacer uso de la función llamada *regsubsets*, esta función junto con las opciones de *method = "exhaustive* y *nbest = 1* realizará una búsqueda exhaustiva del mejor atributo (el que produce menor error cuadrático), la mejor pareja de atributos, el mejor trío, etcétera.

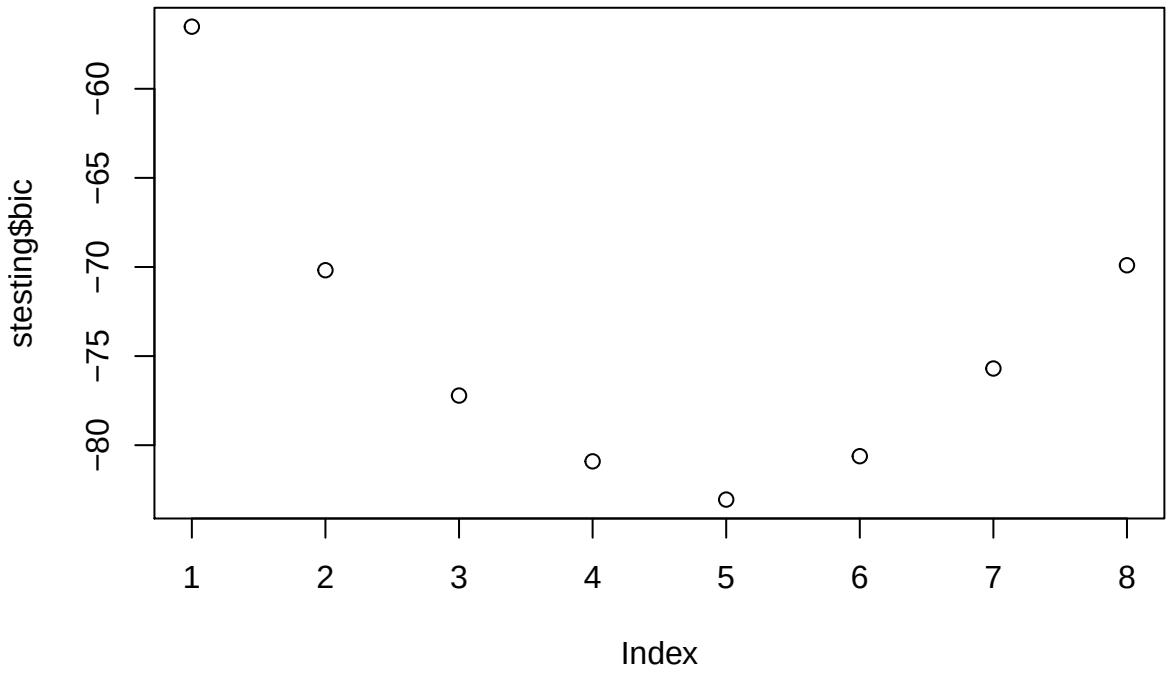
El método nos proporciona el siguiente esquema en el que podemos apreciar las combinaciones de atributos elegidos.

```
##          sbp tobacco ldl adiposity present_famhist typea obesity alcohol
## 1  ( 1 ) " " " "     " " " "       " "              " "   " "     " "
## 2  ( 1 ) " " " "     " " " "       "*"              " "   " "     " "
## 3  ( 1 ) " " "*"     " " " "       "*"              " "   " "     " "
## 4  ( 1 ) " " "*"     " " " "       "*"              "*"   " "     " "
## 5  ( 1 ) " " "*"     "*" " "       "*"              "*"   " "     " "
## 6  ( 1 ) " " "*"     "*" " "       "*"              "*"   "*"     " "
## 7  ( 1 ) " " "*"     "*" "*"       "*"              "*"   "*"     " "
## 8  ( 1 ) "*" "*"     "*" "*"       "*"              "*"   "*"     " "
##          age
## 1  ( 1 ) "*"
## 2  ( 1 ) "*"
## 3  ( 1 ) "*"
## 4  ( 1 ) "*"
## 5  ( 1 ) "*"
## 6  ( 1 ) "*"
## 7  ( 1 ) "*"
## 8  ( 1 ) "*"
```

En este punto nos planteamos cuántos atributos utilizar para nuestro modelo. Claramente, cuantos más atributos utilicemos menor será el error producido. Esto lo podemos corroborar dibujando la gráfica del error por mínimos cuadrados en función del número de atributos elegidos.



Sin embargo, debemos de plantearnos qué número de atributos sería el óptimo si penalizáramos también el número de atributos utilizados. Es decir, cuándo una función que combine el error producido junto con el número de atributos utilizado se minimice. Para ello utilizamos la función *BIC*, cuya gráfica en función del número de atributos podemos observar en la siguiente gráfica: <- EXPLICAR CÓMO FUNCIONA BIC, NO LO ENTIENDO NADA BIEN



Dicha función alcanza un mínimo con 5 atributos, por tanto, este será el número de atributos que utilizaremos para nuestro modelo.

Transformación de atributos

A continuación, nos planteamos la búsqueda de una transformación polinómica de los atributos para la cual nuestro modelo tenga una mayor capacidad de aprender y predecir nuestro conjunto de datos. Para ello utilizamos un algoritmo de búsqueda greedy. Para cada atributo, vamos eligiendo distintos exponentes y nos quedamos con el exponente que proporcione menor error de validación. Cuando llegamos al siguiente atributo, aplicamos el mismo procedimiento, manteniendo para los atributos anteriores el mejor exponente encontrado. Además, como cuando dos modelos proporcionan resultados similares siempre es mejor quedarse con el más simple, fijaremos una tolerancia para la cual, si no hay mejoras significativas en la nueva transformación, nos quedemos con la transformación mejor obtenida previamente, que tendrá un exponente menor y por tanto será más simple el ajuste.

Aplicamos el algoritmo. Los resultados obtenidos son:

```
## Attr =  1 , Exp =  1 ,  Ein =  0.2519195 Eout =  0.2604317
## Attr =  1 , Exp =  2 ,  Ein =  0.2577709 Eout =  0.2764029
## Attr =  1 , Exp =  3 ,  Ein =  0.2595046 Eout =  0.275036
## Attr =  1 , Exp =  4 ,  Ein =  0.2628793 Eout =  0.2694964
## Attr =  1 , Exp =  5 ,  Ein =  0.2603096 Eout =  0.2758993
## Attr =  1 , Exp =  6 ,  Ein =  0.2588545 Eout =  0.2768345
## Attr =  2 , Exp =  1 ,  Ein =  0.2510836 Eout =  0.2667626
## Attr =  2 , Exp =  2 ,  Ein =  0.2625697 Eout =  0.2817986
## Attr =  2 , Exp =  3 ,  Ein =  0.2580495 Eout =  0.2742446
## Attr =  2 , Exp =  4 ,  Ein =  0.2675851 Eout =  0.2836691
## Attr =  2 , Exp =  5 ,  Ein =  0.2583901 Eout =  0.283741
## Attr =  2 , Exp =  6 ,  Ein =  0.2706192 Eout =  0.2838849
## Attr =  3 , Exp =  1 ,  Ein =  0.2540867 Eout =  0.2577698
## Attr =  3 , Exp =  2 ,  Ein =  0.2531269 Eout =  0.2573381
## Attr =  3 , Exp =  3 ,  Ein =  0.2508359 Eout =  0.2663309
## Attr =  3 , Exp =  4 ,  Ein =  0.2497523 Eout =  0.2682734
```

```
## Attr =  3 , Exp =  5 ,  Ein =  0.2502477 Eout =  0.2674101
## Attr =  3 , Exp =  6 ,  Ein =  0.25 Eout =  0.268777
## Attr =  4 , Exp =  1 ,  Ein =  0.250774 Eout =  0.2628058
## Attr =  4 , Exp =  2 ,  Ein =  0.256192 Eout =  0.2576259
## Attr =  4 , Exp =  3 ,  Ein =  0.253839 Eout =  0.2654676
## Attr =  4 , Exp =  4 ,  Ein =  0.251517 Eout =  0.271223
## Attr =  4 , Exp =  5 ,  Ein =  0.2566254 Eout =  0.2581295
## Attr =  4 , Exp =  6 ,  Ein =  0.2532198 Eout =  0.2671942
## Attr =  5 , Exp =  1 ,  Ein =  0.2502786 Eout =  0.2646763
## Attr =  5 , Exp =  2 ,  Ein =  0.2703715 Eout =  0.2873381
## Attr =  5 , Exp =  3 ,  Ein =  0.253839 Eout =  0.2727338
## Attr =  5 , Exp =  4 ,  Ein =  0.2732508 Eout =  0.2903597
## Attr =  5 , Exp =  5 ,  Ein =  0.2613932 Eout =  0.2667626
## Attr =  5 , Exp =  6 ,  Ein =  0.2762848 Eout =  0.2856115

## Vector de exponentes:  1 1 1 1 1

## Eout estimado:  0.2646763
```

Vemos que, para exponentes de hasta tamaño 6 no se aprecian mejoras significativas con respecto a los coeficientes lineales iniciales.

Conclusiones.

Por tanto, el modelo óptimo escogido es un modelo lineal que utiliza los atributos tobacco, ldl, present_famhist, typea y age. A este conjunto de datos le aplicamos una regresión de Poisson obteniendo un error estimado del 26.46763%. -> CONCLUSION: EL MODELO ES UN MOJONSILLO. Aqui hay que añadir mucha mucha más literatura.

Regresión.

Para el problema de regresión hemos elegido la base de datos de *Los Ángeles Ozone*, la cual se centra en medir el nivel de concentración de ozono en la atmósfera. Para ello, se realizaban 8 mediciones hechas diariamente en Los Ángeles durante el año 1976. Aunque la idea era obtener el nivel de ozono para todos los días del año, algunos datos se han perdido así que no contiene todos los días del año (en concreto contiene 330 días). La base de datos consta de 10 atributos que son los siguientes:

- **ozone**: Es la variable de respuesta, mide la elevación máxima del ozono.
- **vh**: Vandenberg 500 mb Height
- **wind**: Velocidad del viento, medida en mph.
- **humidity:** Tanto por ciento de humedad.
- **temp:** Temperatura
- **ibh:**
- **dpg:** Gradiente de presión de Daggot.
- **ibt:**
- **vis:** La visibilidad medida en millas.
- **day:** Día del año en el que se realizó la medición.