Trabajo 3

Francisco Javier Caracuel Beltrán

27 de Mayo de 2017

A continuación, se detalla el resultado del trabajo 3 de Aprendizaje Automático. Este documento incluye todo el código utilizado en el archivo *trabajo3.R*, así como los comentarios añadidos durante su creación.

Como se entrega el fichero *trabajo3.Rmd*, no se incluyen puntos de parada en el código fuente que se encuentra en el fichero *trabajo3.R*.

Se establece el directorio de trabajo

```
setwd("/mnt/A0DADA47DADA18FC/Fran/Universidad/3º/2 Cuatrimestre/AA/Mis prácticas/Práctica 3/datos")
```

Se establece la semilla para la generación de datos aleatorios

```
set.seed(1822)
```

Número de decimales que tendrán los porcentajes de error en la salida

```
perDec = 3
```

Ajuste de Modelos Lineales

Problema de clasificación: South African Heart Disease

1. Comprender al problema a resolver.

El problema a resolver consiste en predecir la posibilidad de padecer una enfermedad cardíaca de alto riesgo siendo varón en la provincia Western Cape, en Sudáfrica. Para poder predecir esta enfermedad se tiene una muestra de datos de *463* varones. Hay muestras que se han recogido antes del tratamiento y otras que han sido recogidas después.

Las distintas características que se obtienen en cada muestra son:

- *sbp*: (presión arterial sistólica).
- *tobacco*: tabaco acumulado (kg).
- *ldl*: (lipoproteína de baja densidad).
- *adiposity*: grasa.
- *famhist*: antecendentes familiares de riesgo cardiovascular (presente/ausente).
- *typea*: (type-A behavior)
- *obesity*: IMC (Índice de masa corporal).
- *alcohol*: consumo actual de alcohol.
- *age*: edad del varón de la muestra.
- *chd*: respuesta obtenida (0: no presente, 1: presente).

2. Los conjuntos de training, validación y test usados en su caso.

Para realizar la predicción solo se cuenta con un conjunto de datos con *463* muestras. No se tienen conjuntos de datos de *train*, *validación* o *test*, por lo que se va a separar toda la muestra de datos en dos conjuntos. El conjunto de *train* contendrá el *70%* de las muestras y el conjunto de *test* contendrá el *30%*. Antes de hacer la separación de los datos en los dos conjuntos se va a realizar su preprocesado.

3. Preprocesado los datos: Falta de datos, categorización, normalización, reducción de dimensionalidad, etc.

Se pretende controlar la cantidad de datos que se tienen, el tipo de muestra (*train* o *test*), comprobar si existen datos cualitativos. Cuando se ajuste el modelo final se determinará la reducción de dimensionalidad y demás procesamiento necesario. En este caso, no se normalizarán las características porque no se va a trabajar por similaridad, como en *K-NN*.

```
# Se pueden leer los datos desde la url
#read.table("https://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data",
#sep=",",head=T,row.names=1)

# En este caso, se han descargado los datos a un fichero del directorio de
# trabajo.
# Se indica que la separación entre datos de cada muestra viene identificada
# por "," y que el fichero de datos tiene cabecera.
chd = read.table("SAHD.data", sep=",", head=T, row.names=1)

# Hay que comprobar si algún valor del conjunto de datos es de carácter
# cualitativo, por lo que se muestran las primeras muestras con la función
# "head".
head(chd)
```

```
##   sbp tobacco  ldl adiposity famhist typea obesity alcohol age chd
## 1 160   12.00 5.73     23.11 Present    49   25.30   97.20  52   1
## 2 144    0.01 4.41     28.61  Absent    55   28.87    2.06  63   1
## 3 118    0.08 3.48     32.28 Present    52   29.14    3.81  46   0
## 4 170    7.50 6.41     38.03 Present    51   31.99   24.26  58   1
## 5 134   13.60 3.50     27.78 Present    60   25.99   57.34  49   1
## 6 132    6.20 6.47     36.21 Present    62   30.77   14.14  45   0
```

```
# De los datos obtenidos, la única columna que tiene etiquetas es la columna
# "famhist".
# Hay que eliminar las etiquetas y sustituiregas por números. Se pondrá 1 a
# los valores "Present" y 2 a los valores "Absent".
chd = within(chd, famhist <- as.numeric(factor(famhist, labels = c(1, 2))))

# Se comprueba que se han cambiado los valores de la columna "famhist".
head(chd)
```

```
##   sbp tobacco  ldl adiposity famhist typea obesity alcohol age chd
## 1 160   12.00 5.73     23.11       2    49   25.30   97.20  52   1
## 2 144    0.01 4.41     28.61       1    55   28.87    2.06  63   1
## 3 118    0.08 3.48     32.28       2    52   29.14    3.81  46   0
## 4 170    7.50 6.41     38.03       2    51   31.99   24.26  58   1
## 5 134   13.60 3.50     27.78       2    60   25.99   57.34  49   1
## 6 132    6.20 6.47     36.21       2    62   30.77   14.14  45   0
```

```
# Se comprueba si existe algún valor perdido. Para eso se utiliza la función
# which() y la función is.na(), que nos indican, respectivamente, la posición
# en la que ocurre algún hecho y si alguna posición no es correcta.
which(is.na(chd))
```

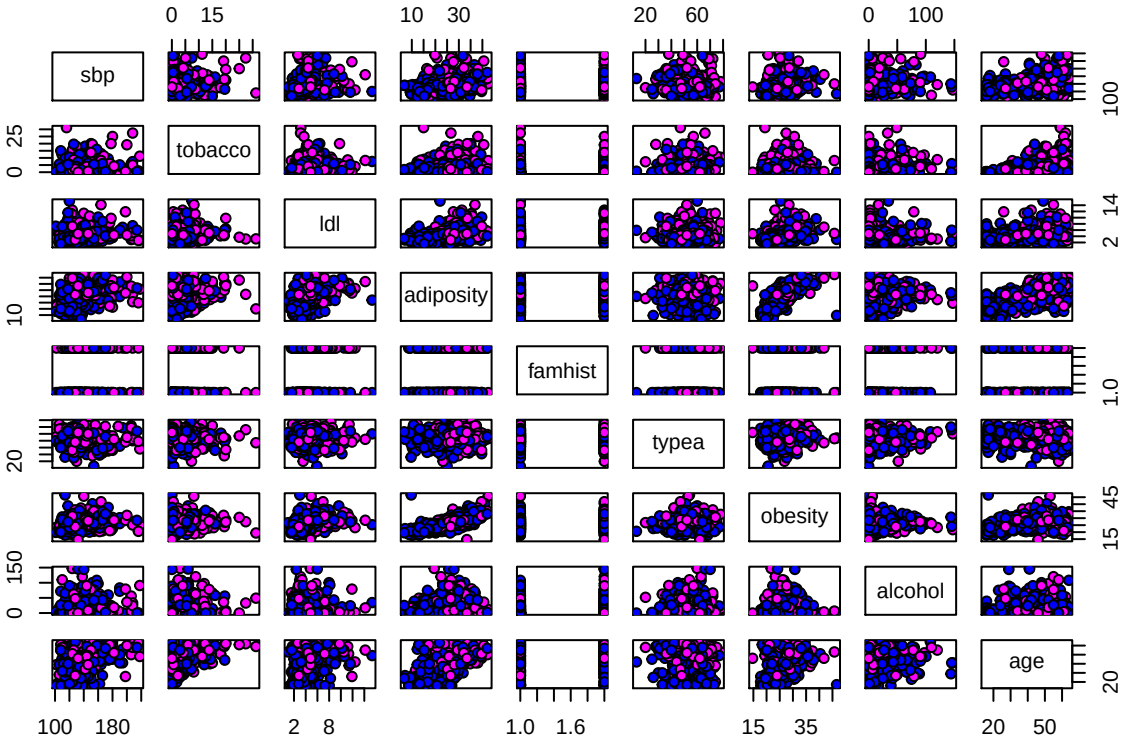
```
## integer(0)
```

```
# En este caso no se obtiene ninguna respuesta, por lo que no se encuentran
# valores perdidos en este conjunto de datos.
```

```r
# Se simplifica el prefijo chd en las operaciones para evitar tener que
# escribir continuamente el conjunto de datos.
attach(chd)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     chd
```

```r
# En este punto sí se pueden separar todos los datos en train y test.

# Se guarda el número de columnas para utilizarlo al sacar los datos y así
# no tener que repetir la misma operación en varias ocasiones.
numAttrChd = ncol(chd)

# Se va a utilizar como train una muestra del 70% de los datos del conjunto,
# por lo que se generan númeroDeMuestras*0.7 posiciones aleatoriamente.
posTrain = sample(nrow(chd), round(nrow(chd)*0.7))

# Se guardan los datos que forman parte del train.
#chd.train = chd[posTrain, c(1:numAttrChd-1)]
chd.train = chd[posTrain, c(1:numAttrChd)]

# Se crea un vector con las etiquetas de los datos de entrenamiento
#chd.train.label = chd[posTrain, c(numAttrChd-1, numAttrChd)]

# Se guardan los datos que forman parte del test. Serán todos aquellos que no
# sean del train.
#chd.test = chd[-posTrain, c(1:numAttrChd-1)]
chd.test = chd[-posTrain, c(1:numAttrChd)]

# Se crea un vector con las etiquetas de los datos de test
#chd.test.label = chd[-posTrain, c(numAttrChd-1, numAttrChd)]

# Se muestra un gráfico con la relación de los atributos entre ellos.
pairs(chd[1:9], pch=21, bg=c(4, 6)[factor(chd$chd)])
```



4. Selección de clases de funciones a usar.

Para predecidr datos se puede utilizar:

- *Regresión Lineal Simple*: con la función *lm()*.
- *Regresión Logística*: con la función *glm()*. Cuenta a su vez con una serie de familias a utilizar. En esta práctica se utilizará *binomial*, *gaussian* y *poisson*.
- *Predicción*: *predict()* permitirá predecir el resultado ajustado por la regresión realizada.
- *Disminución de atributos*: se usa la función *step()* que devuelve un set de atributos con los que se puede representar la mayor parte del conjunto de datos.
- *Regularización*: para realizar la regularización se utilizará la función *potencial*, *exponencial* y *logarítmica*.

5. Discutir la necesidad de regularización y en su caso la función usada.

Se debe comprobar primero qué tipo de muestras se tienen y en base a cómo respondan se realizará la regularización. Si los resultados que se obtienen no se pueden clasificar linealmente, se procederá a aplicar transformaciones para separar todo el conjunto como interese y comprobar si de este modo sí se encuentran las muestras distintas diferenciadas entre ellas. Las transformaciones habituales suelen ser x^n, x^-n, $log(x)$ y $log(x+n)$.

6. Definir los modelos a usar y estimar sus parámetros e hyperparámetros.

El modelo a utilizar es regresión logística. Se podría utilizar *regresión lineal*, *SVM*, *LDA*, *QDA*, u otras técnicas no lineales más avanzadas, pero escapan del interés de esta práctica, por lo que se va a realizar *regresión logística*. Los parámetros a utilizar se comprobarán durante la resolución del problema.

7. Selección y ajuste modelo final.

Se hace regresión logística de la muestra de entrenamiento teniendo en cuenta todos los atributos. La intención es comprobar cuáles de ellos no representan la mayor parte de la muestra y así tener una solución con el menor número de atributos posibles, lo que hará que sea de mayor calidad.

```
chd.glm = glm(chd ~ ., family = gaussian, data = chd.train)
```

Para obtener el menor número de atributos posibles se podría haber utilizado *regsubsets()*, *step()* o *PCA* con el paquete *caret*. En este caso se ha utilizado la función *step()* que, directamente devuelve los atributos mas representativos del conjunto de muestras. Además, indica qué peso tiene cada uno de ellos, por lo que se puede hacer una valoración más exhaustiva del resultado. Se ha inspeccionado la función *step()* para comprobar su funcionamiento y se puede determinar que el pseudocódigo relacionado es:

```
#initial_setting
#for i in steps
#start
#  set_type_scan
#  get_attributes
#  evaluate_attributes
#  result += list_evaluation
#end
#return result
```

El algoritmo de la función *step()* comienza ajustando los parámetros iniciales por defecto que tiene definidos o los que el usuario quiera utilizar. De manera iterativa, en una serie de *pasos* previamente establecidos, obtiene los atributos con los que va a realizar la prueba. En cada iteración selecciona una serie de ellos y evalúa el resultado de la *regresión logística* (en este caso). Finalmente, para cada evaluación realizada, devuelve su resultado, mostrando la *desviación* de cada atributo y el *criterio de información de Akaike*, así como los atributos con los que se han obtenido estos resultados.

Para obtener más información de la que se obtiene solo con la función *step()*, se utiliza *summary()* para ver un informe más completo.

```
summary(step(chd.glm))
```

```
## Start:  AIC=373.99
## chd ~ sbp + tobacco + ldl + adiposity + famhist + typea + obesity +
##     alcohol + age
##
##             Df Deviance    AIC
## - adiposity  1   56.240 372.02
## - alcohol    1   56.241 372.03
## <none>           56.234 373.99
## - obesity    1   56.709 374.70
## - tobacco    1   56.919 375.90
## - sbp        1   57.208 377.53
## - typea      1   57.455 378.93
## - age        1   57.612 379.81
## - ldl        1   57.857 381.18
## - famhist    1   58.698 385.84
##
## Step:  AIC=372.02
## chd ~ sbp + tobacco + ldl + famhist + typea + obesity + alcohol +
##     age
##
##           Df Deviance    AIC
## - alcohol  1   56.247 370.06
## <none>         56.240 372.02
```

```
## - tobacco  1   56.933 373.98
## - obesity  1   57.131 375.10
## - sbp      1   57.218 375.59
## - typea    1   57.457 376.94
## - ldl      1   57.965 379.78
## - age      1   58.226 381.23
## - famhist  1   58.702 383.86
##
## Step:  AIC=370.06
## chd ~ sbp + tobacco + ldl + famhist + typea + obesity + age
##
##           Df Deviance    AIC
## <none>         56.247 370.06
## - tobacco  1   56.938 372.01
## - obesity  1   57.138 373.14
## - sbp      1   57.221 373.61
## - typea    1   57.458 374.94
## - ldl      1   57.999 377.97
## - age      1   58.239 379.30
## - famhist  1   58.703 381.87
##
## Call:
## glm(formula = chd ~ sbp + tobacco + ldl + famhist + typea + obesity +
##     age, family = gaussian, data = chd.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7414  -0.3311  -0.1125   0.3613   1.0251
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.818212   0.243868  -3.355 0.000890 ***
## sbp          0.002953   0.001264   2.336 0.020124 *
## tobacco      0.011944   0.006070   1.968 0.049985 *
## ldl          0.041034   0.013097   3.133 0.001892 **
## famhist      0.186716   0.050343   3.709 0.000246 ***
## typea        0.006295   0.002417   2.604 0.009640 **
## obesity     -0.014359   0.006428  -2.234 0.026195 *
## age          0.006853   0.002052   3.340 0.000937 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1785603)
##
##     Null deviance: 73.467  on 322  degrees of freedom
## Residual deviance: 56.247  on 315  degrees of freedom
## AIC: 370.06
##
## Number of Fisher Scoring iterations: 2
```

Los atributos que se deben tener en cuenta según los resultados obtenidos son:

- *sbp.*
- *tobacco.*

- *ldl.*
- *famhist.*
- *typea.*
- *obesity.*
- *age.*

Si se analizan los coeficientes que han obtenido cada uno de ellos, se puede no tener en cuenta el atributo "*sbp*", "*typea*" y "*age*". Se realizarán dos pruebas, con estos atributos y sin ellos. Si se obtiene el mismo resultado interesa utilizar el menor número de atributos posibles.

Prueba 1

Los coeficientes que se deben tener en cuenta son: *sbp*, *tobacco*, *ldl*, *famhist*, *typea*, *obesity* y *age.*

Se van a utilizar tres familias de funciones para realizar la regresión logística: *binomial*, *gaussian* y *poisson.* Con cada una de ellas se comprobará su resultado y se elegirá la mejor.

Se hace de nuevo regresión lineal con estos coeficientes.

- *binomial*

```
chd.glm = glm(chd ~ sbp + tobacco + ldl + famhist + typea + obesity + age,
              family = binomial,
              data = chd.train)

# Se predicen los resultados de entrenamiento con la regresión logística
predictTrain = predict(chd.glm, chd.train)

# Se calcula la media de error de los datos de entrenamiento
meanTrain.glm = mean((predictTrain > 0.5) != (chd.train[, 'chd'] > 0.5))

# Se predicen los resultados de test con la regresión logística
predictTest = predict(chd.glm, chd.test)

# Se calcula la media de error de los datos de test
meanTest.glm = mean((predictTest > 0.5) != (chd.test[, 'chd'] > 0.5))

print(paste("Ein calculado con Regresión Logística: ",
            round(meanTrain.glm*100, digits = perDec), "%"))
```

```
## [1] "Ein calculado con Regresión Logística:  26.316 %"
```

```
print(paste("Etest calculado con Regresión Logística: ",
            round(meanTest.glm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística:  25.18 %"
```

- *gaussian*

```
chd.glm = glm(chd ~ sbp + tobacco + ldl + famhist + typea + obesity + age,
              family = gaussian,
              data = chd.train)

# Se predicen los resultados de entrenamiento con la regresión logística
predictTrain = predict(chd.glm, chd.train)

# Se calcula la media de error de los datos de entrenamiento
meanTrain.glm = mean((predictTrain > 0.5) != (chd.train[, 'chd'] > 0.5))
```

```
# Se predicen los resultados de test con la regresión logística
predictTest = predict(chd.glm, chd.test)

# Se calcula la media de error de los datos de test
meanTest.glm = mean((predictTest > 0.5) != (chd.test[, 'chd'] > 0.5))

print(paste("Ein calculado con Regresión Logística: ",
            round(meanTrain.glm*100, digits = perDec), "%"))
```

```
## [1] "Ein calculado con Regresión Logística:  26.316 %"
```

```
print(paste("Etest calculado con Regresión Logística: ",
            round(meanTest.glm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística:  25.18 %"
```

- *poisson*

```
chd.glm = glm(chd ~ sbp + tobacco + ldl + famhist + typea + obesity + age,
              family = poisson,
              data = chd.train)

# Se predicen los resultados de entrenamiento con la regresión logística
predictTrain = predict(chd.glm, chd.train)

# Se calcula la media de error de los datos de entrenamiento
meanTrain.glm = mean((predictTrain > 0.5) != (chd.train[, 'chd'] > 0.5))

# Se predicen los resultados de test con la regresión logística
predictTest = predict(chd.glm, chd.test)

# Se calcula la media de error de los datos de test
meanTest.glm = mean((predictTest > 0.5) != (chd.test[, 'chd'] > 0.5))

print(paste("Ein calculado con Regresión Logística: ",
            round(meanTrain.glm*100, digits = perDec), "%"))
```

```
## [1] "Ein calculado con Regresión Logística:  34.985 %"
```

```
print(paste("Etest calculado con Regresión Logística: ",
            round(meanTest.glm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística:  33.813 %"
```

Prueba 2

Los coeficientes que se deben tener en cuenta son: *tobacco*, *ldl*, *famhist* y *obesity*.

Se van a utilizar tres familias de funciones para realizar la regresión logística: *binomial*, *gaussian* y *poisson*. Con cada una de ellas se comprobará su resultado y se elegirá la mejor.

Se hace de nuevo regresión lineal con estos coeficientes.

- *binomial*

```
chd.glm = glm(chd ~ tobacco + ldl + famhist + obesity,
              family = binomial,
              data = chd.train)
```

```
# Se predicen los resultados de entrenamiento con la regresión logística
predictTrain = predict(chd.glm, chd.train)

# Se calcula la media de error de los datos de entrenamiento
meanTrain.glm = mean((predictTrain > 0.5) != (chd.train[, 'chd'] > 0.5))

# Se predicen los resultados de test con la regresión logística
predictTest = predict(chd.glm, chd.test)

# Se calcula la media de error de los datos de test
meanTest.glm = mean((predictTest > 0.5) != (chd.test[, 'chd'] > 0.5))

print(paste("Ein calculado con Regresión Logística: ",
            round(meanTrain.glm*100, digits = perDec), "%"))
```

```
## [1] "Ein calculado con Regresión Logística:  28.483 %"
```

```
print(paste("Etest calculado con Regresión Logística: ",
            round(meanTest.glm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística:  29.496 %"
```

- *gaussian*

```
chd.glm = glm(chd ~ tobacco + ldl + famhist + obesity,
              family = gaussian,
              data = chd.train)

# Se predicen los resultados de entrenamiento con la regresión logística
predictTrain = predict(chd.glm, chd.train)

# Se calcula la media de error de los datos de entrenamiento
meanTrain.glm = mean((predictTrain > 0.5) != (chd.train[, 'chd'] > 0.5))

# Se predicen los resultados de test con la regresión logística
predictTest = predict(chd.glm, chd.test)

# Se calcula la media de error de los datos de test
meanTest.glm = mean((predictTest > 0.5) != (chd.test[, 'chd'] > 0.5))

print(paste("Ein calculado con Regresión Logística: ",
            round(meanTrain.glm*100, digits = perDec), "%"))
```

```
## [1] "Ein calculado con Regresión Logística:  26.935 %"
```

```
print(paste("Etest calculado con Regresión Logística: ",
            round(meanTest.glm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística:  25.18 %"
```

- *poisson*

```
chd.glm = glm(chd ~ tobacco + ldl + famhist + obesity,
              family = poisson,
              data = chd.train)

# Se predicen los resultados de entrenamiento con la regresión logística
```

```
predictTrain = predict(chd.glm, chd.train)

# Se calcula la media de error de los datos de entrenamiento
meanTrain.glm = mean((predictTrain > 0.5) != (chd.train[, 'chd'] > 0.5))

# Se predicen los resultados de test con la regresión logística
predictTest = predict(chd.glm, chd.test)

# Se calcula la media de error de los datos de test
meanTest.glm = mean((predictTest > 0.5) != (chd.test[, 'chd'] > 0.5))

print(paste("Ein calculado con Regresión Logística: ",
            round(meanTrain.glm*100, digits = perDec), "%"))
```

```
## [1] "Ein calculado con Regresión Logística:  34.985 %"
```

```
print(paste("Etest calculado con Regresión Logística: ",
            round(meanTest.glm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística:  33.813 %"
```

De los resultados obtenidos, se descarta la familia "*poisson*" por ser la peor de las tres en ambas pruebas. En la prueba 1, la familia "*binomial*" y "*gaussian*" han obtenido el mismo resultado, pero la familia "*gaussian*" se ha ajustado mejor a la reducción de atributos. Teniendo en cuenta que el error de test obtenido en la prueba 1 y la prueba 2 que se ha obtenido ha sido igual, se confirma la utilización de los atributos tobacco, ldl, famhist y obesity, obteniendo así una reducción de la dimensionalidad del *44%*.

8. Estimacion del error E out del modelo lo más ajustada posible.

El error de *test/out* obtenido ha sido demasiado alto. Este hecho hace pensar que la muestra de datos no es separable linealmente, por lo que habría que intentar buscar transformaciones que permitan separar los datos en dos grupos bien definidos, como ya se hizo en la entrega 2.

Las funciones utilizadas para realizar las transformaciones han sido:

x^n, x^-n, $log(x)$, $log(x+n)$, $exp(x)$, $poly()$

Para aplicar estas transformaciones se han implementado *cinco* bucles *for* en el que se han probado iterativamente las distintas funciones potenciales que se iban generando. No se ha conseguido mejora ninguna con este método, empeorando en muchos casos el error obtenido. De manera *Greedy* se ha utilizado una combinación de las funciones anteriores entre todos los atributos, obteniendo solo en un caso una leve mejora. La mejora obtenida viene dada por la transformación de la función logarítmica en el atributo "*obesity*" de la muestra, que se muestra a continuación:

```
# Se utiliza la familia "gaussian" al ser la que mejor resultado ha ofrecido y
# se aplica la transformación de la función log() al atributo "obesity".
chd.glm = glm(chd ~ tobacco + ldl + famhist + log(obesity),
              family = gaussian,
              data = chd.train)

# Se predicen los resultados de entrenamiento con la regresión logística
predictTrain = predict(chd.glm, chd.train)

# Se calcula la media de error de los datos de entrenamiento
meanTrain.glm = mean((predictTrain > 0.5) != (chd.train[, 'chd'] > 0.5))

# Se predicen los resultados de test con la regresión logística
```

```r
predictTest = predict(chd.glm, chd.test)

# Se calcula la media de error de los datos de test
meanTest.glm = mean((predictTest > 0.5) != (chd.test[, 'chd'] > 0.5))

print(paste("Ein calculado con Regresión Logística: ",
            round(meanTrain.glm*100, digits = perDec), "%"))
```

```
## [1] "Ein calculado con Regresión Logística:  26.935 %"
```

```r
print(paste("Etest calculado con Regresión Logística: ",
            round(meanTest.glm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística:  24.46 %"
```

Se ha podido disminuir el error de *test/out* del *25.18%* al *24.46%*, lo que sigue siendo un resultado alejado de uno de calidad.

9. Discutir y justificar la calidad del modelo encontrado y las razones por las que considera que dicho modelo es un buen ajuste que representa adecuadamente los datos muestrales.

El conjunto de datos que se tiene no es separable linealmente. Se ha intentado un ajuste no lineal para poder transformar la muestra y así clasificarla, pero no se han conseguido grandes mejoras. Considero que el modelo encontrado no es bueno por la muestra en concreto de la que se dispone. Son necesarias técnicas no lineales más avanzadas que sean capaz de ajustar mejor los datos.

Problema de regresión: Prostate Data Info

1. Comprender al problema a resolver.

Estos datos proceden de un estudio que examina la correlación entre el nivel de *antígeno prostático específico* (*PSA*) y un número de medidas clínicas en los hombres a los que se les ha realizado una *prostatectomía radical*.

Para ello se cuenta con *8* atributos:

- *lcavol*: volumen de cáncer.
- *lWeight*: peso de la próstata.
- *age*: edad del paciente.
- *lbph*: cantidad de hiperplasia beningna de próstata.
- *svi*: invasión de vesículas seminales.
- *lcp*: penetración capsular.
- *gleason*: puntuación de Gleason.
- *pgg45*: porcentaje de Gleason.

La salida que se tiene es *lpsam*, que es el antígeno prostático específico.

2. Los conjuntos de training, validación y test usados en su caso.

Para realizar la predicción solo se cuenta con un conjunto de datos con *97* muestras. Ya se tiene en el conjunto de datos una columna que indica si la muestra es de tipo *train* o *test*, por lo que solo se dispondrá a leer a qué tipo de conjunto pertenece cada muestra para agruparlas.

3. Preprocesado los datos: Falta de datos, categorización, normalización, reducción de dimensionalidad, etc.

Se pretende controlar la cantidad de datos que se tienen, el tipo de muestra (*train* o *test*), comprobar si existen datos cualitativos. Cuando se ajuste el modelo final se determinará la reducción de dimensionalidad y demás procesamiento necesario. En este caso, se normalizarán los datos de la columna *pgg45* que viene dado en %.

```
# Se han descargado los datos a un fichero del directorio de trabajo.
# Se indica que la separación entre datos de cada muestra viene identificada
# por una tabulación y que el fichero de datos tiene cabecera.
prost = read.table("prostate.data", sep="\t", head=T, row.names=1)

# En caso de querer cargar los datos sin utilizar este fichero, se pueden
# utilizar las tres instrucciones siguientes, pero no se asegura que funcione
# todo el proceso correctamente.
# data("prostate", package = "ElemStatLearn")
# str(prostate)
# prostate

# Hay que comprobar si algún valor del conjunto de datos es de carácter
# cualitativo, por lo que se muestran las primeras muestras con la función
# "head".
head(prost)
```

```
##       lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
##   train
## 1  TRUE
## 2  TRUE
## 3  TRUE
## 4  TRUE
## 5  TRUE
## 6  TRUE
```

```
# De los datos obtenidos, la única columna con una etiqueta es la última,
# "train", que indica si la muestra pertenece a train o test, por lo que
# cuando se separen los datos y se trabajen con los correspondientes
# subgrupos, ya no se tendrá en cuenta.

# Se comprueba si existe algún valor perdido. Para eso se utiliza la función
# which() y la función is.na(), que nos indican, respectivamente, la posición
# en la que ocurre algún hecho y si alguna posición no es correcta.
which(is.na(prost))
```

```
## integer(0)
```

```
# En este caso no se obtiene ninguna respuesta, por lo que no se encuentran
# valores perdidos en este conjunto de datos.

# Se van a normalizar los datos de la columna pgg45, que se encuentran
# en %. Realmente solo se cambia el valor a su probabilidad correspondiente.
prost[, 'pgg45'] = prost[, 'pgg45']/100
```

```
# Se simplifica el prefijo prost en las operaciones para evitar tener que
# escribir continuamente el conjunto de datos.
attach(prost)
```

```
## The following object is masked from chd:
##
##     age
```

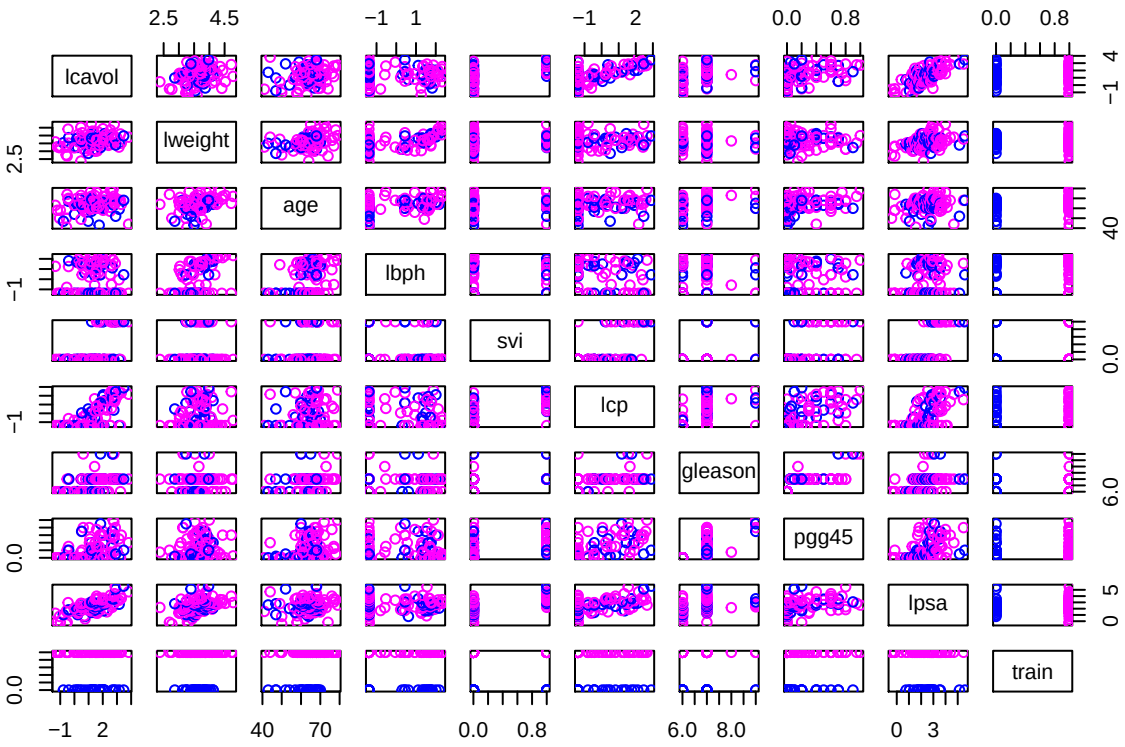
```
# En este punto sí se pueden separar todos los datos en train y test.

# Se guardan los datos que forman parte del train.
prost.train = prost[which(prost[, 'train']) , 1:9]

# Se guardan los datos que forman parte del test. Serán todos aquellos que no
# sean del train.
prost.test = prost[which(!prost[, 'train']) , 1:9]

# Se comprueba que coincida el número de train y test
print(paste("Número de muestras de train: ", nrow(prost.train)))
```

```
## [1] "Número de muestras de train:  67"
```

```
print(paste("Número de muestras de test: ", nrow(prost.test)))
```

```
## [1] "Número de muestras de test:  30"
```

```
# Se muestra un gráfico con la relación de los atributos entre ellos.
pairs(prost, col = c(4, 6)[prost$train + 1])
```



4. Selección de clases de funciones a usar.

Para predecidr datos se puede utilizar:

- *Regresión Lineal Simple*: con la función *lm()*.
- *Regresión Logística*: con la función *glm()*. Cuenta a su vez con una serie de familias a utilizar.
- *Predicción*: *predict()* permitirá predecir el resultado ajustado por la regresión realizada.
- *Disminución de atributos*: se usa la función *step()* que devuelve un set de atributos con los que se puede representar la mayor parte del conjunto de datos.
- *Regularización*: para hacer la regresión con regularización se va a utilizar el paquete *glmnet*, que ya la aplica a una *regresión logística*.

5. Discutir la necesidad de regularización y en su caso la función usada.

Directamente se van a hacer dos pruebas para comprobar la evolución de las muestras. La primera prueba se hará con *regresión lineal*. En la segunda prueba, que será donde se haga la regularización, se usará la función *cv.glmnet()*, que realiza una regresión logística aplicando generalización y haciendo *cross-validation*. La segunda prueba a su vez, se divide en dos, en las que en una se hará la regularización *Ridge* y en otra la regularización *Lasso*.

6. Definir los modelos a usar y estimar sus parámetros e hyperparámetros.

El modelo a utilizar es *regresión lineal*. Se ajustarán los atributos que se van a aplicar en la r*egresión lineal* final con la función *step()*. Para hacer la segunda prueba se utiliza regresión logística y directamente la función *cv.glmnet()* ajusta los parámetros que mejor resultado ofrecen. Sí se cambiará el tipo de generalización que va a realizar con el parámetro alpha de la función.

7. Selección y ajuste modelo final.

```
###################
# Función que devuelve el error en las etiquetas de una muestra
#
# La función comprueba la diferencia entre el valor que se ha predecido y el
# que realmente le corresponde, hace la suma de todas las diferencias y las
# divide por por la máxima diferencia que se puede encontrar. Finalmente se
# hace la media y es el valor que se devuelve.
#
getErrorReg = function(y, yHat){

  (sum(abs(y - yHat))/(max(y)-min(y)))*(1/length(y))

}

# Se hace regresión lineal de la muestra de entrenamiento teniendo en
# cuenta todos los atributos. La intención es comprobar cuáles de ellos no
# representan la mayor parte de la muestra y así tener una solución con el
# menor número de atributos posibles, lo que hará que sea de mayor calidad.
prost.lm = lm(lpsa ~ ., data = prost.train)
```

Para obtener el menor número de atributos posibles se podría haber utilizado *regsubsets()*, *step()* o *PCA* con el paquete *caret*. En este caso se ha utilizado la función *step()* que, directamente devuelve los atributos mas representativos del conjunto de muestras. Además, indica qué peso tiene cada uno de ellos, por lo que se puede hacer una valoración más exhaustiva del resultado. Se ha inspeccionado la función *step()* para comprobar su funcionamiento y se puede determinar que el pseudocódigo relacionado es:

```
# initial_setting
# for i in steps
# start
```

```
#   set_type_scan
#   get_attributes
#   evaluate_attributes
#   result += list_evaluation
# end
# return result
```

El algoritmo de la función *step()* comienza ajustando los parámetros iniciales por defecto que tiene definidos o los que el usuario quiera utilizar. De manera iterativa, en una serie de pasos previamente establecidos, obtiene los atributos con los que va a realizar la prueba. En cada iteración selecciona una serie de ellos y evalúa el resultado de la *regresión logística* (en este caso). Finalmente, para cada evaluación realizada, devuelve su resultado, mostrando la *desviación* de cada atributo y el *criterio de información de Akaike*, así como los atributos con los que se han obtenido estos resultados.

Para obtener más información de la que se obtiene solo con la función *step()*, se utiliza *summary()* para ver un informe más completo.

```
summary(step(prost.lm))
```

```
## Start:  AIC=-37.13
## lpsa ~ lcavol + lweight + age + lbph + svi + lcp + gleason +
##     pgg45
##
##           Df Sum of Sq    RSS     AIC
## - gleason  1    0.0109 29.437 -39.103
## <none>                 29.426 -37.128
## - age      1    0.9886 30.415 -36.914
## - pgg45    1    1.5322 30.959 -35.727
## - lcp      1    1.7683 31.195 -35.218
## - lbph     1    2.1443 31.571 -34.415
## - svi      1    3.0934 32.520 -32.430
## - lweight  1    3.8390 33.265 -30.912
## - lcavol   1   14.6102 44.037 -12.118
##
## Step:  AIC=-39.1
## lpsa ~ lcavol + lweight + age + lbph + svi + lcp + pgg45
##
##           Df Sum of Sq    RSS     AIC
## <none>                 29.437 -39.103
## - age      1    1.1025 30.540 -38.639
## - lcp      1    1.7583 31.196 -37.216
## - lbph     1    2.1354 31.573 -36.411
## - pgg45    1    2.3755 31.813 -35.903
## - svi      1    3.1665 32.604 -34.258
## - lweight  1    4.0048 33.442 -32.557
## - lcavol   1   14.8873 44.325 -13.681
##
## Call:
## lm(formula = lpsa ~ lcavol + lweight + age + lbph + svi + lcp +
##     pgg45, data = prost.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.65425 -0.34471 -0.05615  0.44380  1.48952
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.25906    1.02517   0.253   0.8014
## lcavol       0.57393    0.10507   5.462 9.88e-07 ***
## lweight      0.61921    0.21856   2.833   0.0063 **
## age         -0.01948    0.01310  -1.486   0.1425
## lbph         0.14443    0.06981   2.069   0.0430 *
## svi          0.74178    0.29445   2.519   0.0145 *
## lcp         -0.20542    0.10942  -1.877   0.0654 .
## pgg45        0.89450    0.40994   2.182   0.0331 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7064 on 59 degrees of freedom
## Multiple R-squared:  0.6943, Adjusted R-squared:  0.658
## F-statistic: 19.14 on 7 and 59 DF,  p-value: 4.496e-13
```

Los atributos que se deben tener en cuenta según los resultados obtenidos son:

- *lcavol.*
- *lweight.*
- *age.*
- *lbph.*
- *svi.*
- *lcp.*
- *pgg45.*

```
# Se predice los valores que va a obtener cada elemento del conjunto de test.
prost.yHat <- predict(prost.lm, prost.test)

# Se calcula el error que tiene esta predicción
meanTest.lm = getErrorReg(prost.test$lpsa, prost.yHat)

print(paste("Etest calculado con Regresión Lineal: ",
            round(meanTest.lm*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Lineal:  10.864 %"
```

Para calcular el error haciendo generalización se usa el paquete *glmnet*

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-10
```

```
# La función cv.glmnet() recibe como parámetros los datos sin etiqueta y sus
# etiquetas correspondientes.

# Se separan los datos, cogiendo todos menos la columna que contiene el
# resultado
prost.train.x <- as.matrix(subset(prost.train, select = -lpsa))

# Se obtiene solo el resultado de cada muestra
prost.train.y <- prost.train[, "lpsa"]
```

```r
# Se realiza la regresión logísitca con regularización Ridge
prost.glmnet <- cv.glmnet(prost.train.x, prost.train.y, alpha = 0)

# Se vuelven a separar los datos como anteriormente pero con test
prost.test.x <- as.matrix(subset(prost.test, select = -lpsa))
prost.test.y <- prost.test[, "lpsa"]

# Se predice el resultado que obtendrá cada muestra
prost.yHat <- predict(prost.glmnet, prost.test.x)

# Se calcula el error obtenido
meanTest.lm.ridge = getErrorReg(prost.test.y, prost.yHat)

print(paste("Etest calculado con Regresión Logística con regularización Ridge: ",
            round(meanTest.lm.ridge*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística con regularización Ridge:  11.386 %"
```

Se repite la misma operación realizada con *glmnet()*, pero se hace con generalización *Lasso*. Como los datos ya están separados, no se repite.

```r
# Se realiza la regresión logísitca con regularización Lasso.
prost.glmnet <- cv.glmnet(prost.train.x, prost.train.y, alpha = 1)

# Se predice el resultado que obtendrá cada muestra
prost.yHat <- predict(prost.glmnet, prost.test.x)

# Se calcula el error obtenido
meanTest.lm.lasso = getErrorReg(prost.test.y, prost.yHat)

print(paste("Etest calculado con Regresión Logística con regularización Lasso: ",
            round(meanTest.lm.lasso*100, digits = perDec), "%"))
```

```
## [1] "Etest calculado con Regresión Logística con regularización Lasso:  10.564 %"
```

8. Estimacion del error E out del modelo lo más ajustada posible.

Se ha realizado una primera aproximación con *regresión lineal*, reduciendo la dimensionalidad y se ha obtenido un error del *10.864%*. En un segundo intento, con *regresión logística* con regularización *Ridge*, el error ha aumentado al *11.386%*. El tercer intento, con *regresión logística* con regularización *Lasso*, ha obtenido el mejor porcentaje, con *10.564%*, por lo que el error más ajustado posible ha sido el obtenido con el último método mencionado.

9. Discutir y justificar la calidad del modelo encontrado y las razones por las que considera que dicho modelo es un buen ajuste que representa adecuadamente los datos muestrales.

Al contrario de lo que ha ocurrido con el problema de clasificación, el error obtenido ha sido bueno. Pese a que se ha utilizado una técnica más avanzada a la *regresión lineal*, apenas se ha conseguido aumentar la calidad del ajuste, probablemente gracias a los datos que se tienen. Estos datos son fáciles de aprender y cuentan con unas características que representan la muestra en su totalidad. El modelo ha ajustado bien, posiblemente, por la mínima cantidad de muestras que se tienen en el conjunto de datos, pero se puede determinar que es correcto al obtener un error de *test/out* del *10.564%*.

Bibliografía

La bibliografía y contenido web consultado para la realización de la práctica ha sido el siguiente:

https://stats.stackexchange.com/questions/239078/how-to-do-cross-validation-with-cv-glmnet-lasso-regression-in-r

http://rafalab.github.io/pages/649/prostate.html

https://stackoverflow.com/questions/33472095/logistic-regression-with-regularization-in-r-language

https://stackoverflow.com/questions/10085806/extracting-specific-columns-from-a-data-frame

https://stats.stackexchange.com/questions/99738/how-do-i-train-a-logistic-regression-in-r-using-l1-loss-function

https://stackoverflow.com/questions/8234972/how-to-set-the-coefficient-value-in-regression-r

https://stackoverflow.com/questions/12323859/how-to-manually-set-coefficients-for-variables-in-linear-model

https://stackoverflow.com/questions/25695565/predict-with-arbitrary-coefficients-in-r

https://stats.stackexchange.com/questions/51547/how-to-use-ridge-cv-in-r

https://stats.stackexchange.com/questions/228763/regularization-methods-for-logistic-regression

https://scholar.google.es/scholar?q=regularization+logistic+regression&hl=es&as_sdt=0&as_vis=1&oi=scholart&sa=X&sqi=2&ved=0ahUKEwi1ppPjhIzUAhUII8AKHf5qDNUQgQMIJTAA

https://stackoverflow.com/questions/27540345/how-do-i-prevent-r-function-step-from-outputing-to-the-console

https://stackoverflow.com/questions/31196176/perform-model-selection-with-step-function-and-write-outcome-into-0-1-vector

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/stepfun.html

http://www.astrostatistics.psu.edu/su07/R/html/stats/html/stepfun.html

https://www.math.ucla.edu/~anderson/rw1001/library/base/html/step.html

http://grokbase.com/t/r/r-help/166f4a2nb5/r-cv-glm-problem

http://r.789695.n4.nabble.com/function-cv-glm-in-library-boot-td790577.html

https://stackoverflow.com/questions/25391623/how-to-predict-with-cv-glm-in-r

https://stats.stackexchange.com/questions/250809/confusion-about-cv-glm-in-r

https://stats.stackexchange.com/questions/48766/intepretation-of-crossvalidation-result-cv-glm

https://www.r-bloggers.com/cross-validation-estimating-prediction-error/

https://www.r-bloggers.com/cross-validation-for-predictive-analytics-using-r/

http://r.789695.n4.nabble.com/How-to-do-cross-validation-with-glm-td3765994.html

https://dlegorreta.wordpress.com/2015/03/07/regresion-lineal/

https://stat.ethz.ch/pipermail/r-help/2002-March/019682.html

http://r.789695.n4.nabble.com/help-with-linear-model-td792142.html

https://github.com/lme4/lme4/issues/323

https://rdrr.io/rforge/caret/

https://rdrr.io/rforge/caret/man/models.html

https://www.analyticsvidhya.com/blog/2014/12/caret-package-stop-solution-building-predictive-models/

https://rdrr.io/cran/caret/man/models.html

https://topepo.github.io/caret/available-models.html

https://stats.stackexchange.com/questions/70801/how-to-normalize-data-to-0-1-range

https://stats.stackexchange.com/questions/174823/how-to-apply-standardization-normalization-to-train-and-testset-if-predict

https://stat.ethz.ch/pipermail/r-help//2012-October/336676.html

http://r.789695.n4.nabble.com/Error-in-J-time-invalid-subscript-type-closure-td4659523.html

https://stackoverflow.com/questions/17685502/in-r-getting-the-following-error-attempt-to-replicate-an-object-of-type-clos

http://r.789695.n4.nabble.com/Specify-model-with-polynomial-interaction-terms-up-to-degree-n-td4635130.html

https://stat.ethz.ch/pipermail/r-devel/2010-August/058061.html

https://stats.stackexchange.com/questions/244062/r-polynomial-expansion-error-on-unique-points-and-degree

https://ueb.vhir.org/dl656

https://tgmstat.wordpress.com/2013/11/21/introduction-to-principal-component-analysis-pca/

https://statweb.stanford.edu/~tibs/ElemStatLearn/data.html

http://www.statmethods.net/input/missingdata.html

https://stat.ethz.ch/R-manual/R-devel/library/Matrix/html/is.na-methods.html

http://www.gestiondeoperaciones.net/proyeccion-de-demanda/como-utilizar-una-regresion-lineal-para-realizar-un-pronostico-

http://www.sc.ehu.es/sbweb/fisica/cinematica/regresion/regresion.htm

http://www.um.es/ae/FEIR/10/

https://stackoverflow.com/questions/18496659/k-fold-cross-validation-how-to-get-the-prediction-automatically

https://gerardnico.com/wiki/lang/r/cross_validation

https://www.jstatsoft.org/article/view/v077i09

http://rstatistics.net/linear-regression-advanced-modelling-algorithm-example-with-r/

https://cran.r-project.org/web/packages/mlbench/mlbench.pdf

https://www.r-bloggers.com/logistic-regression-regularized-with-optimization/

https://www.rdocumentation.org/packages/pla/versions/0.2/topics/pla.fit

https://cran.r-project.org/web/packages/pla/pla.pdf

https://www.researchgate.net/post/What_is_the_difference_between_the_general_linear_model_GLMand_generalized_linear_model_GZLM

https://www.reddit.com/r/rstats/comments/2izyw1/difference_between_glm_and_lm_lmyxz_and_glmyxz/

https://stats.stackexchange.com/questions/122103/why-is-glm-different-than-an-lm-with-transformed-variable

https://stats.stackexchange.com/questions/43930/choosing-between-lm-and-glm-for-a-log-transformed-response-variable

http://www.joserodriguez.info/como-utilizar-r-para-hacer-regresiones-lineales-bonitos-graficos-y-regresiones-multivariables-y-l

https://stats.stackexchange.com/questions/94852/glm-gaussian-vs-glm-binomial-vs-log-link-glm-gaussian

https://stats.idre.ucla.edu/r/dae/poisson-regression/

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/family.html

http://r.789695.n4.nabble.com/Equivalence-between-lm-and-glm-td4668220.html

https://stat.ethz.ch/pipermail/r-help/1999-October/005013.html

https://www.quora.com/What-is-the-difference-between-a-linear-model-using-glm-and-a-linear-model-using-lm-in-R

https://stackoverflow.com/questions/8212063/glm-starting-values-not-accepted-log-link

https://stats.stackexchange.com/questions/181113/is-there-any-difference-between-lm-and-glm-for-the-gaussian-family-of-glm

http://www.endmemo.com/program/R/lm.php

https://www.r-bloggers.com/r-tutorial-series-simple-linear-regression/

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/lm.html

https://stackoverflow.com/questions/27455948/in-r-whats-the-simplest-way-to-scale-a-vector-to-a-unit-vector

https://stackoverflow.com/questions/11099272/r-standard-error-output-from-lm-object

https://stackoverflow.com/questions/40163807/r-error-in-linear-regression-model

https://stackoverflow.com/questions/1801487/include-error-terms-in-linear-regression-model-with-r

https://stats.stackexchange.com/questions/186578/error-in-a-linear-regression

http://www-edlab.cs.umass.edu/cs689/lectures/glm.pdf

http://feliperego.github.io/blog/2015/10/23/Interpreting-Model-Output-In-R

https://stat.ethz.ch/pipermail/r-help/2008-April/160538.html

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/coef.html

https://stats.stackexchange.com/questions/26650/how-do-i-reference-a-regression-models-coefficients-standard-errors

https://stats.stackexchange.com/questions/27511/extract-standard-errors-of-coefficient-linear-regression-r

http://www.montefiore.ulg.ac.be/~kvansteen/GBIO0009-1/ac20092010/Class8/Using%20R%20for%20linear%20regression.pdf

http://www.r-tutor.com/elementary-statistics/simple-linear-regression

http://www.statmethods.net/stats/regression.html

https://stat.ethz.ch/R-manual/R-devel/library/base/html/Extract.data.frame.html

https://stackoverflow.com/questions/14222632/change-values-in-row-based-on-a-column-value-r

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/step.html

https://stats.stackexchange.com/questions/214682/stepwise-regression-in-r-how-does-it-work

https://stats.stackexchange.com/questions/4640/interpreting-the-step-output-in-r

http://www.mathstat.dal.ca/~aarms2014/StatLearn/docs/09%20_annotated.pdf

https://www.rdocumentation.org/packages/glmpath/versions/0.97/topics/heart.data

http://www.moneyscience.com/pg/blog/StatAlgo/read/361153/stanford-ml-4-logistic-regression-and-classification

https://rpubs.com/rvempati/86226

http://finzi.psych.upenn.edu/library/catdata/html/heart.html

http://www.ivcampus.sdsu.edu/faculty/jcramirez/

http://www.math.chalmers.se/Stat/Grundutb/GU/MSG500/A09/RegSummary09.pdf

https://gist.github.com/smc77/1315162

https://gist.github.com/smc77/1315162/00727351f7e150a3f5ddb7fdfb696715ea1a545b

https://artax.karlin.mff.cuni.cz/r-help/library/catdata/html/heart.html

https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&sqi=2&ved=0ahUKEwjyssiPpYbUAhVHWxoKHdw
sQFghUMAY&url=http%3A%2F%2Fstat.duke.edu%2F~cr173%2FSta102_Sp14%2FProject%2Fheart.pdf&usg=AFQjCNGDfE

https://rdrr.io/cran/ElemStatLearn/man/SAheart.html

https://blogs.oracle.com/taylor22/finding-datasets-for-south-african-heart-disease

https://cran.r-project.org/web/packages/ElemStatLearn/ElemStatLearn.pdf

https://www.rdocumentation.org/packages/leaps/versions/2.1-1/topics/regsubsets