ProyectoFinal

1.- DefiniciÃ³n del problema a resolver y enfoque elegido

2.- CodificaciÃ³n de los datos de entrada para hacerlos Ãºtiles a los algoritmos.

Creo un vector con los nombres de cada columna para agregÃ¡rselo a cada data.frame.

Imprimo los valores de los niveles de factor en la etiqueta de cada base de datos, ya que al principio me he rallado un poco porque las 6 bases de datos son exactamente iguales y lo Ãºnico que var??a son las etiquetas. Al final he deducido que los pacientes son los mismos en las 6 bases de datos (hay mÃ¡s pero todav??a no las he mirado, las que mas confianza me han dado han sido Ã©stas), pero cada base de datos tiene una etiqueta diferente, supongo que se trata de predecir con los mismos datos 6 propiedades diferentes de la enfermedad.

```
getInvalidCols <- function (data) {
  invalids <- c()
  for (i in 1:ncol(data))
    if (length(levels(as.factor(as.character(data[,i])))) < 2)
      invalids <- c(invalids, i)
  return(invalids)
}


getVariablesRelevantesLASSO <- function(data, label, umbral = 0.002) {
  mat <- data.matrix(data)
  cv.lasso <- cv.glmnet(mat, c(label), family = "gaussian", alpha = 1)
  plot(cv.lasso)
  lambda <- cv.lasso$lambda.min
  print(paste("El lambda que minimiza el error es", lambda))
  res.lasso <- glmnet(mat, c(label), family = "gaussian", alpha = 1)
  pesos <- predict(res.lasso, type = "coefficients", s = lambda)
  print(pesos)
  relevantes <- which(abs(pesos)[2:length(pesos)] >= umbral)
  return (relevantes)
}
```

Conjunto de datos de entrenamiento

```
data.train.bp <- read.csv("./data/allbp.data.txt", header=FALSE, na.strings="?", comment.char = "|")
data.train.hyper <- read.csv("./data/allhyper.data.txt", header=FALSE, na.strings="?", comment.char = "
data.train.hypo <- read.csv("./data/allhypo.data.txt", header=FALSE, na.strings="?", comment.char = "|"
data.train.rep <- read.csv("./data/allrep.data.txt", header=FALSE, na.strings="?", comment.char = "|")
data.train.dis <- read.csv("./data/dis.data.txt", header=FALSE, na.strings="?", comment.char = "|")
data.train.sick <- read.csv("./data/sick.data.txt", header=FALSE, na.strings="?", comment.char = "|")
```

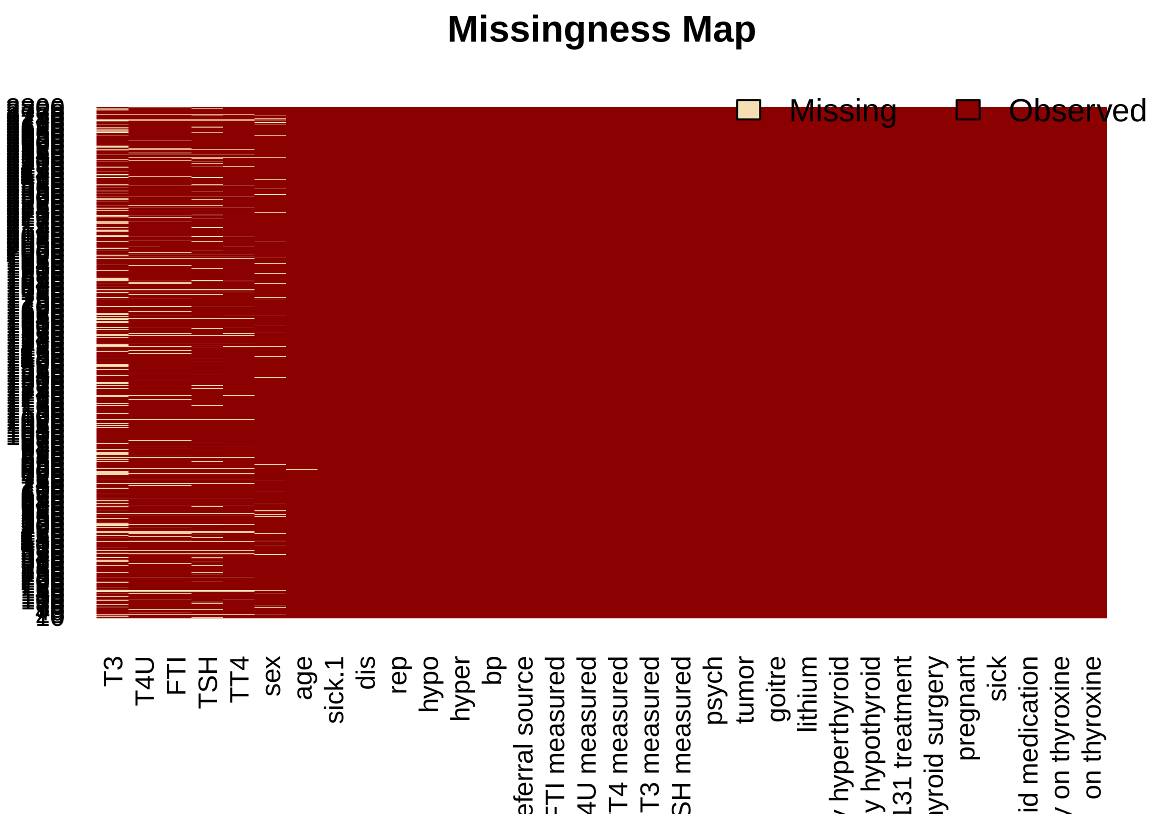
Conjunto de datos de test

```
data.test.bp <- read.csv("./data/allbp.test.txt", header=FALSE, na.strings="?", comment.char = "|")
data.test.hyper <- read.csv("./data/allhyper.test.txt", header=FALSE, na.strings="?", comment.char = "|
data.test.hypo <- read.csv("./data/allhypo.test.txt", header=FALSE, na.strings="?", comment.char = "|")
data.test.rep <- read.csv("./data/allrep.test.txt", header=FALSE, na.strings="?", comment.char = "|")
data.test.dis <- read.csv("./data/dis.test.txt", header=FALSE, na.strings="?", comment.char = "|")
data.test.sick <- read.csv("./data/sick.test.txt", header=FALSE, na.strings="?", comment.char = "|")
```

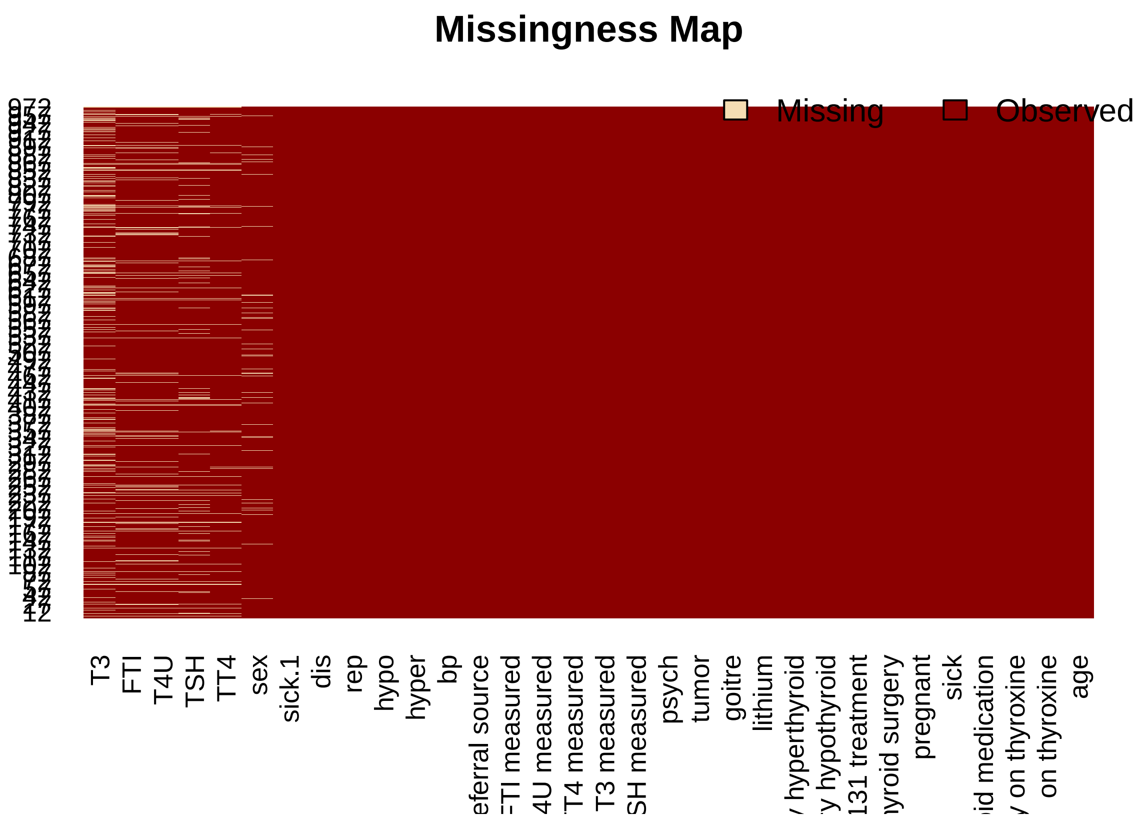
3.- ValoraciÃ³n del interÃ©s de la variables para el problema y selecciÃ³n de un subconjunto (en su caso).

NUEVO: Vamos a ver como se distribuyen los missing values en los dataset. Para eso usamos la función missmap de la librería Amelia

```
nombres <- c("age", "sex", "on thyroxine", "query on thyroxine", "on antithyroid medication",
             "sick", "pregnant", "thyroid surgery", "I131 treatment", "query hypothyroid",
             "query hyperthyroid", "lithium", "goitre", "tumor", "hypopituitary", "psych",
             "TSH measured", "TSH", "T3 measured", "T3", "TT4 measured", "TT4", "T4U measured",
             "T4U", "FTI measured", "FTI", "TBG measured", "TBG", "referral source", "bp",
             "hyper", "hypo",
             "rep", "dis", "sick")


data.train.clean <- cbind(data.train.bp, data.train.hyper[,ncol(data.train.hyper)],
                    data.train.hypo[,ncol(data.train.hypo)], data.train.rep[,ncol(data.train.rep)],
                    data.train.dis[,ncol(data.train.dis)], data.train.sick[,ncol(data.train.sick)])


colnames(data.train.clean) <- nombres
data.test.clean <- cbind(data.test.bp, data.test.hyper[,ncol(data.test.hyper)],
                    data.test.hypo[,ncol(data.test.hypo)], data.test.rep[,ncol(data.test.rep)],
                    data.test.dis[,ncol(data.test.dis)], data.test.sick[,ncol(data.test.sick)])


colnames(data.test.clean) <- nombres

invalidCols <- unique(c(getInvalidCols(data.train.clean), getInvalidCols(data.test.clean)))


if (!is.null(invalidCols)) {
  data.train.clean <- data.train.clean[,-invalidCols]
  data.test.clean <- data.test.clean[,-invalidCols]
}

##NUEVOOOOO
missmap(data.train.clean)
```



```r
missmap(data.test.clean)
```



```r
#Para intentar mejorar el modelo anterior, en vez de eliminar los datos
#con missing values, los sustituiremos por el promedio de estas variables en el data set de
#training. Crearemos un dataframe
data.aux <- na.omit(data.train.clean)
means <- sapply(matrix(data.aux), mean)
```

```
## Warning in mean.default(X[[2L]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[3L]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[4L]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[5L]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[6L]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[7L]], ...): argument is not numeric or logical:
## returning NA
## Warning in mean.default(X[[8L]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[9L]], ...): argument is not numeric or logical:
## returning NA

## Warning in mean.default(X[[10L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[11L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[12L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[13L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[14L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[15L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[16L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[18L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[20L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[22L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[24L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[26L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[27L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[28L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[29L]], ...): argument is not numeric or
## logical: returning NA
## Warning in mean.default(X[[30L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[31L]], ...): argument is not numeric or
## logical: returning NA

## Warning in mean.default(X[[32L]], ...): argument is not numeric or
## logical: returning NA
```

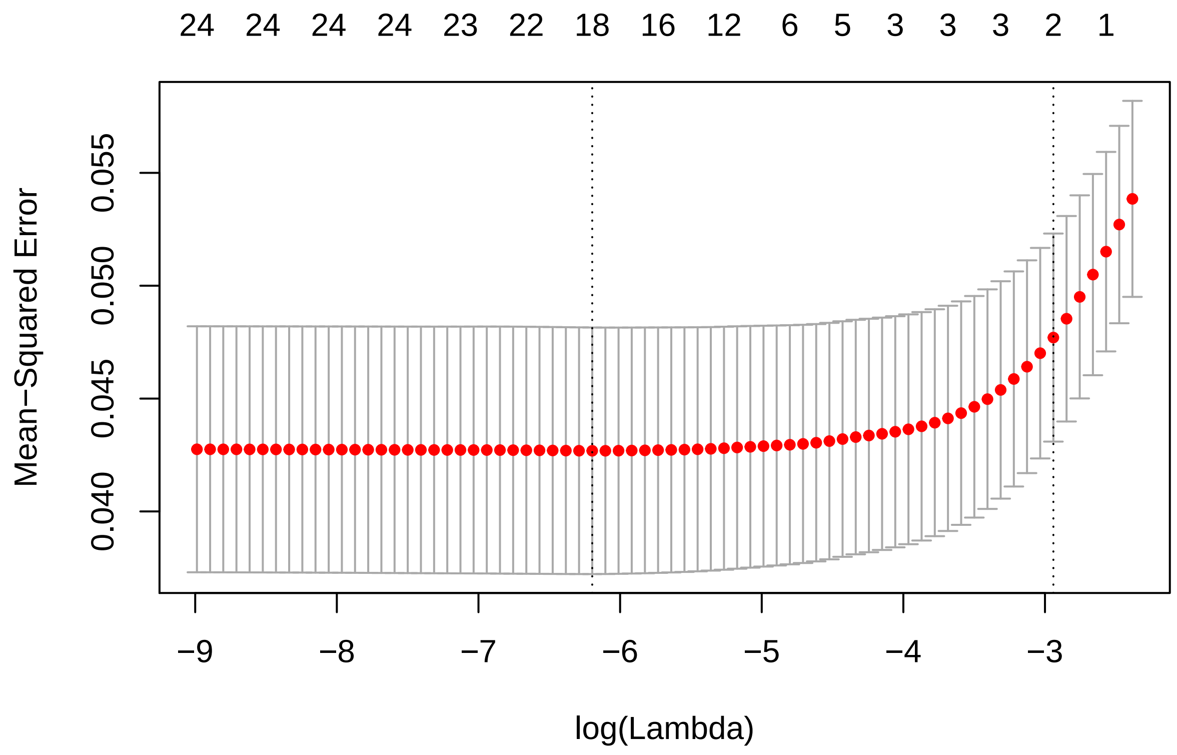
```
#Ahora iremos mirando dato a dato si tiene alg?n NA y en el caso de que lo tenga,
#lo cambiaremos por el promedio
repararDatos <- function(datos, medias)
{
  for(i in 1:nrow(data.train.clean))
  {
    for(j in 1:ncol(datos))
    {
      if(is.na(datos[i,j]))
      {
        datos[i,j] <- medias[j]
      }
    }
  }

  return (datos)
}


#Ahora, si nos sigue quedando alg?n NA m?s, entonces esos datos ya los omitimos
data.train.clean <- repararDatos(data.train.clean, means)
data.test.clean <- repararDatos(data.test.clean, means)



data.train.clean <- na.omit(data.train.clean)
data.test.clean <- na.omit(data.test.clean)

###NUEVOOOOO

print (data.train.clean$age[data.train.clean$age > 100])
```

```
## [1] 455
```

```
data.train.clean <- data.train.clean[data.train.clean$age < 100,]
data.test.clean<- data.test.clean[data.test.clean$hyper != 'secondary toxic.',]
data.test.clean$hyper <- as.factor(as.numeric(data.test.clean$hyper))
levels(data.test.clean$hyper) <- levels(data.train.clean$hyper)
levels(data.test.clean$hypo) <- levels(data.train.clean$hypo)
end <- ncol(data.train.clean)
ini <- end - 5
label.train <- data.train.clean[ini:end]
data.train.clean <- data.train.clean[-(ini:end)]
```

```
label.test <- data.test.clean[ini:end]
data.test.clean <- data.test.clean[-(ini:end)]



invalidCols <- unique(c(getInvalidCols(data.train.clean), getInvalidCols(data.test.clean)))
if (!is.null(invalidCols)) {
  data.train.clean <- data.train.clean[,-invalidCols]
  data.test.clean <- data.test.clean[,-invalidCols]
}
```

```
relevantes.bp <- getVariablesRelevantesLASSO(data.train.clean, label.train$bp)
```

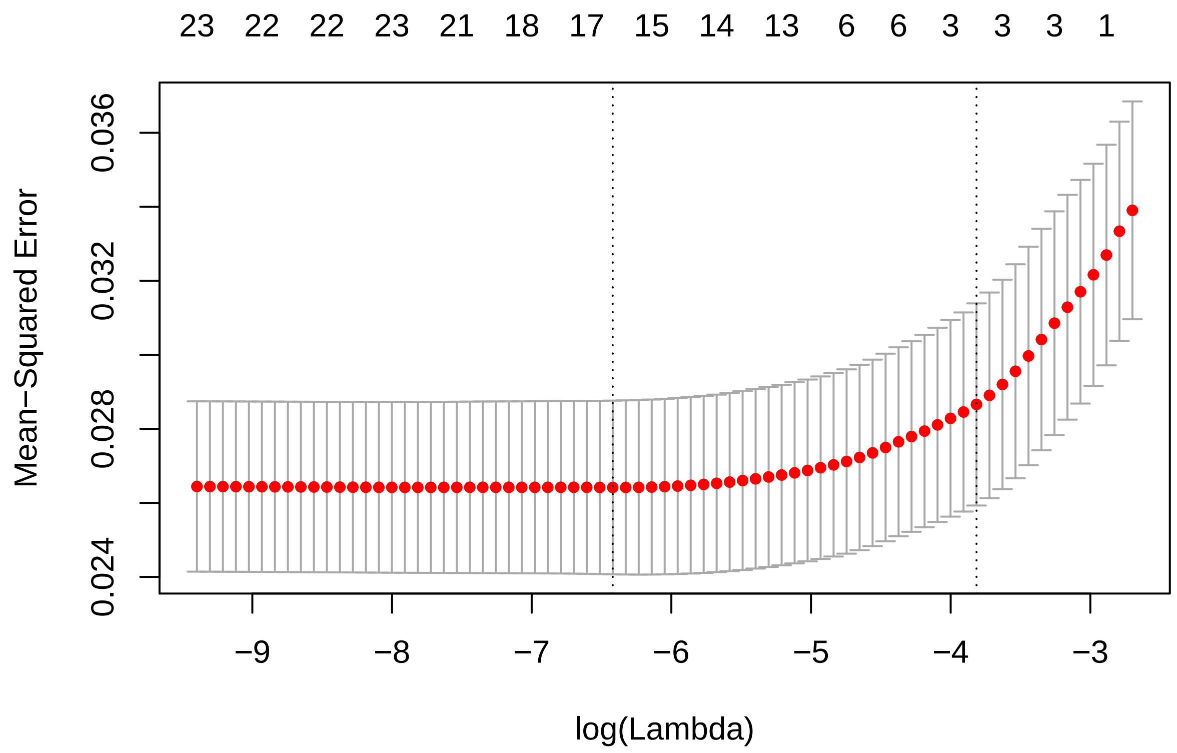


```
## [1] "El lambda que minimiza el error es 0.00203576846532222"
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                                     1
## (Intercept)                3.642680e+00
## age                        3.274875e-04
## sex                        5.495591e-03
## on thyroxine               3.600777e-02
## query on thyroxine        -7.756674e-03
## on antithyroid medication  .
## sick                       .
## pregnant                  -3.818830e-01
## thyroid surgery            3.218739e-02
## I131 treatment              7.788428e-03
## query hypothyroid            .
## query hyperthyroid           .
## lithium                      2.616915e-02
## goitre                       .
## tumor                        4.283650e-02
## psych                        2.676455e-02
## TSH measured                -1.441309e-02
## TSH                          1.388014e-04
## T3 measured                  .
## T3                          -3.837631e-02
## TT4 measured                -5.026983e-03
## TT4                         -1.551339e-05
## T4U measured                -2.563449e-02
## T4U                         -3.322834e-01
## FTI measured                 .
## FTI                          .
## referral source             -2.226395e-03
```

```
print(colnames(data.train.clean)[relevantes.bp])
```

```
##  [1] "sex"                 "on thyroxine"       "query on thyroxine"
##  [4] "pregnant"            "thyroid surgery"    "I131 treatment"
##  [7] "lithium"             "tumor"              "psych"
## [10] "TSH measured"        "T3"                 "TT4 measured"
## [13] "T4U measured"        "T4U"                "referral source"
```

```
relevantes.hyper <- getVariablesRelevantesLASSO(data.train.clean, label.train$hyper)
```

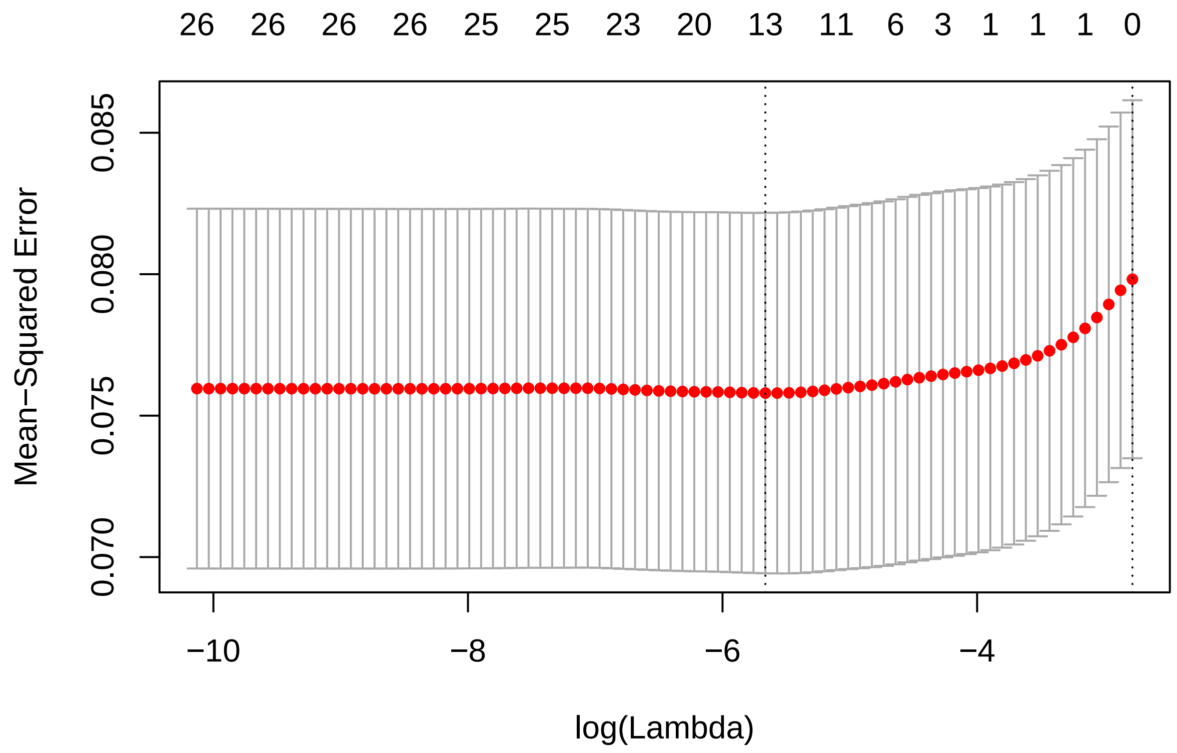


```
## [1] "El lambda que minimiza el error es 0.00162895429459921"
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                                         1
## (Intercept)                  3.462936e+00
## age                         -1.169978e-05
## sex                          1.376150e-02
## on thyroxine                 6.234951e-02
## query on thyroxine           .
## on antithyroid medication    4.356414e-02
## sick                         3.140336e-03
## pregnant                    -1.125050e-01
## thyroid surgery              .
## I131 treatment               .
## query hypothyroid            .
## query hyperthyroid          -1.247769e-02
## lithium                      .
## goitre                       .
## tumor                       -2.028280e-01
## psych                        2.836155e-02
## TSH measured                 .
## TSH                         -9.610789e-04
## T3 measured                 -5.624655e-03
## T3                          -4.643870e-02
## TT4 measured                -1.409120e-02
## TT4                          .
## T4U measured                 .
## T4U                           5.445748e-02
## FTI measured                 -8.216718e-03
## FTI                          -1.989828e-03
## referral source               .
```

```
print(colnames(data.train.clean)[relevantes.hyper])
```

```
##  [1] "sex"                          "on thyroxine"
##  [3] "on antithyroid medication" "sick"
##  [5] "pregnant"                     "query hyperthyroid"
##  [7] "tumor"                        "psych"
##  [9] "T3 measured"                  "T3"
## [11] "TT4 measured"                 "T4U"
## [13] "FTI measured"
```

```
relevantes.hypo <- getVariablesRelevantesLASSO(data.train.clean, label.train$hypo)
```

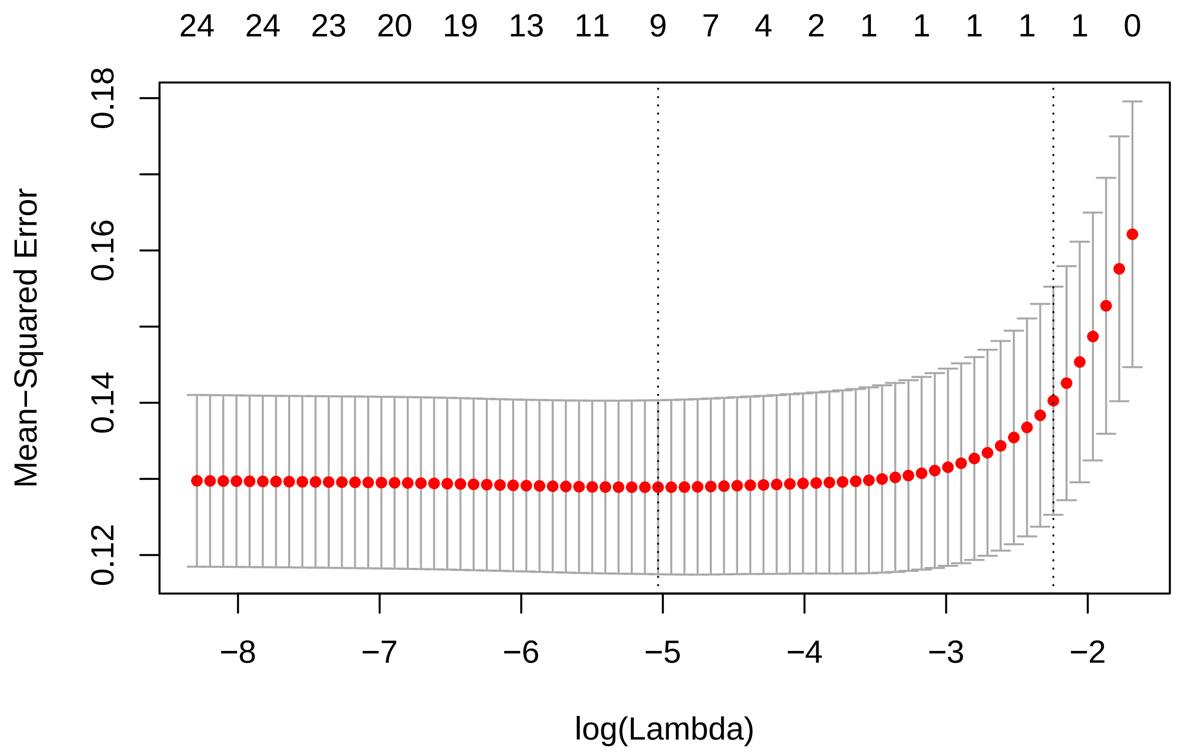


```
## [1] "El lambda que minimiza el error es 0.00346962898493411"
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                                          1
## (Intercept)                  2.0320325510
## age                           .
## sex                          0.0043367109
## on thyroxine                 0.0558240306
## query on thyroxine        -0.0204134189
## on antithyroid medication  .
## sick                      -0.0457219683
## pregnant                   0.0292636740
## thyroid surgery            0.0268406877
## I131 treatment             .
## query hypothyroid         -0.0480858072
## query hyperthyroid         .
## lithium                    .
## goitre                     .
## tumor                      .
## psych                      .
## TSH measured              -0.0308476362
## TSH                        0.0030251341
## T3 measured                0.0211058588
## T3                         .
## TT4 measured              -0.0204142182
## TT4                       -0.0002729903
## T4U measured               .
## T4U                        .
## FTI measured               .
## FTI                        .
## referral source            0.0029227637
```

```
print(colnames(data.train.clean)[relevantes.hypo])
```

```
##  [1] "sex"                "on thyroxine"       "query on thyroxine"
##  [4] "sick"               "pregnant"           "thyroid surgery"
##  [7] "query hypothyroid"  "TSH measured"       "TSH"
## [10] "T3 measured"        "TT4 measured"       "referral source"
```

```
relevantes.rep <- getVariablesRelevantesLASSO(data.train.clean, label.train$rep)
```

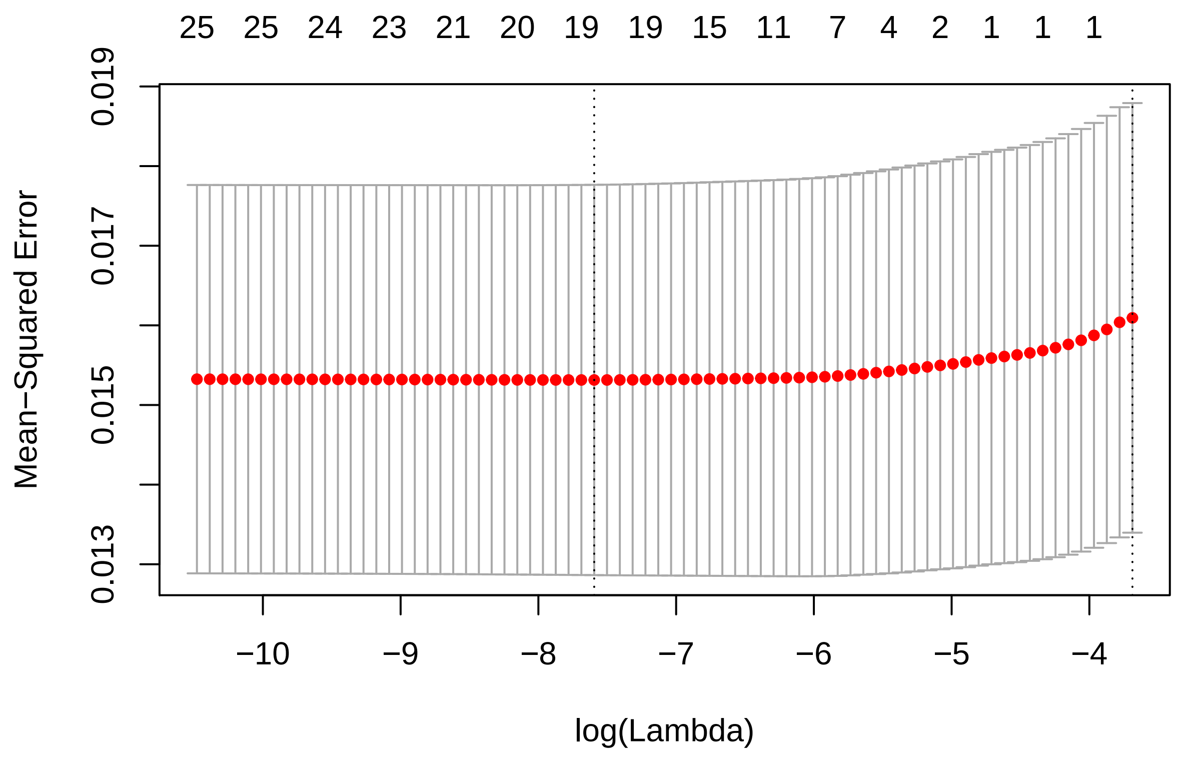


```
## [1] "El lambda que minimiza el error es 0.00651407507255239"
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                                        1
## (Intercept)                0.3597622924
## age                        .
## sex                        .
## on thyroxine               0.5429864360
## query on thyroxine         .
## on antithyroid medication  .
## sick                       .
## pregnant                   .
## thyroid surgery            .
## I131 treatment            -0.0148938861
## query hypothyroid          0.0553629299
## query hyperthyroid         .
## lithium                    .
## goitre                     .
## tumor                      0.0074882441
## psych                      .
## TSH measured               0.0101512939
## TSH                        0.0005842618
## T3 measured                .
## T3                        -0.0138479046
## TT4 measured               0.0096821423
## TT4                        .
## T4U measured               .
## T4U                          .
## FTI measured                 .
## FTI                          0.0003324119
## referral source              .
```

```
print(colnames(data.train.clean)[relevantes.rep])
```

```
## [1] "on thyroxine"      "I131 treatment"    "query hypothyroid"
## [4] "tumor"             "TSH measured"      "T3"
## [7] "TT4 measured"
```

```
relevantes.dis <- getVariablesRelevantesLASSO(data.train.clean, label.train$dis)
```

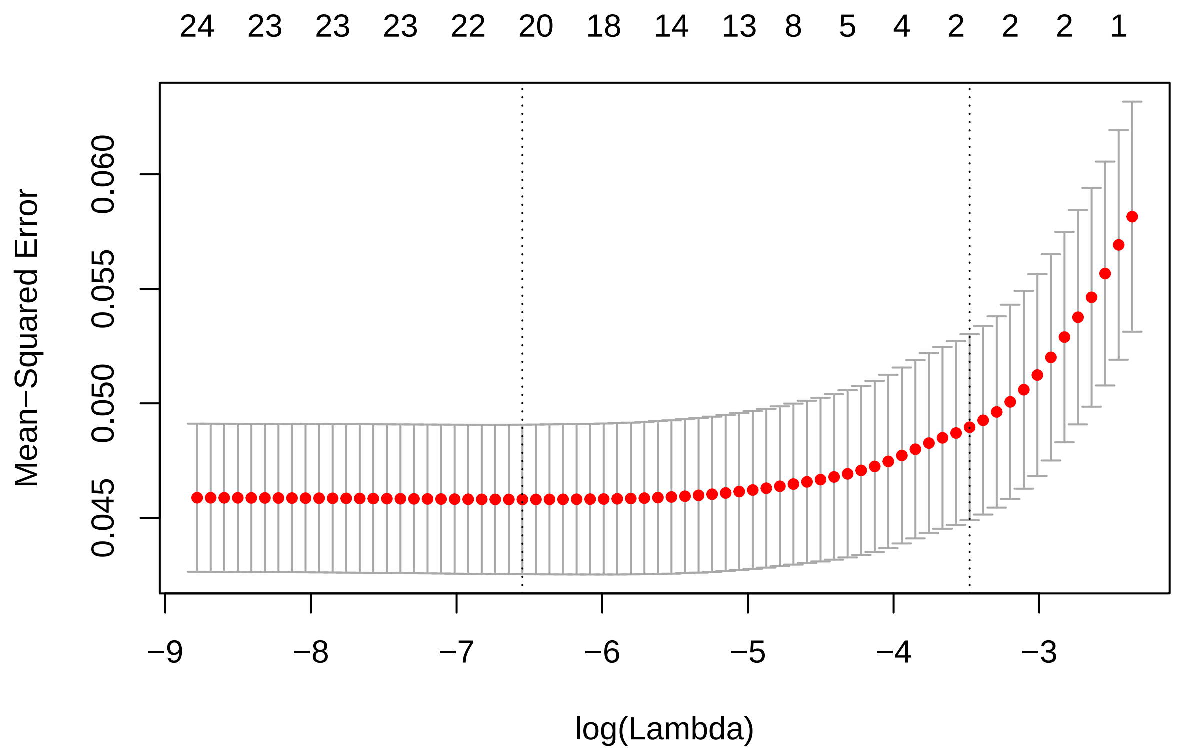


```
## [1] "El lambda que minimiza el error es 0.000503078867182642"
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                                         1
## (Intercept)                  2.0103147572
## age                          0.0001556584
## sex                         -0.0053103848
## on thyroxine                 0.0355949816
## query on thyroxine           0.0084957440
## on antithyroid medication    .
## sick                         0.0180282063
## pregnant                     .
## thyroid surgery            0.0051002050
## I131 treatment             0.0019656146
## query hypothyroid         -0.0051371533
## query hyperthyroid         0.0049487847
## lithium                    .
## goitre                     .
## tumor                      0.0121963647
## psych                      0.0107230978
## TSH measured              -0.0131220684
## TSH                       -0.0002338423
## T3 measured               -0.0069379295
## T3                         0.0135557392
## TT4 measured               .
## TT4                       -0.0001033570
## T4U measured              -0.0014538150
## T4U                        .
## FTI measured               .
## FTI                       -0.0009286581
## referral source            0.0008642794
```

```r
print(colnames(data.train.clean)[relevantes.dis])
```

```
##  [1] "sex"                "on thyroxine"       "query on thyroxine"
##  [4] "sick"               "thyroid surgery"    "query hypothyroid"
##  [7] "query hyperthyroid" "tumor"              "psych"
## [10] "TSH measured"       "T3 measured"        "T3"
```

```r
relevantes.sick <- getVariablesRelevantesLASSO(data.train.clean, label.train$sick)
```



```
## [1] "El lambda que minimiza el error es 0.0014326613420736"
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                                       1
## (Intercept)                8.891554e-01
## age                        2.672791e-04
## sex                        .
## on thyroxine              -2.455934e-02
## query on thyroxine         1.915373e-02
## on antithyroid medication  1.804406e-02
## sick                       2.673648e-02
## pregnant                   1.002214e-01
## thyroid surgery           -3.693422e-02
## I131 treatment             .
## query hypothyroid          5.908655e-02
## query hyperthyroid         2.067463e-02
## lithium                    2.747767e-03
## goitre                     2.518538e-02
## tumor                      .
## psych                     -2.448700e-02
## TSH measured              -7.021070e-03
## TSH                       -5.360018e-05
## T3 measured                4.420769e-02
## T3                        -1.257474e-01
## TT4 measured               .
## TT4                        .
## T4U measured               8.042913e-03
## T4U                       -2.657467e-02
## FTI measured               .
## FTI                        1.084491e-03
## referral source            2.091067e-02
```

```
print(colnames(data.train.clean)[relevantes.sick])
```

```
##  [1] "on thyroxine"               "query on thyroxine"
##  [3] "on antithyroid medication" "sick"
##  [5] "pregnant"                   "thyroid surgery"
##  [7] "query hypothyroid"          "query hyperthyroid"
##  [9] "lithium"                    "goitre"
## [11] "psych"                      "TSH measured"
## [13] "T3 measured"                "T3"
## [15] "T4U measured"               "T4U"
## [17] "referral source"
```

4.- NormalizaciÃ³n de las variables (en su caso)

5.- SelecciÃ³n de las tÃ©cnica (parÃ¡metrica) y valoraciÃ³n de la idoneidad de la misma frente a otras alternativas

6.- AplicaciÃ³n de la tÃ©cnica especificando claramente que algoritmos se usan en la estimaciÃ³n de los parÃ¡metros, los hiperparÃ¡metros y el error de generalizaciÃ³n.

```
modeloSVM <- function(tune.out, kernel, data, label) {
  bestMod <- tune.out$best.model
  print(paste("cost:", bestMod$cost))
  print(paste("gamma:", bestMod$gamma))
  print(paste("NÃºmero de vectores soporte:", bestMod$tot.nSV))
  pred <- predict(bestMod, data)
  print("Matriz de confusiÃ³n:")
  print(table(predict = pred, truth = label))
  error.svm <- sum(pred != label)/length(label)
  print (paste("El error Ein en SVM con kernel", kernel, "es", error.svm))
}
dat <- data.frame(x = data.train.clean[,relevantes.bp], y = label.train$bp)
tune.out.linear.bp <- tune(svm, y~., data = dat, kernel = "linear", type = "C-classification",
                           ranges = list(cost = c(1e-3, 1e-2, 1e-1, 1, 5, 10, 50, 100)))
modeloSVM(tune.out.linear.bp, "lineal", dat, dat$y)
```

```
## [1] "cost: 0.1"
## [1] "gamma: 0.0526315789473684"
## [1] "NÃºmero de vectores soporte: 226"
## [1] "Matriz de confusiÃ³n:"
##                               truth
## predict                        decreased binding protein.
##   decreased binding protein.                          0
##   increased binding protein.                          0
##   negative.                                          8
##                              truth
## predict                       increased binding protein. negative.
##   decreased binding protein.                         0         0
##   increased binding protein.                        56        13
##   negative.                                         64      2548
## [1] "El error Ein en SVM con kernel lineal es 0.0316102640386761"
```

```r
dat <- data.frame(x = data.train.clean[,relevantes.hyper], y = label.train$hyper)
tune.out.linear.hyper <- tune(svm, y~., data = dat, kernel = "linear", type = "C-classification",
                           ranges = list(cost = c(1e-3, 1e-2, 1e-1, 1, 5, 10, 50, 100)))
modeloSVM(tune.out.linear.hyper, "lineal", dat, label.train$hyper)
```

```
## [1] "cost: 10"
## [1] "gamma: 0.0714285714285714"
## [1] "NÃºmero de vectores soporte: 172"
## [1] "Matriz de confusiÃ³n:"
##                truth
## predict          goitre. hyperthyroid. negative. T3 toxic.
##   goitre.              6             2         1         0
##   hyperthyroid.        1            13         3         0
##   negative.            0            44      2613         6
##   T3 toxic.            0             0         0         0
## [1] "El error Ein en SVM con kernel lineal es 0.0211974711788769"
```

```r
dat <- data.frame(x = data.train.clean[,relevantes.hypo], y = label.train$hypo)
tune.out.linear.hypo <- tune(svm, y~., data = dat, kernel = "linear", type = "C-classification",
                           ranges = list(cost = c(1e-3, 1e-2, 1e-1, 1, 5, 10, 50, 100)))
modeloSVM(tune.out.linear.hypo, "lineal", dat, label.train$hypo)
```

```
## [1] "cost: 100"
## [1] "gamma: 0.0625"
## [1] "NÃºmero de vectores soporte: 222"
## [1] "Matriz de confusiÃ³n:"
##                             truth
## predict                      compensated hypothyroid. negative.
##   compensated hypothyroid.                        121         5
##   negative.                                        24      2470
##   primary hypothyroid.                              4         3
##   secondary hypothyroid.                            0         0
##                             truth
## predict                      primary hypothyroid. secondary hypothyroid.
##   compensated hypothyroid.                     11                      0
##   negative.                                     5                      2
##   primary hypothyroid.                         44                      0
##   secondary hypothyroid.                        0                      0
## [1] "El error Ein en SVM con kernel lineal es 0.0200818148010413"
```

```r
dat <- data.frame(x = data.train.clean[,relevantes.rep], y = label.train$rep)
tune.out.linear.rep <- tune(svm, y~., data = dat, kernel = "linear", type = "C-classification",
                           ranges = list(cost = c(1e-3, 1e-2, 1e-1, 1, 5, 10, 50, 100)))
modeloSVM(tune.out.linear.rep, "lineal", dat, label.train$rep)
```

```
## [1] "cost: 0.001"
## [1] "gamma: 0.125"
## [1] "NÃºmero de vectores soporte: 139"
## [1] "Matriz de confusiÃ³n:"
##                        truth
## predict                 negative. overreplacement. replacement therapy.
##   negative.                  2603               23                   28
##   overreplacement.              0                0                    0
##   replacement therapy.          0                0                    0
##   underreplacement.             0                0                    0
##                        truth
## predict                 underreplacement.
##   negative.                            35
##   overreplacement.                      0
##   replacement therapy.                  0
##   underreplacement.                     0
## [1] "El error Ein en SVM con kernel lineal es 0.0319821494979546"
```

```
dat <- data.frame(x = data.train.clean[,relevantes.dis], y = label.train$dis)
tune.out.linear.dis <- tune(svm, y~., data = dat, kernel = "linear", type = "C-classification",
                           ranges = list(cost = c(1e-3, 1e-2, 1e-1, 1, 5, 10, 50, 100)))
modeloSVM(tune.out.linear.dis, "lineal", dat, label.train$dis)
```

```
## [1] "cost: 0.001"
## [1] "gamma: 0.0769230769230769"
## [1] "NÃºmero de vectores soporte: 98"
## [1] "Matriz de confusiÃ³n:"
##              truth
## predict       discordant. negative.
##   discordant.           0         0
##   negative.            44      2645
## [1] "El error Ein en SVM con kernel lineal es 0.0163629602082559"
```

```
dat <- data.frame(x = data.train.clean[,relevantes.sick], y = label.train$sick)
```

```
## Warning in `$.data.frame`(label.train, sick): Name partially matched in
## data frame
```

```
tune.out.linear.sick <- tune(svm, y~., data = dat, kernel = "linear", type = "C-classification",
                           ranges = list(cost = c(1e-3, 1e-2, 1e-1, 1, 5, 10, 50, 100)))
modeloSVM(tune.out.linear.sick, "lineal", dat, label.train$sick)
```

```
## [1] "cost: 10"
## [1] "gamma: 0.0476190476190476"
## [1] "NÃºmero de vectores soporte: 277"
## [1] "Matriz de confusiÃ³n:"
##            truth
## predict     negative. sick.
##   negative.      2493    67
##   sick.            29   100
## [1] "El error Ein en SVM con kernel lineal es 0.03570100409074"
```

7.- Argumentar sobre la idoneidad de la funciÃ³n regularizaciÃ³n usada (en su caso)

8.- ValoraciÃ³n de los resultados (grÃ¡ficas, mÃ©tricas de error, anÃ¡lisis de residuos, etc)

```
getError <- function(data.train, data.test, label.train, label.test, tune.out, relevantes) {
  dat <- data.frame(x = data.train[,relevantes], y = label.train)
  test <- data.frame(x = data.test[,relevantes], y = label.test)
  modelo <- svm(y~., data = dat, kernel = "linear", type = "C-classification",
  cost = tune.out$best.model$cost)
  pred <- predict(modelo, dat)
  ein <- sum(pred != dat$y)/nrow(dat)
  pred <- predict(modelo, test)
  eout<- sum(pred != test$y)/nrow(test)
  return (c(ein, eout))
}
print(getError(data.train.clean, data.test.clean, label.train$bp, label.test$bp, tune.out.linear.bp, rel
```

```
## [1] 0.03161026 0.02685285
```

```
print(getError(data.train.clean, data.test.clean, label.train$hyper, label.test$hyper, tune.out.linear.h
```

```
## [1] 0.02119747 0.01718582
```

```
print(getError(data.train.clean, data.test.clean, label.train$hypo, label.test$hypo, tune.out.linear.hyp
```

```
## [1] 0.02008181 0.02792696
```

```
print(getError(data.train.clean, data.test.clean, label.train$rep, label.test$rep, tune.out.linear.rep,
```

```
## [1] 0.03198215 0.03651987
```

```
print(getError(data.train.clean, data.test.clean, label.train$dis, label.test$dis, tune.out.linear.dis,
```

```
## [1] 0.01636296 0.01288937
```

```
print(getError(data.train.clean, data.test.clean, label.train$sick, label.test$sick, tune.out.linear.sic
```

```
## [1] 0.03570100 0.04296455
```

9.- Justificar que se ha obtenido la mejor de las posibles soluciones con la tÃ©cnica elegida y la muestra dada. Argumentar en tÃ©rminos de la dimensiÃ³n VC del modelo, el error de generalizaciÃ³n y las curvas de aprendizaje.