Disfonía en enfermos de Parkinson

Rubén Ibarrondo López y Miren Hayet Otero

28/5/2021

Índice

1 Objetivo

El objetivo principal de este trabajo consiste en encontrar un modelo de clasificación capaz de diferenciar a enfermos de Parkinson de pacientes sanos, en base a registros de voz. Para ello, primero se van a analizar las características de los datos de los que se dispone, y después se ajustarán y compararán diferentes técnicas de clasificación.

2 Análisis preliminar

Es necesario realizar un análisis preliminar de los datos para después obtener un modelo lo más fácil de interpretar y mejor posible.

La base de datos utilizada se puede consultar aquí. Se dispone de 195 registros de voz correspondientes a 31 pacientes, de los cuales hay 23 enfermos de Parkinson. Para cada registro se han recogido 23 medidas relacionadas con la voz:

- MDVP.Fo.Hz: Frecuencia vocal fundamental media.
- MDVP.Fhi.Hz : Frecuencia vocal fundamental máxima.
- MDVP.Flo.Hz: Frecuencia vocal fundamental mínima.
- MDVP.Jitter, MDVP.Jitter.Abs, MDVP.RAP, MDVP.PPQ, Jitter.DDP: Medidas de variación en la frecuencia fundamental.
- MDVP.Shimmer, MDVP.Shimmer.dB, Shimmer.APQ3, Shimmer.APQ5, MDVP.APQ, Shimmer.DDA: Medidas de variación en la amplitud.
- NHR,HNR: Medidas del ratio entre el ruido y las componentes tonales de la voz.
- status: Estado de salud del paciente. 1-Enfermo de Parkinson, 0-Sano.
- RPDE, D2: Medidas no-lineales de complejidad dinámica.
- DFA: Exponente escalador de fractal de señal.
- spread1, spread2, PPE: Medidas no-lineales de la variación de la frecuencia fundamental.

En este caso no hay ningún dato ausente por lo que no va a ser necesaria ninguna estrategia de imputación.

A continuación se puede ver un resumen de las diferentes variables:

```
##      name            MDVP.Fo.Hz.      MDVP.Fhi.Hz.    MDVP.Flo.Hz.
##  Length:195         Min.   : 88.33   Min.   :102.1   Min.   : 65.48
##  Class :character   1st Qu.:117.57   1st Qu.:134.9   1st Qu.: 84.29
##  Mode  :character   Median :148.79   Median :175.8   Median :104.31
##                     Mean   :154.23   Mean   :197.1   Mean   :116.32
##                     3rd Qu.:182.77   3rd Qu.:224.2   3rd Qu.:140.02
##                     Max.   :260.11   Max.   :592.0   Max.   :239.17
##  MDVP.Jitter...     MDVP.Jitter.Abs.       MDVP.RAP           MDVP.PPQ
##  Min.   :0.001680   Min.   :7.000e-06   Min.   :0.000680   Min.   :0.000920
##  1st Qu.:0.003460   1st Qu.:2.000e-05   1st Qu.:0.001660   1st Qu.:0.001860
##  Median :0.004940   Median :3.000e-05   Median :0.002500   Median :0.002690
##  Mean   :0.006220   Mean   :4.396e-05   Mean   :0.003306   Mean   :0.003446
##  3rd Qu.:0.007365   3rd Qu.:6.000e-05   3rd Qu.:0.003835   3rd Qu.:0.003955
```

```
##  Max.   :0.033160   Max.   :2.600e-04   Max.   :0.021440   Max.   :0.019580
##    Jitter.DDP        MDVP.Shimmer     MDVP.Shimmer.dB.  Shimmer.APQ3
##  Min.   :0.002040   Min.   :0.00954   Min.   :0.0850   Min.   :0.004550
##  1st Qu.:0.004985   1st Qu.:0.01650   1st Qu.:0.1485   1st Qu.:0.008245
##  Median :0.007490   Median :0.02297   Median :0.2210   Median :0.012790
##  Mean   :0.009920   Mean   :0.02971   Mean   :0.2823   Mean   :0.015664
##  3rd Qu.:0.011505   3rd Qu.:0.03789   3rd Qu.:0.3500   3rd Qu.:0.020265
##  Max.   :0.064330   Max.   :0.11908   Max.   :1.3020   Max.   :0.056470
##   Shimmer.APQ5        MDVP.APQ        Shimmer.DDA           NHR
##  Min.   :0.00570   Min.   :0.00719   Min.   :0.01364   Min.   :0.000650
##  1st Qu.:0.00958   1st Qu.:0.01308   1st Qu.:0.02474   1st Qu.:0.005925
##  Median :0.01347   Median :0.01826   Median :0.03836   Median :0.011660
##  Mean   :0.01788   Mean   :0.02408   Mean   :0.04699   Mean   :0.024847
##  3rd Qu.:0.02238   3rd Qu.:0.02940   3rd Qu.:0.06080   3rd Qu.:0.025640
##  Max.   :0.07940   Max.   :0.13778   Max.   :0.16942   Max.   :0.314820
##       HNR              status         RPDE              DFA
##  Min.   : 8.441   Sano     : 48   Min.   :0.2566   Min.   :0.5743
##  1st Qu.:19.198   Parkinson:147   1st Qu.:0.4213   1st Qu.:0.6748
##  Median :22.085                   Median :0.4960   Median :0.7223
##  Mean   :21.886                   Mean   :0.4985   Mean   :0.7181
##  3rd Qu.:25.076                   3rd Qu.:0.5876   3rd Qu.:0.7619
##  Max.   :33.047                   Max.   :0.6852   Max.   :0.8253
##     spread1          spread2              D2             PPE
##  Min.   :-7.965   Min.   :0.006274   Min.   :1.423   Min.   :0.04454
##  1st Qu.:-6.450   1st Qu.:0.174350   1st Qu.:2.099   1st Qu.:0.13745
##  Median :-5.721   Median :0.218885   Median :2.362   Median :0.19405
##  Mean   :-5.684   Mean   :0.226510   Mean   :2.382   Mean   :0.20655
##  3rd Qu.:-5.046   3rd Qu.:0.279234   3rd Qu.:2.636   3rd Qu.:0.25298
##  Max.   :-2.434   Max.   :0.450493   Max.   :3.671   Max.   :0.52737
```

No parece haber ningún dato disparatado por lo que parecen ser variables con valores coherentes. La distribución de la variable de clasificación nos indica que en torno a un 75% de los registros corresponden a enfermos de Parkinson.

A la hora de crear cualquier modelo de clasificación es importante que la cantidad de variables que lo forman sea lo menor posible, ya que esto facilita su aplicación e interpretación. Muchas veces las medidas/variables de las que se dispone no suelen aportar demasiada información a la hora de clasificar, ya sea por que no están relacionadas con la variable de clasificación o porque no presentan gran variabilidad. También puede ocurrir que algunas variables estén altamente correladas entre sí, por lo que si se incluyen todas en el modelo, no van a aportar nueva información a la hora de clasificar.

Comencemos por ver si hay alguna variable con poca variabilidad:

```
##                  freqRatio percentUnique zeroVar   nzv
## MDVP.Fo.Hz.       1.000000     100.00000   FALSE FALSE
## MDVP.Fhi.Hz.      1.000000     100.00000   FALSE FALSE
## MDVP.Flo.Hz.      1.000000     100.00000   FALSE FALSE
## MDVP.Jitter...    1.000000      88.71795   FALSE FALSE
## MDVP.Jitter.Abs.  1.642857       9.74359   FALSE FALSE
## MDVP.RAP          1.666667      79.48718   FALSE FALSE
## MDVP.PPQ          1.333333      84.61538   FALSE FALSE
## Jitter.DDP        1.500000      92.30769   FALSE FALSE
## MDVP.Shimmer      1.000000      96.41026   FALSE FALSE
## MDVP.Shimmer.dB.  1.250000      76.41026   FALSE FALSE
## Shimmer.APQ3      1.000000      94.35897   FALSE FALSE
## Shimmer.APQ5      1.000000      96.92308   FALSE FALSE
## MDVP.APQ          1.000000      96.92308   FALSE FALSE
## Shimmer.DDA       1.000000      96.92308   FALSE FALSE
## NHR               1.000000      94.87179   FALSE FALSE
## HNR               1.000000     100.00000   FALSE FALSE
```

```
## RPDE              1.000000     100.00000   FALSE FALSE
## DFA               1.000000     100.00000   FALSE FALSE
## spread1           1.000000     100.00000   FALSE FALSE
## spread2           2.000000      99.48718   FALSE FALSE
## D2                1.000000     100.00000   FALSE FALSE
## PPE               1.000000     100.00000   FALSE FALSE
```

Todas las variables presentan una variabilidad suficiente como para poder aportar información en la clasificación.

Veamos que importancia tiene cada variable en relación con la variable de clasificación:

```
##             DFA     MDVP.Fhi.Hz.     MDVP.Flo.Hz.      MDVP.Fo.Hz.
##       0.6498016        0.6748866        0.6972789        0.7006803
##            RPDE               D2              HNR      Shimmer.DDA
##       0.7071995        0.7249150        0.7379535        0.7546769
##     Shimmer.APQ3     Shimmer.APQ5              NHR         MDVP.RAP
##       0.7548186        0.7699121        0.7731718        0.7769274
##       Jitter.DDP   MDVP.Jitter...     MDVP.Shimmer MDVP.Shimmer.dB.
##       0.7774235        0.7777069        0.7827381        0.7850765
##        MDVP.PPQ MDVP.Jitter.Abs.          spread2         MDVP.APQ
##       0.7872024        0.7889739        0.8136338        0.8258929
##         spread1              PPE
##       0.8969671        0.8969671
```

Ninguna variable obtiene una puntuación que nos asegure que no es lo suficientemente importante como para no incluirla en el modelo.

Por último, nos queda comprobar si existe correlación entre las variables. En la figura 1 se muestran las correlaciones más altas entre variables. Concrétamente se distinguen en 4 tonalidades que van desde el azul oscuro al claro las correlaciones mayores a 0.95, 0.9, 0.85 y 0.8 respectivamente.

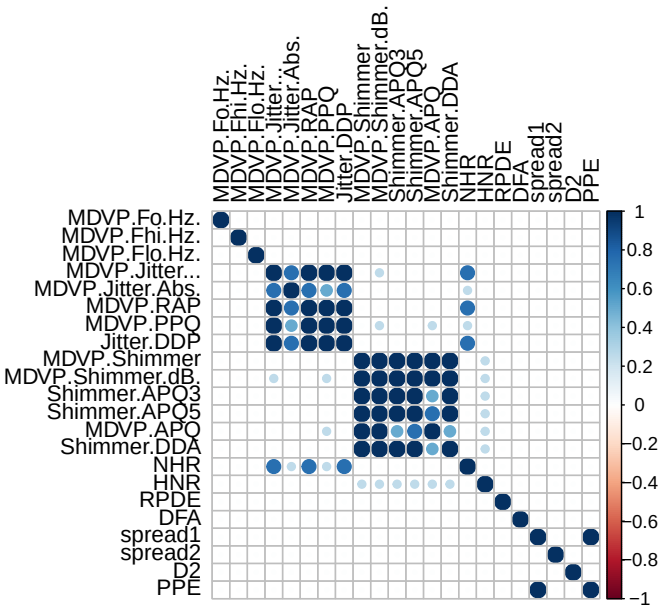

Figure 1: Correlación entre variables

Por lo tanto, si establecemos 0.95 como la máxima correlación que pueden tener dos variables en el modelo, tendremos que escoger una variable entre MDVP.Jitter, MDVP.RAP, MDVP.PPQ y Jitter.DDP, entre MDVP.Shimmer y MDVP.Shimmer.dB y entre spread1 y PPE. Basándonos en la importancia de las variables nos quedaremos con MVDP.PPQ, MDVP.Shimmer.dB y spread1.

Por último, analicemos la estructura de asociación entre las variables seleccionadas mediante el Análisis de Componentes Principales. Vemos que los cuatro primer componentes recogen en torno al 80% de la variabilidad:

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     2.4188391 1.4990492 1.1488098 0.98966759 0.88775987
## Proportion of Variance 0.4500602 0.1728576 0.1015203 0.07534169 0.06062443
## Cumulative Proportion  0.4500602 0.6229178 0.7244381 0.79977980 0.86040423
##                            Comp.6     Comp.7     Comp.8     Comp.9    Comp.10
## Standard deviation     0.73897244 0.66502957 0.50220130 0.47543920 0.39551240
## Proportion of Variance 0.04200617 0.03402033 0.01940047 0.01738788 0.01203308
## Cumulative Proportion  0.90241040 0.93643074 0.95583121 0.97321909 0.98525217
##                           Comp.11     Comp.12     Comp.13
## Standard deviation     0.310618050 0.244322628 0.188532894
## Proportion of Variance 0.007421813 0.004591811 0.002734204
## Cumulative Proportion  0.992673985 0.997265796 1.000000000
```

En la figura 2 podemos ver la estructura de los primeros cuatro componentes principales respecto al status del paciente. En los dos primeros componentes parece haber una pequeña diferencia entre el comportamiento de los pacientes sanos y enfermos, aunque en general no hay una distinción clara.

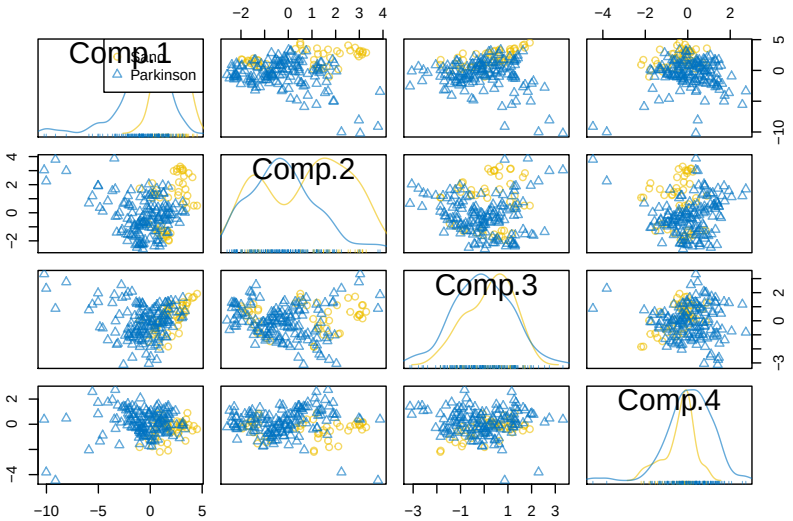

Figure 2: Estructura de los primeros cinco componentes principales por status del paciente

Una vez realizado en análisis preliminar que nos ha permitido conocer mejor los datos y hacer una limpia de aquellas variables innecesarias, vamos a dividir el conjunto de datos

en un conjunto de entrenamiento y validación que contengan el 75% y el 25% de los datos, respectivamente.

3 Clasificador tipo Bagging

El primer clasificador escogido ha sido uno de tipo árbol, concretamente el metaclasificador Bagging. Este método crea varios modelos a partir de diferentes muestras, y clasifica cada individuo de acuerdo a lo que diga la mayoría de los modelos.

En la figura 3 podemos observar la importancia de las variables a la hora de clasificar las muestras. Vemos que las variables spread1 y MDVP.Fo.Hz son, con diferencia, las más importantes. También resulta curioso que la variable MDVP.Jitter.Abs no parece tener ninguna importancia.

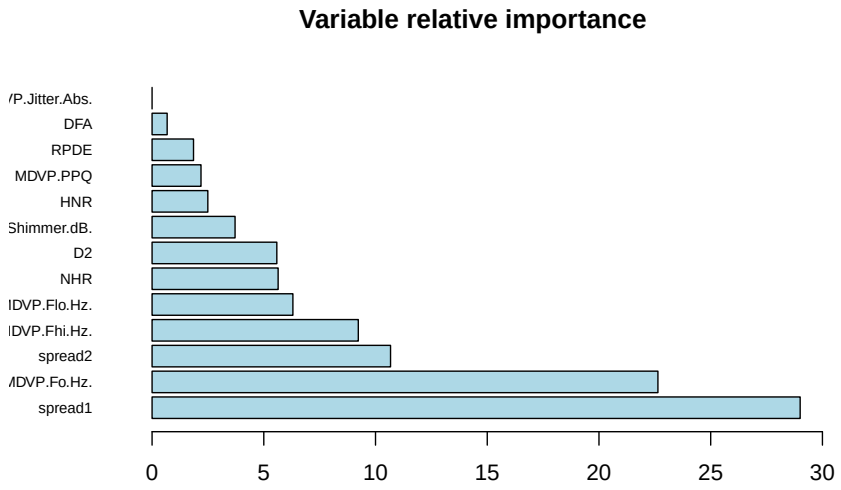

Figure 3: Importancia de las variables en la clasificación de tipo Bagging

En la figura 4 podemos observar el error frente al número de árboles. Vemos que en torno a las 40 iteraciones el error se minimiza.

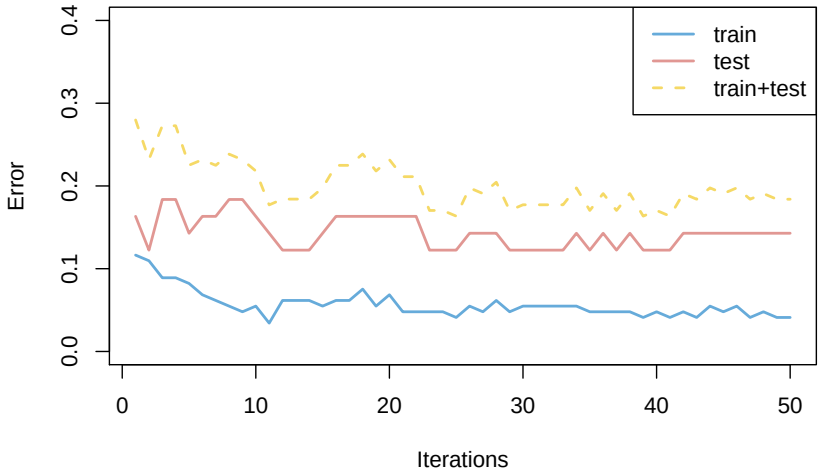

Figure 4: Error del método Bagging frente al número de árboles

4 Clasificador tipo SVM

El segundo clasificador escogido ha sido uno de tipo SVM (máquina de vector soporte). Se va a hacer uso de la librería `e1071`, ya que tiene implementados los métodos de tipo SVM. Para reducir la dimensionalidad del problema vamos a trabajar con los primeros cuatro componentes principales, ya que como hemos visto recogen una gran parte de la variabilidad.

Uno de los métodos implementados permite, mediante validación cruzada, dar con el clasificador que mejor se ajusta a los datos:

```
##    degree coef0 cost
## 61      1     0   10
```

Nos indica que el mejor modelo es un clasificador de grado 1, coeficiente 0 y coste 10. En la figura 5 podemos visualizar la clasificación que realiza ese modelo sobre los primeros dos componentes principales. Aunque este gráfico nos da una idea de como se realiza la clasificación sobre unas dimensiones concretas (en este caso PC1 y PC2), no podemos visualizar del todo bien si las muestras están bien o mal clasificadas.

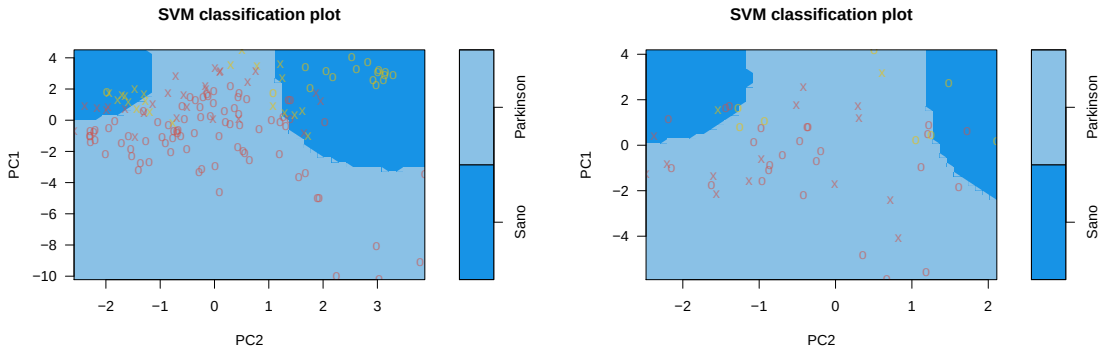

Figure 5: Clasificación del modelo SVM sobre los primeros dos componentes principales (izquerda:muestra de entrenamiento, derecha: muestra de validación) y clasificación real (amarillo: sano, rojo:parkinson)

5 Clasificador tipo Boosting

El último clasificador a analizar va a ser el de tipo Boosting. Este clasificador combina los resultados de clasificadores blandos, adjudicando un peso a cada uno de ellos en función de su historial de aciertos-fallos. Con tal de intentar utilizar los clasificadroes más blandos posibles, se suelen utilizar árboles de profundidad 1 (stumps), y es lo que haremos en este caso. En R hay más de un paquete que implementa este método, entre los que hemos escogido `gbm`.

En la figura 6 podemos observar la importancia de las variables para el clasificador de tipo Boosting. En este caso también, las variables spread1 y MDVP.Fo.Hz son las más importantes, aunque no se alejan de las demás tanto como lo hacían en el modelo Bagging.

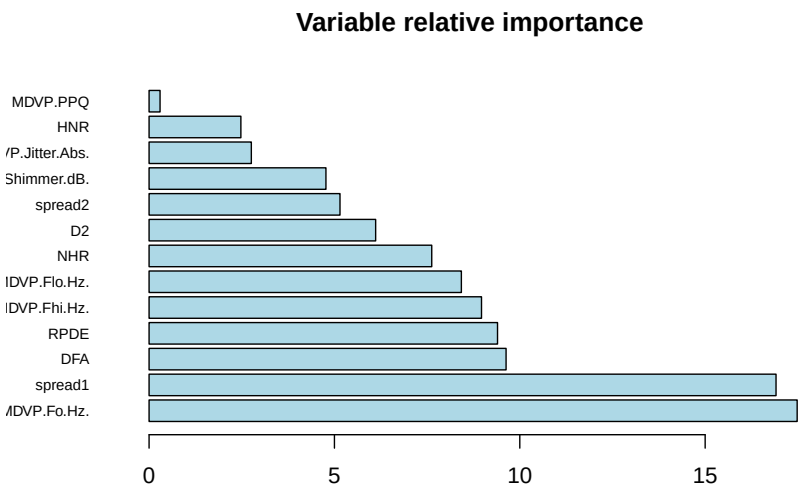

Figure 6: Importancia de las variables en la clasificación de tipo Boosting

Por otro lado, podemos analizar el error para las muestras de entrenamiento y validación en base al número de árboles utilizados. Tal y como vemos en la figura 7a partir de entorno a 65-70 árboles el error cometido en la muestra de entrenamiento es nulo, por lo que se ajusta muy bien a los datos. Sin embargo hacen falta más árboles para que el error en la muestra de validación decrezca.

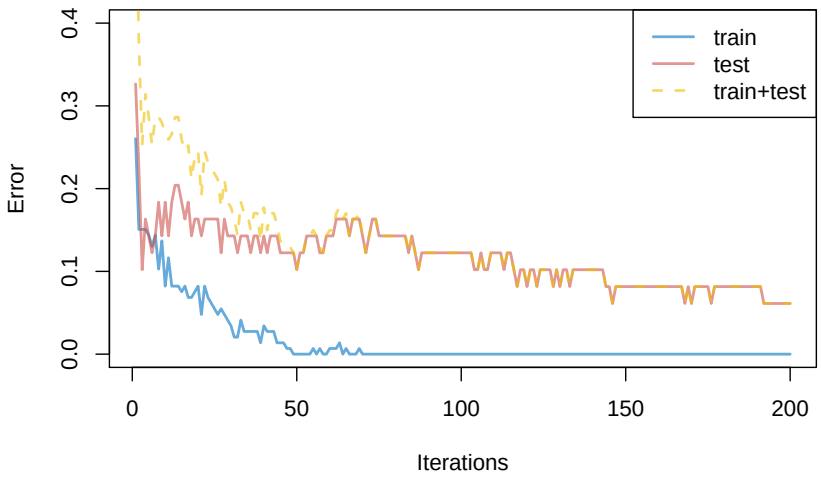

Figure 7: Error del método Boosting frente al número de árboles

6 Comparación de clasificadores

Disponemos de varias herramientas para analizar y comparar la calidad de los modelos seleccionados.

6.1 Tablas de confusión

Uno de los indicadores más fáciles de interpretar son las tablas de confusión, que resumen los aciertos y errores cometidos durante la clasificación.

Comencemos por ver las predicciones del primer modelo, el metaclasificador Bagging.

```
##                            bagging.pred.train
## parkinson.fil.train$status Parkinson Sano
##                   Sano               6   32
##                   Parkinson        108    0
```

```
##                           bagging.pred.test
## parkinson.fil.test$status Parkinson Sano
##                  Sano               5    5
##                  Parkinson         37    2
```

Para el modelo SVM las tablas de confusión correspondientes a la muestra de entrenamiento y validación son las siguientes:

```
##                            svm.pred.train
## parkinson.fil.train$status Sano Parkinson
##                   Sano        34         4
##                   Parkinson    1       107
```

```
##                           svm.pred.test
## parkinson.fil.test$status Sano Parkinson
##                  Sano         7         3
##                  Parkinson    1        38
```

Por último veamos las tablas de confusión para el modelo Boosting.

```
##                            boost.pred.train
## parkinson.fil.train$status Parkinson Sano
##                   Sano               0   38
##                   Parkinson        108    0
```

```
##                           boost.pred.test
## parkinson.fil.test$status Parkinson Sano
##                  Sano               2    8
##                  Parkinson         38    1
```

El clasificador Boosting es el que mejor se adapta a la muestra de entrenamiento, ya que consigue clasificar correctamente todos los registros de voz, y el SVM el que mejor clasifica en la muestra de validación, ya que detecta un enfermo más.

6.2 Curva ROC

Otra herramienta para medir la validez de un modelo es la curva ROC, la cual nos da una idea sobre la habilidad del modelo a la hora de distinguir entre positivos y negativos. Representa la probabilidad de predecir un positivo real (sensibilidad) frente a la probabilidad de predecir un negativo real(especificidad). Un buen modelo tiene que conseguir que estos porcentajes sean lo más altos posibles, por lo que una forma de medir la calidad del modelo es calcular el area bajo la curva (AUC) ROC.

En el caso del modelo Bagging las curvas ROC para las muestras de entrenamiento y validación se muestran en la figura 8.

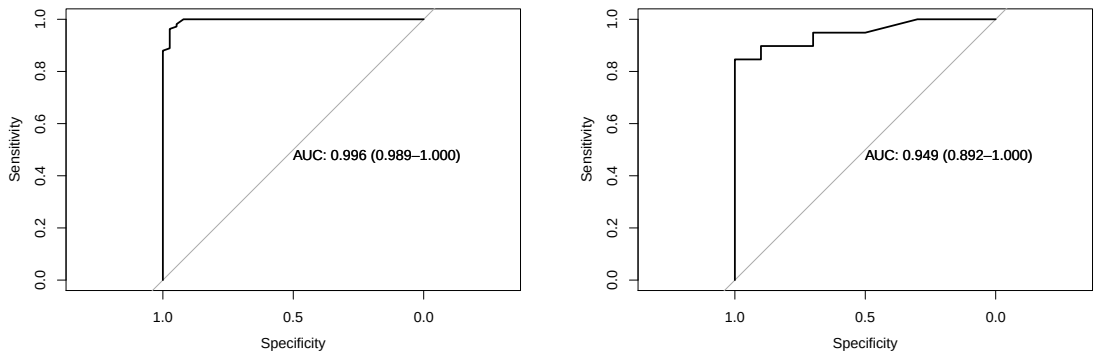

Figure 8: Curvas ROC para el modelo Bagging (izquierda: entrenamiento, validación)

Por su parte, en la figura 9 podemos observar las curvas para el modelo SVM sobre la muestra de entrenamiento y validación.

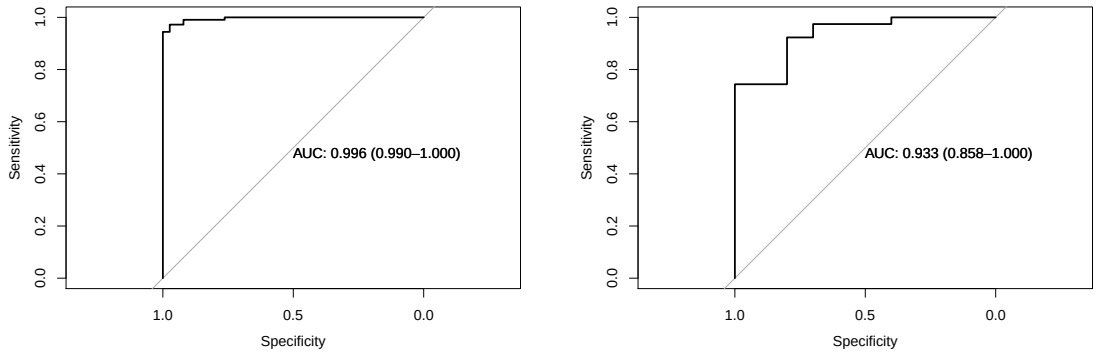

Figure 9: Curvas ROC para el modelo SVM e(izquierda: entrenamiento, validación)

Por último, las curvas ROC para las muestras de entrenamiento y validación del modelo Boosting las podemos encontrar en la figura 10.

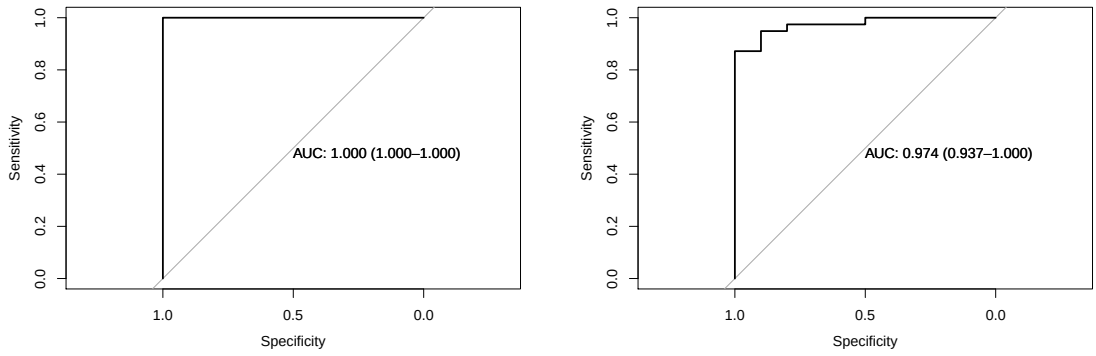

Figure 10: Curvas ROC para el modelo Boosting(izquierda: entrenamiento, validación)

Atendiendo al valor AUC de las curvas ROC de cada modelo, podríamos concluir que el modelo Boosting es el más adecuado, ya que consigue el mayor valor tanto en la muestra de entrenamiento como en la de validación.

6.3 Curva de ganancia

Las curvas Lift y de ganancia surgen del interés del marketing para calcular a qué extensión de la población se debe llegar para garantizar que se alcanza un cierto porcentaje de la población que se quiere cubrir. Concretamente, la curva de ganancia mide en el eje horizontal el porcentaje de muestra de entrenamiento que hay que tomar para conseguir el porcentaje de casos positivos que marca el eje vertical. Por lo tanto cuanto menor porcentaje de la muestra de entrenamiento utilice para detectar todos los positivos, mejor será el modelo.

Las figuras 11 y 12 muestran las curvas de ganancia en el modelo de entrenamiento y de validación para los tres modelos propuestos. La parte inferior del triangulo sombreado corresponde a la curva que obtendría un modelo aleatorio, y la parte superior corresponde al modelo perfecto que acertaría el 100% de los casos en el porcentaje real de casos en la muestra. Por lo tanto, el modelo Boosting es el que mejor se ajusta a la muestra de entrenamiento, ya que realiza una clasificación perfecta. En el caso de la muestra de validación el SVM parece ser el peor con diferencia ya que es el que necesita mayor porcentaje de muestra para conseguir el 100% de aciertos. Entre el Bagging y el Boosting, el primero alcanza un poco antes el 100% de los aciertos, pero es verdad que el Boosting alcanza un alto porcentaje de aciertos antes que el Bagging.

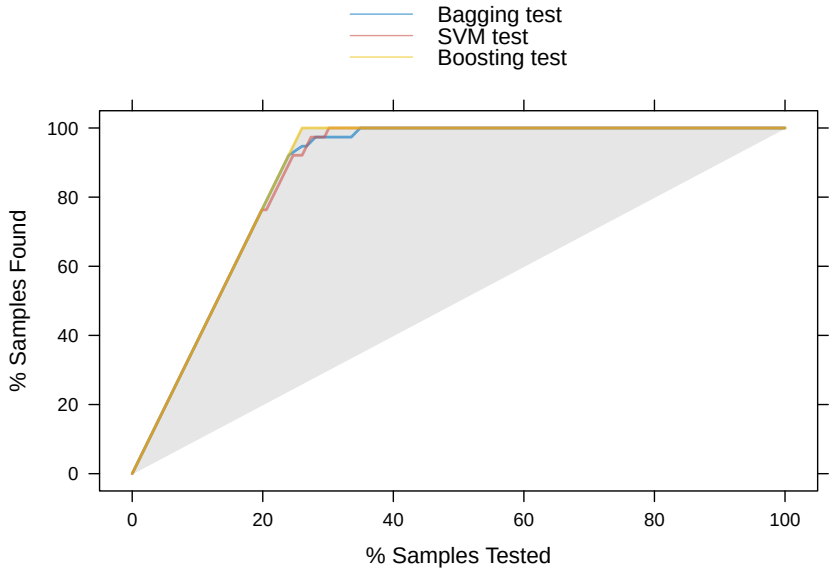

Figure 11: Curvas Lift de la muestra de entrenamiento para los diferentes modelos

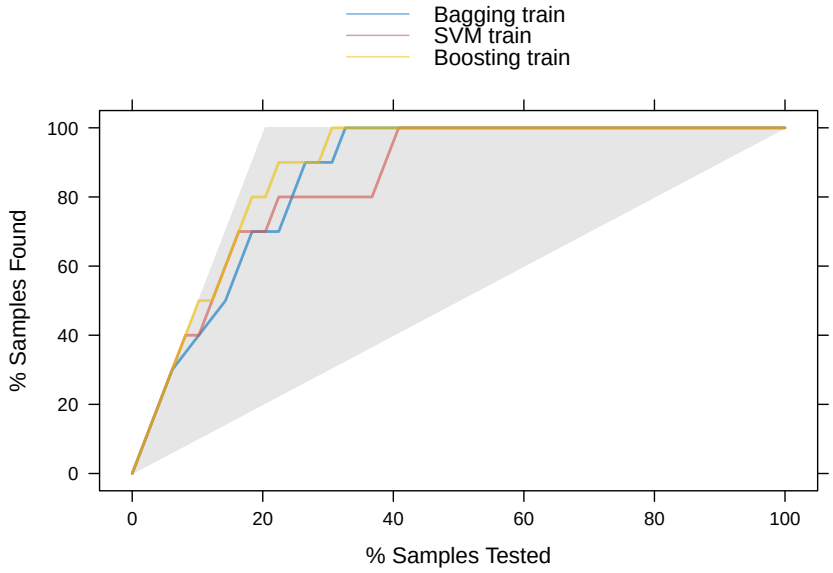

Figure 12: Curvas Lift de la muestra de validación para los diferentes modelos

7 Conclusiones

Durante este trabajo se han analizado diferentes técnicas de clasificación con el objetivo de identificar enfermos de Parkinson en base a las características de su voz. Concretamente, se han analizado un metaclasificador de tipo Bagging, un clasificador de tipo SVM y otro de tipo Boosting, gracias a las distintas librerías de las que dispone R.

El conjunto de datos disponía de 22 características de los registros de voz de los pacientes, por lo que, con tal de reducir la dimensión del ajuste y obtener modelos más simples, se ha intentado reducir esa cantidad. Existía gran correlación entre varias variables, por lo que se ha decidido descartar aquellas que menos importancia han mostrado frente a la variable de clasificación. De esta manera se ha conseguido reducir el conjunto de datos de 22 variables descriptivas a 13.

También se han calculado los componentes principales de este nuevo grupo de variables, y se ha visto que los cuatro primeros recogen en torno al 80% de la variabilidad del conjunto de datos. Ésto ha resultado ser muy útil ya que haciendo uso sólo de estos cuatro componentes se ha construido un modelo SVM, que, aunque no ha resultado ser el mejor, y a costa de clasificar 7 de los 48 registros de pacientes sanos como si lo fueran de enfermos, ha detectado 145 de los 147 registros de enfermos de Parkinson.

Para medir la validez de los modelos y compararlos entre ellos se han analizado las tablas de confusión, curvas ROC y curvas de ganancia de los modelos. Las tablas de confusión han mostrado que el modelo tipo Boosting ha sido el que más ha acertado. Mediante el análisis de las curvas ROC de los diferentes modelos se ha podido ver que las curvas del modelo tipo Boosting han conseguido un mayor AUC, por lo que este modelo es el más capaz de detectar el máximo de positivos reales sin diagnosticar falsos positivos. Por último, las curvas Lift han mostrado que el modelo tipo Boosting tiene mayor capacidad de detectar positivos para un tamaño de muestra concreto. Por lo tanto, se puede concluir que el mejor modelo obtenido ha sido el de tipo Boosting. Este modelo consigue clasificar correctamente 145 de los 147 registros correspondientes a enfermos, adjudicando Parkinson a tan solo 3 de las 48 grabaciones de pacientes sanos.

El modelo Boosting ha mostrado que las variables más importantes a la hora de realizar la clasificación han sido spread1 y MDVP.Fo.Hz. Además, tanto este modelo como el Bagging sitúan entre las cinco características más importantes a spread1, MDVP.Fo.Hz, MDVP.Fhi.Hz y MDVP.Flo.Hz. Esto conduce a concluir que el tener Parkinson influye en los valores mínimos, medios, máximos y las medidas no-lineales de la variación de la frecuencia fundamental, y por lo tanto, en la voz de un enfermo.

8 Apéndice: Código

```
# Cargar librerías necesarias a priori
library(Rcmdr)
library(tidyverse)
library(colorspace)

# Cargar fichero de datos
path="D:/Miren/Master I/MD/Trabajo final/parkinsons.data"
parkinson <- read.table(path, sep=',', header = TRUE)
parkinson$status<-factor(parkinson$status,
                         labels=c('Sano','Parkinson'))
# Resumen
summary(parkinson)

# Comprobar varianza de variables
require(caret)
nearZeroVar(parkinson[-c(1,18)], saveMetrics= TRUE)

# Importancia de las variables
rocvarimp2<-filterVarImp(x = parkinson[-c(1,18)],
                         y = as.factor(parkinson$status))
apply(rocvarimp2, 1, mean) %>% sort()

# Gráfico de correlaciones
require("corrplot")
corrplot((abs(cor(parkinson[-c(1,18)]))>0.95)*0.25+
           (abs(cor(parkinson[-c(1,18)]))>0.9)*0.25+
           (abs(cor(parkinson[-c(1,18)]))>0.85)*0.25+
           (abs(cor(parkinson[-c(1,18)]))>0.8)*0.25, method="circle",
         tl.col = "black")
```

```
# Crear nuevo dataset con las variables que interesan
# (Status es el elemento número 9)
parkinson.fil <- parkinson[-c(1, 5, 7, 9, 10, 12, 13, 14, 15, 24)]
parkinson.fil<-parkinson.fil%>%relocate(status)

# Calcular componentes principales
parkinson.PC <- princomp(parkinson.fil[-c(1)], cor=TRUE, scores=TRUE)
summary(parkinson.PC)

# Añadir los PC al dataset
parkinson.fil$PC1<-parkinson.PC$scores[,1]
```

```
parkinson.fil$PC2<-parkinson.PC$scores[,2]
parkinson.fil$PC3<-parkinson.PC$scores[,3]
parkinson.fil$PC4<-parkinson.PC$scores[,4]

# Graficar CP por status
scatterplotMatrix(~parkinson.PC$scores[,1:4] | status, regLine=FALSE,
                  smooth=FALSE,
                  diagonal=list(method="density"), by.groups=TRUE,
                  data=parkinson.fil, col=c('#EFC00099','#0073C299'))
```

```
# Crear muestras
set.seed(725)
n_data<-195
train<-sample(c(1:n_data), round(0.75 *n_data)) # Muestra entrenamiento
test<- setdiff(c(1:n_data), train) # Muestra validación
parkinson.fil.train<-parkinson.fil[train,]
parkinson.fil.test<-parkinson.fil[test,]

# Crear modelo bagging
library(adabag)
set.seed(200323)
parkinson.bagging <- bagging(status~., data=parkinson.fil.train[,1:14],
                             mfinal=50)
# Plotear importancia
a<-rev(sort(parkinson.bagging$importance))
importanceplot(parkinson.bagging,cex.names=0.7, horiz=TRUE)
```

```
# Plotear error
errorevol.train <-errorevol(parkinson.bagging,
                            parkinson.fil.train[,1:14] )
errorevol.test <-errorevol(parkinson.bagging,
                            parkinson.fil.test[,1:14] )
plot(errorevol.train[[1]], type = "l", xlab = "Iterations",
     ylab = "Error", col = '#0073C299', lwd = 2, ylim=c(0,0.4))
lines(errorevol.test[[1]], cex = 0.5, col = '#CD534C99', lty = 1,
      lwd = 2)
lines(errorevol.test[[1]]+errorevol.train[[1]], cex = 0.5,
col = '#EFC00099', lty = 2,lwd = 2)
legend("topright", c("train", "test","train+test"),
       col=c('#0073C299','#CD534C99','#EFC00099'), lty = c(1,1,2),
       lwd = 2)
```

```
# SVM
# Calcular modelo svm óptimo mediante CV
require(e1071)
parkinson.svm.PC<-tune.svm(status~PC1+PC2+PC3+PC4,
                           data=parkinson.fil.train,
              coef0=c(0, 0.5, 1,1.5,2.5,5 ), degree=1:5,
              cost=c(0.1,1,10))
parkinson.svm.PC$best.parameters
parkinson.svm.PC.1<-svm(status~PC1+PC2+PC3+PC4,
                        data=parkinson.fil.train,
                        coef0=0, cost=10, degree=1, probability = TRUE)

# Plotear modelo
plot(parkinson.svm.PC.1,data=parkinson.fil.train,
     formula=PC1~PC2,symbolPalette=c('#EFC00099','#CD534C99'),
     color.palette=hsv_palette(h = 204/360, from = 0.9, to = 0.4, v =0.9))
```

```
plot(parkinson.svm.PC.1,data=parkinson.fil.test,
     formula=PC1~PC2,symbolPalette=c('#EFC00099','#CD534C99'),
     color.palette=hsv_palette(h = 204/360, from = 0.9, to = 0.4, v =0.9))
```

```
# Modelo boosting
library(gbm)
cntrl<-rpart.control(maxdepth=1)
parkinson.boost <- boosting(status ~ .,
                            data=parkinson.fil.train[,1:14],
                            mfinal=200, control=cntrl)

# Plotear importancia
a<-rev(sort(parkinson.boost$importance))
importanceplot(parkinson.boost,cex.names=0.7, horiz=TRUE)
```

```
# Plotear error
boost.errorevol.train <-errorevol(parkinson.boost,
                            parkinson.fil.train[,1:14] )
boost.errorevol.test <-errorevol(parkinson.boost,
                            parkinson.fil.test[,1:14] )
plot(boost.errorevol.train[[1]], type = "l", xlab = "Iterations",
     ylab = "Error", col = '#0073C299', lwd = 2, ylim=c(0,0.4))
lines(boost.errorevol.test[[1]], cex = 0.5, col = '#CD534C99', lty = 1,
      lwd = 2)
lines(boost.errorevol.test[[1]]+boost.errorevol.train[[1]],
      cex = 0.5, col = '#EFC00099', lty = 2,lwd = 2)
```

```
legend("topright", c("train", "test","train+test"),
       col=c('#0073C299','#CD534C99','#EFC00099'), lty = c(1,1,2),
       lwd = 2)
```

```
# Tablas de confusión bagging
# Calcular valores y probabilidades para bagging
bagging.pred.train<-predict(parkinson.bagging,
                            parkinson.fil.train[,1:14],
                            probability = TRUE)
bagging.pred.train.prob<-bagging.pred.train$prob
bagging.pred.train<-bagging.pred.train$class
bagging.pred.test<-predict(parkinson.bagging,
                            parkinson.fil.test[,1:14],
                            probability = TRUE)
bagging.pred.test.prob<-bagging.pred.test$prob
bagging.pred.test<-bagging.pred.test$class
# Tablas
xtabs(~parkinson.fil.train$status+bagging.pred.train)
xtabs(~parkinson.fil.test$status+ bagging.pred.test)

# Tablas de confusión SVM
# Calcular valores y probabilidades para SVM
svm.pred.train<-predict(parkinson.svm.PC.1, parkinson.fil.train,
                        probability = TRUE)
svm.pred.train.prob<-attr(svm.pred.train,"probabilities")
svm.pred.test<-predict(parkinson.svm.PC.1, parkinson.fil.test,
                        probability = TRUE)
svm.pred.test.prob<-attr(svm.pred.test,"probabilities")
xtabs(~parkinson.fil.train$status+svm.pred.train)
xtabs(~parkinson.fil.test$status+ svm.pred.test)

# Tablas de confusión Boosting
# Probabilidades y predicciones para el modelo Boosting
boost.pred.train<-predict(parkinson.boost,parkinson.fil.train)
boost.pred.test<-predict(parkinson.boost,parkinson.fil.test)
boost.pred.train.prob<-boost.pred.train$prob
boost.pred.test.prob<-boost.pred.test$prob
boost.pred.train<-boost.pred.train$class
boost.pred.test<-boost.pred.test$class
# Tablas
xtabs(~parkinson.fil.train$status+boost.pred.train)
xtabs(~parkinson.fil.test$status+ boost.pred.test)

# Curvas ROC para Bagging
```

```
library(pROC)
ROC_bagging_train<-roc(as.numeric(parkinson.fil.train$status),
             bagging.pred.train.prob[,2],
             ci=TRUE)
plot(ROC_bagging_train, print.auc=TRUE)
```

```
ROC_bagging_test<-roc(as.numeric(parkinson.fil.test$status),
             bagging.pred.test.prob[,2],
             ci=TRUE)
plot(ROC_bagging_test, print.auc=TRUE)
```

```
# Curvas ROC para SVM
ROC_svm_train<-roc(as.numeric(parkinson.fil.train$status),
             svm.pred.train.prob[,2],
             ci=TRUE)
plot(ROC_svm_train, print.auc=TRUE)
```

```
ROC_svm_test<-roc(as.numeric(parkinson.fil.test$status),
             svm.pred.test.prob[,2],
             ci=TRUE)
plot(ROC_svm_test, print.auc=TRUE)
```

```
# Curvas ROC para Boosting
library(pROC)
ROC_boost_train<-roc(as.numeric(parkinson.fil.train$status),
             boost.pred.train.prob[,2],
             ci=TRUE)
plot(ROC_boost_train, print.auc=TRUE)
```

```
ROC_boost_test<-roc(as.numeric(parkinson.fil.test$status),
             boost.pred.test.prob[,2],
             ci=TRUE)
plot(ROC_boost_test, print.auc=TRUE)
```

```
# Curvas Lift
lift_train<-lift(status ~ bagging.pred.train.prob[,1]+
                   svm.pred.train.prob[,1]+
                   boost.pred.train.prob[,1],
                 data=parkinson.fil.train, class="Sano")
lift_test<-lift(status ~ bagging.pred.test.prob[,1]+
                   svm.pred.test.prob[,1]+
                   boost.pred.test.prob[,1],
```

```
                  data=parkinson.fil.test, class="Sano")
xyplot(lift_train, auto.key=TRUE,
       plot="gain",col=c('#0073C299','#CD534C99','#EFC00099'),
       lwd=c(2,2,2),
       key=list(space="top",
         lines=list(col=c('#0073C299','#CD534C99','#EFC00099')),
         text=list(c("Bagging test", "SVM test", "Boosting test"))))
```

```
xyplot(lift_test,
       plot="gain",col=c('#0073C299','#CD534C99','#EFC00099'),
       lwd=c(2,2,2),
       key=list(space="top",
         lines=list(col=c('#0073C299','#CD534C99','#EFC00099')),
         text=list(c("Bagging train", "SVM train",
                     "Boosting train"))))
```