Clasificación de tumores con Machine Learning

Joaquín Amat Rodrigo

Mayo, 2018

Tabla de contenidos

Versión PDF: Github

Introducción

Lograr un tratamiento eficaz contra el cáncer depende en gran medida de una correcta clasificación del tumor, ya que, en base a esto, se debe tratar al paciente con un fármaco u otro. Durante mucho tiempo, el principal criterio de clasificación se ha basado en el tipo de tejido u órgano en el que se detectaba el tumor (criterios clínicos e histopatológicos), sin embargo, dada las innumerables condiciones en las que se diagnostica el cáncer, no siempre es trivial y robusto seguir este criterio. Una alternativa que ha emergido gracias a las nuevas tecnologías de microarrays consiste en clasificar los tumores en función de su perfil molecular. Esta idea parte de la premisa de que, tumores con un mismo origen, tienen un patrón molecular (expresión genética, mutaciones, proteoma…) similar o, al menos, más similar que si se comparan con tumores de otro tipo. A lo largo de este documento, se analiza la capacidad de distintos algoritmos de *machine learning* para encontrar patrones en los niveles que expresión genética que permitan clasificar distintos tipos de tumores de forma correcta.

Objetivos del estudio

Antes de realizar cualquier análisis, y a pesar de que a lo largo del proceso siempre aparecen nuevos interrogantes, es importante establecer qué preguntas se quieren responder con los datos disponibles. En este estudio, los puntos principales son:

- ¿Existe algún método de *machine learning* capaz de predecir, con un porcentaje de acierto alto, el tipo de tumor en función sus niveles de expresión?
- De entre los varios miles de genes disponibles ¿Son todos importantes en vista a la clasificación o son solo unos pocos los que aportan información relevante?

Se trata, por lo tanto, de un problema de clasificación multiclase supervisado.

A diferencia de otros documentos, este pretende ser un ejemplo práctico con menos desarrollo teórico. El lector podrá darse cuenta de lo sencillo que es aplicar un gran abanico de métodos predictivos con R y sus librerías. Sin embargo, es crucial que cualquier analista entienda los fundamentos teóricos en los que se basa cada uno de ellos para que un proyecto de este tipo tenga éxito. Aunque aquí solo se describan brevemente, estarán acompañados de links donde encontrar información detallada.

Algunas partes de este análisis, por ejemplo, el filtrado *Signal to Noise* o el ajuste de las redes neuronales, pueden tardar más de una hora (Intel® Core™ i5-4210U CPU @ 1.70GHz × 4 7,7 GiB RAM).

Datos

Los datos de expresión empleados en este documento se han obtenido de la publicación Multi-Class Cancer Diagnosis Using Tumor Gene Expression Signatures. Siguiendo el link anterior se puede acceder a un repositorio que contiene tanto el artículo como a los *datasets* empleados.

- *GCM_Training.res*: contiene los datos de entrenamiento (144 muestras tumorales).
- *GCM_Test.res*: contiene los datos de test (54 muestras, 46 tumores primarios y 8 metastásicos).
- *GCM_Total.res*: contiene los datos de entrenamiento y test, excepto los 8 tumores metastásicos, y 90 muestras de tejido no tumoral.

El formato .res es una tipo de archivo de texto *tab-delimited* en el que cada fila representa un gen (con identificador único y descripción) y cada columna una muestra. Los valores numéricos se corresponden con la expresión de cada gen en cada muestra. Además, para cada muestra, se incluye una columna adicional que indica si la lectura está presente (P) o ausente (A). Con la finalidad de representar mejor lo que ocurre en la práctica cuando un analista se enfrenta a un nuevo set de datos, se emplea el archivo *GCM_Total.res* que contiene todas las muestras juntas.

```
# Lectura de fichero GCM_Total.res
library(tidyverse)
datos <- read_delim(file = "GCM_Total.res", delim = "\t", col_names = TRUE,
                    progress = FALSE)
```

Reestructuración de los datos

Si bien la información viene dada en una estructura bastante organizada, se realizan una serie de modificaciones para almacenarla en un *dataframe* en el que cada fila representa una muestra y las columnas contienen la información relativa a ellas (información descriptiva sobre el tipo de tumor y la expresión de los genes).

```
# Exploración de los nombres de las columnas
colnames(datos) %>% head(10)

# Se eliminan las columnas que indican si el valor está presente o ausente.
# Como estas columnas están identificadas por números, al importarlas con
# read_delim() se les añade delante la letra X. Este patrón común permite
# excluirlas de forma sencilla.
datos <- datos %>% select(-starts_with("X"))
```

```
# Las dos primeras filas no contienen información
datos <- datos[-c(1:2), ]

# Las columnas "Description" y "Accession" contienen la descripción de cada gen
# y un identificador único. Dado que la descripción puede ser bastante larga, se
# almacena en un dataframe adicional y se excluye del set de datos principal.

descripcion_genes <- datos %>% select(Description, Accession)
datos             <- datos %>% select(-Description)

# Se pivota la tabla de forma que cada fila es una muestra y cada columna un gen.
datos <- datos %>% gather(key = "muestra", value = "expresion", -Accession)
datos <- datos %>% spread(key = Accession, value = expresion)

# El nombre de cada muestra contiene información descriptiva concatenada en una
# única frase. Se añade un identificador numérico a cada muestra y se separa la
# información contenida en el nombre en varias columnas nuevas.
datos %>% select(muestra) %>% head()
datos <- datos %>% mutate(id_muestra = 1:nrow(datos))
datos <- datos %>% select(id_muestra, everything())
datos <- datos  %>% separate(col = muestra, sep = "__",
                             into = c("tipo_muestra", "resto_variable"))
datos <- datos  %>% separate(col = resto_variable, sep = "_",
                             into = c("tipo_tumor", "subtipo_tumor"))

# Escritura de los dos nuevos sets de datos
write_delim(x = datos, path = "GCM_Total_clean.txt", delim = "\t")
write_delim(x = descripcion_genes, path = "GCM_Total_descrip_genes.txt",
            delim = "\t")
```

Los archivos resultantes del proceso reestructuración (*GCM_Total_clean.txt* y *GCM_Total_descrip_genes.txt*) pueden descargarse directamente del repositorio Github.

```
library(tidyverse)

# Al tratarse de un archivo con tantas columnas, se agiliza su lectura si se
# especifica el tipo de cada columna. En este caso, excepto las primeras 4
# columnas, todas son de tipo numérico.
datos <- read_delim(file = "GCM_Total_clean.txt", delim = "\t",
                    col_types = paste(c("i","c","c","c", rep("d",16063)),
collapse=""),
                    col_names = TRUE)

descripcion_genes <- read_delim(file = "GCM_Total_descrip_genes.txt",
                                delim = "\t")
```

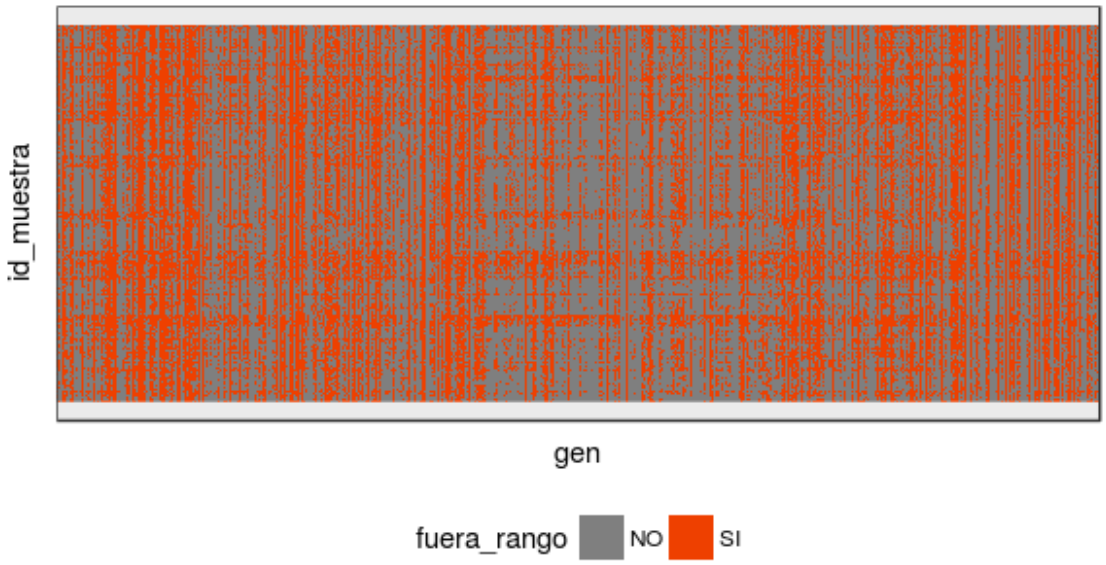
Filtrado de calidad de datos

Acorde a la información aportada por los autores de la publicación, la tecnología microarray empleada para cuantificar los niveles de expresión tiene unos límites de detección, fuera de los cuales, la lectura no es fiable. En concreto, si el experimento ha salido bien pero la señal es inferior a 20 unidades, se considera ruido y debe interpretarse como que no hay expresión de ese gen. En el extremo opuesto, el detector se satura en las 16000 unidades, por lo que este es el valor máximo. Antes de proceder a sustituir los valores inferiores a 20 por 0 y los superiores a 16000 por 16000, se analiza cuantas observaciones están fuera de rango.

```
datos_long <- datos %>% select(-tipo_muestra, -tipo_tumor, -subtipo_tumor) %>%
              gather(key = "gen", value = "expresion", -id_muestra) %>%
              mutate(fuera_rango = if_else(expresion < 20 | expresion > 16000,
                                           "SI", "NO"))

# Proporción de valores por debajo del mínimo de detección
nrow(datos_long %>% filter(expresion < 20)) / nrow(datos_long)
```

```
## [1] 0.3927066
```

```
# Proporción de valores por encima del máximo de detección
nrow(datos_long %>% filter(expresion > 16000)) / nrow(datos_long)

## [1] 0.000824877

# Proporción de valores fuera de rango de detección
nrow(datos_long %>% filter(fuera_rango == "SI")) / nrow(datos_long)

## [1] 0.3935315

# Representación gráfica de los valores fuera de rango de detección
ggplot(data = datos_long, aes(x = gen, y = id_muestra, fill = fuera_rango)) +
  geom_raster() +
  scale_fill_manual(values = c("gray50", "orangered2")) +
  theme_bw() +
  theme(legend.position = "bottom",
        axis.text = element_blank(),
        axis.ticks = element_blank())
```



Un 39% de las lecturas están fuera de rango, lo que parece un porcentaje muy alto. Prácticamente la totalidad de estos valores se corresponden a lecturas por debajo del límite de detección (ruido).

Se procede a remplazar los datos fuera de los límites de detección.

```
remplazo <- function(x){
  x[x < 20] <- 0
  x[x > 16000] <- 16000
  return(x)
}

datos <- datos %>% map_at(.at = 5:ncol(datos), .f = remplazo) %>% as.tibble()
```

Es importante remarcar que esta trasformación es un paso de limpieza, no un preprocesado de datos para mejorar el modelo, por lo tanto, sí puede hacerse sobre todas las observaciones antes de separarlas en conjunto de entrenamiento y test.

Exploración de los datos

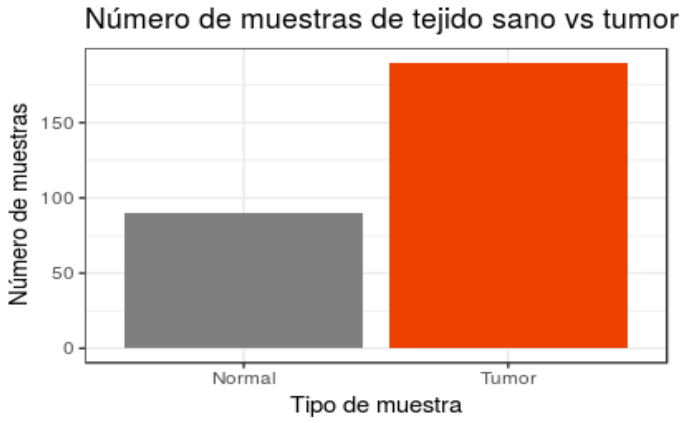
Cantidad y tipo de muestras

La información adjunta al set de datos *GCM_Total.res* indica que se dispone 280 muestras, 190 procedentes de distintos tipos de tumores y 90 de tejido sano (normal). A continuación, se analizan los tumores disponibles así como el número de muestras de cada uno.

```
info_muestras <- datos %>% select(id_muestra, tipo_muestra, tipo_tumor,
                                  subtipo_tumor)
# Muestras normales y tumorales
info_muestras %>%
  group_by(tipo_muestra) %>%
  count()
```

```
## # A tibble: 2 x 2
## # Groups:   tipo_muestra [2]
##   tipo_muestra     n
##   <chr>        <int>
## 1 Normal          90
## 2 Tumor          190
```

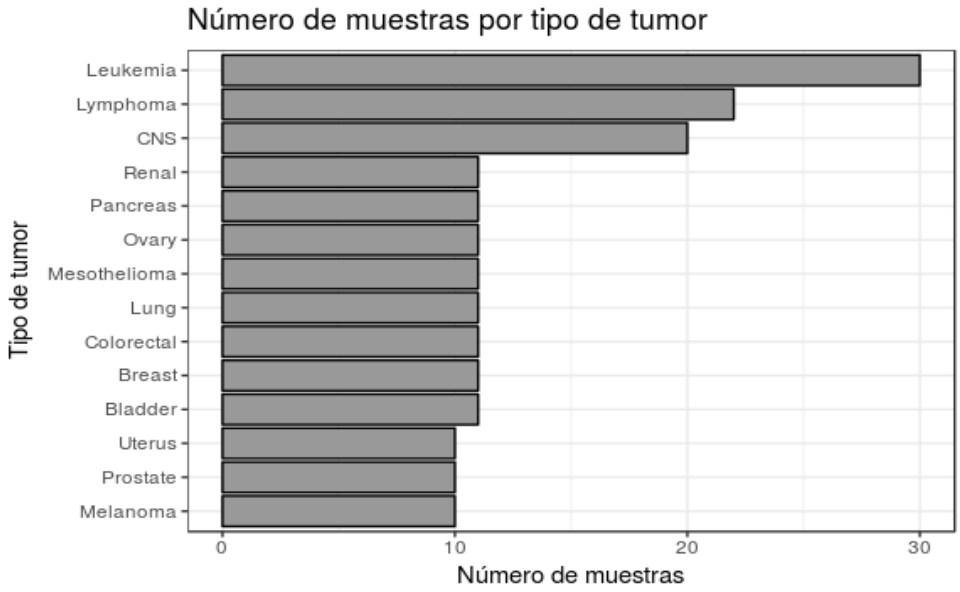
```
info_muestras %>%
  group_by(tipo_muestra) %>%
  count() %>%
  ggplot(aes(x = tipo_muestra, y = n, fill = tipo_muestra)) +
    geom_col() +
    scale_fill_manual(values = c("gray50", "orangered2")) +
    theme_bw() +
    labs(x = "Tipo de muestra", y = "Número de muestras",
         title = "Número de muestras de tejido sano vs tumoral") +
    theme(legend.position = "none")
```



```r
# Tipos de tumores
info_muestras %>%
  filter(tipo_muestra == "Tumor") %>%
  group_by(tipo_tumor) %>%
  count()
```

```
## # A tibble: 14 x 2
## # Groups:   tipo_tumor [14]
##    tipo_tumor       n
##    <chr>        <int>
##  1 Bladder         11
##  2 Breast          11
##  3 CNS             20
##  4 Colorectal      11
##  5 Leukemia        30
##  6 Lung            11
##  7 Lymphoma        22
##  8 Melanoma        10
##  9 Mesothelioma    11
## 10 Ovary           11
## 11 Pancreas        11
## 12 Prostate        10
## 13 Renal           11
## 14 Uterus          10
```

```r
info_muestras %>%
  filter(tipo_muestra == "Tumor") %>%
  group_by(tipo_tumor) %>%
  count() %>%
  ggplot(aes(x = reorder(tipo_tumor, n), y = n)) +
    geom_col(fill = "gray60", color = "black") +
    coord_flip() +
    theme_bw() +
    labs(x = "Tipo de tumor", y = "Número de muestras",
         title = "Número de muestras por tipo de tumor") +
    theme(legend.position = "bottom")
```



El número de muestras tumorales duplica al de muestras normales. Dentro de los tumores, los tipos *Leukemia*, *Lymphoma* y *CN* están bastante más representados. Cuando el número de observaciones por clase no es el mismo (clases desbalanceadas), los métodos estadísticos y de *machine learning* pueden verse afectados, por lo que es un aspecto a tener en cuenta en el análisis.

El objetivo de este estudio es poder predecir el tipo de tumor, por lo que se seleccionan los datos correspondientes a los tumores y se descartan las columnas que no son necesarias.

```
datos <- datos %>% filter(tipo_muestra == "Tumor")
datos <- datos %>% select(-tipo_muestra, -subtipo_tumor)
```

Valores ausentes

Se comprueba que para todas las muestras se dispone del valor de expresión de los 16063 genes.

```
na_por_columna <- map_dbl(.x = datos, .f = function(x){sum(is.na(x))})
any(na_por_columna > 0)
```

```
## [1] FALSE
```

El set de datos está completo, no hay valores ausentes.

División de datos

Evaluar la capacidad predictiva de un modelo consiste en comprobar cómo de próximas son sus predicciones a los verdaderos valores de la variable respuesta. Para poder cuantificar de forma correcta este error, se necesita disponer de un conjunto de observaciones, de las que se conozca la variable respuesta, pero que el modelo no haya "visto", es decir, que no hayan participado en su ajuste. Con esta finalidad, se separan los datos disponibles en un conjunto de entrenamiento y un conjunto de test. El tamaño adecuado de las particiones depende en gran medida de la cantidad de datos disponibles y la seguridad que se necesite en la estimación del error, 80%-20% suele dar buenos resultados. El reparto debe hacerse de forma aleatoria o aleatoria-estratificada.

```
library(caret)
# Se crean los índices de las observaciones de entrenamiento (80%)
set.seed(123)
train <- createDataPartition(y = datos$tipo_tumor, p = 0.8, list = FALSE, times= 1)
datos_train <- datos[train, ]
datos_test  <- datos[-train, ]


# Se eliminan los dataframe datos y datos_long para no ocupar tanta memoria
rm(list = c("datos", "datos_long"))
```

Es importante verificar que la distribución de la variable respuesta es similar en el conjunto de entrenamiento y en el de test. Por defecto, la función createDataPartition() garantiza una distribución aproximada (reparto estratificado).

```
distribucion_train <- prop.table(table(datos_train$tipo_tumor)) %>% round(3)
distribucion_test  <- prop.table(table(datos_test$tipo_tumor)) %>% round(3)
data.frame(train = distribucion_train, test = distribucion_test )
```

```
##      train.Var1 train.Freq    test.Var1 test.Freq
## 1       Bladder      0.058      Bladder     0.056
## 2        Breast      0.058       Breast     0.056
## 3           CNS      0.104          CNS     0.111
## 4    Colorectal      0.058   Colorectal     0.056
## 5      Leukemia      0.156     Leukemia     0.167
## 6          Lung      0.058         Lung     0.056
## 7      Lymphoma      0.117     Lymphoma     0.111
## 8      Melanoma      0.052     Melanoma     0.056
## 9  Mesothelioma      0.058 Mesothelioma     0.056
## 10       Ovary      0.058        Ovary     0.056
## 11     Pancreas      0.058     Pancreas     0.056
## 12     Prostate      0.052     Prostate     0.056
## 13        Renal      0.058        Renal     0.056
## 14       Uterus      0.052       Uterus     0.056
```

Este tipo de reparto estratificado asegura que el conjunto de entrenamiento y el de test sean similares en cuanto a la variable respuesta, sin embargo, no garantiza que ocurra lo mismo con los predictores. Es importante tenerlo en cuenta sobre todo cuando los predictores son cualitativos Machine Learning con R.

El tipo de tumor más frecuente es *Leukemia* (15.6%), este es el porcentaje aproximado de aciertos que se espera si siempre se predice *tipo_tumor = "Leukemia"*. Por lo tanto, este es el porcentaje de aciertos (*accuracy*) que deben ser capaces de superar los modelos predictivos para considerarse mínimamente útiles.

```
# Aciertos si se emplea la clase mayoritaria como predictor
mean(datos_train$tipo_tumor == "Leukemia")
```

```
## [1] 0.1558442
```

Preprocesado

El preprocesado de datos engloba aquellas transformaciones hechas sobre los datos con la finalidad de que puedan ser aceptados por el algoritmo de *machine learning* o que mejoren sus resultados. Todo preprocesado de datos debe aprenderse de las observaciones de entrenamiento y luego aplicarse al conjunto de entrenamiento y al de test. Esto es muy importante para no violar la condición de que ninguna información procedente de las observaciones de test puede participar o influir en el ajuste del modelo.

Genes con varianza próxima a cero

En la mayoría de análisis discriminantes (diferenciación de grupos), el número de observaciones disponibles es mucho mayor que el número de variables, sin embargo, los estudios de expresión genética suelen caracterizarse por justo lo contrario. Por lo general, se dispone de un número bajo de muestras en comparación a los varios miles de genes disponibles como predictores. Esto dificulta en gran medida la creación de modelos predictivos por dos razones. En primer lugar, algunos algoritmos de *machine learning*, por ejemplo el análisis discriminante lineal (LDA), no puede aplicarse si el número de observaciones es inferior al número de predictores. En segundo lugar, aun cuando todos los genes pueden incorporarse en el modelo (*SVM*), muchos de ellos no aportan más que ruido al modelo, lo que

disminuye su capacidad predictiva cuando se aplica a nuevas observaciones (*overfitting*). A este problema se le conoce como "alta dimensionalidad".

Una forma de reducir este problema consiste en eliminar aquellos genes cuya expresión apenas varía en el conjunto de observaciones, y que, por lo tanto, no aportan información. Con este objetivo, se procede a eliminar aquellos genes cuya expresión máxima no supere 5 veces la expresión mínima ($\frac{max}{min} < 5$) y cuya diferencia absoluta entre máximo y mínimo no supere 500 unidades ($max - min < 500$).

```
filtrado_varianza <- function(x){
  # Esta función devuelve TRUE para todas las columnas no numéricas y, en el caso
  # de las numéricas, aquellas que superen el las condiciones de varianza mínima.
  if(is.numeric(x)){
    maximo <- max(x)
    minimo <- min(x)
    ratio <- maximo / minimo
    rango <- maximo - minimo
    return(ratio >= 5 & rango >= 500)
  }else{
    return(TRUE)
  }
}
# Se identifican las columnas que cumplen la condición
genes_varianza <- map_lgl(.x = datos_train, .f = filtrado_varianza)

# Número de columnas (genes) excluidos
sum(genes_varianza == FALSE)
```

```
## [1] 4730
```

```
datos_train <- datos_train[, genes_varianza]
```

Aplicando el filtro de varianza mínima se excluyen 4730 genes. Estos mismos genes identificados en el conjunto de entrenamiento, tienen que ser excluidos también del conjunto de test.

```
datos_test <- datos_test[, genes_varianza]
```

Si bien la eliminación de predictores no informativos podría considerarse un paso propio del proceso de *selección de predictores*, dado que consiste en un filtrado por varianza, tiene que realizarse antes de estandarizar los datos, ya que después, todos los predictores tienen varianza 1.

Estandarización

Cuando los predictores son numéricos, la escala en la que se miden, así como la magnitud de su varianza, pueden influir en gran medida en el modelo. Muchos algoritmos de *machine learning* (*SVM*, redes neuronales, *lasso*...) son sensibles a esto, de forma que, si no se igualan de alguna forma los predictores, aquellos que se midan en una escala mayor o que tengan más varianza, dominarán el modelo aunque no sean los que más relación tienen con la variable respuesta. Medidas de distancia y escalado de variables, Estandarización y escalado.

Se procede a normalizar la expresión de cada gen (columna) para que tengan media 0 y varianza 1.

```
estandarizador <- preProcess(x = datos_train, method = c("center", "scale"))
datos_train    <- predict(object = estandarizador, newdata = datos_train)
datos_test     <- predict(object = estandarizador, newdata = datos_test)
```

Selección de genes y reducción de dimensionalidad

A pesar de haber filtrado aquellos genes con poca varianza, sigue habiendo varios miles como predictores. Es necesario aplicar una estrategia que permita identificar el subconjunto de ellos que están realmente relacionados con la variable respuesta, es decir, genes que se expresan de forma diferente en una clase de tumor respecto al resto de clases. Este problema a sido foco de investigación en el ámbito de la bioinformática durante años, lo que ha dado lugar a una amplia variedad de métodos conocidos como *feature selection*, cuyo objetivo es reducir el número de predictores para mejorar los modelos.

Métodos wrapper

Los métodos wrapper evalúan múltiples modelos, generados mediante la incorporación o eliminación de predictores, con la finalidad de identificar la combinación óptima que consigue maximizar la capacidad del modelo. Pueden entenderse como algoritmos de búsqueda que tratan a los predictores disponibles como valores de entrada y utilizan una métrica del modelo, por ejemplo, su error de predicción, como objetivo de la optimización.

Métodos de filtrado

Los métodos basados en filtrado evalúan la relevancia de los predictores fuera del modelo para, posteriormente, incluir únicamente aquellos que pasan un determinado criterio. Se trata por lo tanto de analizar la relación que tiene cada predictor con la variable respuesta. Por ejemplo, en

problemas de clasificación con predictores continuos, se puede aplicar un ANOVA a cada predictor para identificar aquellos que varían dependiendo de la variable respuesta. Finalmente, se incorporan al modelo aquellos predictores con un *p-value* inferior a un determinado límite o los *n* mejores.

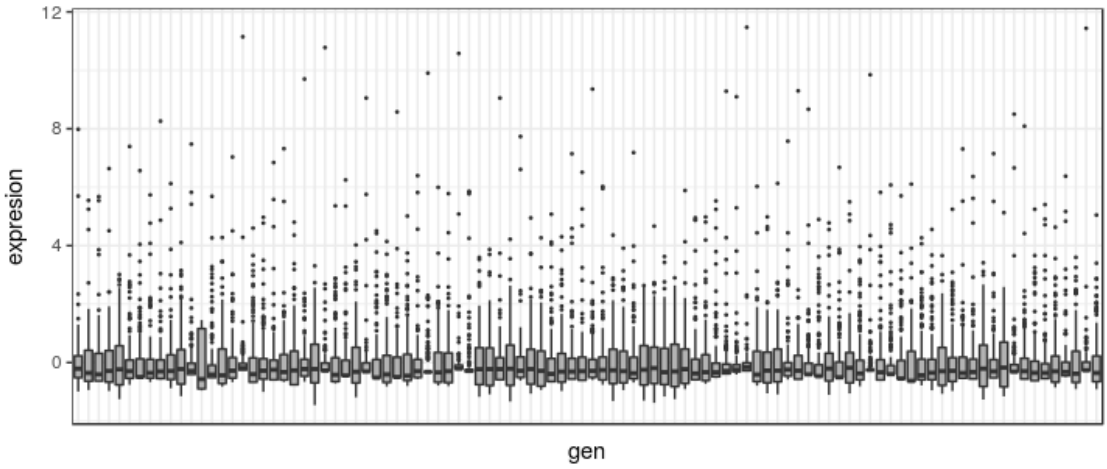
Ambas estrategias, *wrapper* y filtrado, tienen ventajas y desventajas. Los métodos de filtrado son computacionalmente más rápidos, por lo que suelen ser la opción factible cuando hay cientos o miles de predictores, sin embargo, el criterio de selección no está directamente relacionada con la efectividad del modelo. Además, en la mayoría de casos, los métodos de filtrado evalúan cada predictor de forma individual, por lo que no contemplan interacciones y pueden incorporar predictores redundantes (correlacionados). Los métodos *wrapper*, además de ser computacionalmente más costosos, para evitar *overfitting*, necesitan recurrir a validación cruzada o *bootstrapping*, por lo que requieren un número alto de observaciones. A pesar de ello, si se cumplen las condiciones, suelen conseguir una mejor selección.

En este caso, al existir varios miles de posibles predictores, se recurre a métodos de filtrado.

Anova p-value

El contraste de hipótesis ANOVA compara la media de una variable continua entre dos o más grupos. Para este estudio, la idea es que el ANOVA permita identificar aquellos genes cuya expresión varía significativamente entre los distintos tipos de tumor. Dos de las condiciones para que este test de hipótesis sea válido son: que la variable respuesta (nivel de expresión génica) se distribuya de forma normal y que tenga varianza constante en todos los grupos.

```
# Representación de la expresión de 100 genes seleccionados de forma aleatoria.
set.seed(123)
datos_train %>% select_at(sample(4:ncol(datos_train), 100)) %>%
  gather(key = "gen", value = "expresion") %>%
  ggplot(aes(x = gen, y = expresion)) +
    geom_boxplot(outlier.size = 0.3, fill = "gray70") +
    theme_bw() +
    theme(axis.text.x = element_blank(),
          axis.ticks.x = element_blank())
```



Un rápido análisis gráfico muestra que la expresión de los genes no se distribuye de forma normal ni con varianza constante. Esto significa que los resultados del ANOVA no son precisos, por lo que no se deben emplear los *p-value* para determinar significancia estadística. Sin embargo, sí pueden resultar útiles como criterio para ordenar los genes de mayor a menor potencial importancia.

Se aplica un análisis ANOVA para cada uno de los genes.

```
custom_anova <- function(x,y){
  anova <- summary(aov(x ~ as.factor(y)))
  return(unlist(anova)["Pr(>F)1"])
}

p_values <- datos_train %>%
            select(-tipo_tumor) %>%
            map_dbl(.f = custom_anova, y = datos_train$tipo_tumor) %>%
            sort()
p_values %>% head(10)
```

```
##                    L20688_at     AFFX-HSAC07/X00351_5_at
##                 1.884272e-55                1.157357e-53
##   AFFX-HSAC07/X00351_5_at-2     AFFX-HSAC07/X00351_M_at
##                 1.157357e-53                1.799200e-44
##   AFFX-HSAC07/X00351_M_at-2                 X03689_s_at
##                 1.799200e-44                1.828046e-39
##                    Z50781_at   AFFX-HUMGAPDH/M33197_5_at
##                 5.318899e-39                8.708696e-39
## AFFX-HUMGAPDH/M33197_5_at-2              RC_AA280630_at
##                 8.708696e-39                5.122258e-38
```

Para evitar que la selección de genes esté excesivamente influenciada por los datos de entrenamiento, y así minimizar el riesgo de *overfitting*, se implementa un proceso de *bootstraping*. El algoritmo seguido es el siguiente:

---

1. Para cada iteración de *bootstraping*:
   1.1 Se genera una nueva pseudo-muestra por muestreo repetido con reposición, del mismo tamaño que la muestra original.
   1.2 Se calcula el *p-value* asociado a cada gen mediante un ANOVA.
2. Se calcula el *p-value* promedio de cada gen.
3. Se seleccionan los top *n* genes con menor *p-value* promedio.

```
# VERSIÓN NO PARALELIZADA
# ==============================================================================
# Se emplea un número de resampling bajo para que no tarde demasiado. Para valores
# más elevados emplear la versión paralelizada que se describe más adelante.

n_boot <- 3
resultados_anova <- vector(mode = "list", length = n_boot)

# Semillas para que los muestreos sean reproducibles
set.seed(123)
seeds = sample.int(1000, size = n_boot)

# Función ANOVA
custom_anova <- function(x,y){
  anova <- summary(aov(x ~ as.factor(y)))
  return(unlist(anova)["Pr(>F)1"])
}

for (i in 1:n_boot){
  # Se crea una muestra bootstrapping
  set.seed(seeds[i])
  indices <- sample(1:nrow(datos_train), size = nrow(datos_train), replace = TRUE)
  pseudo_muestra <- datos_train[indices, ]

  # Se calculan los p-values con la nueva muestra
  resultados_anova[[i]] <- pseudo_muestra %>%
                           select(-tipo_tumor) %>%
                           map_dbl(.f = custom_anova, y= pseudo_muestra$tipo_tumor)
}
```

```
# Los resultados almacenados en forma de lista se convierten en dataframe
names(resultados_anova) <-  paste("resample", 1:n_boot, sep = "_")
resultados_anova <- data.frame(resultados_anova)
resultados_anova <- resultados_anova %>% rownames_to_column(var = "gen")
resultados_anova <- resultados_anova %>%
                      mutate(pvalue_medio = rowMeans(resultados_anova[, -1])) %>%
                      arrange(pvalue_medio)
head(resultados_anova)
```

Para agilizar el proceso, es recomendable paralelizar el *loop* externo.

```
#==============================================================================
# VERSIÓN PARALELIZADA DE BOOTSTRAPPING PARA FILTRADO POR ANOVA
#==============================================================================
library(doParallel)
# Se especifica el número de cores a utilizar (esto depende del ordenador empleado)
registerDoParallel(cores = 3)
getDoParWorkers()

# Número de iteraciones bootstrapping
n_boot <- 100

# Semillas para que los muestreos sean reproducibles
set.seed(123)
seeds = sample.int(1000, size = n_boot)

# Función ANOVA
custom_anova <- function(x,y){
  anova <- summary(aov(x ~ as.factor(y)))
  return(unlist(anova)["Pr(>F)1"])
}

# LOOP PARALELIZADO
#==============================================================================
# La función foreach devuelve los resultados de cada iteración en una lista

resultados_anova_pvalue <- foreach(i = 1:n_boot) %dopar% {

  # Se crea una muestra por bootstrapping
  set.seed(seeds[i])
  indices <- sample(1:nrow(datos_train), size = nrow(datos_train), replace = TRUE)
  pseudo_muestra <- datos_train[indices, ]

  # Se calculan los p-values para la nueva muestra
  p_values <- pseudo_muestra %>%
                select(-tipo_tumor) %>%
                map_dbl(.f = custom_anova, y = pseudo_muestra$tipo_tumor)
```

```
  # Se devuelven los p-value
  p_values
}
options(cores = 1)

# Los resultados almacenados en forma de lista se convierten en dataframe
names(resultados_anova_pvalue) <-  paste("resample", 1:n_boot, sep = "_")
resultados_anova_pvalue <- data.frame(resultados_anova_pvalue)
resultados_anova_pvalue <- resultados_anova_pvalue %>% rownames_to_column(var="gen")
resultados_anova_pvalue <- resultados_anova_pvalue %>%
                    mutate(pvalue_medio=rowMeans(resultados_anova_pvalue[, -1])) %>%
                    arrange(pvalue_medio)

# Se guarda en disco el objeto creado para no tener que repetir de nuevo toda la
# computación.
saveRDS(object = resultados_anova_pvalue, file = "resultados_anova_pvalue.rds")
```

```
resultados_anova_pvalue %>% select(1,2,3,4) %>% head()
```

```
##                         gen   resample_1   resample_2   resample_3
## 1                 L20688_at 3.963230e-62 5.664129e-56 4.423574e-51
## 2   AFFX-HSAC07/X00351_5_at 2.687954e-60 2.044244e-47 2.526420e-52
## 3 AFFX-HSAC07/X00351_5_at-2 2.687954e-60 2.044244e-47 2.526420e-52
## 4   AFFX-HSAC07/X00351_M_at 1.060545e-47 4.944489e-40 9.888806e-40
## 5 AFFX-HSAC07/X00351_M_at-2 1.060545e-47 4.944489e-40 9.888806e-40
## 6                 X78136_at 4.042074e-39 4.773703e-33 8.561465e-34
```

```
# Se filtran los 100, 50 y 25 genes identificados como más relevantes mediante
# anova
filtrado_anova_pvalue_100 <- resultados_anova_pvalue %>% pull(gen) %>% head(100)
filtrado_anova_pvalue_50  <- resultados_anova_pvalue %>% pull(gen) %>% head(50)
filtrado_anova_pvalue_25 <-  resultados_anova_pvalue %>% pull(gen) %>% head(25)
```

Signal to noise (S2N)

Otra forma de identificar genes característicos de un tipo de tumor es ordenándolos acorde al valor de estadístico *Signal-to-Noise (S2N)*. Este estadístico se calcula con la siguiente ecuación:

$$S2N = \frac{\mu_{\text{grupo i}} - \mu_{\text{resto de grupos}}}{\sigma_{\text{grupo i}} + \sigma_{\text{resto de grupos}}}$$

Cuanto mayor es la diferencia entre la expresión promedio de un gen en un grupo respecto a los demás, mayor es el valor absoluto de *S2N*. Puede considerarse que, para un determinado grupo, los genes con un valor alto de *S2N* son buenos representantes.

El algoritmo seguido para calcular los valores *S2N* es:

Para cada grupo *i*:

- Se separan los valores del grupo *i* y los del resto de grupos en dos *dataframe* distintos.
- Se calcula la media y la desviación típica de cada gen en el grupo *i*.
- Se calcula la media y la desviación típica de cada gen en resto de grupos (de forma conjunta).
- Empleando las medias y desviaciones típicas calculadas en los dos puntos anteriores, se calcula el estadístico *Signal-to-Noise* de cada gen para el grupo *i*.

```
# Se identifica el nombre de los distintos grupos (tipos de tumor)
grupos <- unique(datos_train$tipo_tumor)

# Se crea una lista donde almacenar los resultados para cada grupo
s2n_por_grupo <- vector(mode = "list", length = length(grupos))
names(s2n_por_grupo) <- grupos

# Se calcula el valor S2N de cada gen en cada grupo
for (grupo in grupos){

  # Media y desviación de cada gen en el grupo i
  datos_grupo <- datos_train %>% filter(tipo_tumor == grupo) %>% select(-c(1:3))
  medias_grupo <- map_dbl(datos_grupo, .f = mean)
  sd_grupo <- map_dbl(datos_grupo, .f = sd)
```

```
  # Media y desviación de cada gen en el resto de grupos
  datos_otros <- datos_train %>% filter(tipo_tumor != grupo) %>% select(-c(1:3))
  medias_otros <- map_dbl(datos_otros, .f = mean)
  sd_otros <- map_dbl(datos_otros, .f = sd)

  # Calculo S2N
  s2n <- (medias_grupo - medias_otros)/(sd_grupo + sd_otros)
  s2n_por_grupo[[grupo]] <- s2n
}
```

Como resultado de este algoritmo, se ha obtenido el valor del estadístico *Signal-to-Noise* para cada uno de los genes, en cada uno de los tipos de tumor. Los resultados se han almacenado en una lista. A continuación, se seleccionan los 10 genes con mayor valor absoluto en cada uno de los grupos.

```
extraer_top_genes <- function(x, n=10, abs=TRUE){
  if (abs == TRUE) {
  x <- abs(x)
  x <- sort(x)
  x <- x[1:n]
  return(names(x))
  }else{
  x <- sort(x)
  x <- x[1:n]
  return(names(x))
  }
}

s2n_por_grupo <- s2n_por_grupo %>% map(.f = extraer_top_genes)
```

Si se cumple que el estadístico *signal-to-noise* es capaz de identificar en cada grupo genes cuya expresión es particularmente alta o baja en comparación al resto de grupos (genes representativos del tipo de tumor), cabe esperar que, los genes con mayor valor absoluto *signal-to-noise*, sean distintos en cada tipo de tumor. Si esto es cierto, la intersección de los top 10 genes de los 14 grupos, debería contener aproximadamente 140 genes.

```
genes_seleccionados_s2n <- unique(unlist(s2n_por_grupo))
length(genes_seleccionados_s2n)
```

```
## [1] 140
```

Al igual, que en el filtrado por ANOVA, para evitar que la selección este excesivamente influenciada por la muestra de entrenamiento, es conveniente recurrir a un proceso de *resampling* y agregar los resultados. Esta vez, como método de agregación se emplea la media.

```
#================================================================================
# VERSIÓN PARALELIZADA DE BOOTSTRAPPING PARA FILTRADO POR SIGNAL TO NOISE
#================================================================================
# Warning: Este cálculo puede tardar varias horas.

library(doParallel)
# Se especifica el número de cores a utilizar (esto depende del ordenador)
registerDoParallel(cores = 3)

# Número de iteraciones bootstrapping
n_boot <- 100

# Semillas para que los muestreos sean reproducibles
set.seed(123)
seeds = sample.int(1000, size = n_boot)

# LOOP PARALELIZADO
#================================================================================
resultados_s2n <- foreach(i = 1:n_boot) %dopar% {

  # Se crea una nueva muestra por bootstrapping
  set.seed(seeds[i])
  indices <- sample(1:nrow(datos_train),
                    size = nrow(datos_train),
                    replace = TRUE)
  pseudo_muestra <- datos_train[indices, ]

  # Se identifica el nombre de los distintos grupos (tipos de tumor)
  grupos <- unique(pseudo_muestra$tipo_tumor)

  # Se crea una lista donde almacenar los resultados para cada grupo
  s2n_por_grupo <- vector(mode = "list", length = length(grupos))
  names(s2n_por_grupo) <- grupos

  # Se calcula el valor S2N de cada gen en cada grupo
  for (grupo in grupos){
    # Media y desviación de cada gen en el grupo i
    datos_grupo  <- pseudo_muestra %>% filter(tipo_tumor == grupo) %>% select(-c(1:3))
    medias_grupo <- map_dbl(datos_grupo, .f = mean)
    sd_grupo     <- map_dbl(datos_grupo, .f = sd)

    # Media y desviación de cada gen en el resto de grupos
    datos_otros  <- pseudo_muestra %>% filter(tipo_tumor != grupo) %>% select(-c(1:3))
    medias_otros <- map_dbl(datos_otros, .f = mean)
    sd_otros     <- map_dbl(datos_otros, .f = sd)

    # Calculo S2N
    s2n <- (medias_grupo - medias_otros)/(sd_grupo + sd_otros)
    s2n_por_grupo[[grupo]] <- s2n
```

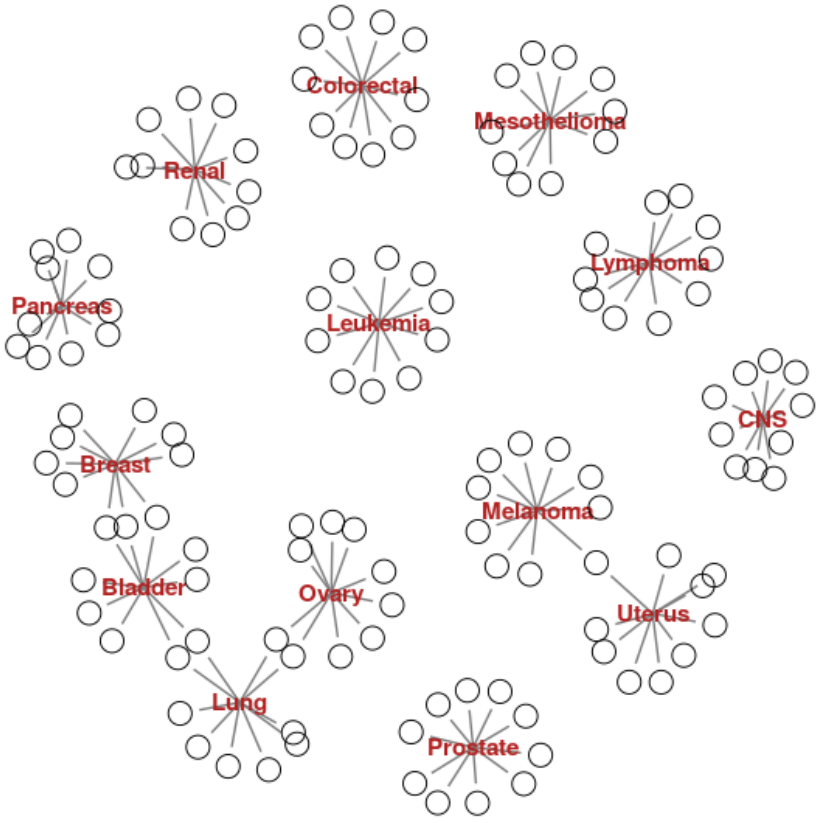
```
  }

  s2n_por_grupo

}
options(cores = 1)

names(resultados_s2n) <- paste("resample", 1:n_boot, sep = "_")

# Se guarda en disco el objeto creado
saveRDS(object = resultados_s2n, file = "resultados_s2n.rds")
```

En cada elemento de la lista resultados_s2n se ha almacenado el resultado de una repetición *bootstrapping*, que a su vez, es otra lista con los valores *S2N* de cada gen en cada grupo. Para obtener un único listado final por tipo de tumor, se tienen que agregar los valores obtenidos en las diferentes repeticiones.

```
# Se crea un dataframe con todos los resultados y se agrupa por tipo de tumor
resultados_s2n_grouped <- resultados_s2n %>%
  unlist() %>%
  as.data.frame() %>%
  rownames_to_column(var = "id") %>%
  separate(col = id, sep = "[.]",
           remove = TRUE,
           into = c("resample", "tipo_tumor", "gen")) %>%
  rename(s2n = ".") %>%
  group_by(tipo_tumor) %>%
  nest()

# Para cada tipo de tumor se calcula el s2n medio de los genes y se devuelven
# los 10 genes con mayor S2N absoluto
extraer_top_genes <- function(df, n=10){
  df <- df %>% spread(key = "resample", value = s2n)
  df <- df %>% mutate(s2n_medio = abs(rowMeans(df[, -1])))
  top_genes <- df %>% arrange(desc(s2n_medio)) %>% pull(gen) %>% head(n)
  return(as.character(top_genes))
}

resultados_s2n_grouped <- resultados_s2n_grouped %>%
                          mutate(genes = map(.x = data, .f = extraer_top_genes))
resultados_s2n_grouped %>% head()
saveRDS(object = resultados_s2n_grouped, file = "resultados_s2n_grouped.rds")
```

El siguiente gráfico muestra los grupos de genes característicos de cada tipo de tumor.

```
library(igraph)
library(ggnetwork)

datos_graph <- resultados_s2n_grouped %>% select(genes, tipo_tumor) %>% unnest()

# Se convierte el dataframe en un objeto igraph
datos_graph <- graph.data.frame(d = datos_graph, directed = TRUE)
# Se objeto igraph en un objeto tipo ggnetwork
datos_graph_tidy <- ggnetwork(datos_graph)
# Se convierte es un dataframe estandar
datos_graph_tidy <- datos_graph_tidy %>% map_df(as.vector)
datos_graph_tidy <- distinct(datos_graph_tidy)

# Se separan los nodos que representan los tipos de tumores de los nodos que
# representan genes

datos_graph_tidy_tumores <- datos_graph_tidy %>%
  filter(vertex.names %in% c(unique(datos_train$tipo_tumor)) &
         x == xend)

datos_graph_tidy_genes <- datos_graph_tidy %>%
  filter(!vertex.names %in% c(unique(datos_train$tipo_tumor)))

# Se crea el gráfico
ggplot(datos_graph_tidy, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(color = "grey50") +
  geom_nodetext(data = datos_graph_tidy_tumores, aes(label = vertex.names),
                size = 4, color = "firebrick", fontface = "bold") +
  geom_nodes(data = datos_graph_tidy_genes, size = 5, shape = 1) +
  #geom_nodetext(data = datos_graph_tidy_genes, aes(label = vertex.names),
  #              size = 2) +
  theme_blank()
```



La representación en forma de *network* permite identificar que algunos de los genes seleccionados son comunes entre varios tipos de tumor. Estos no son por lo tanto buenos candidatos para diferencias los grupos.

Finalmente, se identifica la intersección entre los genes seleccionados para cada tipo de tumor (*10x14=140*), y se eliminan aquellos que son comunes para varios tumores (aparecen más de dos veces).

```
genes_repetidos <-  resultados_s2n_grouped %>%
                    pull(genes) %>%
                    unlist() %>%
                    table() %>%
                    as.data.frame() %>%
                    filter(Freq > 1) %>%
                    pull(".") %>% as.character()
filtrado_s2n_140 <- resultados_s2n_grouped %>%
                    pull(genes) %>% unlist()
filtrado_s2n_140 <- filtrado_s2n_140[!(filtrado_s2n_140 %in% genes_repetidos)]
saveRDS(object = filtrado_s2n_140, file = "filtrado_s2n_140.rds")
```

El mismo proceso se repite pero seleccionando únicamente los top 5 genes por grupo (*5x14 = 70*).

Comparación de filtrados

Se estudia cuantos genes en común se han seleccionado con cada uno de los métodos.

```
length(intersect(filtrado_anova_pvalue_100, filtrado_s2n_140))
```

```
## [1] 15
```

```
length(intersect(filtrado_anova_pvalue_50, filtrado_s2n_70))
```

```
## [1] 9
```

La selección de genes resultante con ambos métodos es muy distinta.

Reducción de dimensionalidad

Los métodos de reducción de dimensionalidad permiten simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conservan su información. Supóngase que existe una muestra con n individuos cada uno con p variables (X_1, X_2, ..., X_p), es decir, el espacio muestral tiene p dimensiones. El objetivo de estos métodos es encontrar un número de factores subyacentes ($z < p$) que expliquen aproximadamente lo mismo que las p variables originales. Donde antes se necesitaban p valores para caracterizar a cada individuo, ahora bastan z valores. Dos de las técnicas más utilizadas son: PCA y t-SNE.

PCA

Se aplica un *PCA* a los niveles de expresión y se conservan las componentes principales hasta alcanzar un 95% de varianza explicada.

```
transformacion_pca <- preProcess(x = datos_train, method = "pca", thresh = 0.95)
transformacion_pca
```

```
## Created from 154 samples and 11335 variables
## Pre-processing:
##   - centered (11334)
##   - ignored (1)
##   - principal component signal extraction (11334)
##   - scaled (11334)
##
## PCA needed 78 components to capture 95 percent of the variance
```

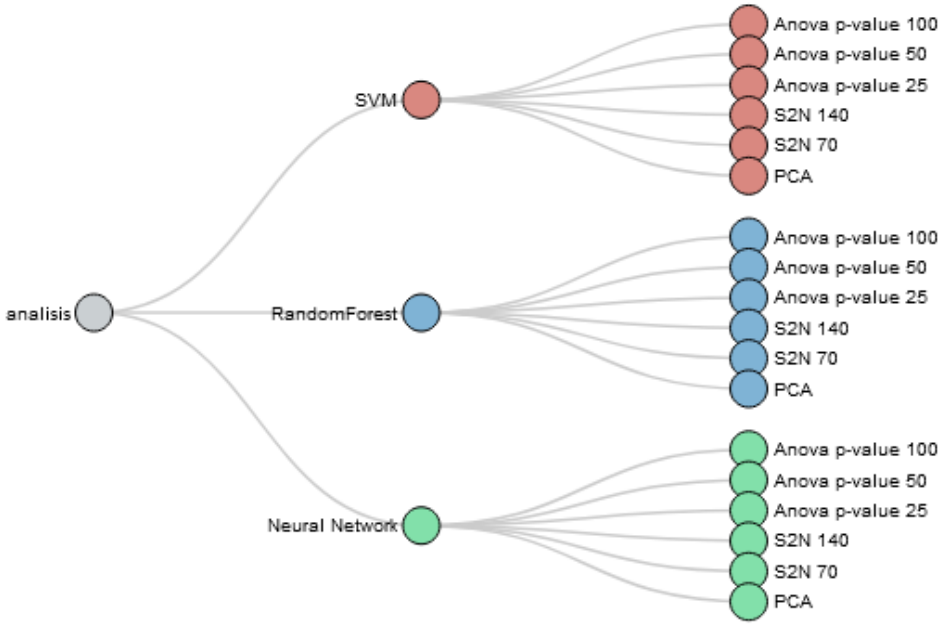
```
datos_train_pca    <- predict(object = transformacion_pca, newdata = datos_train)
datos_test_pca     <- predict(object = transformacion_pca, newdata = datos_test)
```

Modelos

En los siguientes apartados se entrenan diferentes modelos de *machine learning* con el objetivo de compararlos e identificar el que mejor resultado obtiene clasificando los tumores. Además, se comparan los diferentes filtrados de genes. Los modelos se entrenan, optimizan y comparan empleando las funcionalidades que ofrece el paquete caret. Para más detalle sobre su funcionamiento, consultar Machine learning con R y caret.

Diagrama de los modelos ajustados y los genes empleados

```
library(collapsibleTree)
analisis <- data.frame(
  modelo   = rep(c("SVM", "RandomForest", "Neural Network"), each = 6),
  filtrado = rep(c("Anova p-value 100", "Anova p-value 50", "Anova p-value 25",
                   "S2N 140", "S2N 70", "PCA"), times = 3)
)

collapsibleTree(df = analisis,
                hierarchy = c("modelo", "filtrado"),
                collapsed = FALSE,
                zoomable = FALSE,
                fill = c("#cacfd2", "#d98880", "#7fb3d5", "#82e0aa",
                         rep(c("#d98880", "#7fb3d5", "#82e0aa"), each = 6)))
```



SVM

Información detallada sobre *SVM* en Máquinas de Vector Soporte (Support Vector Machines, SVMs)

El método *svmRadial* de caret emplea la función ksvm() del paquete kernlab. Este algoritmo tiene 2 hiperparámetros:

- sigma: coeficiente del kernel radial.
- C: penalización por violaciones del margen del hiperplano.

Filtrado por ANOVA p-value 100

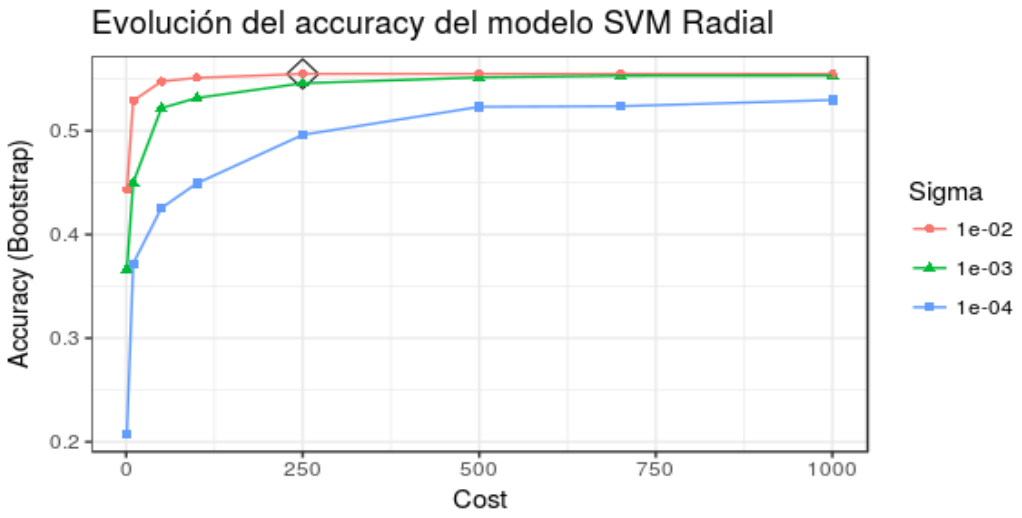
```
# PARALELIZACIÓN DE PROCESO
#================================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#================================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(sigma = c(0.0001, 0.001, 0.01),
                               C = c(1, 10, 50, 100, 250, 500, 700, 1000))
set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#================================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)
```

```
# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
svmrad_pvalue_100 <- train(
                      form = tipo_tumor ~ .,
                      data = datos_train[c("tipo_tumor",
filtrado_anova_pvalue_100)],
                      method = "svmRadial",
                      tuneGrid = hiperparametros,
                      metric = "Accuracy",
                      trControl = control_train
                     )
registerDoMC(cores = 1)
saveRDS(object = svmrad_pvalue_100, file = "svmrad_pvalue_100.rds")

svmrad_pvalue_100
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 154 samples
## 100 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   1e-04     1  0.2078489  0.1233001
##   1e-04    10  0.3720670  0.3224959
##   1e-04    50  0.4255896  0.3786851
##   1e-04   100  0.4493067  0.4032674
##   1e-04   250  0.4961537  0.4521074
##   1e-04   500  0.5231385  0.4799832
##   1e-04   700  0.5236956  0.4803652
##   1e-04  1000  0.5297145  0.4867274
##   1e-03     1  0.3665119  0.3165698
##   1e-03    10  0.4499877  0.4040677
##   1e-03    50  0.5219251  0.4787343
##   1e-03   100  0.5317013  0.4888668
##   1e-03   250  0.5458972  0.5038667
##   1e-03   500  0.5514966  0.5095493
##   1e-03   700  0.5530981  0.5112303
##   1e-03  1000  0.5532590  0.5114521
##   1e-02     1  0.4429619  0.3964551
##   1e-02    10  0.5289988  0.4862089
##   1e-02    50  0.5476313  0.5052872
```

```
##   1e-02   100  0.5511265  0.5090178
##   1e-02   250  0.5548717  0.5129301
##   1e-02   500  0.5548717  0.5129322
##   1e-02   700  0.5548717  0.5129322
##   1e-02  1000  0.5548717  0.5129322
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 250.
```

```r
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(svmrad_pvalue_100, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo SVM Radial") +
  theme_bw()
```



Mejor modelo: *sigma* = 0.01, *C* = 300, *accuracy* = 0.5548717.

Filtrado por ANOVA p-value 50

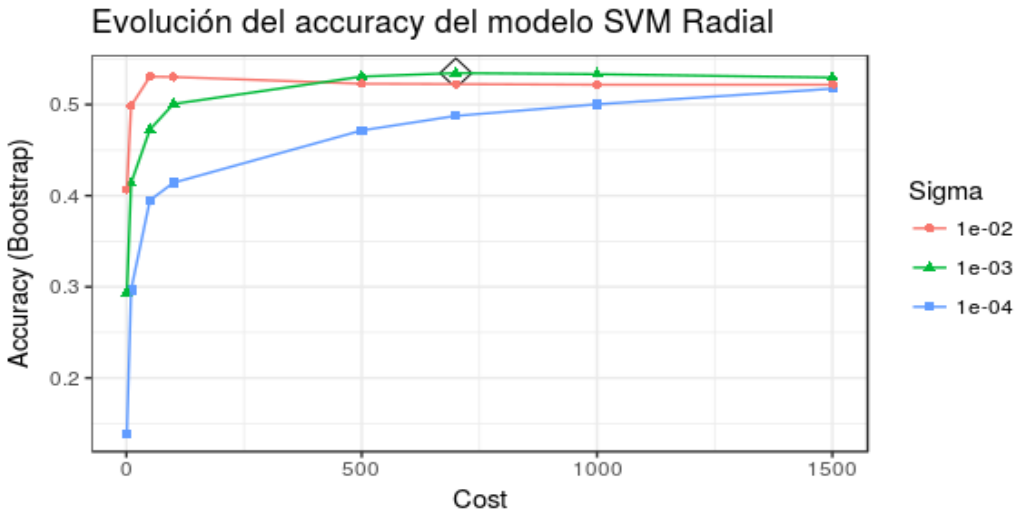
```r
# PARALELIZACIÓN DE PROCESO
#===============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#===============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(sigma = c(0.0001, 0.001, 0.01),
                               C = c(1, 10, 50, 100, 500, 700, 1000, 1500))
```

```
set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)
# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
svmrad_pvalue_50 <- train(
                      form = tipo_tumor ~ .,
                      data = datos_train[c("tipo_tumor",
filtrado_anova_pvalue_50)],
                      method = "svmRadial",
                      tuneGrid = hiperparametros,
                      metric = "Accuracy",
                      trControl = control_train
                    )
registerDoMC(cores = 1)
saveRDS(object = svmrad_pvalue_50, file = "svmrad_pvalue_50.rds")

svmrad_pvalue_50
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 154 samples
##  50 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   1e-04     1  0.1390977  0.02830175
##   1e-04    10  0.2966844  0.23878841
##   1e-04    50  0.3950232  0.34651538
##   1e-04   100  0.4142378  0.36655585
##   1e-04   500  0.4716077  0.42566727
##   1e-04   700  0.4875991  0.44253810
##   1e-04  1000  0.5003152  0.45559975
```

```
##   1e-04  1500  0.5173425  0.47341922
##   1e-03     1  0.2935448  0.23475393
##   1e-03    10  0.4138532  0.36611315
##   1e-03    50  0.4725888  0.42673610
##   1e-03   100  0.5006731  0.45606176
##   1e-03   500  0.5306822  0.48691896
##   1e-03   700  0.5345244  0.49088543
##   1e-03  1000  0.5332763  0.48919440
##   1e-03  1500  0.5296332  0.48517338
##   1e-02     1  0.4067941  0.35857448
##   1e-02    10  0.4985066  0.45357254
##   1e-02    50  0.5306271  0.48681524
##   1e-02   100  0.5301999  0.48589534
##   1e-02   500  0.5227895  0.47757749
##   1e-02   700  0.5224213  0.47721776
##   1e-02  1000  0.5217335  0.47647196
##   1e-02  1500  0.5217335  0.47647196
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.001 and C = 700.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(svmrad_pvalue_50, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo SVM Radial") +
  theme_bw()
```



Mejor modelo: *sigma* = 0.001, *C* = 700, *accuracy* = 0.5345244.

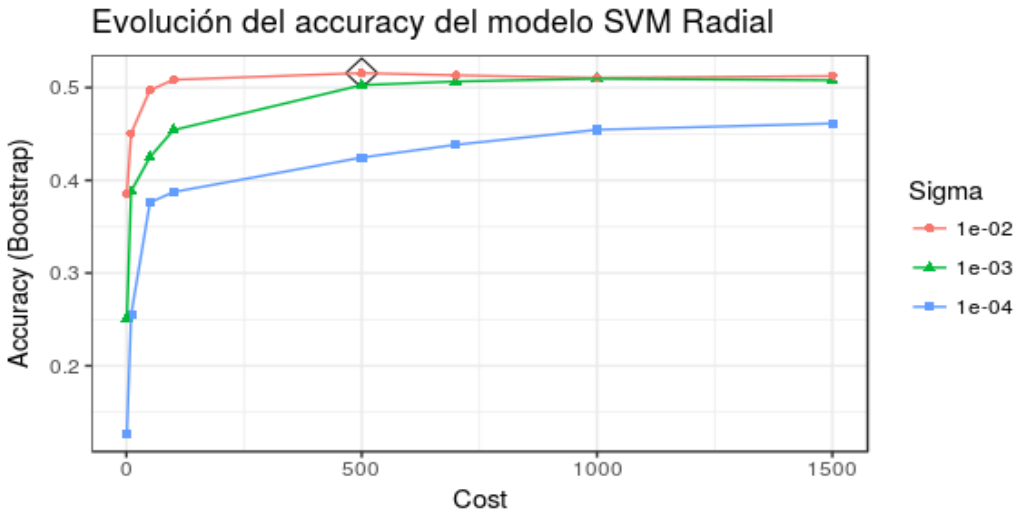
Filtrado por ANOVA p-value 25

```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(sigma = c(0.0001, 0.001, 0.01),
                               C = c(1, 10, 50, 100, 500, 700, 1000, 1500))
set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)
# AJUSTE DEL MODELO
# =============================================================================
set.seed(342)
svmrad_pvalue_25 <- train(
                      form = tipo_tumor ~ .,
                      data = datos_train[c("tipo_tumor",
filtrado_anova_pvalue_25)],
                      method = "svmRadial",
                      tuneGrid = hiperparametros,
                      metric = "Accuracy",
                      trControl = control_train
                    )
registerDoMC(cores = 1)
saveRDS(object = svmrad_pvalue_25, file = "svmrad_pvalue_25.rds")

svmrad_pvalue_25
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 154 samples
##  25 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
```

```
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   1e-04     1  0.1270298  0.01414768
##   1e-04    10  0.2547005  0.18489362
##   1e-04    50  0.3761014  0.32610453
##   1e-04   100  0.3873704  0.33813366
##   1e-04   500  0.4245915  0.37653606
##   1e-04   700  0.4383743  0.39013379
##   1e-04  1000  0.4545405  0.40742349
##   1e-04  1500  0.4612799  0.41416196
##   1e-03     1  0.2514322  0.17992749
##   1e-03    10  0.3880542  0.33881192
##   1e-03    50  0.4249363  0.37685132
##   1e-03   100  0.4541839  0.40698794
##   1e-03   500  0.5026673  0.45690021
##   1e-03   700  0.5065387  0.46080582
##   1e-03  1000  0.5096155  0.46381666
##   1e-03  1500  0.5078675  0.46154168
##   1e-02     1  0.3855600  0.33604254
##   1e-02    10  0.4505680  0.40308300
##   1e-02    50  0.4971511  0.45105583
##   1e-02   100  0.5084018  0.46243160
##   1e-02   500  0.5156146  0.46977048
##   1e-02   700  0.5132201  0.46702454
##   1e-02  1000  0.5104436  0.46394475
##   1e-02  1500  0.5124460  0.46607677
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 500.
```

```r
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(svmrad_pvalue_25, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo SVM Radial") +
  theme_bw()
```



Mejor modelo: *sigma* = 0.01, *C* = 500, *accuracy* = 0.5156146.

Filtrado por S2N 140

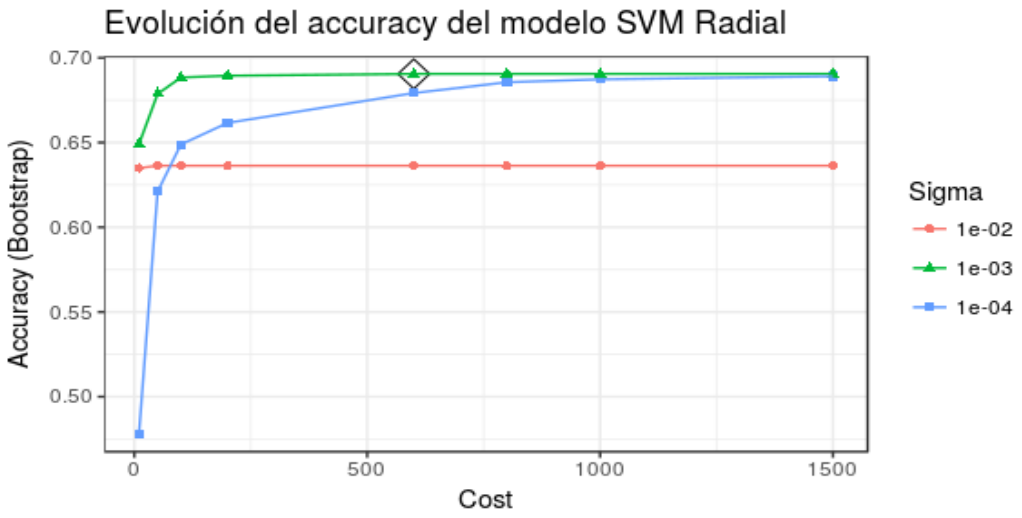
```
# PARALELIZACIÓN DE PROCESO
#=============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#=============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(sigma = c(0.0001, 0.001, 0.01),
                               C = c(10, 50, 100, 200, 600, 800, 1000, 1500))
set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#=============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)
```

```
# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
svmrad_s2n_140 <- train(
                     form = tipo_tumor ~ .,
                     data = datos_train[c("tipo_tumor", filtrado_s2n_140)],
                     method = "svmRadial",
                     tuneGrid = hiperparametros,
                     metric = "Accuracy",
                     trControl = control_train
                     )
registerDoMC(cores = 1)
saveRDS(object = svmrad_s2n_140, file = "svmrad_s2n_140.rds")

svmrad_s2n_140
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 154 samples
## 124 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   1e-04    10  0.4781852  0.4340077
##   1e-04    50  0.6210650  0.5864427
##   1e-04   100  0.6488673  0.6160512
##   1e-04   200  0.6615939  0.6295683
##   1e-04   600  0.6792164  0.6484499
##   1e-04   800  0.6855678  0.6552814
##   1e-04  1000  0.6872621  0.6570890
##   1e-04  1500  0.6890610  0.6590254
##   1e-03    10  0.6490217  0.6163052
##   1e-03    50  0.6786705  0.6479404
##   1e-03   100  0.6883862  0.6584406
##   1e-03   200  0.6894373  0.6595522
##   1e-03   600  0.6904871  0.6607011
##   1e-03   800  0.6904871  0.6607011
##   1e-03  1000  0.6904871  0.6607011
##   1e-03  1500  0.6904871  0.6607011
##   1e-02    10  0.6349287  0.5992857
##   1e-02    50  0.6363235  0.6007940
##   1e-02   100  0.6363235  0.6007940
##   1e-02   200  0.6363235  0.6007940
```

```
##   1e-02   600  0.6363235  0.6007940
##   1e-02   800  0.6363235  0.6007940
##   1e-02  1000  0.6363235  0.6007940
##   1e-02  1500  0.6363235  0.6007940
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.001 and C = 600.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(svmrad_s2n_140, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo SVM Radial") +
  theme_bw()
```



Mejor modelo: *sigma* = 0.001, *C* = 600, *accuracy* = 0.6904871.

Filtrado por S2N 70

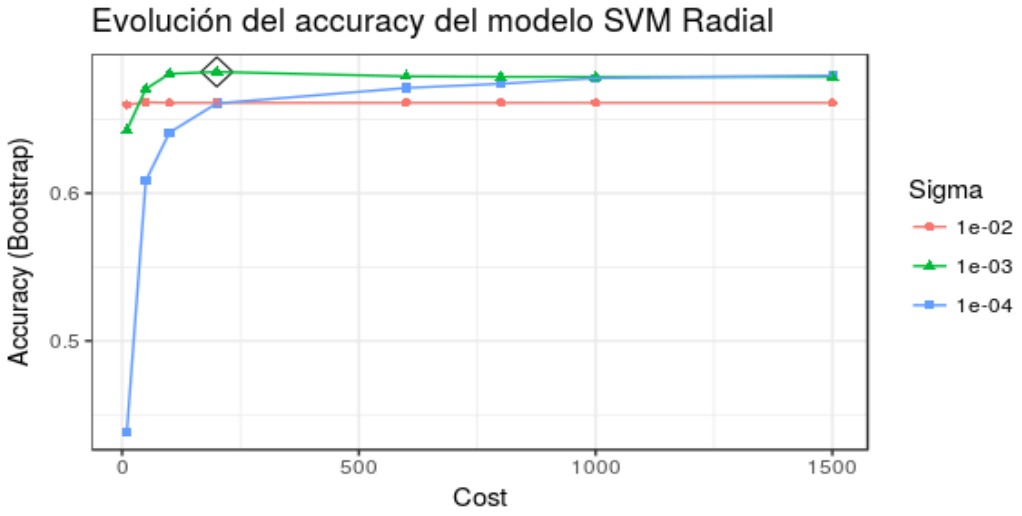
```
# PARALELIZACIÓN DE PROCESO
#===============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#===============================================================================
repeticiones_boot <- 50
# Hiperparámetros
hiperparametros <- expand.grid(sigma = c(0.0001, 0.001, 0.01),
                               C = c(10, 50, 100, 200, 600, 800, 1000, 1500))
```

```
set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
svmrad_s2n_70 <- train(
                      form = tipo_tumor ~ .,
                      data = datos_train[c("tipo_tumor", filtrado_s2n_70)],
                      method = "svmRadial",
                      tuneGrid = hiperparametros,
                      metric = "Accuracy",
                      trControl = control_train
                      )
registerDoMC(cores = 1)
saveRDS(object = svmrad_s2n_70, file = "svmrad_s2n_70.rds")

svmrad_s2n_70
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 154 samples
##  70 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   1e-04    10  0.4385375  0.3898072
##   1e-04    50  0.6084647  0.5726624
##   1e-04   100  0.6410914  0.6075662
##   1e-04   200  0.6609118  0.6287008
##   1e-04   600  0.6713126  0.6395965
##   1e-04   800  0.6741261  0.6426495
##   1e-04  1000  0.6780060  0.6468396
```

```
##   1e-04  1500  0.6796936  0.6486076
##   1e-03    10  0.6427156  0.6093710
##   1e-03    50  0.6706113  0.6389579
##   1e-03   100  0.6809044  0.6500056
##   1e-03   200  0.6821623  0.6513337
##   1e-03   600  0.6791659  0.6480151
##   1e-03   800  0.6788269  0.6476404
##   1e-03  1000  0.6788269  0.6476404
##   1e-03  1500  0.6788269  0.6476404
##   1e-02    10  0.6599166  0.6269438
##   1e-02    50  0.6615932  0.6286637
##   1e-02   100  0.6612484  0.6282862
##   1e-02   200  0.6612484  0.6282862
##   1e-02   600  0.6612484  0.6282862
##   1e-02   800  0.6612484  0.6282862
##   1e-02  1000  0.6612484  0.6282862
##   1e-02  1500  0.6612484  0.6282862
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.001 and C = 200.
```

```r
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(svmrad_s2n_70, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo SVM Radial") +
  theme_bw()
```



Mejor modelo: *sigma* = 0.001 , *C* = 200, *accuracy* = 0.6821623.

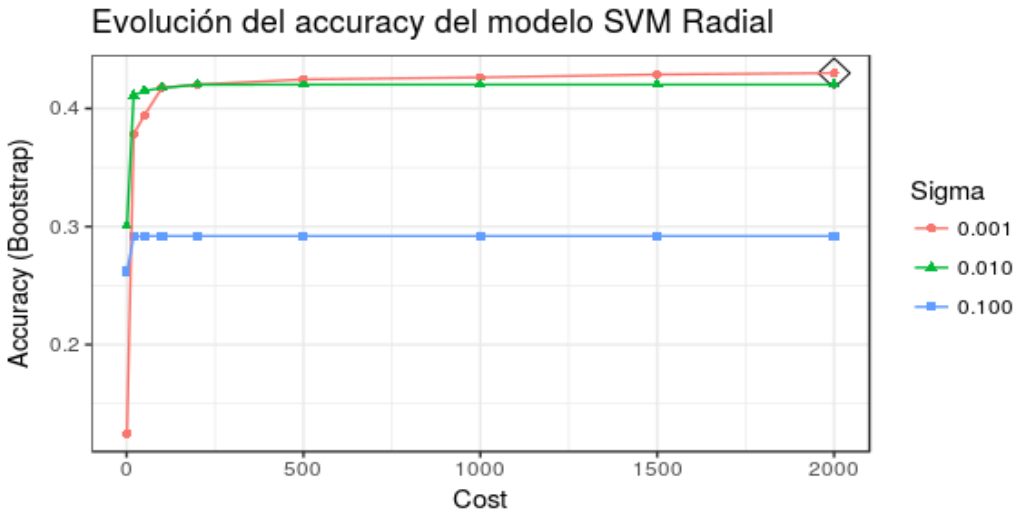
Reducción PCA

```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(sigma = c(0.001, 0.01, 0.1),
                               C = c(1, 20, 50, 100, 200, 500, 1000, 1500, 2000))
set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# =============================================================================
set.seed(342)
svmrad_pca <- train(form = tipo_tumor ~ .,
                    data = datos_train_pca,
                    method = "svmRadial",
                    tuneGrid = hiperparametros,
                    metric = "Accuracy",
                    trControl = control_train)
registerDoMC(cores = 1)
saveRDS(object = svmrad_pca, file = "svmrad_pca.rds")

svmrad_pca
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 154 samples
##  78 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
```

```
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   0.001     1  0.1246485  0.008111059
##   0.001    20  0.3782270  0.325006441
##   0.001    50  0.3940651  0.341365025
##   0.001   100  0.4175254  0.366526890
##   0.001   200  0.4203110  0.369277169
##   0.001   500  0.4246361  0.373355469
##   0.001  1000  0.4263804  0.375210888
##   0.001  1500  0.4288099  0.377766491
##   0.001  2000  0.4300400  0.379116379
##   0.010     1  0.3004183  0.229277714
##   0.010    20  0.4109745  0.347746440
##   0.010    50  0.4149865  0.351727490
##   0.010   100  0.4174088  0.354342853
##   0.010   200  0.4201938  0.357412455
##   0.010   500  0.4202380  0.357386814
##   0.010  1000  0.4202380  0.357386814
##   0.010  1500  0.4202380  0.357386814
##   0.010  2000  0.4202380  0.357386814
##   0.100     1  0.2620953  0.172961535
##   0.100    20  0.2912640  0.205288247
##   0.100    50  0.2919543  0.205984017
##   0.100   100  0.2919543  0.205984017
##   0.100   200  0.2919543  0.205984017
##   0.100   500  0.2919543  0.205984017
##   0.100  1000  0.2919543  0.205984017
##   0.100  1500  0.2919543  0.205984017
##   0.100  2000  0.2919543  0.205984017
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.001 and C = 2000.
```

```r
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(svmrad_pca, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo SVM Radial") +
  theme_bw()
```



Mejor modelo: *sigma* = 0.001, *C* = 2000, *accuracy* = 0.4300400.

RandomForest

Información detallada sobre *random forest* en Árboles de predicción: bagging, random forest, boosting y C5.0

El método *ranger* de caret emplea la función ranger() del paquete ranger. Este algoritmo tiene 3 hiperparámetros:

- mtry: número predictores seleccionados aleatoriamente en cada árbol.
- min.node.size: tamaño mínimo que tiene que tener un nodo para poder ser dividido.
- splitrule: criterio de división.

Filtrado por ANOVA p-value 100

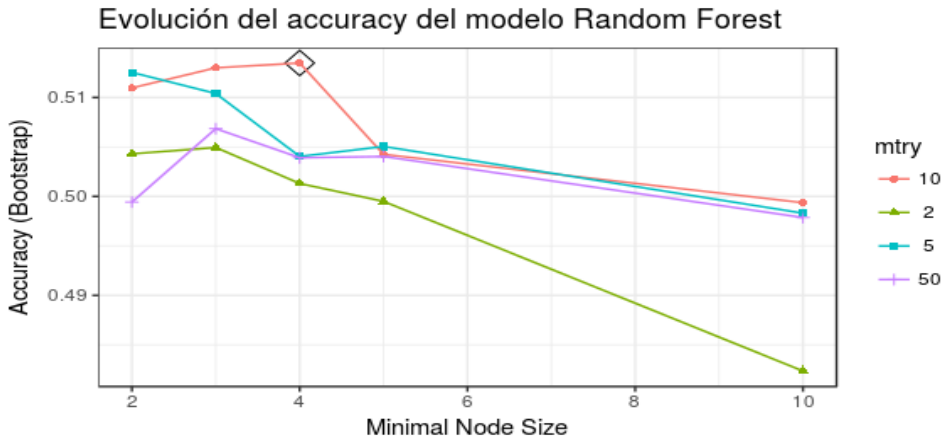
```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(mtry = c(2, 5, 10, 50),
                               min.node.size = c(2, 3, 4, 5, 10),
                               splitrule = "gini")

set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# =============================================================================
```

```
set.seed(342)
rf_pvalue_100 <- train(
                  form = tipo_tumor ~ .,
                  data = datos_train[c("tipo_tumor", filtrado_anova_pvalue_100)],
                  method = "ranger",
                  tuneGrid = hiperparametros,
                  metric = "Accuracy",
                  trControl = control_train,
                  # Número de árboles ajustados
                  num.trees = 500)

saveRDS(object = rf_pvalue_100, file = "rf_pvalue_100.rds")
registerDoMC(cores = 1)

rf_pvalue_100
```

```
## Random Forest
##
## 154 samples
## 100 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  Accuracy   Kappa
##    2     2             0.5043083  0.4580127
##    2     3             0.5049427  0.4587542
##    2     4             0.5013050  0.4548919
##    2     5             0.4994943  0.4530367
##    2    10             0.4823426  0.4340357
##    5     2             0.5125346  0.4670717
##    5     3             0.5104007  0.4648117
##    5     4             0.5040432  0.4577901
##    5     5             0.5050499  0.4590714
##    5    10             0.4983020  0.4516733
##   10     2             0.5109691  0.4652424
##   10     3             0.5130031  0.4677764
##   10     4             0.5134874  0.4680756
##   10     5             0.5042510  0.4580470
##   10    10             0.4993608  0.4528191
##   50     2             0.4994304  0.4531282
##   50     3             0.5068500  0.4609431
##   50     4             0.5039106  0.4576299
##   50     5             0.5040375  0.4578440
##   50    10             0.4978588  0.4511572
```

```
##
## Tuning parameter 'splitrule' was held constant at a value of gini
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 10, splitrule = gini
##  and min.node.size = 4.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(rf_pvalue_100, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo Random Forest") +
  guides(color = guide_legend(title = "mtry"),
         shape = guide_legend(title = "mtry")) +
  theme_bw()
```



Mejor modelo: *mtry* = 10, *splitrule* = gini, *min.node.size* = 4, *accuracy* = 0.5138110.

Filtrado por ANOVA p-value 50

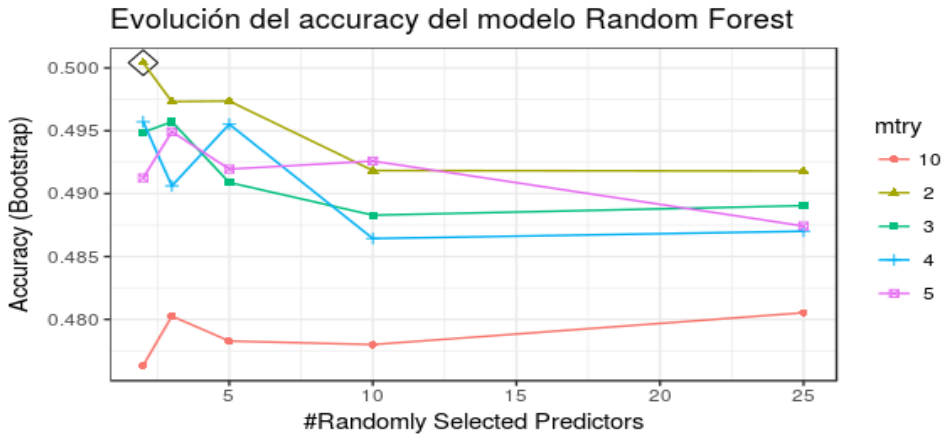
```
# PARALELIZACIÓN DE PROCESO
#===============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#===============================================================================
repeticiones_boot <- 50
# Hiperparámetros
hiperparametros <- expand.grid(mtry = c(2, 3, 5, 10, 25),
                               min.node.size = c(2, 3, 4, 5, 10),
                               splitrule = "gini")
set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
```

```
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
rf_pvalue_50 <- train(
                  form = tipo_tumor ~ .,
                  data = datos_train[c("tipo_tumor", filtrado_anova_pvalue_50)],
                  method = "ranger",
                  tuneGrid = hiperparametros,
                  metric = "Accuracy",
                  trControl = control_train,
                  # Número de árboles ajustados
                  num.trees = 500)

saveRDS(object = rf_pvalue_50, file = "rf_pvalue_50.rds")
registerDoMC(cores = 1)

rf_pvalue_50
```

```
## Random Forest
##
## 154 samples
##  50 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  Accuracy   Kappa
##    2     2             0.5004133  0.4542043
##    2     3             0.4948841  0.4481763
##    2     4             0.4957192  0.4491591
##    2     5             0.4912498  0.4443905
##    2    10             0.4763288  0.4280723
##    3     2             0.4973192  0.4507884
##    3     3             0.4957213  0.4489977
```

```
##     3     4             0.4906199  0.4434662
##     3     5             0.4949264  0.4481590
##     3    10             0.4802537  0.4323296
##     5     2             0.4973605  0.4507795
##     5     3             0.4908822  0.4436949
##     5     4             0.4955231  0.4488022
##     5     5             0.4919533  0.4450656
##     5    10             0.4782734  0.4299805
##    10     2             0.4918353  0.4448368
##    10     3             0.4882872  0.4409017
##    10     4             0.4864286  0.4388859
##    10     5             0.4925907  0.4454656
##    10    10             0.4779956  0.4296955
##    25     2             0.4917994  0.4448925
##    25     3             0.4890461  0.4416856
##    25     4             0.4870100  0.4395091
##    25     5             0.4874216  0.4399493
##    25    10             0.4805246  0.4324839
##
## Tuning parameter 'splitrule' was held constant at a value of gini
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 2, splitrule = gini
##  and min.node.size = 2.
```

```r
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(rf_pvalue_50, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo Random Forest") +
  guides(color = guide_legend(title="mtry"), shape = guide_legend(title ="mtry")) +
  theme_bw()
```



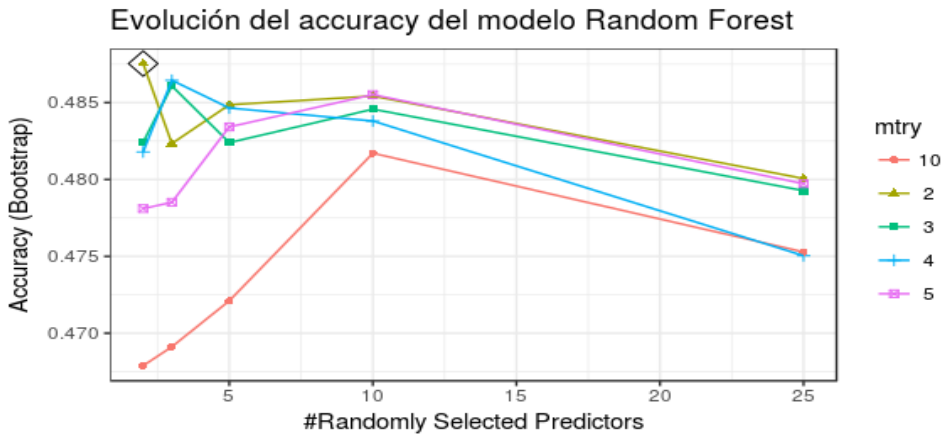
Mejor modelo: *mtry* = 2, *splitrule* = gini, *min.node.size* = 2, *accuracy* = 0.5017468.

Filtrado por ANOVA p-value 25

```
# PARALELIZACIÓN DE PROCESO
#============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(mtry = c(2, 3, 5, 10, 25),
                               min.node.size = c(2, 3, 4, 5, 10),
                               splitrule = "gini")

set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# ============================================================================
set.seed(342)
rf_pvalue_25 <- train(
                  form = tipo_tumor ~ .,
                  data = datos_train[c("tipo_tumor", filtrado_anova_pvalue_25)],
                  method = "ranger",
                  tuneGrid = hiperparametros,
                  metric = "Accuracy",
                  trControl = control_train,
                  # Número de árboles ajustados
                  num.trees = 500)

saveRDS(object = rf_pvalue_25, file = "rf_pvalue_25.rds")
registerDoMC(cores = 1)

rf_pvalue_25
```

```
## Random Forest
##
## 154 samples
##  25 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  Accuracy   Kappa
##    2     2             0.4875376  0.4401626
##    2     3             0.4824233  0.4347869
##    2     4             0.4817937  0.4339623
##    2     5             0.4781071  0.4300189
##    2    10             0.4678878  0.4189576
##    3     2             0.4823391  0.4344177
##    3     3             0.4860837  0.4386411
##    3     4             0.4864514  0.4390062
##    3     5             0.4785148  0.4303742
##    3    10             0.4691193  0.4201802
##    5     2             0.4848605  0.4372919
##    5     3             0.4823998  0.4346132
##    5     4             0.4846470  0.4369560
##    5     5             0.4834203  0.4355383
##    5    10             0.4721101  0.4235134
##   10     2             0.4854259  0.4375376
##   10     3             0.4845721  0.4366514
##   10     4             0.4838030  0.4356810
##   10     5             0.4855180  0.4376095
##   10    10             0.4817062  0.4335955
##   25     2             0.4800582  0.4316212
##   25     3             0.4792709  0.4306269
##   25     4             0.4750390  0.4260252
##   25     5             0.4797227  0.4312433
##   25    10             0.4752853  0.4263989
##
## Tuning parameter 'splitrule' was held constant at a value of gini
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 2, splitrule = gini
##  and min.node.size = 2.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(rf_pvalue_25, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo Random Forest") +
  guides(color = guide_legend(title = "mtry"),
         shape = guide_legend(title = "mtry")) +
  theme_bw()
```



Mejor modelo: *mtry* = 2, *splitrule* = gini, *min.node.size* = 2, *accuracy* = 0.4875376.

Filtrado por S2N 140

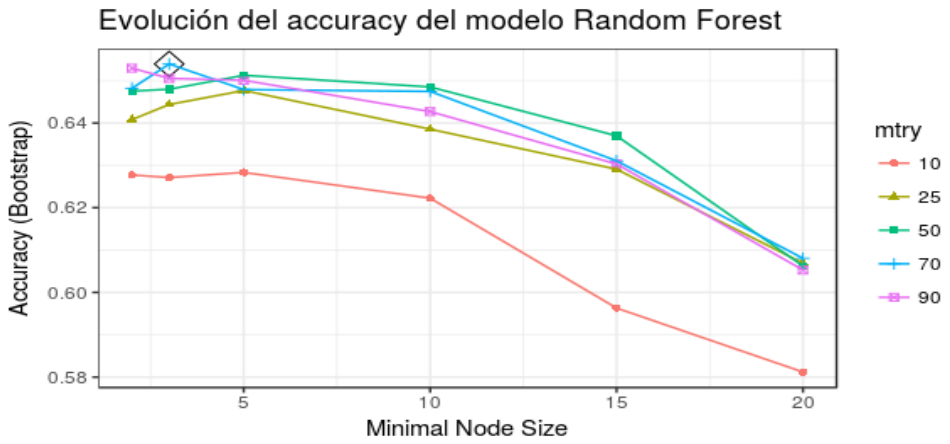
```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(mtry = c(10, 25, 50, 70, 90),
                               min.node.size = c(2, 3, 5, 10, 15, 20),
                               splitrule = "gini")

set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)
```

```
# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
rf_s2n_140 <- train(
                form = tipo_tumor ~ .,
                data = datos_train[c("tipo_tumor", filtrado_s2n_140)],
                method = "ranger",
                tuneGrid = hiperparametros,
                metric = "Accuracy",
                trControl = control_train,
                # Número de árboles ajustados
                num.trees = 500
              )

saveRDS(object = rf_s2n_140, file = "rf_s2n_140.rds")
registerDoMC(cores = 1)

rf_s2n_140
```

```
## Random Forest
##
## 154 samples
## 124 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  Accuracy   Kappa
##   10     2             0.6277318  0.5918506
##   10     3             0.6271157  0.5910643
##   10     5             0.6283339  0.5924796
##   10    10             0.6222340  0.5858138
##   10    15             0.5962925  0.5576043
##   10    20             0.5811859  0.5415062
##   25     2             0.6408063  0.6059969
##   25     3             0.6443675  0.6099375
##   25     5             0.6476676  0.6136094
##   25    10             0.6385296  0.6034048
```

```
##   25    15             0.6290701  0.5932242
##   25    20             0.6068839  0.5695300
##   50     2             0.6475310  0.6135568
##   50     3             0.6479637  0.6139701
##   50     5             0.6512624  0.6174337
##   50    10             0.6485047  0.6143930
##   50    15             0.6369608  0.6020802
##   50    20             0.6062254  0.5687919
##   70     2             0.6481825  0.6143484
##   70     3             0.6539251  0.6204492
##   70     5             0.6478853  0.6138232
##   70    10             0.6474529  0.6133836
##   70    15             0.6310577  0.5957920
##   70    20             0.6080442  0.5708778
##   90     2             0.6529365  0.6196086
##   90     3             0.6505284  0.6167967
##   90     5             0.6500701  0.6163078
##   90    10             0.6426831  0.6082070
##   90    15             0.6303011  0.5950468
##   90    20             0.6052728  0.5680226
##
## Tuning parameter 'splitrule' was held constant at a value of gini
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 70, splitrule = gini
##  and min.node.size = 3.
```

```r
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(rf_s2n_140, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo Random Forest") +
  guides(color = guide_legend(title = "mtry"),
         shape = guide_legend(title = "mtry")) +
  theme_bw()
```



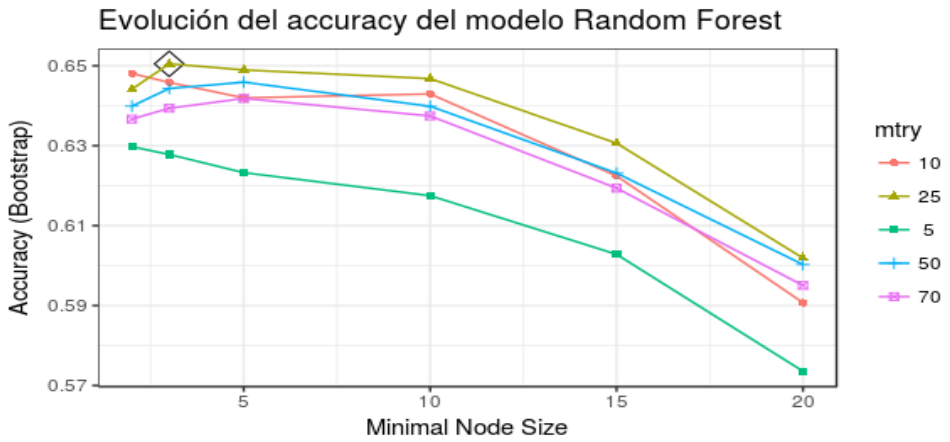
Mejor modelo: *mtry* = 70, *splitrule* = gini, *min.node.size* = 3, *accuracy* = 0.6539251.

Filtrado por S2N 70

```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(mtry = c(5, 10, 25, 50, 70),
                               min.node.size = c(2, 3, 5, 10, 15, 20),
                               splitrule = "gini")

set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# =============================================================================
set.seed(342)
rf_s2n_70 <- train(
                form = tipo_tumor ~ .,
                data = datos_train[c("tipo_tumor", filtrado_s2n_70)],
                method = "ranger",
                tuneGrid = hiperparametros,
                metric = "Accuracy",
                trControl = control_train,
                # Número de árboles ajustados
                num.trees = 500
              )

saveRDS(object = rf_s2n_70, file = "rf_s2n_70.rds")
registerDoMC(cores = 1)

rf_s2n_70
```

```
## Random Forest
##
## 154 samples
##  70 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  Accuracy   Kappa
##    5     2             0.6297421  0.5940515
##    5     3             0.6277989  0.5920068
##    5     5             0.6232470  0.5870947
##    5    10             0.6174872  0.5807316
##    5    15             0.6028104  0.5649050
##    5    20             0.5735735  0.5334133
##   10     2             0.6480453  0.6140921
##   10     3             0.6458197  0.6116533
##   10     5             0.6419567  0.6074243
##   10    10             0.6429442  0.6083867
##   10    15             0.6224156  0.5860989
##   10    20             0.5907217  0.5518363
##   25     2             0.6441308  0.6095050
##   25     3             0.6504726  0.6164269
##   25     5             0.6489504  0.6146812
##   25    10             0.6467766  0.6121925
##   25    15             0.6306140  0.5948419
##   25    20             0.6018785  0.5638122
##   50     2             0.6398970  0.6048228
##   50     3             0.6443156  0.6095326
##   50     5             0.6458920  0.6112227
##   50    10             0.6398517  0.6045603
##   50    15             0.6231348  0.5866322
##   50    20             0.6002573  0.5619946
##   70     2             0.6366616  0.6011182
##   70     3             0.6393682  0.6042619
##   70     5             0.6418298  0.6068039
##   70    10             0.6374341  0.6020411
##   70    15             0.6193718  0.5825284
##   70    20             0.5950341  0.5563674
##
## Tuning parameter 'splitrule' was held constant at a value of gini
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 25, splitrule = gini
##  and min.node.size = 3.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(rf_s2n_70, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo Random Forest") +
  guides(color = guide_legend(title = "mtry"),
         shape = guide_legend(title = "mtry")) +
  theme_bw()
```



Mejor modelo: *mtry* = 25, *splitrule* = gini, *min.node.size* = 3, *accuracy* = 0.6504726.

Reducción PCA

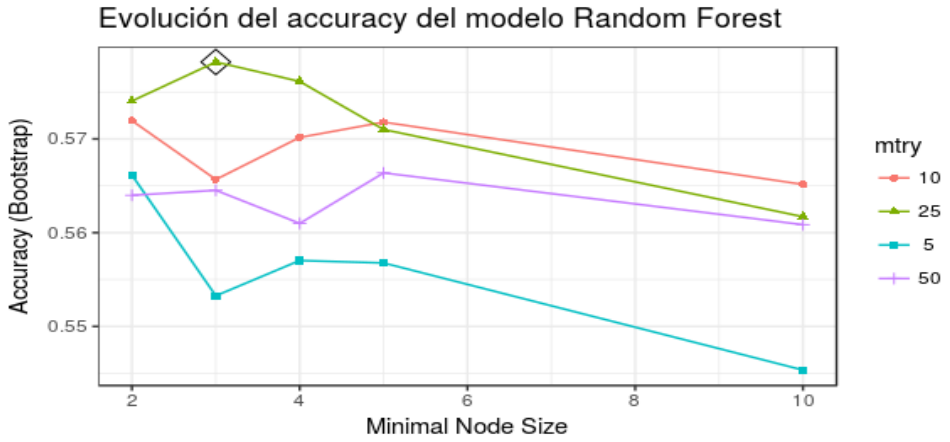
```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(mtry = c(5, 10, 25, 50),
                               min.node.size = c(2, 3, 4, 5, 10),
                               splitrule = "gini")

set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)
```

```
# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# =============================================================================
set.seed(342)
rf_pca <- train(
            form   = tipo_tumor ~ .,
            data   = datos_train_pca,
            method = "ranger",
            tuneGrid = hiperparametros,
            metric = "Accuracy",
            trControl = control_train,
            # Número de árboles ajustados
            num.trees = 500)

saveRDS(object = rf_pca, file = "rf_pca.rds")
registerDoMC(cores = 1)

rf_pca
```

```
## Random Forest
##
## 154 samples
##  78 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  Accuracy   Kappa
##    5     2             0.5661429  0.5218283
##    5     3             0.5532681  0.5073647
##    5     4             0.5570295  0.5120235
##    5     5             0.5567665  0.5112416
##    5    10             0.5453416  0.4993187
##   10     2             0.5719472  0.5288984
##   10     3             0.5656711  0.5221776
##   10     4             0.5701599  0.5270407
##   10     5             0.5717906  0.5286839
##   10    10             0.5651421  0.5214604
##   25     2             0.5740460  0.5321156
```

```
##   25     3             0.5782047  0.5365518
##   25     4             0.5761293  0.5346299
##   25     5             0.5710010  0.5290098
##   25    10             0.5616947  0.5186507
##   50     2             0.5639869  0.5217320
##   50     3             0.5645222  0.5223521
##   50     4             0.5609861  0.5187161
##   50     5             0.5663742  0.5243813
##   50    10             0.5608526  0.5184299
##
## Tuning parameter 'splitrule' was held constant at a value of gini
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 25, splitrule = gini
##  and min.node.size = 3.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(rf_pca, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo Random Forest") +
  guides(color = guide_legend(title = "mtry"),
         shape = guide_legend(title = "mtry")) +
  theme_bw()
```



Mejor modelo: *mtry* = 25, *splitrule* = gini, *min.node.size* = 3, *accuracy* = 0.5782047.

Neural Network

El método *nnet* de caret emplea la función nnet() del paquete nnet para crear redes neuronales con una capa oculta. Este algoritmo tiene 2 hiperparámetros:

- size: número de neuronas en la capa oculta.
- decay: controla la regularización durante el entrenamiento de la red.

En vista de los resultados obtenidos con los algoritmos anteriores, se emplean únicamnete los filtrados *filtrado_s2n_140* y *filtrado_s2n_70*.

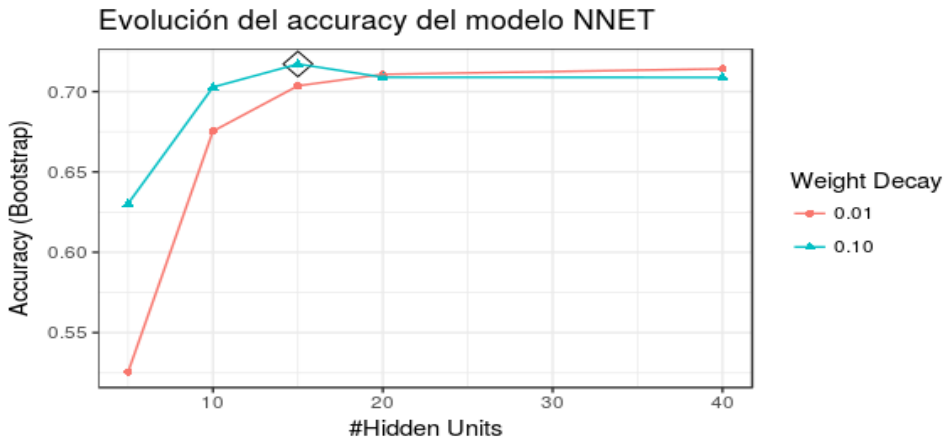
Filtrado por S2N 140

```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(size = c(5, 10, 15, 20, 40),
                               decay = c(0.01, 0.1))

set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)
```

```
# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
nnet_s2n_140 <- train(
                form = tipo_tumor ~ .,
                data = datos_train[c("tipo_tumor", filtrado_s2n_140)],
                method = "nnet",
                tuneGrid = hiperparametros,
                metric = "Accuracy",
                trControl = control_train,
                # Rango de inicialización de los pesos
                rang = c(-0.7, 0.7),
                # Número máximo de pesos
                MaxNWts = 10000,
                # Para que no se muestre cada iteración por pantalla
                trace = FALSE
               )

saveRDS(object = nnet_s2n_140, file = "nnet_s2n_140.rds")
registerDoMC(cores = 1)

nnet_s2n_140
```

```
## Neural Network
##
## 154 samples
## 124 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy   Kappa
##    5    0.01   0.5254675  0.4811417
##    5    0.10   0.6302975  0.5943459
##   10    0.01   0.6753978  0.6436091
##   10    0.10   0.7028002  0.6739635
##   15    0.01   0.7036079  0.6747143
##   15    0.10   0.7171688  0.6897189
##   20    0.01   0.7107725  0.6826762
##   20    0.10   0.7089224  0.6804983
##   40    0.01   0.7141754  0.6862463
##   40    0.10   0.7088533  0.6804054
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 15 and decay = 0.1.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(nnet_s2n_140, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo NNET") +
  theme_bw()
```



Mejor modelo: *size* = 15, *decay* = 0.1, *accuracy* = 0.7171688.

Filtrado por S2N 70

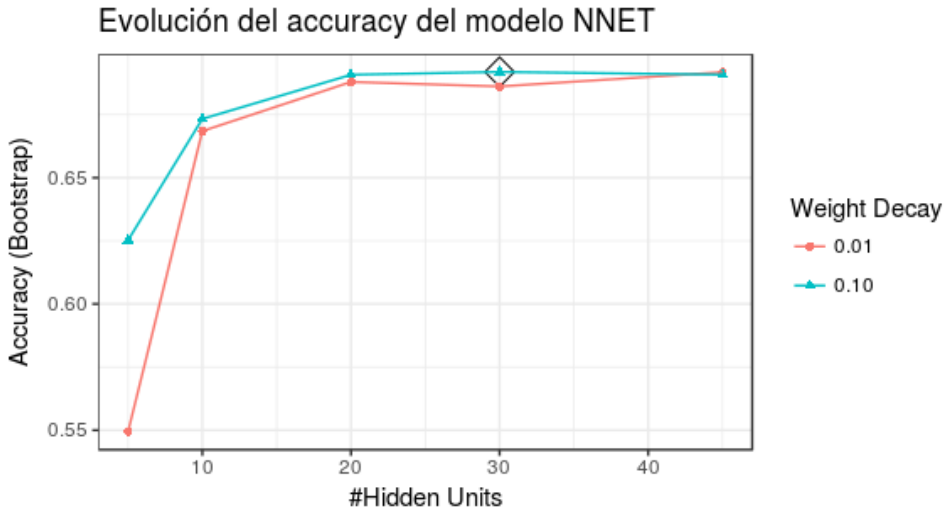
```
# PARALELIZACIÓN DE PROCESO
#==============================================================================
library(doMC)
registerDoMC(cores = 3)

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
#==============================================================================
repeticiones_boot <- 50

# Hiperparámetros
hiperparametros <- expand.grid(size = c(5, 10, 20, 30, 45),
                               decay = c(0.01, 0.1))

set.seed(123)
seeds <- vector(mode = "list", length = repeticiones_boot + 1)
for (i in 1:repeticiones_boot) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[repeticiones_boot + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
#==============================================================================
```

```
control_train <- trainControl(method = "boot", number = repeticiones_boot,
                              seeds = seeds, returnResamp = "final",
                              verboseIter = FALSE, allowParallel = TRUE)

# AJUSTE DEL MODELO
# ==============================================================================
set.seed(342)
nnet_s2n_70 <- train(
                form = tipo_tumor ~ .,
                data = datos_train[c("tipo_tumor", filtrado_s2n_70)],
                method = "nnet",
                tuneGrid = hiperparametros,
                metric = "Accuracy",
                trControl = control_train,
                # Rango de inicialización de los pesos
                rang = c(-0.7, 0.7),
                # Número máximo de pesos
                MaxNWts = 10000,
                # Para que no se muestre cada iteración por pantalla
                trace = FALSE
              )

saveRDS(object = nnet_s2n_70, file = "nnet_s2n_70.rds")
registerDoMC(cores = 1)

nnet_s2n_70
```

```
## Neural Network
##
## 154 samples
##  70 predictors
##  14 classes: 'Bladder', 'Breast', 'CNS', 'Colorectal', 'Leukemia', 'Lung',
'Lymphoma', 'Melanoma', 'Mesothelioma', 'Ovary', 'Pancreas', 'Prostate', 'Renal',
'Uterus'
##
## No pre-processing
## Resampling: Bootstrapped (50 reps)
## Summary of sample sizes: 154, 154, 154, 154, 154, 154, ...
## Resampling results across tuning parameters:
##
##   size  decay  Accuracy   Kappa
##    5    0.01   0.5494064  0.5068716
##    5    0.10   0.6252516  0.5892019
##   10    0.01   0.6685003  0.6362067
##   10    0.10   0.6733199  0.6412894
##   20    0.01   0.6879614  0.6573582
##   20    0.10   0.6908082  0.6604591
##   30    0.01   0.6861679  0.6555563
##   30    0.10   0.6919208  0.6617316
##   45    0.01   0.6918023  0.6616542
```

```
##   45    0.10   0.6908365  0.6606609
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 30 and decay = 0.1.
```

```
# REPRESENTACIÓN GRÁFICA
# ==============================================================================
ggplot(nnet_s2n_70, highlight = TRUE) +
  labs(title = "Evolución del accuracy del modelo NNET") +
  theme_bw()
```



Mejor modelo: *size* = 30, *decay* = 0.1, *accuracy* = 0.6919208.

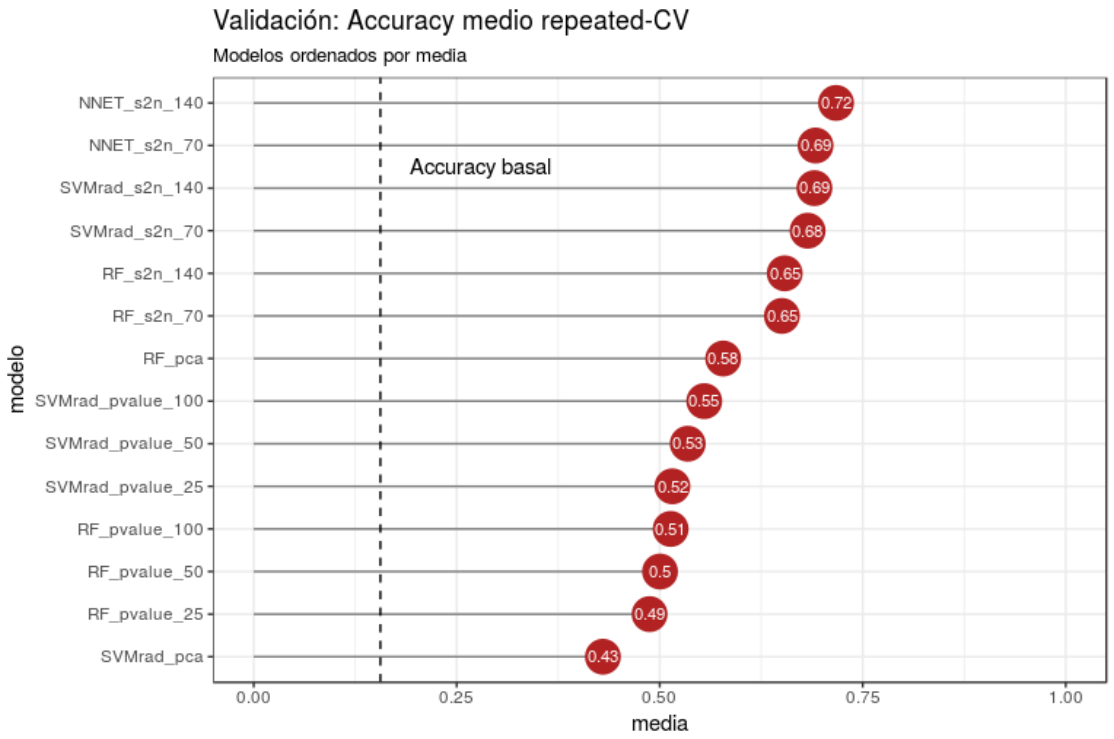
Comparación de modelos

Error de validación

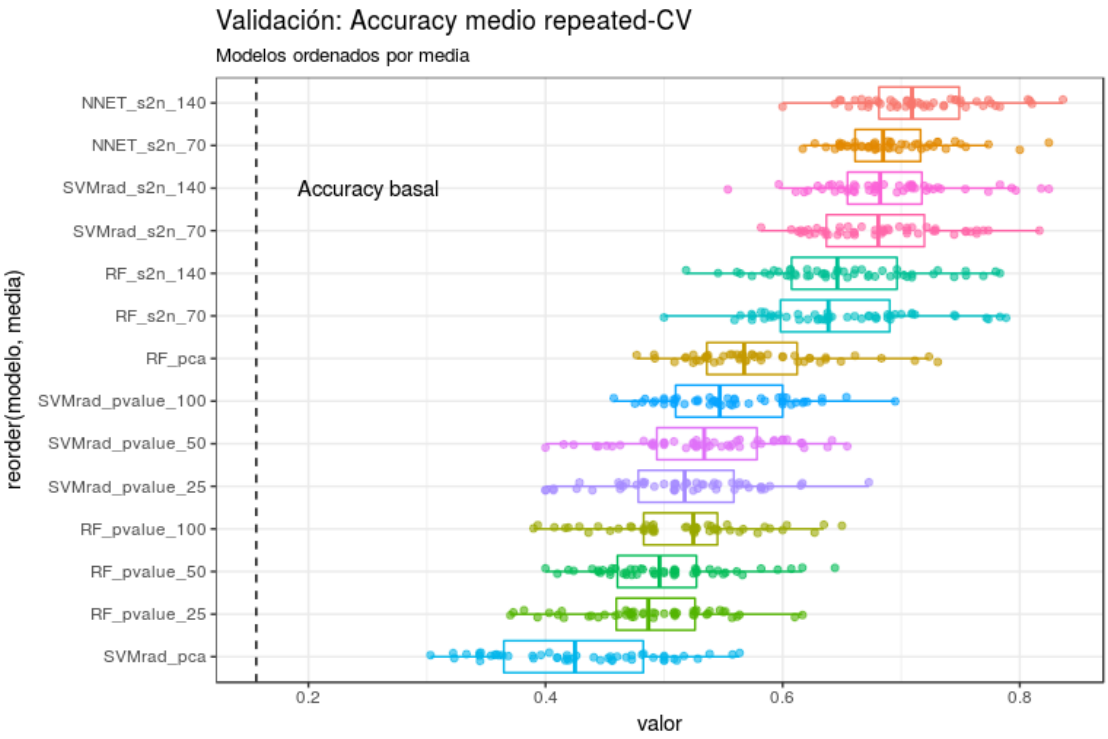
```
modelos <- list(
  SVMrad_pvalue_100 = svmrad_pvalue_100,
  SVMrad_pvalue_50  = svmrad_pvalue_50,
  SVMrad_pvalue_25  = svmrad_pvalue_25,
  SVMrad_s2n_140    = svmrad_s2n_140,
  SVMrad_s2n_70     = svmrad_s2n_70,
  SVMrad_pca        = svmrad_pca,
  RF_pvalue_100     = rf_pvalue_100,
  RF_pvalue_50      = rf_pvalue_50,
  RF_pvalue_25      = rf_pvalue_25,
  RF_s2n_140        = rf_s2n_140,
  RF_s2n_70         = rf_s2n_70,
  RF_pca            = rf_pca,
  NNET_s2n_140      = nnet_s2n_140,
  NNET_s2n_70       = nnet_s2n_70
  )

resultados_resamples <- resamples(modelos)

# Se trasforma el dataframe devuelto por resamples() para separar el nombre del
# modelo y las métricas en columnas distintas.
metricas_resamples <- resultados_resamples$values %>%
                        gather(key = "modelo", value = "valor", -Resample) %>%
                        separate(col = "modelo", into = c("modelo", "metrica"),
                                 sep = "~", remove = TRUE)
# Accuracy y Kappa promedio de cada modelo
promedio_metricas_resamples <- metricas_resamples %>%
  group_by(modelo, metrica) %>%
  summarise(media = mean(valor)) %>%
  spread(key = metrica, value = media) %>%
  arrange(desc(Accuracy))

promedio_metricas_resamples
```

```
## # A tibble: 14 x 3
## # Groups:   modelo [14]
##    modelo            Accuracy Kappa
##    <chr>                <dbl> <dbl>
##  1 NNET_s2n_140         0.717 0.690
##  2 NNET_s2n_70          0.692 0.662
##  3 SVMrad_s2n_140       0.690 0.661
##  4 SVMrad_s2n_70        0.682 0.651
##  5 RF_s2n_140           0.654 0.620
##  6 RF_s2n_70            0.650 0.616
##  7 RF_pca               0.578 0.537
##  8 SVMrad_pvalue_100    0.555 0.513
##  9 SVMrad_pvalue_50     0.535 0.491
## 10 SVMrad_pvalue_25     0.516 0.470
## 11 RF_pvalue_100        0.513 0.468
## 12 RF_pvalue_50         0.500 0.454
## 13 RF_pvalue_25         0.488 0.440
## 14 SVMrad_pca           0.430 0.379
```

```r
metricas_resamples %>%
  filter(metrica == "Accuracy") %>%
  group_by(modelo) %>%
  summarise(media = mean(valor)) %>%
  ggplot(aes(x = reorder(modelo, media), y = media, label = round(media, 2))) +
    geom_segment(aes(x = reorder(modelo, media), y = 0,
                     xend = modelo, yend = media),
                     color = "grey50") +
    geom_point(size = 8, color = "firebrick") +
    geom_text(color = "white", size = 3) +
    scale_y_continuous(limits = c(0, 1)) +
    # Accuracy basal
    geom_hline(yintercept = 0.156, linetype = "dashed") +
    annotate(geom = "text", y = 0.28, x = 12.5, label = "Accuracy basal") +
    labs(title = "Validación: Accuracy medio repeated-CV",
         subtitle = "Modelos ordenados por media",
         x = "modelo") +
    coord_flip() +
    theme_bw()
```



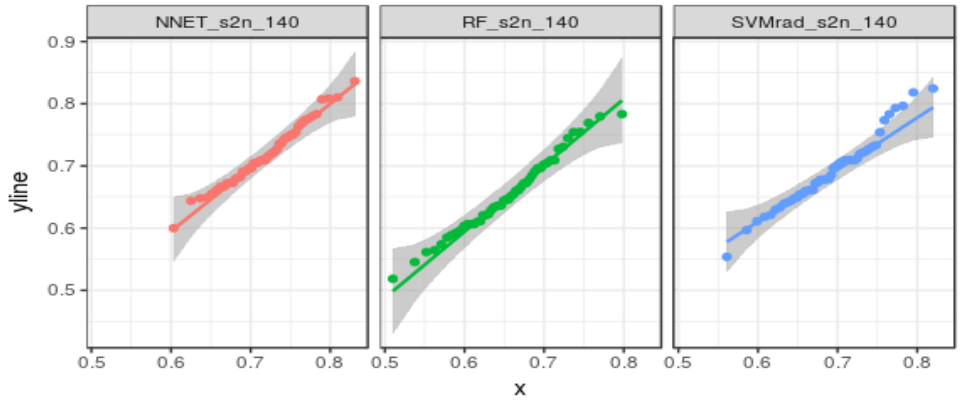

```r
metricas_resamples %>% filter(metrica == "Accuracy") %>%
  group_by(modelo) %>%
  mutate(media = mean(valor)) %>%
  ungroup() %>%
  ggplot(aes(x = reorder(modelo, media), y = valor, color = modelo)) +
    geom_boxplot(alpha = 0.6, outlier.shape = NA) +
    geom_jitter(width = 0.1, alpha = 0.6) +
    # Accuracy basal
    geom_hline(yintercept = 0.156, linetype = "dashed") +
    annotate(geom = "text", y = 0.25, x = 12, label = "Accuracy basal") +
    theme_bw() +
    labs(title = "Validación: Accuracy medio repeated-CV",
         subtitle = "Modelos ordenados por media") +
    coord_flip() +
    theme(legend.position = "none")
```



Una de las ventajas de los métodos de validación es que, si se han empleado exactamente los mismos datos en todos los modelos (en este caso es así gracias a las semillas), se pueden aplicar test de hipótesis que permitan determinar si las diferencias observadas entre modelos son significativas o si solo son debidas a variaciones aleatorias. Acorde a los errores de validación obtenidos por *bootstrapping*, los mejores modelos se consiguen empleando los genes seleccionados por el estadístico *S2N* (140 y 70), y con los algoritmos de redes neuronales, *SVM* radial y *Random Forest*. Se compara el mejor de cada uno de estos 3 algoritmos para determinar si hay evidencias de que uno de ellos sea superior a los demás.

Si se cumple la condición de normalidad, se pueden aplicar un t-test de datos pareados para comparar el *accuracy* medio de cada modelo.

```
library(qqplotr)
metricas_resamples %>%
  filter(modelo %in% c("NNET_s2n_140", "SVMrad_s2n_140", "RF_s2n_140") &
         metrica == "Accuracy") %>%
  ggplot(aes(sample = valor, color = modelo)) +
    stat_qq_band(alpha = 0.5, color = "gray") +
    stat_qq_line() +
    stat_qq_point() +
    theme_bw() +
  theme(legend.position = "none") +
    facet_wrap(~modelo)
```



El análisis gráfico no muestra grandes desviaciones de la normal, además, dado que se dispone de más de 30 valores por grupo, el *t-test* tiene cierta robustez. Se procede a comparar los 3 modelos.

```
metricas_ttest <- metricas_resamples %>%
  filter(modelo %in% c("NNET_s2n_140", "SVMrad_s2n_140", "RF_s2n_140") &
         metrica == "Accuracy") %>%
  select(-metrica)

pairwise.t.test(x = metricas_ttest$valor,
                g = metricas_ttest$modelo,
                paired = TRUE,
                # Al ser solo 3 comparaciones, no se añade ajuste de p.value
                p.adjust.method = "none")
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  metricas_ttest$valor and metricas_ttest$modelo
##
##                NNET_s2n_140 RF_s2n_140
## RF_s2n_140     5.0e-10      -
## SVMrad_s2n_140 4.5e-05      0.00011
##
## P value adjustment method: none
```

Para un nivel de significancia $\alpha = 0.05$, se encuentran evidencias para aceptar que el *accuracy* promedio de los modelos es distinto. Por lo tanto, según el proceso de validación por *bootstraping*, el mejor modelo es *NNET_s2n_140*.

Error de test

```
predic_svmrad_pvalue_100 <- predict(object = svmrad_pvalue_100, newdata=datos_test)
predic_svmrad_pvalue_50  <- predict(object = svmrad_pvalue_50, newdata= datos_test)
predic_svmrad_pvalue_25  <- predict(object = svmrad_pvalue_25, newdata= datos_test)
predic_svmrad_s2n_140    <- predict(object = svmrad_s2n_140, newdata = datos_test)
predic_svmrad_s2n_70     <- predict(object = svmrad_s2n_70, newdata = datos_test)
predic_svmrad_pca        <- predict(object = svmrad_pca, newdata = datos_test_pca)
predic_rf_pvalue_100     <- predict(object = rf_pvalue_100, newdata = datos_test)
predic_rf_pvalue_50      <- predict(object = rf_pvalue_50, newdata = datos_test)
predic_rf_pvalue_25      <- predict(object = rf_pvalue_25, newdata = datos_test)
predic_rf_s2n_140        <- predict(object = rf_s2n_140, newdata = datos_test)
predic_rf_s2n_70         <- predict(object = rf_s2n_70, newdata = datos_test)
predic_rf_pca            <- predict(object = rf_pca, newdata = datos_test_pca)
predic_nnet_s2n_140      <- predict(object = nnet_s2n_140, newdata = datos_test)
predic_nnet_s2n_70       <- predict(object = nnet_s2n_70, newdata = datos_test)

predicciones <- data.frame(
      SVMrad_pvalue_100 = predic_svmrad_pvalue_100,
      SVMrad_pvalue_50  = predic_svmrad_pvalue_50,
      SVMrad_pvalue_25  = predic_svmrad_pvalue_25,
      SVMrad_s2n_140    = predic_svmrad_s2n_140,
      SVMrad_s2n_70     = predic_svmrad_s2n_70,
      SVMrad_pca        = predic_svmrad_pca,
      RF_pvalue_100     = predic_rf_pvalue_100,
      RF_pvalue_50      = predic_rf_pvalue_50,
      RF_pvalue_25      = predic_rf_pvalue_25,
      RF_s2n_140        = predic_rf_s2n_140,
      RF_s2n_70         = predic_rf_s2n_70,
      RF_pca            = predic_rf_pca,
      NNET_s2n_140      = predic_nnet_s2n_140,
      NNET_s2n_70       = predic_nnet_s2n_70,
      valor_real        = datos_test$tipo_tumor
    )

predicciones %>% head()
```

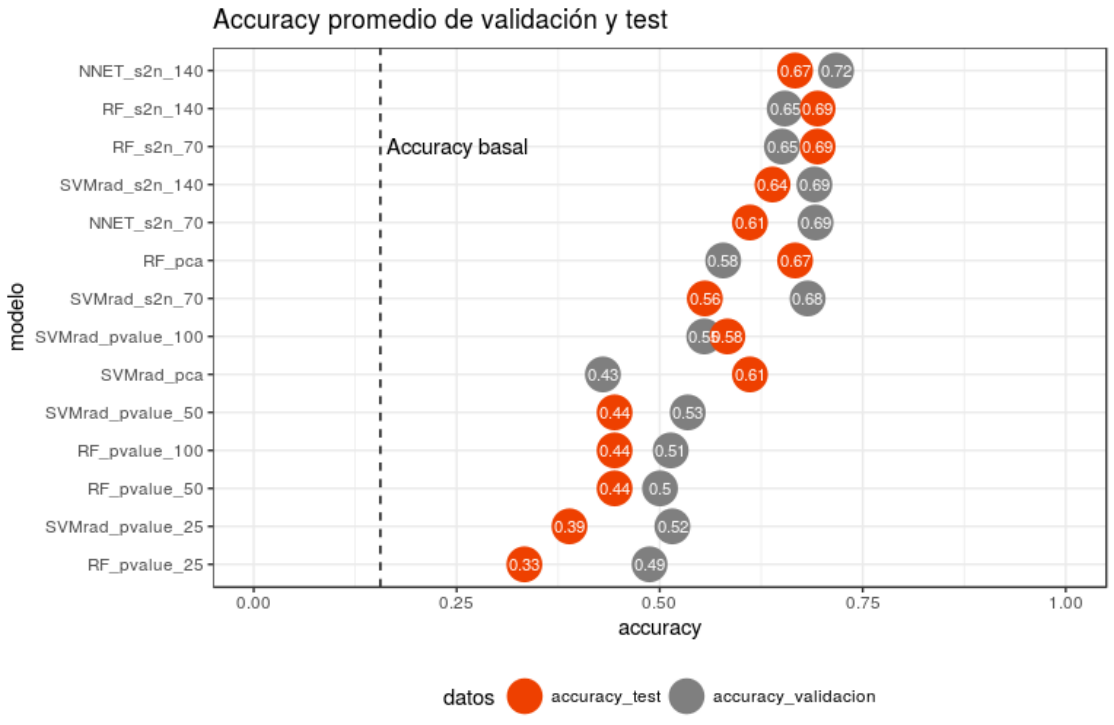
```
##   SVMrad_pvalue_100 SVMrad_pvalue_50 SVMrad_pvalue_25 SVMrad_s2n_140
## 1              Lung         Prostate         Prostate     Colorectal
## 2             Ovary            Ovary            Ovary          Ovary
## 3             Ovary             Lung             Lung       Pancreas
## 4          Pancreas         Melanoma             Lung         Breast
## 5               CNS              CNS              CNS            CNS
## 6               CNS              CNS              CNS            CNS
##   SVMrad_s2n_70 SVMrad_pca RF_pvalue_100 RF_pvalue_50 RF_pvalue_25
## 1    Colorectal Colorectal      Pancreas     Pancreas     Pancreas
## 2         Ovary      Ovary         Ovary        Ovary        Ovary
## 3      Pancreas      Ovary         Ovary        Ovary        Ovary
```

```
## 4        Breast     Breast      Pancreas     Pancreas     Pancreas
## 5           CNS        CNS           CNS          CNS          CNS
## 6           CNS    Bladder           CNS          CNS          CNS
##   RF_s2n_140    RF_s2n_70  RF_pca NNET_s2n_140 NNET_s2n_70 valor_real
## 1 Colorectal Mesothelioma Bladder   Colorectal  Colorectal    Bladder
## 2      Ovary      Bladder   Ovary      Bladder       Ovary    Bladder
## 3    Bladder      Bladder   Ovary     Pancreas    Pancreas     Breast
## 4 Colorectal   Colorectal  Breast       Breast      Breast     Breast
## 5        CNS          CNS     CNS          CNS         CNS        CNS
## 6        CNS          CNS     CNS          CNS         CNS        CNS
```

```
calculo_accuracy <- function(x, y){
  return(mean(x == y))
}
accuracy_test <- map_dbl(.x = predicciones[, -15],
                         .f = calculo_accuracy,
                         y = predicciones[, 15]) %>%
                 as.data.frame() %>%
                 rename(accuracy_test = ".") %>%
                 rownames_to_column(var = "modelo") %>%
                 arrange(desc(accuracy_test))

metricas_resamples %>%
  group_by(modelo, metrica) %>%
  summarise(media = mean(valor)) %>%
  spread(key = metrica, value = media) %>%
  select(accuracy_validacion = Accuracy) %>%
  left_join(accuracy_test, by = "modelo") %>%
  arrange(desc(accuracy_test))
```

```
## # A tibble: 14 x 3
## # Groups:   modelo [14]
##    modelo            accuracy_validacion accuracy_test
##    <chr>                           <dbl>         <dbl>
##  1 RF_s2n_140                      0.654         0.694
##  2 RF_s2n_70                       0.650         0.694
##  3 NNET_s2n_140                    0.717         0.667
##  4 RF_pca                          0.578         0.667
##  5 SVMrad_s2n_140                  0.690         0.639
##  6 NNET_s2n_70                     0.692         0.611
##  7 SVMrad_pca                      0.430         0.611
##  8 SVMrad_pvalue_100               0.555         0.583
##  9 SVMrad_s2n_70                   0.682         0.556
## 10 RF_pvalue_100                   0.513         0.444
## 11 RF_pvalue_50                    0.500         0.444
## 12 SVMrad_pvalue_50                0.535         0.444
## 13 SVMrad_pvalue_25                0.516         0.389
## 14 RF_pvalue_25                    0.488         0.333
```

```
metricas_resamples %>%
  group_by(modelo, metrica) %>%
  summarise(media = mean(valor)) %>%
  spread(key = metrica, value = media) %>%
  select(accuracy_validacion = Accuracy) %>%
  left_join(accuracy_test, by = "modelo") %>%
  gather(key = "datos", value = "accuracy", -modelo) %>%
  ggplot(aes(x = reorder(modelo, accuracy), y = accuracy,
             color = datos, label = round(accuracy, 2))) +
  geom_point(size = 8) +
  ylim(0, 1) +
  scale_color_manual(values = c("orangered2", "gray50")) +
  geom_text(color = "white", size = 3) +
  # Accuracy basal
  geom_hline(yintercept = 0.156, linetype = "dashed") +
  annotate(geom = "text", y = 0.25, x = 12, label = "Accuracy basal") +
  coord_flip() +
  labs(title = "Accuracy promedio de validación y test",
       x = "modelo") +
  theme_bw() +
  theme(legend.position = "bottom")
```



Acorde al *accuracy* de test, los mejores modelos son *RF_s2n_140* y *RF_s2n_70*, seguidos por *NNET_s2n_140*.

```
predicciones %>%
 select(RF_s2n_140, valor_real) %>%
 filter(RF_s2n_140 != valor_real)
```

```
##    RF_s2n_140 valor_real
## 1  Colorectal    Bladder
## 2       Ovary    Bladder
## 3     Bladder     Breast
## 4  Colorectal     Breast
## 5      Breast Colorectal
## 6     Bladder   Melanoma
## 7         CNS      Ovary
## 8  Colorectal      Ovary
## 9        Lung   Prostate
## 10    Bladder   Prostate
## 11 Colorectal      Renal
```

```
confusionMatrix(data=predicciones$RF_s2n_140,
                reference=as.factor(datos_test$tipo_tumor))
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Bladder Breast CNS Colorectal Leukemia Lung Lymphoma
##   Bladder            0      1   0          0        0    0        0
##   Breast             0      0   0          1        0    0        0
##   CNS                0      0   4          0        0    0        0
##   Colorectal         1      1   0          1        0    0        0
##   Leukemia           0      0   0          0        6    0        0
##   Lung               0      0   0          0        0    2        0
##   Lymphoma           0      0   0          0        0    0        4
##   Melanoma           0      0   0          0        0    0        0
##   Mesothelioma       0      0   0          0        0    0        0
##   Ovary              1      0   0          0        0    0        0
##   Pancreas           0      0   0          0        0    0        0
##   Prostate           0      0   0          0        0    0        0
##   Renal              0      0   0          0        0    0        0
##   Uterus             0      0   0          0        0    0        0
```

```
##               Reference
## Prediction     Melanoma Mesothelioma Ovary Pancreas Prostate Renal Uterus
##   Bladder             1            0     0        0        1     0      0
##   Breast              0            0     0        0        0     0      0
##   CNS                 0            0     1        0        0     0      0
##   Colorectal          0            0     1        0        0     1      0
##   Leukemia            0            0     0        0        0     0      0
##   Lung                0            0     0        0        1     0      0
##   Lymphoma            0            0     0        0        0     0      0
##   Melanoma            1            0     0        0        0     0      0
##   Mesothelioma        0            2     0        0        0     0      0
##   Ovary               0            0     0        0        0     0      0
##   Pancreas            0            0     0        2        0     0      0
##   Prostate            0            0     0        0        0     0      0
##   Renal               0            0     0        0        0     1      0
##   Uterus              0            0     0        0        0     0      2
##
## Overall Statistics
##
##                Accuracy : 0.6944
##                  95% CI : (0.5189, 0.8365)
##     No Information Rate : 0.1667
##     P-Value [Acc > NIR] : 3.104e-12
##
##                   Kappa : 0.665
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Bladder Class: Breast Class: CNS
## Sensitivity                 0.00000       0.00000     1.0000
## Specificity                 0.91176       0.97059     0.9688
## Pos Pred Value              0.00000       0.00000     0.8000
## Neg Pred Value              0.93939       0.94286     1.0000
## Prevalence                  0.05556       0.05556     0.1111
## Detection Rate              0.00000       0.00000     0.1111
## Detection Prevalence        0.08333       0.02778     0.1389
## Balanced Accuracy           0.45588       0.48529     0.9844
##                      Class: Colorectal Class: Leukemia Class: Lung
## Sensitivity                    0.50000          1.0000     1.00000
## Specificity                    0.88235          1.0000     0.97059
## Pos Pred Value                 0.20000          1.0000     0.66667
## Neg Pred Value                 0.96774          1.0000     1.00000
## Prevalence                     0.05556          0.1667     0.05556
## Detection Rate                 0.02778          0.1667     0.05556
## Detection Prevalence           0.13889          0.1667     0.08333
## Balanced Accuracy              0.69118          1.0000     0.98529
```

```
##                      Class: Lymphoma Class: Melanoma Class: Mesothelioma
## Sensitivity                   1.0000         0.50000             1.00000
## Specificity                   1.0000         1.00000             1.00000
## Pos Pred Value                1.0000         1.00000             1.00000
## Neg Pred Value                1.0000         0.97143             1.00000
## Prevalence                    0.1111         0.05556             0.05556
## Detection Rate                0.1111         0.02778             0.05556
## Detection Prevalence          0.1111         0.02778             0.05556
## Balanced Accuracy             1.0000         0.75000             1.00000
##                      Class: Ovary Class: Pancreas Class: Prostate
## Sensitivity               0.00000         1.00000         0.00000
## Specificity               0.97059         1.00000         1.00000
## Pos Pred Value            0.00000         1.00000             NaN
## Neg Pred Value            0.94286         1.00000         0.94444
## Prevalence                0.05556         0.05556         0.05556
## Detection Rate            0.00000         0.05556         0.00000
## Detection Prevalence      0.02778         0.05556         0.00000
## Balanced Accuracy         0.48529         1.00000         0.50000
##                      Class: Renal Class: Uterus
## Sensitivity               0.50000        1.00000
## Specificity               1.00000        1.00000
## Pos Pred Value            1.00000        1.00000
## Neg Pred Value            0.97143        1.00000
## Prevalence                0.05556        0.05556
## Detection Rate            0.02778        0.05556
## Detection Prevalence      0.02778        0.05556
## Balanced Accuracy         0.75000        1.00000
```

La matriz de confusión muestra que el modelo clasifica peor unos tipos de tumor que otros, en concreto, los tumores de *Bladder*, *Breast*, *Ovary* y *Prostate* tienen mayor error.

Mejor modelo

Para identificar cuál de todos es el mejor modelo, conviene tener en cuenta los dos tipos de error: el de validación, en este caso *bootstrapping*, y el de test. En vista de los resultados obtenidos, no puede afirmarse que, para este problema, exista un modelo que supere claramente a todos los demás, de ahí que, pequeñas diferencias, provoquen cambios en el orden de los modelos entre validación y de test. Tanto *RF_s2n_140*, *NNET_s2n_140* y *SVMrad_s2n_140* son buenos candidatos.

Model ensembling

```
moda <- function(x){
  return(names(which.max(table(x))))
}

predicciones_ensemble <- predicciones %>%
                         select(NNET_s2n_140, RF_s2n_140, SVMrad_s2n_140) %>%
                         mutate(moda = apply(X = select(.data = predicciones,
                                                        NNET_s2n_140,
                                                        RF_s2n_140,
                                                        SVMrad_s2n_140),
                                             MARGIN = 1,
                                             FUN = moda))
predicciones_ensemble %>% head()
```

```
##   NNET_s2n_140 RF_s2n_140 SVMrad_s2n_140       moda
## 1   Colorectal Colorectal     Colorectal Colorectal
## 2      Bladder      Ovary          Ovary      Ovary
## 3     Pancreas    Bladder       Pancreas   Pancreas
## 4       Breast Colorectal         Breast     Breast
## 5          CNS        CNS            CNS        CNS
## 6          CNS        CNS            CNS        CNS
```

```
mean(predicciones_ensemble$moda == datos_test$tipo_tumor)
```

```
## [1] 0.6666667
```

En este caso, el *ensemble* de los modelos no consigue una mejora.

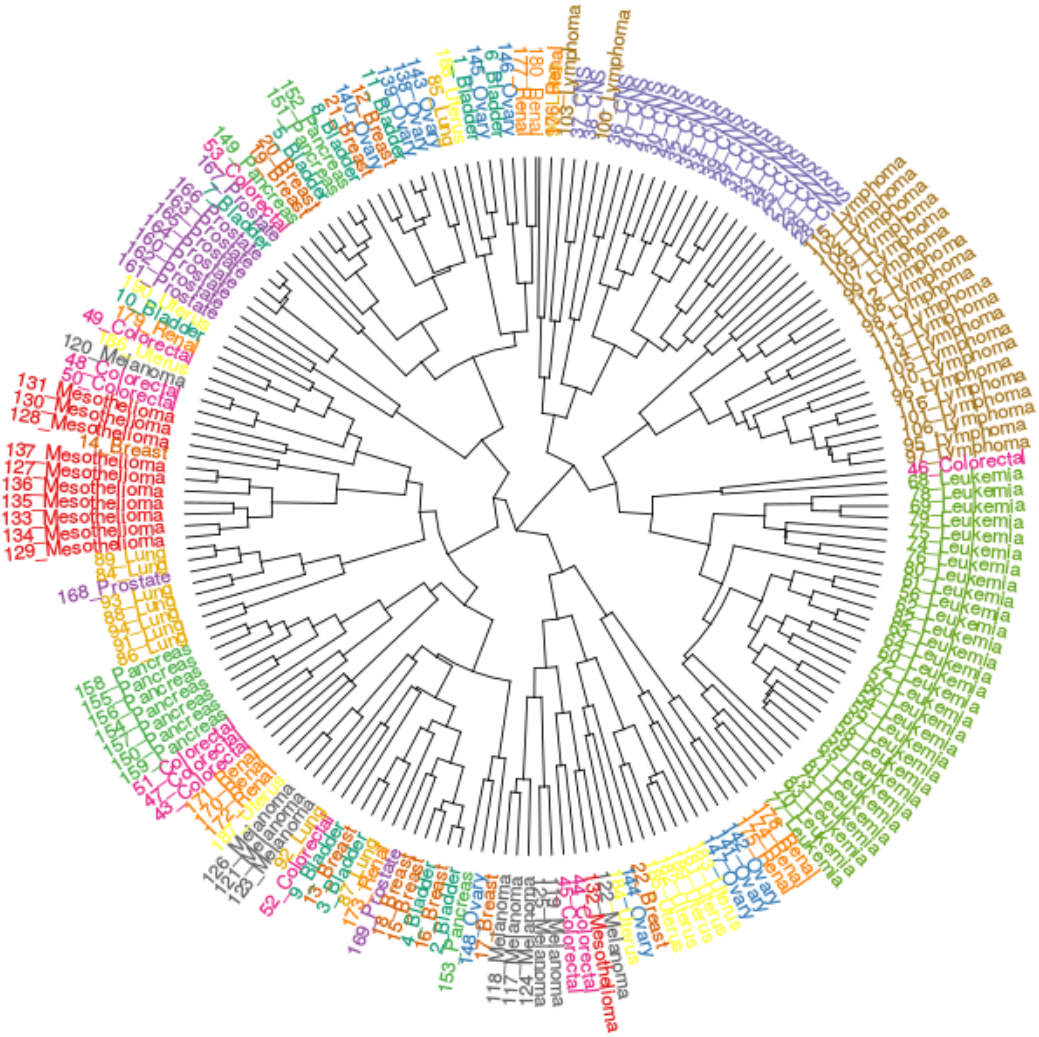
Clustering

Una de las premisas en la que se basa el análisis realizado es la idea de que, tumores de distintos tejidos, tienen un perfil de expresión genética distinto y, por lo tanto, puede emplearse esta información para clasificarlos. Una forma de explorar si esto es cierto, es mediante el uso de técnicas de aprendizaje no supervisado, en concreto el clustering.

Se procede a agrupar los tumores mediante clustering aglomerativo empleando la selección de genes *filtrado_s2n_140*.

```
# Se unen de nuevo todos los datos en un único dataframe
datos_clustering <- bind_rows(datos_train, datos_test)
datos_clustering <- datos_clustering %>% arrange(tipo_tumor)

# La librería factoextra emplea el nombre de las filas del dataframe para
# identificar cada observación.
datos_clustering <- datos_clustering %>% as.data.frame()
rownames(datos_clustering) <- paste(1:nrow(datos_clustering),
                                    datos_clustering$tipo_tumor,
                                    sep = "_")

# Se emplean únicamente los genes filtrados
datos_clustering <- datos_clustering %>% select(tipo_tumor, filtrado_s2n_140)

# Se calculan las distancias en base a la correlación de Pearson
mat_distancias <- get_dist(datos_clustering[, -1],
                           method = "pearson",
                           stand = FALSE)

# HIERARCHICAL CLUSTERING
# ==============================================================================
library(cluster)
library(factoextra)
set.seed(101)
hc_average <- hclust(d = mat_distancias, method = "complete")

# VISUALIZACIÓN DEL DENDOGRAMA
# ==============================================================================
# Vector de colores para cada observacion: Se juntan dos paletas para tener
# suficientes colores
library(RColorBrewer)
colores <- c(brewer.pal(n=8, name="Dark2"),brewer.pal(n = 8, name = "Set1")) %>%
           unique()
# Se seleccionan 14 colores, uno para cada tipo de tumor
colores <- colores[1:14]
# Se asigna a cada tipo de tumor uno de los colores. Para conseguirlo de forma
```

```
# rápida, se convierte la variable tipo_tumor en factor y se emplea su codificación
# numérica interna para asignar los colores.
colores <- colores[as.factor(datos_clustering$tipo_tumor)]
# Se reorganiza el vector de colores según el orden en que se han agrupado las
# observaciones en el clustering
colores <- colores[hc_average$order]
fviz_dend(x = hc_average, label_cols = colores, cex = 0.6, lwd = 0.3,
          main = "Linkage completo", type = "circular")
```



El algoritmo de *clustering* no es capaz de diferenciar los 14 grupos de tumores, lo que pone de manifiesto la dificultad de la clasificación. El dendograma muestra que los tumores de *Breast*, *Blader* y *Colorectal* tienen un perfil de expresión menos definido. Estos tipos de tumor coinciden con los tipos en los que más ha fallado el modelo predictivo (ver matriz de confusión).

Bibliografía

Applied Predictive Modeling By Max Kuhn and Kjell Johnson

Multiclass cancer diagnosis using tumor gene expression signatures, Ramaswamy, Tamayo P, Rifkin R, Mukherjee S, Yeang CH, Angelo M, Ladd C, Reich M, Latulippe E, Mesirov JP, Poggio T, Gerald W, Loda M, Lander ES, Golub TR.

Evaluating Microarray-based Classifi ers: An Overview A.-L. Boulesteix, C. Strobl, T. Augustin and M. Daumer

Data Mining for Bioinformatics by Sumeet Dua, Pradeep Chowriappa

Selection bias in gene extraction on the basis of microarray gene-expression data, Christophe Ambroise and Geoffrey J. McLachlan

Machine learning applications in cancer prognosis and prediction Konstantina Kourou a, Themis P. Exarchos, Kons

http://qpcrupdate.info/haeseleer-bioinf-2005.pdf