Ensamblado

Los métodos de ensamblado o métodos Ensemble (conjunto) consisten en la construcción de predicciones a partir de la combinación de varios modelos.

Una primera aproximación a esta idea la puede dar el siguiente ejemplo suponiendo un problema de regresión:

	y	x_1	x_2	...			$\hat{y}_{red}$	$\hat{y}_{regresión}$	$\hat{y}_{rf}$	...		$\hat{y}_{total}$=media($\hat{y}_{red}$, $\hat{y}_{regresión}$, ...)
DATOS												

1) Se construye un modelo de redes que da lugar a la predicción y1 (sobre los datos test).

2) Se construye otro modelo con regresión, que da lugar a la predicción y2.

3) Se construye un modelo random forest que da lugar a la predicción y3.

4) Se estudia la performance de y1,y2,y3, y del promedio de ellas, y4. Esta variable y4 es una predicción nueva, que a veces puede funcionar mejor que cualquiera de las predicciones y1,y2,y3.

5) O bien, en lugar de promediar y1,y2,y3, se contruye, por ejemplo, un modelo de red neuronal en el que las variables de entrada sean y1,y2,y3. Se obtendría una nueva predicción y4 a partir de este modelo.

Técnicas básicas de ensamblado de modelos

En realidad ya se han visto técnicas de ensamblado: bagging, random forest y gradient boosting (en este último es iterativo pero hay ensamblado enmascarado).

Bagging

=Bootstrap Averaging

1) Se extraen muestras Bootstrap de los datos

2) Se construye el modelo (estimación de parámetros y opcionalmente se seleccionan variables también) con esa muestra bootstrap

3) Se predicen los datos test con cada una de esas muestras bootstrap

4) Se promedia el resultado de las predicciones sobre los datos test

El algoritmo Random Forest es un tipo de Bagging (añadiendo el sorteo de variables en cada nodo)

Boosting

Se trata de un método de ensamblado, pues la predicción final es un sumatorio de predicciones ponderadas obtenidas mediante los arboles como algoritmo de predicción.

El Gradient Boosting es un caso particular. El caso general es utilizar en vez de árboles cualquier otro método de predicción.

Stacking

En realidad hay cierta confusión al aludir a estas técnicas. Se usan también los términos blending o averaging.

Se utiliza en general este término para cualquier tipo de combinación de modelos.

Es decir, dadas las predicciones y1, y2, y3 obtenidas por diferentes algoritmos, se combinan sus resultados. Existen tres opciones básicas:

1) Averaging (promediado): se calcula el promedio de las predicciones, Si se trata de clasificación, se obtiene el promedio de las probabilidades. Se puede utilizar también promedio ponderado, por ejemplo 0.80*predigbm+0.20*predirandomforest

2) Voto (para clasificación): se predice el resultado con mayoría entre las predicciones: y1=0,y2=0,y3=1→predicción=1

3) Combinación a partir de otro algoritmo (esto es estrictamente stacking). Por ejemplo, se introducen en una regresión o árbol y1, y2, y3 como variables independientes. En regresión equivaldría a un promediado de modelos con pesos diferentes.

Justificación Teórica

Se presenta una justificación simplificada de por qué son tan eficientes los métodos de ensamblado. Se puede ver por ejemplo, en el artículo

*Linear and Order Statistics Combiners for Pattern Classification (*Kagan Tumer, Joydeep Ghosh, 1999)

Se puede descargar en:

https://arxiv.org/abs/cs/9905012

Suponiendo las siguientes definiciones:

$\bar{\beta}$ = valor del sesgo **con ensamblado**

β = valor del sesgo **de un único clasificador**

σ_b^2 =término de varianza de un clasificador

s =término fijo relativo a la diferencia entre clases.

δ =término de **correlación** entre clasificadores

$z = \frac{\beta}{\bar{\beta}}$ cociente sesgos clasificador único/ensamblado

El error de un solo clasificador se puede fijar en dos términos relativos al sesgo y varianza:

$$E_{add}(\beta) = \frac{s}{2}(\sigma_b^2 + \beta^2).$$

Y el error del ensamblado se puede aproximar por:

$$E_{add}^{ave}(\bar{\beta}) = \frac{s}{2}\left(\sigma_b^2\left(\frac{1+\delta(N-1)}{N}\right) + \frac{\beta^2}{z^2}\right)$$

Casuística

Error un solo clasificador $$E_{add}(\beta) = \frac{s}{2}(\sigma_b^2 + \beta^2).$$

Error del ensamblado $$E_{add}^{ave}(\bar{\beta}) = \frac{s}{2}\left(\sigma_b^2\left(\frac{1+\delta(N-1)}{N}\right) + \frac{\beta^2}{z^2}\right)$$

1) $\bar{\beta} \leq \beta, \delta < 1$

El error del ensamblado será menor que de un simple clasificador, pues ambos términos del sumatorio son menores .

Manteniendo el sesgo igual o menor, cuanto **menor** sea la **correlación** entre clasificadores δ **menor** será el error global de ensamblado.

2) $\bar{\beta} > \beta, \delta < 1$

Entonces si el sesgo no cambia mucho, todo depende del término de correlación.

Si es muy baja, puede compensar el primer término $\sigma_b^2\left(\frac{1+\delta(N-1)}{N}\right)$ a lo grande que puede llegar a ser el segundo $\frac{\beta^2}{z^2}$ y entonces el error global será menor en el ensamblado que en un clasificador individual.

Lo que ocurre en estos casos es que el sesgo aumenta algo respecto al mejor clasificador pero la varianza se reduce.

3) $\bar{\beta} = \beta, \delta < 1$

Este es el caso por ejemplo al utilizar versiones diferentes del mismo algoritmo (redes con diferente semilla de inicialización de pesos). Como la correlación nunca va a ser 1 y el sesgo es similar, siempre se mejora a una red individual en términos de varianza del error (esta mejora puede no ser significativa).

También es el caso del bagging y random forest, cada árbol tiene un sesgo similar pero al utilizarse diferentes muestras en cada arbol no son idénticos y por tanto la correlación nunca será 1.

Igualmente a menudo diferentes modelos equivalentes (un mismo algoritmo con diferentes variables input pero un sesgo parecido) estarían en este caso.

Consecuencias

1) Siempre, cuanto menor sea la correlación entre clasificadores, más se reducirá el error con el ensamblado. Pero tiene que cumplirse que el sesgo del ensamblado baje, se mantenga, o como mínimo no suba mucho. Desgraciadamente si el sesgo se mejora o no con el ensamblado no se puede saber a priori, son necesarias pruebas empíricas repetidas (CV, train-test) para saberlo.

2) En general, intentaremos unir clasificadores que tengan sesgo suficientemente bajo y similar y con poca correlación entre ellos.

3) Como la correlación no suele ser nunca 1, uniendo clasificadores con sesgo similar aún con correlación alta (0.9), el término $\frac{1+\delta(N-1)}{N}$ siempre va a ser <1 y mejoraremos algo la varianza.

4) Algunos de los métodos de ensamblado integrales como Boosting y RandomForest se aprovechan de estas propiedades combinando árboles diferentes que debido a su construcción (introducir aleatorización en el caso de Random Forest o bien atacar cada vez una diferente construcción de la variable dependiente en el caso de Gradient Boosting) tienen sesgo similar pero correlaciones no demasiado altas entre ellos.

5) Cuando se utilizan muchos clasificadores es difícil establecer reglas generales.

6) Otras técnicas permiten encontrar clasificadores con relativamente baja correlación entre ellos

Por ejemplo:

- Técnicas diferentes (ensemble de gradient boosting+random forest, logística+redes, etc.)
- Sets de variables diferentes
- Aleatorización (entrenar con diferentes datos training para predecir los mismos datos test).

7) Cuando un algoritmo supera claramente a los demás en cuestión de sesgo-varianza, el ensamblado puede que no mejore nada y no merezca la pena. Es el caso del uso de xgboost en Kaggle. Hay que destacar que en este caso xgboost en en sí mismo un método de ensamblado de árboles, más todavía si se utiliza remuestreo de observaciones y/o variables como opciones de xgboost.

Algunas páginas ilustrativas interesantes:

http://mlwave.com/kaggle-ensembling-guide/

http://www.overkillanalytics.net/more-is-always-better-the-power-of-simple-ensembles/

Ventajas de los métodos ensemble

- Bastante robustos, unos modelos corrigen a otros.
- Reducen la varianza del error en general, casi nunca empeoran los modelos.

Desventajas de los métodos ensemble

- Cada modelo tiene sus errores de estimadores de parámetros lo que aumenta aparentemente la complejidad.
- Excesivas posibilidades que a veces llevan al sobreajuste.
- Los resultados no son interpretables.

Construcción de ensamblados con R

En general los pasos para realizar un ensamblado son los siguientes:

1) Obtener en un mismo archivo las probabilidades (si la dependiente es binaria) o valores predichos (si es continua) con cada uno de los algoritmos.

2) Promediar o ponderar estas probabilidades o valores para obtener ensamblados.

3) Construir matriz de confusión para cada modelo incluido los ensamblados, si es dependiente binaria, si es continua no.

El paquete CaretEnsemble

Archivo ejemplo caretEnsemble saheart.R

Este paquete construye ensamblado utilizando un modelo de regresión cuyas variables input son las predicciones de los diferentes algoritmos que se le introducen. Como resultado nos dará unas ponderaciones para cada algoritmo para construir las nuevas predicciones del ensamblado. Aporta también la performance del ensamblado en términos de Accuracy.

Su principal defecto es su rigidez en las ponderaciones, pues no ofrece más soluciones que la que da, y los resultados pueden ser raros o inexplicables, asignando pesos negativos a los algoritmos.

Como siempre, previamente al ensamblado hay que tener decididos y tuneados los modelos y sus parámetros.

```
load ("saheartbis.Rda")
library(caret)
library(caretEnsemble)

# PARÁMETROS EXTRAÍDOS DE TUNING CON CARET de ejemplos anteriores
# A veces hay que cambiar el nombre de los parámetros
# Los de grid hay que ponerlos en un grid
```

```
# avNNet recomienda en este caso size=5 nodos,
# decay=0.1 , iteraciones=200

# gbm
# n.minobsinnode=5,shrinkage=0.001,n.trees=3000,interaction.depth=2

# rforest
# nodesize=10, mtry=6,n.trees=200,replace=TRUE,sampsize=150

# SVM RBF C=5,sigma=0.01

# SVM LINEAL  C=0.03
# SVM Poly  C=0.02,degree=2,scale=2

# ************************************************************
# Estas tres líneas son importantes, sustituir por el nombre
# de variable y archivo
# ************************************************************

formula1<-as.formula(paste("factor(","chd",")","~."))
saheartbis$chd<-as.factor(saheartbis$chd)
levels(saheartbis$chd) <- make.names(levels(factor(saheartbis$chd)))

# Aquí se fijan el número de repeticiones de validación cruzada
# y la semilla
set.seed(3005)
repeticiones=10


# Manera de evaluar los modelos
stackControl <- trainControl(method="repeatedcv",
 number=4, repeats=repeticiones, savePredictions=TRUE,
classProbs=TRUE)

# Parámetros para caret, tunear antes

# Muy importante considerar esto:

# Cada method (algoritmo) tiene parámetros a tunear con Grid
# y parámetros específicos que no se pueden tunear.
# --Los que se pueden tunear hay que ponerlos en un Grid aunque
# solo se les de un valor (ver por ejemplo gbmGrid).
# --Los que no se pueden tunear hay que nombrarlos directamente
# en train (ver por ejemplo rf)


gbmGrid <- expand.grid(n.trees = c(3000),
  interaction.depth = c(2), shrinkage =c(0.001), n.minobsinnode =
c(5))

rfGrid <- expand.grid(mtry=c(6))

svmRadialGrid <- expand.grid(sigma=c(0.01),C=c(5))

svmlinGrid <- expand.grid(C=c(0.03))

svmPolyGrid <- expand.grid(C=c(0.02),degree=c(2),scale=c(2))

set.seed(3005)

models <- caretList(chd~., data=saheartbis, trControl=stackControl,
```

```
tuneList=list(
 parrf=caretModelSpec(method="rf",maxnodes=30,
 n.trees = 200,nodesize=10,sampsize=150,tuneGrid=rfGrid),
 glm=caretModelSpec(method="glm"),
 gbm=caretModelSpec(method="gbm",tuneGrid=gbmGrid),
 svmlinear=caretModelSpec(method="svmLinear",tuneGrid=svmlinGrid),
 svmPoly=caretModelSpec(method="svmPoly",tuneGrid=svmPolyGrid),
 svmradial=caretModelSpec(method="svmRadial",tuneGrid=svmRadialGrid)
 ))

results <- resamples(models)
summary(results)
dotplot(results)

modelCor(results)
splom(results)
results[[2]]

ense <- caretEnsemble(models)

# Aquí se recomiendan los pesos para el ensamblado
# de todos los modelos y se ve la tasa de aciertos
# de cada modelo y ensamblado
summary(ense)


The following models were ensembled: parrf, glm, gbm, svmlinear,
svmPoly, svmradial

They were weighted:

2.3066 3.1481 -1.5802 -4.2925 -0.6839 1.4379 -2.6848

The resulting Accuracy is: 0.7345

The fit for each individual model on the Accuracy is:
    method  Accuracy AccuracySD
     parrf 0.7027792 0.03335384
       glm 0.7289993 0.03121818
       gbm 0.7205266 0.03214976
 svmlinear 0.7387463 0.03015550
   svmPoly 0.6934745 0.02624074
 svmradial 0.7352699 0.02666693
```

Validación cruzada repetida

Archivo ejemplo cruzadas para ensamblado continuas.R

En este caso he creado un esquema para construir ensamblado controlando manualmente los epsos, usando las funciones de validación cruzada repetida orientadas a tener un gráfico de cajas explicativo.

Aquí hay que resaltar que solo se está realizando un promedio ponderado de modelos, no usando predicciones de unos modelos como input para otros. Para realizar este último tipo de ensamblado habría que construirlo manualmente siguiendo los pasos expuestos en el esquemita de "construción de ensamblados en R" anteriormente expuesto.

Kit con validación cruzada repetida y gráfico de cajas, dependiente continua (10 pasos)

1) Leer funciones "cruzadas" preparadas para ensamblado (hay pequeñas diferencias con las cruzadas anteriores)

2) Preparar el archivo, definir variables, semilla y repeticiones de CV

3) Obtener datos de CV repetida para cada algoritmo y procesar el resultado (para tener un vector de predicciones para cada algoritmo).

Previamente hay que haber estudiado bien cada algoritmo/modelo para tener los parámetros bien tuneados.

4) Construcción de ensamblados a partir del archivo unipredi de predicciones (hago promedios pero se pueden cambiar las ponderaciones)

5) Procesado de los ensamblados, hay que construir los promedios de errores por cada repetición de CV

6) Boxplot

7) Tabla ordenada para observar

8) Boxplot ordenado

9) Opcionalmente, selección de modelos a graficar

10) Observar en la lista de ensamblados a cuales corresponden los mejores

1) Leer funciones "cruzadas" preparadas para ensamblado (hay pequeñas diferencias con las cruzadas anteriores)

```
source("cruzadas ensamblado continuas fuente.R")
```

2) Preparar el archivo, definir variables, semilla y repeticiones de CV

```
load ("compressbien.Rda")
dput(names(compressbien))
set.seed(12345)

archivo<-compressbien

vardep<-"cstrength"
listconti<-c("cement", "blast", "age", "water")
listclass<-c("")
grupos<-4
sinicio<-1234
repe<-15
```

3) Obtener datos de CV repetida para cada algoritmo y procesar el resultado (para tener un vector de predicciones para cada algoritmo).

Previamente hay que haber estudiado bien cada algoritmo/modelo para tener los parámetros bien tuneados.

```
# APLICACIÓN CRUZADAS PARA ENSAMBLAR

medias1<-cruzadalin(data=archivo,
 vardep=vardep,listconti=listconti,
 listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe)
medias1bis<-as.data.frame(medias1[1])
medias1bis$modelo<-"regresion"
predi1<-as.data.frame(medias1[2])
predi1$reg<-predi1$pred
…
```

Como se ve, en esta nueva versión de las funciones se guardan las medias de errores en los archivos tipo medias1bis y las predicciones concretas bajo cv repetida en predi1.

medias1bis corresponde a los errores para el modelo de regresión

predi1 corresponde a las predicciones bajo cv repetida para regresión

…

medias2bis corresponde a los errores para el modelo avnnet

predi2 corresponde a las predicciones bajo cv repetida para avnnet

etc.

Como se puede ver en Rstudio en este ejemplo medias1bis tiene 15 observaciones ; es el error en cada una de las repeticiones de validación cruzada (en cada repetición se ordenan los datos aleatorizando mediante la semilla y después se hace validación cruzada y se obtiene el error).

Por otra parte, predi1 contiene 15450 observaciones. En el archivo hay 1030 observaciones. Hay 15 repeticiones de validación cruzada y en cada una de ellas se tiene la predicción de cada observación del archivo (tomada como datos test). Por lo tanto hay 15*1030=15450 predicciones. Estas se guardan para promediarlas después con las predicciones de otros algoritmos y así poder construir el ensamblado.

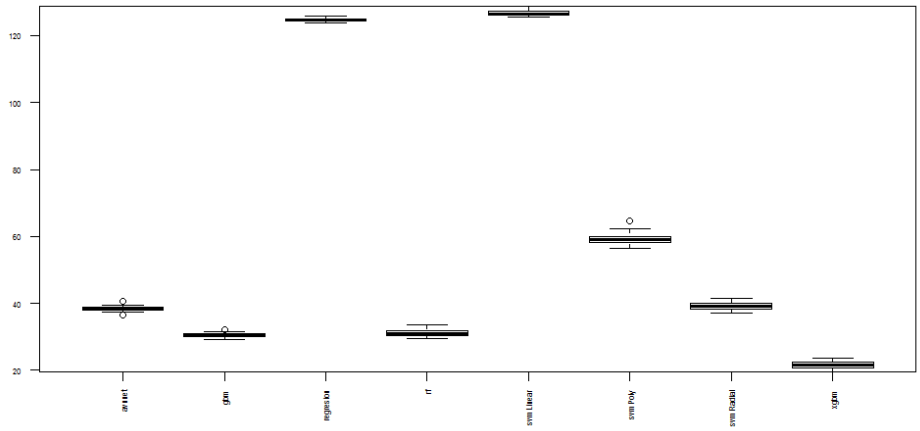
Además en predi1, predi2, etc. creo una columna con el nombre del algoritmo y la predicción (en predi1 es la columna reg).

Se omite el resto del código aquí…

…

En primer lugar se realiza el gráfico habitual comparando algoritmos.

```
union1<-rbind(medias1bis,medias2bis,
 medias3bis,medias4bis,medias5bis,medias6bis,
 medias7bis,medias8bis)

par(cex.axis=0.5)
boxplot(data=union1,error~modelo)
```



4) Construcción de ensamblados a partir del archivo unipredi de predicciones (hago promedios pero se pueden cambiar las ponderaciones)

En este momento se unen en paralelo los archivos de predicciones puntuales predi1, predi2,…, se borran columnas duplicadas y se utilizan las columnas de predicción de cada algoritmo, llamadas reg, avnnet, rf, etc. combinándolas para obtener ensamblados.

Por ejemplo en el archivo unipredi la variable predi9 son las predicciones bajo validación cruzada repetida resultantes de promediar la predicción de regresión con la de la red neuronal avnnet. (aparece en el código unipredi$reg+unipredi$avnnet)/2).

En el archivo unipredi, las predi1…predi8 son los algoritmos simples y de predi9 a predi69 son ensamblados.

Cuantos ensamblados se podrían construir? si hay k=8 algoritmos, se pueden construir 2^k-k-1 ensamblados, y k=8 modelos simples. En este caso: 2^8-8-1=247 ensamblados. 2^k es el sumatorio de los números combinatorios $\binom{8}{0}+\binom{8}{1}+\binom{8}{2}+\binom{8}{3}$…Los dos primeros términos son 1+8, que corersponden al modelo nulo y a los 8 algoritmos simples. El tercer término es el número de ensamblados con dos algoritmos, el cuarto con 3 algoritmos, etc.

Solo se han construido 69 modelos aquí pero se pueden añadir a la lista hasta predi256.

```
# CONSTRUCCIÓN DE TODOS LOS ENSAMBLADOS
# SE UTILIZARÁN LOS ARCHIVOS SURGIDOS DE LAS FUNCIONES LLAMADOS
predi1,...
unipredi<-cbind(predi1,predi2,predi3,predi4,predi5,predi6,predi7,
predi8)
# Esto es para eliminar columnas duplicadas

unipredi<- unipredi[, !duplicated(colnames(unipredi))]

# Construccion de ensamblados, cambiar al gusto

unipredi$predi9<-(unipredi$reg+unipredi$avnnet)/2
unipredi$predi10<-(unipredi$reg+unipredi$rf)/2
unipredi$predi11<-(unipredi$reg+unipredi$gbm)/2
unipredi$predi12<-(unipredi$reg+unipredi$xgbm)/2
unipredi$predi13<-(unipredi$reg+unipredi$svmLinear)/2
unipredi$predi14<-(unipredi$reg+unipredi$svmPoly)/2
unipredi$predi15<-(unipredi$reg+unipredi$svmRadial)/2
unipredi$predi16<-(unipredi$avnnet+unipredi$rf)/2
unipredi$predi17<-(unipredi$avnnet+unipredi$gbm)/2
```

```
unipredi$predi18<-(unipredi$avnnet+unipredi$xgbm)/2
unipredi$predi19<-(unipredi$avnnet+unipredi$svmLinear)/2
unipredi$predi20<-(unipredi$avnnet+unipredi$svmPoly)/2
unipredi$predi21<-(unipredi$avnnet+unipredi$svmRadial)/2
unipredi$predi22<-(unipredi$rf+unipredi$gbm)/2
unipredi$predi23<-(unipredi$rf+unipredi$xgbm)/2
unipredi$predi24<-(unipredi$rf+unipredi$svmLinear)/2
unipredi$predi25<-(unipredi$rf+unipredi$svmPoly)/2
unipredi$predi26<-(unipredi$rf+unipredi$svmRadial)/2
unipredi$predi27<-(unipredi$gbm+unipredi$xgbm)/2
unipredi$predi28<-(unipredi$gbm+unipredi$svmLinear)/2
unipredi$predi29<-(unipredi$gbm+unipredi$svmPoly)/2
unipredi$predi30<-(unipredi$gbm+unipredi$svmRadial)/2
unipredi$predi31<-(unipredi$reg+unipredi$avnnet+unipredi$rf)/3
unipredi$predi32<-(unipredi$reg+unipredi$avnnet+unipredi$gbm)/3
unipredi$predi33<-(unipredi$reg+unipredi$avnnet+unipredi$xgbm)/3
unipredi$predi34<-(unipredi$reg+unipredi$avnnet+unipredi$svmLinear)/3
unipredi$predi35<-(unipredi$reg+unipredi$avnnet+unipredi$svmPoly)/3
unipredi$predi36<-(unipredi$reg+unipredi$avnnet+unipredi$svmRadial)/3
unipredi$predi37<-(unipredi$reg+unipredi$rf+unipredi$gbm)/3
unipredi$predi38<-(unipredi$reg+unipredi$rf+unipredi$xgbm)/3
unipredi$predi39<-(unipredi$reg+unipredi$rf+unipredi$svmLinear)/3
unipredi$predi40<-(unipredi$reg+unipredi$rf+unipredi$svmPoly)/3
unipredi$predi41<-(unipredi$reg+unipredi$rf+unipredi$svmRadial)/3
unipredi$predi42<-(unipredi$reg+unipredi$gbm+unipredi$xgbm)/3
unipredi$predi43<-(unipredi$reg+unipredi$gbm+unipredi$xgbm)/3
unipredi$predi44<-(unipredi$reg+unipredi$gbm+unipredi$svmLinear)/3
unipredi$predi45<-(unipredi$reg+unipredi$gbm+unipredi$svmPoly)/3
unipredi$predi46<-(unipredi$reg+unipredi$gbm+unipredi$svmRadial)/3
unipredi$predi47<-(unipredi$reg+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$predi48<-(unipredi$reg+unipredi$xgbm+unipredi$svmPoly)/3
unipredi$predi49<-(unipredi$reg+unipredi$xgbm+unipredi$svmRadial)/3
unipredi$predi50<-(unipredi$rf+unipredi$gbm+unipredi$svmLinear)/3
unipredi$predi51<-(unipredi$rf+unipredi$gbm+unipredi$svmPoly)/3
unipredi$predi52<-(unipredi$rf+unipredi$gbm+unipredi$svmRadial)/3
unipredi$predi53<-(unipredi$rf+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$predi54<-(unipredi$rf+unipredi$xgbm+unipredi$svmPoly)/3
unipredi$predi55<-(unipredi$rf+unipredi$xgbm+unipredi$svmRadial)/3

unipredi$predi56<-(unipredi$rf+unipredi$avnnet+unipredi$gbm)/3
unipredi$predi57<-(unipredi$rf+unipredi$avnnet+unipredi$xgbm)/3
unipredi$predi58<-(unipredi$rf+unipredi$avnnet+unipredi$svmLinear)/3
unipredi$predi59<-(unipredi$rf+unipredi$avnnet+unipredi$svmPoly)/3
unipredi$predi60<-(unipredi$rf+unipredi$avnnet+unipredi$svmRadial)/3

unipredi$predi61<-(unipredi$avnnet+unipredi$gbm+unipredi$svmLinear)/3
unipredi$predi62<-(unipredi$avnnet+unipredi$gbm+unipredi$svmPoly)/3
unipredi$predi63<-(unipredi$avnnet+unipredi$gbm+unipredi$svmRadial)/3

unipredi$predi64<-
(unipredi$reg+unipredi$rf+unipredi$gbm+unipredi$avnnet)/4
unipredi$predi65<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet)/4
unipredi$predi66<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet)/4

unipredi$predi67<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmLi
near)/5
```

```
unipredi$predi68<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmPo
ly)/5
unipredi$predi69<-
(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmRa
dial)/5
```

5) Procesado de los ensamblados, hay que construir los promedios de errores por cada repetición de CV

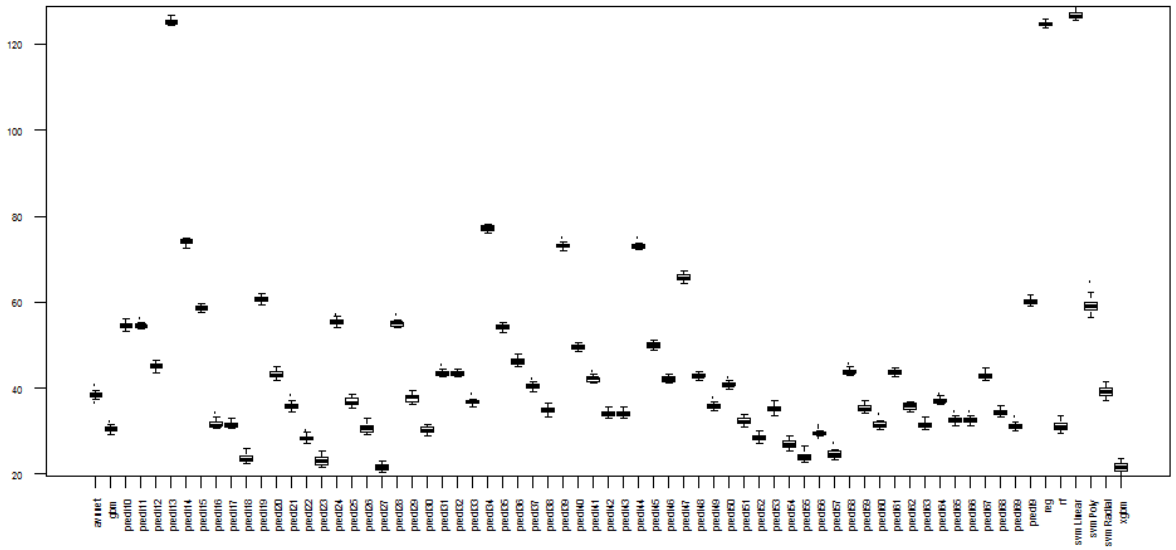
En esta parte nos quedamos solamente con las predicciones de los diferentes modelos y se calcula el error para cada predicción y se promedia en su repetición de validación cruzada.

```
dput(names(unipredi))

# Recorto los modelos de la lista de variables
listado<-c("reg", "avnnet",
"rf", "gbm", "xgbm", "svmLinear", "svmPoly", "svmRadial", "predi9",
"predi10", "predi11", "predi12", "predi13", "predi14", "predi15",
"predi16", "predi17", "predi18", "predi19", "predi20", "predi21",
"predi22", "predi23", "predi24", "predi25", "predi26", "predi27",
"predi28", "predi29", "predi30", "predi31", "predi32", "predi33",
"predi34", "predi35", "predi36", "predi37", "predi38", "predi39",
"predi40", "predi41", "predi42", "predi43", "predi44", "predi45",
"predi46", "predi47", "predi48", "predi49", "predi50", "predi51",
"predi52", "predi53", "predi54", "predi55", "predi56", "predi57",
"predi58", "predi59", "predi60", "predi61", "predi62", "predi63",
"predi64", "predi65", "predi66", "predi67", "predi68", "predi69"
)
repeticiones<-nlevels(factor(unipredi$Rep))
unipredi$Rep<-as.factor(unipredi$Rep)
unipredi$Rep<-as.numeric(unipredi$Rep)
# Calculo el MSE para cada repeticion de validación cruzada
medias0<-data.frame(c())
for (prediccion in listado)
{
paso <-unipredi[,c("obs",prediccion,"Rep")]
paso$error<-(paso[,c(prediccion)]-paso$obs)^2
paso<-paso %>%
  group_by(Rep) %>%
  summarize(error=mean(error))
paso$modelo<-prediccion
medias0<-rbind(medias0,paso)
}
```

6) Boxplot

```
par(cex.axis=0.5,las=2)
boxplot(data=medias0,outcex=0.4,error~modelo)
```

Obviamente el boxplot no está ordenado de mejor modelo a peor, se hará después.



7) Tabla ordenada para observar

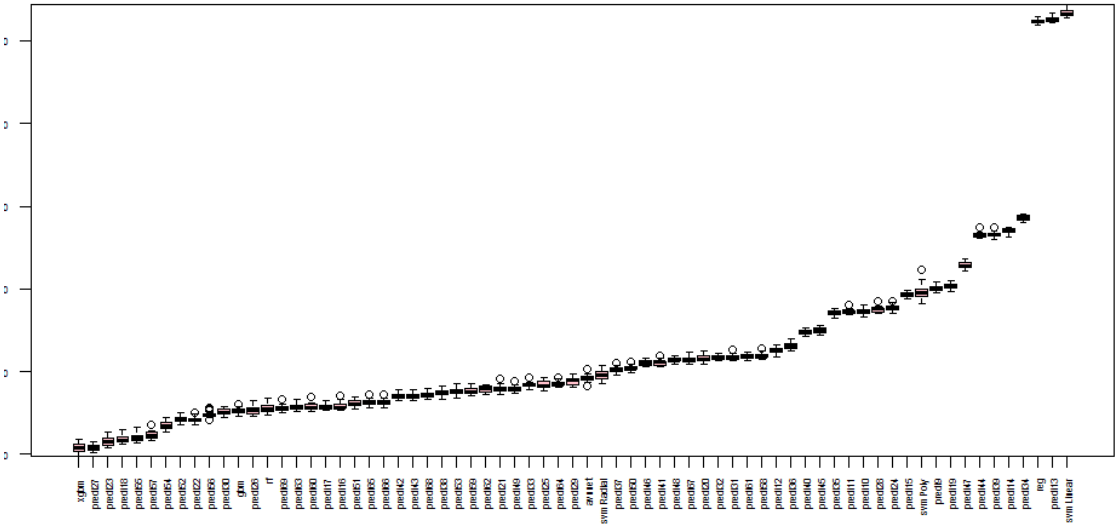
En esta parte se construye una tabla para ordenar modelos de menor error a mayor, en el dataframe tablamedias.

```
# PRESENTACION TABLA MEDIAS

tablamedias<-medias0 %>%
  group_by(modelo) %>%
  summarize(error=mean(error))

tablamedias<-tablamedias[order(tablamedias$error),]
```

	modelo	error
1	xgbm	21.68392
2	predi27	21.76288
3	predi23	23.02302
4	predi18	23.64593
5	predi55	24.03186
6	predi57	24.70089
7	predi54	26.93216
8	predi52	28.48733
9	predi22	28.49261
10	predi56	29.69959
11	predi30	30.22862
12	gbm	30.49247
13	predi26	30.59308
14	rf	31.21659
15	predi69	31.26763

8) Boxplot ordenado

```
# ORDENACIÓN DEL FACTOR MODELO POR LAS MEDIAS EN ERROR
# PARA EL GRÁFICO

medias0$modelo <- with(medias0,
 reorder(modelo,error, mean))
par(cex.axis=0.5,las=2)
boxplot(data=medias0,outcex=0.4,error~modelo,col="pink")
```

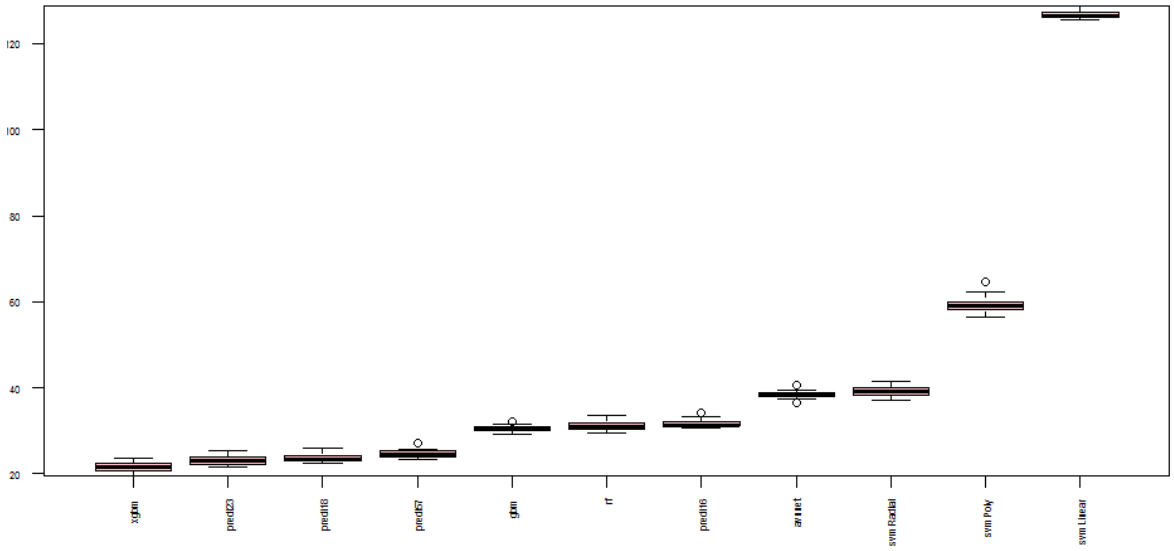


En este gráfico están ordenados los modelos por la media de su error (se puede poner median en lugar de mean, quizá sea mejor).

9) Opcionalmente, selección de modelos a graficar

En el gráfico nos interesa ver los mejorers modelos y los algoritmos simples, seleccionamos solo esos modelos mediante una lista.

```
# Se pueden escoger listas pero el factor hay que pasarlo a character
# para que no salgan en el boxplot todos los niveles del factor
listadobis<-c("regresion", "avnnet",
"rf","gbm",  "xgbm", "svmLinear",  "svmPoly",
"svmRadial","predi23","predi57", "predi18", "predi16")

medias0$modelo<-as.character(medias0$modelo)
mediasver<-medias0[medias0$modelo %in% listadobis,]

mediasver$modelo <- with(mediasver,
 reorder(modelo,error, mean))
par(cex.axis=0.5,las=2)
boxplot(data=mediasver,error~modelo,col="pink")
```



En este caso está claro que xgbm es un algoritmo dominante y no merece la pena el ensamblado.

10) Observar en la lista de ensamblados a cuales corresponden los mejores:

predi23,predi57,predi18:

```
unipredi$predi23<-(unipredi$rf+unipredi$xgbm)/2

unipredi$predi18<-(unipredi$avnnet+unipredi$xgbm)/2

unipredi$predi57<-(unipredi$rf+unipredi$avnnet+unipredi$xgbm)/3
```

11) Conclusiones (en este caso no merece la pena ensamblar).

Ante la duda, no hacerlo.

Kit con validación cruzada repetida y gráfico de cajas, dependiente binaria (10 pasos)

Archivo ejemplo cruzadas para ensamblado bin.R

En este caso el procesado de los modelos incluirá aplicar el punto de corte en las probabilidades y calcular Accuracy a través de la matriz de confusión.

```
# PRUEBAS DE ENSAMBLADO
# ************************************
# IMPORTANTE: AQUÍ HAY QUE DECIDIR ANTES LOS PARÁMETROS A UTILIZAR
# EN CADA ALGORITMO, NO VALE GRID
# Importante, la dependiente en letras Yes, No
# Preparación de archivo, variables y CV.
# Esto se cambia para cada archivo.
# Necesario haber cambiado la var dep a Yes,No.
# ************************************
```

1) Leer funciones "cruzadas" preparadas para ensamblado (hay pequeñas diferencias con las cruzadas anteriores)

```
# LEER LAS CRUZADAS DE ENSAMBLADO, SON LIGERAMENTE DIFERENTES
# A LAS UTILIZADAS ANTERIORMENTE AUNQUE SE LLAMAN IGUAL
source("cruzadas ensamblado binaria fuente.R")
```

2) Preparar el archivo, definir variables, semilla y repeticiones de CV

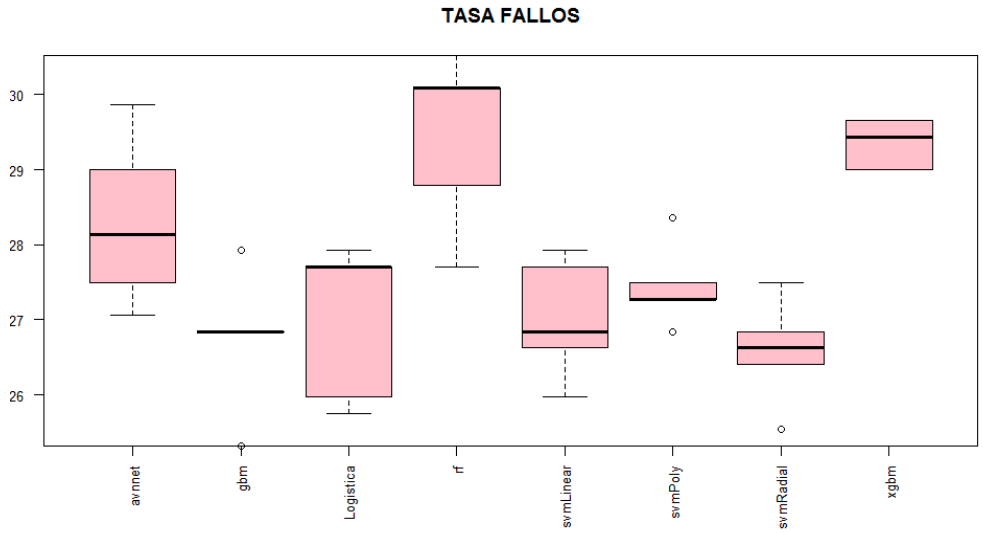
```
load ("saheartbis.Rda")
dput(names(saheartbis))
set.seed(12345)
archivo<-saheartbis
vardep<-"chd"
listconti<-c("sbp", "tobacco",
  "ldl","age", "typea","famhist.Absent")
listclass<-c("")
grupos<-4
sinicio<-1234
repe<-15
```

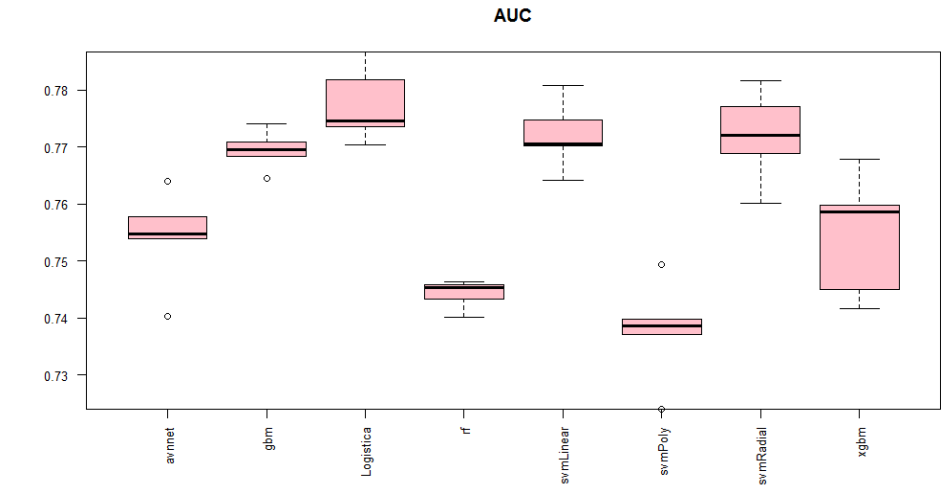
3) Obtener datos de CV repetida para cada algoritmo y procesar el resultado (para tener un vector de predicciones para cada algoritmo).

Previamente hay que haber estudiado bien cada algoritmo/modelo para tener los parámetros bien tuneados.

```
# APLICACIÓN CRUZADAS PARA ENSAMBLAR

medias1<-cruzadalogistica(data=archivo,
 vardep=vardep,listconti=listconti,
 listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe)
medias1bis<-as.data.frame(medias1[1])
medias1bis$modelo<-"Logistica"
predi1<-as.data.frame(medias1[2])
predi1$logi<-predi1$Yes


Se omite el resto del código aquí…

…
```

```
union1<-rbind(medias1bis,medias2bis,medias3bis,medias4bis,medias5bis,
medias6bis,medias7bis,medias8bis)

par(cex.axis=0.5)
boxplot(data=union1,tasa~modelo)
boxplot(data=union1,auc~modelo)
```





4) Construcción de ensamblados a partir del archivo unipredi de predicciones (hago promedios pero se pueden cambiar las ponderaciones)

```
# CONSTRUCCIÓN DE TODOS LOS ENSAMBLADOS

# SE UTILIZARÁN LOS ARCHIVOS SURGIDOS DE LAS FUNCIONES LLAMADOS predi1,...

unipredi<-
cbind(predi1,predi2,predi3,predi4,predi5,predi6,predi7,predi8)

# Esto es para eliminar columnas duplicadas

unipredi<- unipredi[, !duplicated(colnames(unipredi))]

# Construccion de ensamblados, cambiar al gusto
```

Aquí se crean los ensamblados, donde se promedian las probabilidades predichas obtenidas de cada algoritmo.

```
unipredi$predi9<-(unipredi$logi+unipredi$avnnet)/2

…

Omitimos el código aquí

…
```

5) Procesado de los ensamblados, hay que construir los promedios de **tasa de fallos** y **AUC** por cada repetición de CV. Para ello se aplica el punto de corte, calcula tasa de fallos y se promedia por repeticiones de CV.

```
# Listado de modelos a considerar, cambiar al gusto

dput(names(unipredi))

listado<-c("logi", "avnnet",
"rf","gbm",  "xgbm", "svmLinear",  "svmPoly", "svmRadial","predi9",
"predi10", "predi11", "predi12",
"predi13", "predi14", "predi15", "predi16", "predi17", "predi18",
"predi19", "predi20", "predi21", "predi22", "predi23", "predi24",
"predi25", "predi26", "predi27", "predi28", "predi29", "predi30",
"predi31", "predi32", "predi33", "predi34", "predi35", "predi36",
"predi37", "predi38", "predi39", "predi40", "predi41", "predi42",
"predi43", "predi44", "predi45", "predi46", "predi47", "predi48",
"predi49", "predi50", "predi51", "predi52", "predi53", "predi54",
"predi55", "predi56", "predi57", "predi58", "predi59", "predi60",
"predi61", "predi62", "predi63", "predi64", "predi65", "predi66",
"predi67", "predi68", "predi69")
# Cambio a Yes, No, todas las predicciones
for (prediccion in listado)
{
unipredi[,prediccion]<-ifelse(unipredi[,prediccion]>0.5,"Yes","No")
}
# Defino funcion tasafallos
tasafallos<-function(x,y) {
   confu<-confusionMatrix(x,y)
   tasa<-confu[[3]][1]
   return(tasa)
  }
```

```
# Se obtiene el numero de repeticiones CV y se calculan las medias por
repe en
# el data frame medias0
repeticiones<-nlevels(factor(unipredi$Rep))
unipredi$Rep<-as.factor(unipredi$Rep)
unipredi$Rep<-as.numeric(unipredi$Rep)
medias0<-data.frame(c())
for (prediccion in listado)
{
for (repe in 1:repeticiones)
{
 paso <- unipredi[(unipredi$Rep==repe),]
 pre<-factor(paso[,prediccion])
 obs<-paso[,c("obs")]
 tasa=1-tasafallos(pre,obs)
 t<-as.data.frame(tasa)
 t$modelo<-prediccion
 medias0<-rbind(medias0,t)
}
}
```

6) Boxplot

```
# Finalmente boxplot

par(cex.axis=0.5,las=2)
boxplot(data=medias0,tasa~modelo,col="pink",main="TASA FALLOS")

# Para AUC se utiliza la variable auc del archivo medias0
boxplot(data=medias0,auc~modelo,col="pink",main="AUC")
```

No se pone aquí, directamente lo ponemos después de ordenar.

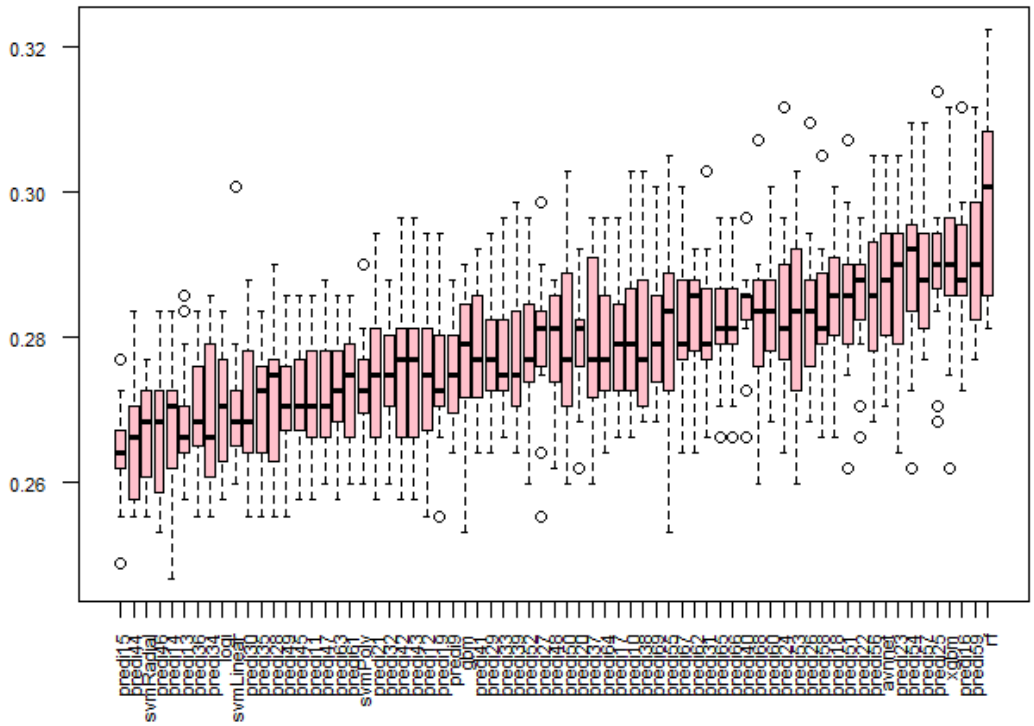
7) Tabla ordenada para observar

```
# PRESENTACION TABLA MEDIAS

tablamedias<-medias0 %>%
  group_by(modelo) %>%
  summarize(tasa=mean(tasa))

tablamedias<-tablamedias[order(tablamedias$tasa),]
```

	modelo	tasa
1	predi15	0.2627706
2	predi28	0.2627706
3	predi30	0.2645022
4	predi46	0.2649351
5	predi36	0.2653680
6	predi44	0.2653680
7	predi34	0.2658009
8	svmRadial	0.2658009
9	predi11	0.2666667
10	predi61	0.2670996
11	predi19	0.2675325
12	gbm	0.2675325
13	predi13	0.2679654
14	predi45	0.2683983
15	predi21	0.2688312
16	predi32	0.2701299
17	logi	0.2701299

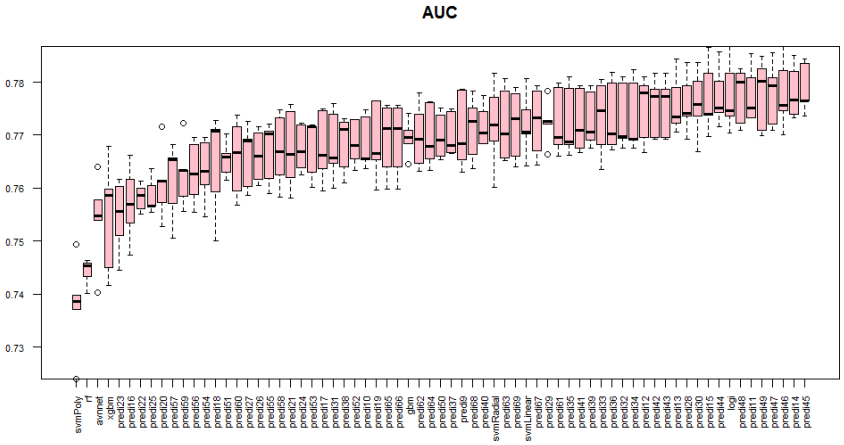
8) Boxplot ordenado

```
# ORDENACIÓN DEL FACTOR MODELO POR LAS MEDIAS EN TASA
# PARA EL GRAFICO

medias0$modelo <- with(medias0,
 reorder(modelo,tasa, mean))
par(cex.axis=0.5,las=2)
boxplot(data=medias0,tasa~modelo,col="pink")
```



```
# ************************************
# PARA AUC
# ************************************

# PRESENTACION TABLA MEDIAS

tablamedias2<-medias0 %>%
  group_by(modelo) %>%
  summarize(auc=mean(auc))

tablamedias2<-tablamedias2[order(-tablamedias2$auc),]
```

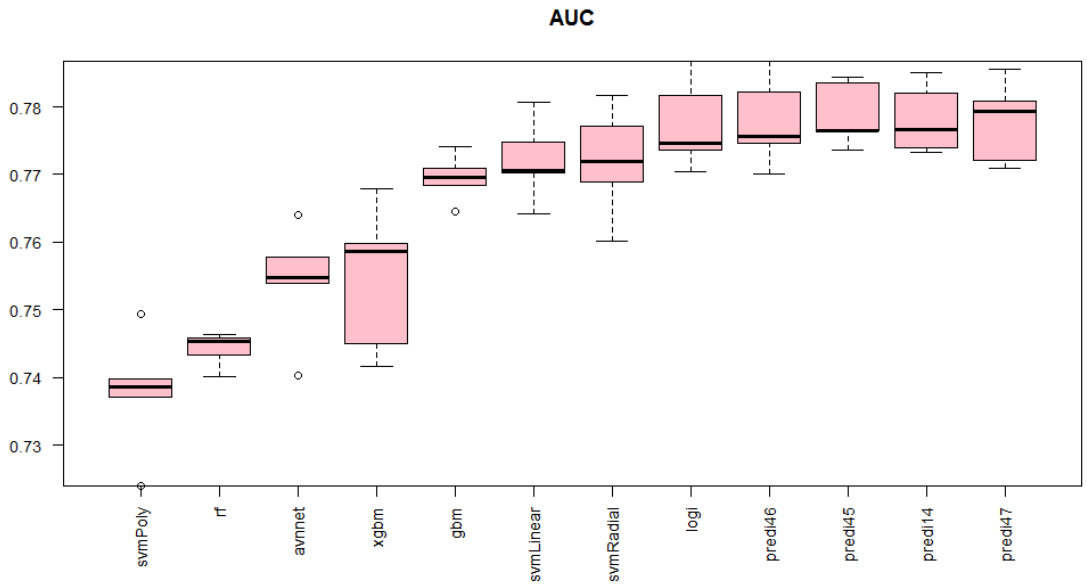
	modelo	auc
1	predi45	0.7788907
2	predi14	0.7781705
3	predi46	0.7778725
4	predi47	0.7777690
5	predi49	0.7776945
6	predi11	0.7775911
7	predi48	0.7774834
8	logi	0.7774172
9	predi44	0.7773924
10	predi15	0.7771896
11	predi30	0.7760224
12	predi28	0.7759934




```
# ORDENACIÓN DEL FACTOR MODELO POR LAS MEDIAS EN AUC
# PARA EL GRAFICO

medias0$modelo <- with(medias0,
 reorder(modelo,auc, mean))
par(cex.axis=0.7,las=2)
boxplot(data=medias0,auc~modelo,col="pink")
```

9) Opcionalmente, selección de modelos a graficar

```
# Se pueden escoger listas pero el factor hay que pasarlo a character
# para que no salgan en el boxplot todos los niveles del factor

listadobis<-c("logi", "avnnet",
"rf","gbm",  "xgbm", "svmLinear",  "svmPoly",
"svmRadial","predi45", "predi14", "predi46", "predi47")

medias0$modelo<-as.character(medias0$modelo)

mediasver<-medias0[medias0$modelo %in% listadobis,]


mediasver$modelo <- with(mediasver,
 reorder(modelo,auc, median))

par(cex.axis=0.5,las=2)
boxplot(data=mediasver,auc~modelo,col="pink",main='AUC')
```



10) Observar los mejores ensamblados

```
unipredi$predi47<-(unipredi$logi+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$predi14<-(unipredi$logi+unipredi$svmPoly)/2
unipredi$predi45<-(unipredi$logi+unipredi$gbm+unipredi$svmPoly)/3
unipredi$predi46<-(unipredi$logi+unipredi$gbm+unipredi$svmRadial)/3
```

11) Conclusiones

1) Los algoritmos simples que mejor funcionan son la regresión logística. SVM Radial y SVM Lineal.

2) Aparentemente ensamblar la logística, xgbm y svmLinear es el ensamblado con mejor AUC.

3) En este caso la diferencia del ensamblado con la regresión logística es pequeña y se preferiría por lo tanto la regresión logística.

4) En estos datos la tasa de fallos es algo incoherente con el AUC, siendo el SVMRadial el mejor algoritmo simple y gbm combinado con SVM Lineal el mejor ensamblado. Preferiremos el AUC como medida de diagnóstico por estas incoherencias.

Ejemplo de gráficos de apoyo para comparar resultados

Se puede utilizar la información de los arcvhivos anteriores para observar algunos detalles:

1) La correlación entre predicciones de los diferentes algoritmos

2) Ver cómo se comportan los algoritmos dos a dos en sus predicciones mediante gráficos.

En primer lugar se construye un archivo con predicciones de los modelos que nos interesan.

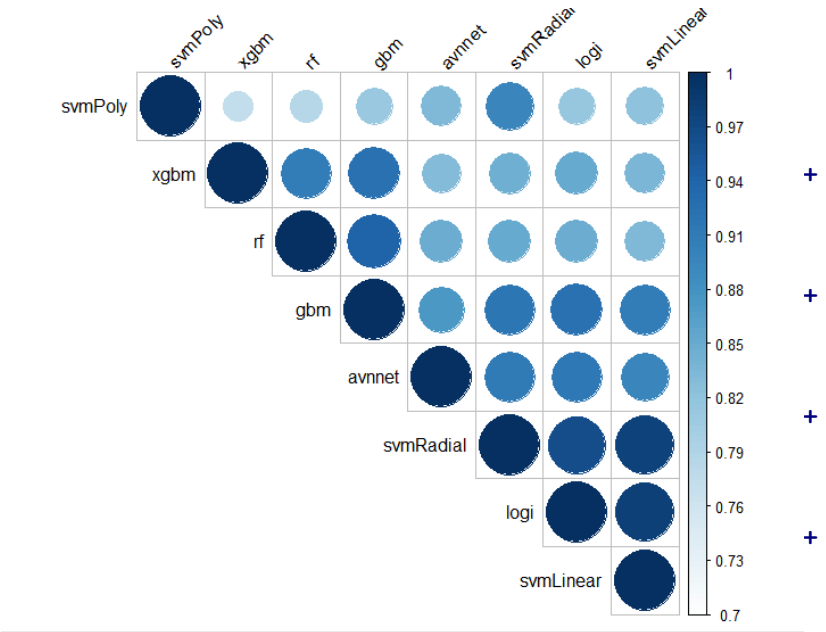
```
unipredi<-
cbind(predi1,predi2,predi3,predi4,predi5,predi6,predi7,predi8)

# Esto es para eliminar columnas duplicadas

unipredi<- unipredi[, !duplicated(colnames(unipredi))]

# Añadir ensamblados

unipredi$predi47<-(unipredi$logi+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$predi14<-(unipredi$logi+unipredi$svmPoly)/2
unipredi$predi45<-(unipredi$logi+unipredi$gbm+unipredi$svmPoly)/3
unipredi$predi46<-(unipredi$logi+unipredi$gbm+unipredi$svmRadial)/3
```

Me quedo con solo una repetición de validación cruzada para ver gráficos de puntos.

Aquí es necesario ver antes cómo están nombradas las Repeticiones de validación cruzada en el archivo unipredi, si son muchas las nombra como Rep001, etc. En este caso son Rep1 a Rep5 porque hay menos de 10.

```
# Me quedo con la primera repetición de validación cruzada para los
análisis
unigraf<-unipredi[unipredi$Rep=="Rep1",]
```

```
# Correlaciones entre predicciones de cada algoritmo individual

solos<-c("logi", "avnnet",
"rf","gbm",  "xgbm", "svmLinear",  "svmPoly",
"svmRadial")

mat<-unigraf[,solos]
matrizcorr<-cor(mat)
matrizcorr
library(corrplot)
corrplot(matrizcorr, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45,cl.lim=c(0.7,1),is.corr=FALSE)

library(ggplot2)
```



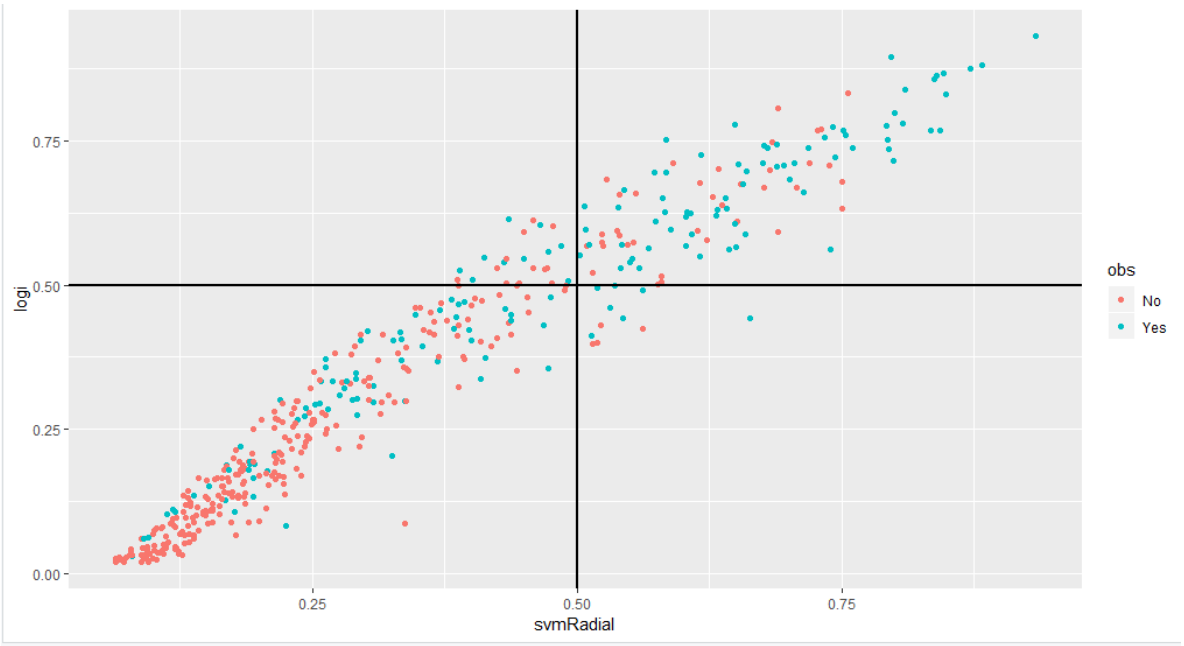
Hay que notar que s ha puesto cl.lim=c(0.7,1) para que el gráfico se vea mejor, pues las correlaciones entre algoritmos siempre son altas (>0.7 en este caso).

Se observa cómo las diferentes familias tienen alta correlación entre sí (los SVM, los algoritmos basados en árboles..). La logística tiene mayor correlación con svmLineal (por supuesto) y SVMRadial.

También se pueden plotear dos a dos cómo se comportan los algoritmos en una repetición de validación cruzada en los siguientes gráficos.

```
library(ggplot2)

qplot(svmRadial,logi,data=unigraf,colour=obs)+
 geom_hline(yintercept=0.5, color="black", size=1)+
 geom_vline(xintercept=0.5, color="black", size=1)
```

Aquí comparamos el comprotamiento de los dos mejores algoritmos simples enfrentados, logística y svmRadial.



En el gráfico se plotean los puntos con su color según sean realmente No (rojo) o Yes (azul).

En los ejes están las probabilidades predichas por cada algoritmo para cada punto. Los puntos con probabilidad predicha alta (>0.50 si usamos ese punto de corte) deberían ser azules. Por ejemplo, según el algoritmo svmRadial los puntos a la derecha de la recta vertical (0.50) deberían ser todos azules (falla en los que son rojos). Según la logística, los puntos por encima de la recta horizontal (0.50) deberían ser azules.

Los dos algoritmos coinciden en las cuadrantes izquierda-abajo y derecha-arriba, y discrepan en los cuadrantes izquierda-arriba y derecha-abajo. Se observa que en las discrepancias lo hace mejor svmRadial y por ello es mejor que la regresión logística en tasa de fallos (esta última identifica como rojos muchos puntos azules en el cuadrante inferior derecho) .

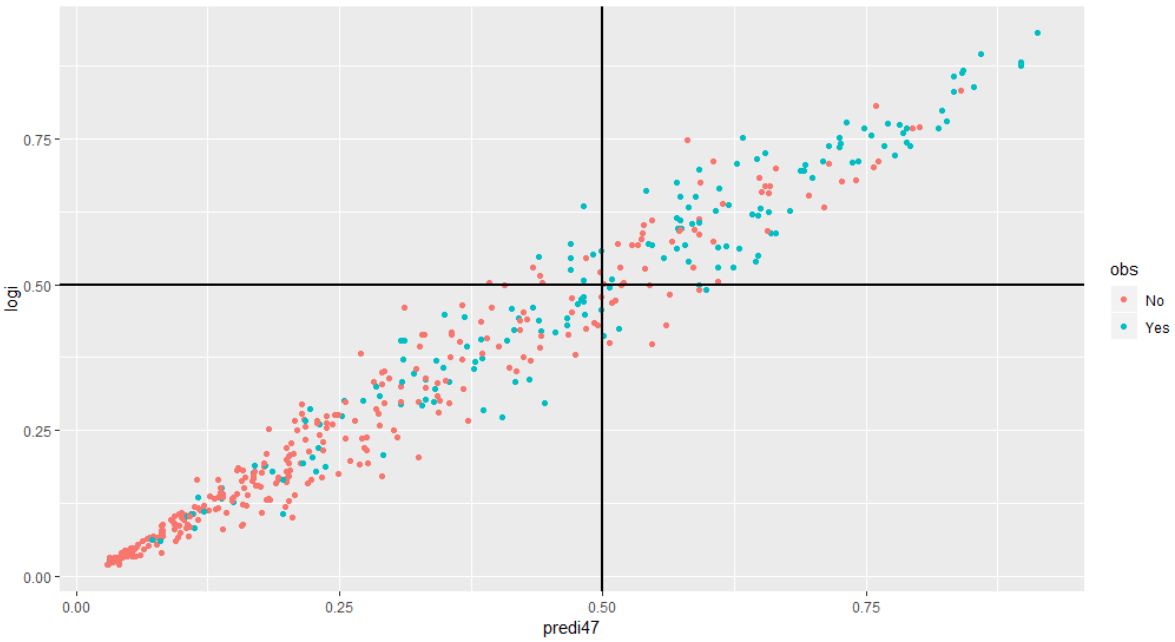
Pero en términos de AUC no es así (es mejor la logística en estos datos), aunque esto no se puede ver tan fácilmente en el gráfico (visualmente habría que desplazar el punto de corte 0.5 en ambos ejes e ir viendo el comportamiento de ambos algoritmos).

También se observa la correlación entre ambos algoritmos, si los puntos están juntos en torno a la diagonal es que hay alta correlación como se verá en un gráfico después.

El siguiente plot compara logi con el mejor ensamblado (en términos de AUC), que es predi47.

En realidad según el boxplot de tasa de fallos logi es mejor que predi47 en tasa de fallos y ligeramente peor en términos de AUC.

```
qplot(predi47,logi,data=unigraf,colour=obs)+
 geom_hline(yintercept=0.5, color="black", size=1)+
 geom_vline(xintercept=0.5, color="black", size=1)
```



Se observa cómo en las discrepancias lo hace mejor logi y falla más predi47 lo que ejemplifica el comportamiento en tasa de fallos que habíamos visto en los diagramas de cajas.

El siguiente gráfico plotea dos algoritmos que funcionan diferentemente (baja correlación).

```
qplot(gbm,svmPoly,data=unigraf,colour=obs)+
 geom_hline(yintercept=0.5, color="black", size=1)+
 geom_vline(xintercept=0.5, color="black", size=1)
```

Bibliografía básica

Libros disponibles en PDF

Hastie, Tibshirani: The Elements of Statistical Learning (PDF)

(En la web hay más información)

http://statweb.stanford.edu/~tibs/ElemStatLearn/

Hastie, Tibshirani: An Introduction to Statistical Learning with Applications in R (PDF)

(básicamente el mismo que el anterior, pero para R)

http://www-bcf.usc.edu/~gareth/ISL/data.html/

Algunas páginas ilustrativas

http://mlwave.com/kaggle-ensembling-guide/

http://www.overkillanalytics.net/more-is-always-better-the-power-of-simple-ensembles/

Paquetes R

https://cran.r-project.org/web/packages/caretEnsemble/index.html