Métodos de ensamblado

Métodos de ensamblado

Los métodos de ensamblado o 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. Es un ensamblado (averaging) de 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. Es un ensamblado (stacking) de y1,y2,y3.

ETC.

Las opciones de combinación son enormes. Pero…¿funciona?

Diferentes algoritmos y técnicas dan lugar a diferentes superficies de predicción.

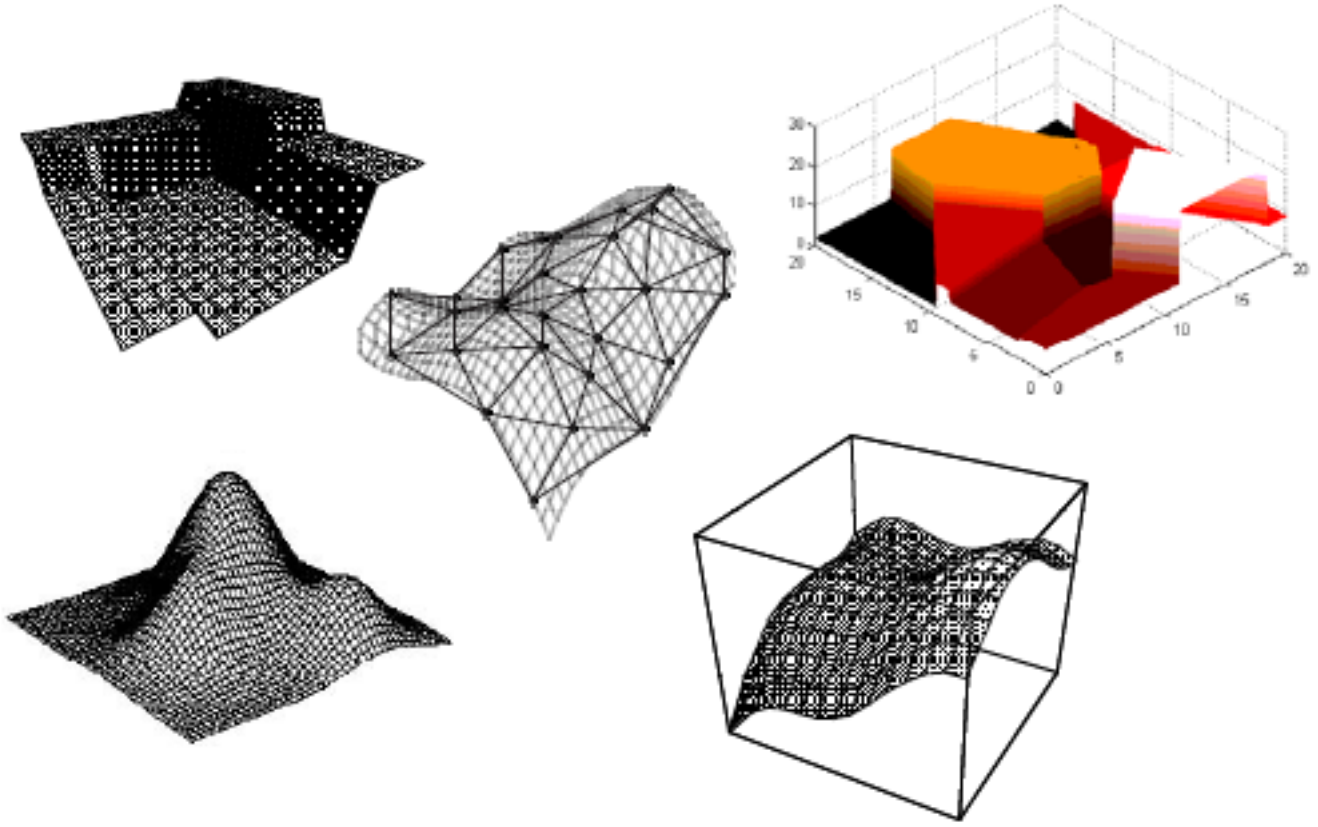


Figure 1.3: Example estimation surfaces for five modeling algorithms. Clockwise from top left: decision tree, Delaunay planes (based on Elder, J. (1993)), nearest neighbor, polynomial network (or neural network), kernel.

Una Aplicación Ilustrativa: The Netflix Prize, 2009

Objetivo: predecir la puntuación que los usuarios dan a una película
(premio: 1.000.000 $)

El premio gordo se otorgaba a los equipos capaces de batir en un 10% sobre datos test, el algoritmo base de Cinematch, que daba, en 2006, un RMSE=0.95.

Es decir, habría que obtener un error de como máximo RMSE=0.85.

Desde el lanzamiento del concurso en 2006, nadie lo consiguió…

…hasta el año 2009

“

It is our great honor to announce the $1M Grand Prize winner of the Netflix Prize contest as team BellKor’s Pragmatic Chaos for their verified submission on July 26, 2009 at 18:18:28 UTC, achieving the winning RMSE of 0.8567 on the test subset.

Algunas ideas sobre la solución

1) Los equipos ganadores BellKor y Big Chaos, participaron en el concurso desde 2006. Los modelos se fueron refinando año tras año.

2) Los primeros modelos se abordan de modo constructivo: un predictor lineal sencillo…

Denote by μ the overall average rating. A baseline prediction for an unknown rating r_{ui} is denoted by b_{ui} and accounts for the user and item effects:

$$b_{ui} = \mu + b_u + b_i \quad (1)$$

The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. For example, suppose that we want a baseline estimate for the rating of the movie Titanic by user Joe. Now, say that the average rating over all movies, μ, is 3.7 stars. Furthermore, Titanic is better than an average movie, so it tends to be rated 0.5 stars above the average. On the other hand, Joe is a critical user, who tends to rate 0.3 stars lower than the average. Thus, the baseline estimate for Titanic's rating by Joe would be 3.9 stars by calculating $3.7 - 0.3 + 0.5$.

3) Se añade el efecto temporal… $b_{ui} = \mu + b_u(t_{ui}) + b_i(t_{ui})$

4) Se incorporan efectos como la frecuencia de votaciones que el usuario hace ese día y la distancia entre la fecha de evaluación y de creación de la película…

$$b_{ui} = \mu + b_u + \alpha_u \cdot \mathrm{dev}_u(t_{ui}) + b_{u,t_{ui}} + (b_i + b_{i,\mathrm{Bin}(t_{ui})}) \cdot c_u(t_{ui}) + b_{i,f_{ui}}. \quad (11)$$

Al añadir las frecuencias el RMSE=0.9555 pasa a 0.9278 (mejor que Cinematch)

5) Primera utilización de combinación de modelos: se incorpora el predictor anterior a una fórmula que tiene en cuenta factores estáticos relativos a las películas y a las películas puntuadas por el usuario, además de considerar un factor dinámico de las puntuaciones de películas por cada usuario:

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(p_u(t_{ui}) + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$

Estamos ya en RMSE=0.8762

6) Combinación de modelos: Se utiliza Gradient Boosting sobre:

- Conjunto de 474 variables: RMSE=0.8603
- Conjunto de 75 variables: RMSE=0.8606 (¡¡el increíble gradient boosting!!)
- Conjunto de 24 variables que son a su vez predicciones obtenidas con otros modelos (incluidos los explicados anteriormente):

Post Progress Prize 2008 predictors

Those were mentioned earlier in this document:

1) PQ1
2) PQ2
3) PQ3
4) PQ4
5) PQ5
6) PQ6
7) PQ7

Progress Prize 2008 predictors

The following is based on our notation in [3]:

8) SVD++$^{(1)}$ ($f = 200$)
9) Integrated ($f = 100$, $k = 300$)
10) SVD++$^{(3)}$ ($f = 500$)
11) First neighborhood model of Sec. 2.2 of [3] (RMSE=0.9002)
12) A neighborhood model mentioned towards the end of Sec. 2.2 of [3] (RMSE=0.8914)

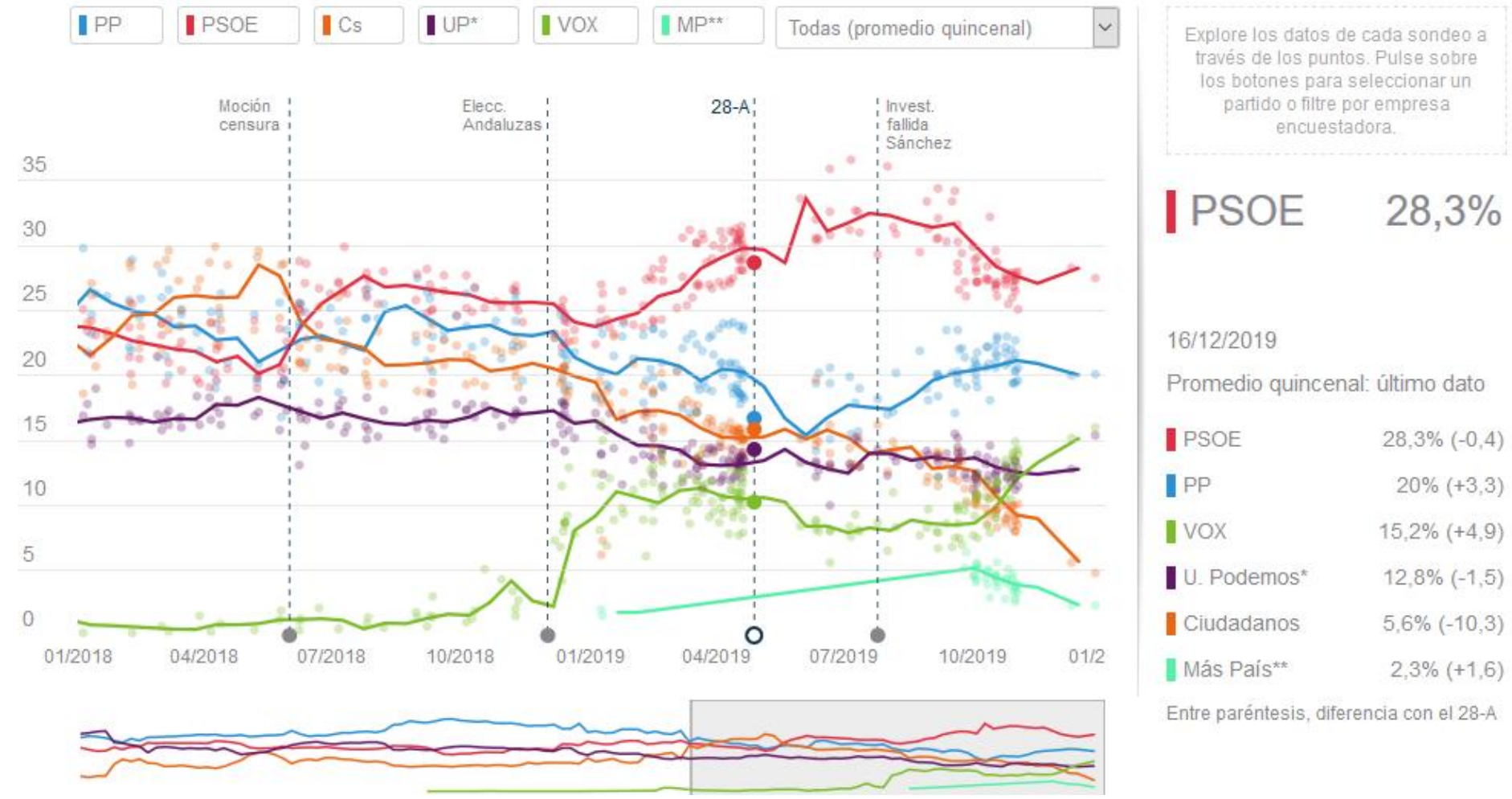
Progress Prize 2007 predictors

The following is based on our notation in [2]:

13) Predictor #40
14) Predictor #35
15) Predictor #67
16) Predictor #75
17) NNMF (60 factors) with adaptive user factors
18) Predictor #81
19) Predictor #73
20) 100 neighbors User-kNN on residuals of all global effects but the last 4
21) Predictor #85
22) Predictor #45

Finalmente se gana el concurso con un RMSE=0.8567

Últimamente se ha hecho habitual en la prensa presentar los **ensamblados** por promedio de diferentes sondeos o encuestas electorales.



Técnicas básicas de combinado de modelos

1) Bagging

2) Boosting

3) Stacking

4) Otros (Bayesian Model Averaging)

Bagging

=Bootstrap **A**vera**ging**

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

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

Stacking (blending, 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 (Tumer-Ghosh y otros**)**

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

β =valor 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

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

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

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

Casuística

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

β =valor 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

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

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

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

1) $\bar{\beta} \leq \beta, \delta < 1$ El error del ensamblado es menor que de un simple clasificador.

Manteniendo el sesgo igual o menor, cuanto **menor** sea la **correlación** entre clasificadores **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 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).

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 relativamente bajas 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).

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 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.

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)

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

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

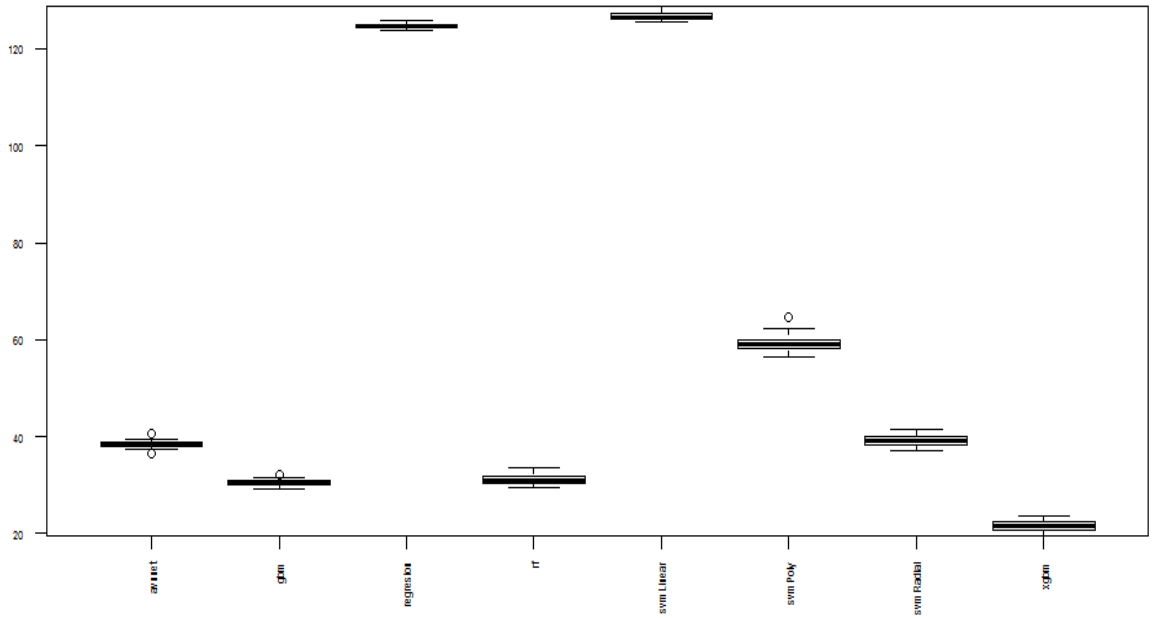
```
load ("compressbien.Rda")
dput(names(compressbien))
set.seed(12345)

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


medias2<-cruzadaavnnet(data=archivo,
 vardep=vardep,listconti=listconti,
 listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
    size=c(5),decay=c(0.1),repeticiones=5,itera=200)

medias2bis<-as.data.frame(medias2[1])
medias2bis$modelo<-"avnnet"
predi2<-as.data.frame(medias2[2])
predi2$avnnet<-predi2$pred
```

```
medias3<-cruzadarf(data=archivo,
  vardep=vardep,listconti=listconti,
  listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
 mtry=3,ntree=600,nodesize=10,replace=TRUE)

medias3bis<-as.data.frame(medias3[1])
medias3bis$modelo<-"rf"
predi3<-as.data.frame(medias3[2])
predi3$rf<-predi3$pred

medias4<-cruzadagbm(data=archivo,
  vardep=vardep,listconti=listconti,
  listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
n.minobsinnode=20,shrinkage=0.10,n.trees=500,interaction.depth=2)

medias4bis<-as.data.frame(medias4[1])
medias4bis$modelo<-"gbm"
predi4<-as.data.frame(medias4[2])
predi4$gbm<-predi4$pred

medias5<-cruzadaxgbm(data=archivo,
  vardep=vardep,listconti=listconti,
  listclass=listclass,grupos=grupos,sinicio=sinicio,repe=repe,
   min_child_weight=10,eta=0.03,nrounds=5000,max_depth=6,
  gamma=0,colsample_bytree=1,subsample=1,
 alpha=0,lambda=0,lambda_bias=0)


medias5bis<-as.data.frame(medias5[1])
medias5bis$modelo<-"xgbm"
predi5<-as.data.frame(medias5[2])
predi5$xgbm<-predi5$pred


medias6<-cruzadaSVM(data=archivo,
   vardep=vardep,listconti=listconti,
   listclass=listclass,grupos=grupos,
 sinicio=sinicio,repe=repe,C=0.05)

medias6bis<-as.data.frame(medias6[1])
medias6bis$modelo<-"svmLinear"
predi6<-as.data.frame(medias6[2])
predi6$svmLinear<-predi6$pred
```

```
medias7<-cruzadaSVMpoly(data=archivo,
   vardep=vardep,listconti=listconti,
   listclass=listclas s,grupos=grupos,sinicio=sinicio,repe=repe,
 C=0.01,degree=2,scale=2)

medias7bis<-as.data.frame(medias7[1])
medias7bis$modelo<-"svmPoly"
predi7<-as.data.frame(medias7[2])
predi7$svmPoly<-predi7$pred

medias8<-cruzadaSVMRBF(data=archivo,
   vardep=vardep,listconti=listconti,
   listclass=listclass,grupos=grupos,
 sinicio=sinicio,repe=repe,
 C=5,sigma=0.01)

medias8bis<-as.data.frame(medias8[1])
medias8bis$modelo<-"svmRadial"
predi8<-as.data.frame(medias8[2])
predi8$svmRadial<-predi8$pred

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)

```
# 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
```

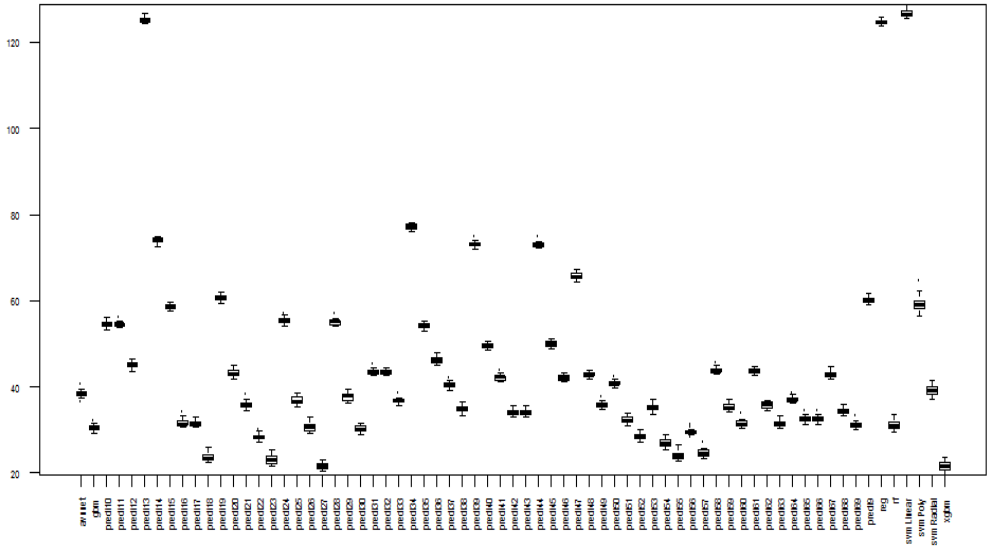
```
unipredi$predi43<-(unipredi$reg+unipredi$gbm+unipredi$xgbm)/3
unipredi$predi44<-(unipredi$reg+unipredi$gbm+unipredi$svmLinear)/3
unipredi$predi45<-(unipredi$reg+unipredi$gbm+unipredi$svmPoly)/3
unipredi$predi46<-(unipredi$reg+unipredi$gbm+unipredi$svmRadial)/3
unipredi$predi47<-(unipredi$reg+unipredi$xgbm+unipredi$svmLinear)/3
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$avnnet+unipredi$svmLinear)/5
unipredi$predi68<-(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmPoly)/5
unipredi$predi69<-(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmRadial)/5
```

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

```
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

```
# Finalmente boxplot (solo he calculado tasa fallos)
par(cex.axis=0.5,las=2)
boxplot(data=medias0,outcex=0.4,error~modelo)
```



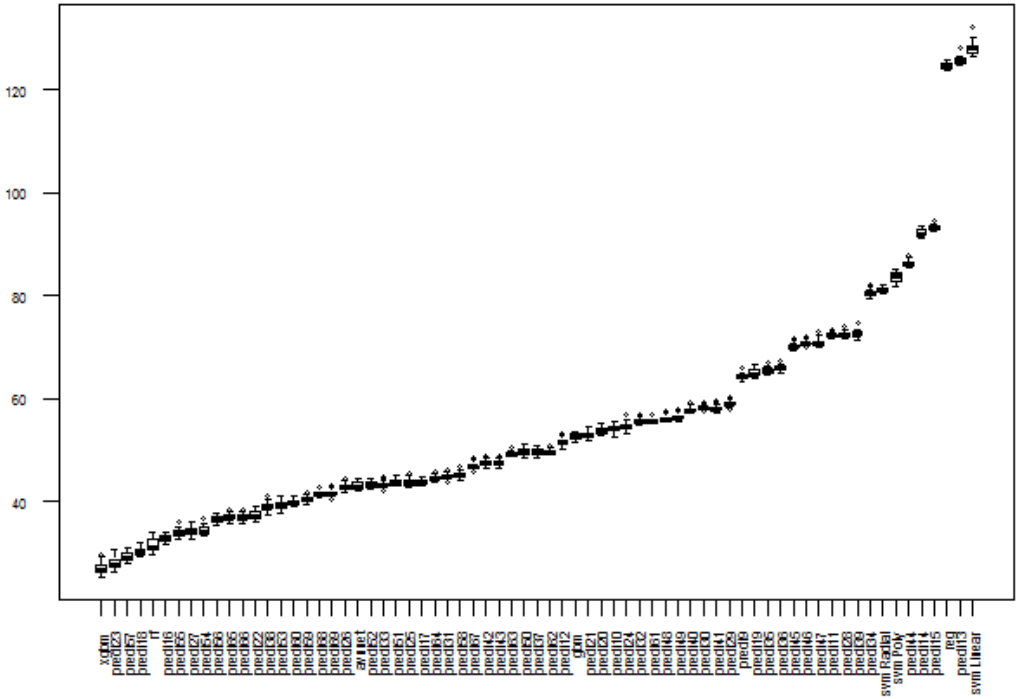
7) Tabla ordenada para observar

```
# 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

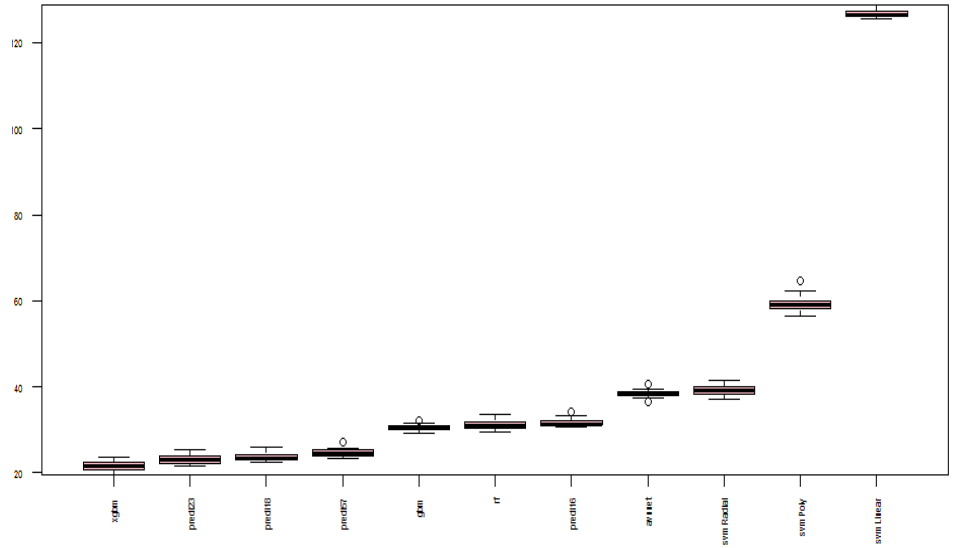
8) Boxplot ordenado

```
# 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")
```



9) Opcionalmente, selección de modelos a graficar

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

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")
```



10) Observar en la lista de ensamblados a cuales corresponden los mejores: predi23,predi57,predi18:

```
unipredi$predi23<-(unipredi$rf+unipredi$xgbm)/2
unipredi$predi57<-(unipredi$rf+unipredi$avnnet+unipredi$xgbm)/3
unipredi$predi18<-(unipredi$avnnet+unipredi$xgbm)/2
```

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 (9 pasos)

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

Omitimos el 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

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

5) Procesado de los ensamblados, hay que construir los promedios de **tasa de fallos** 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")
```

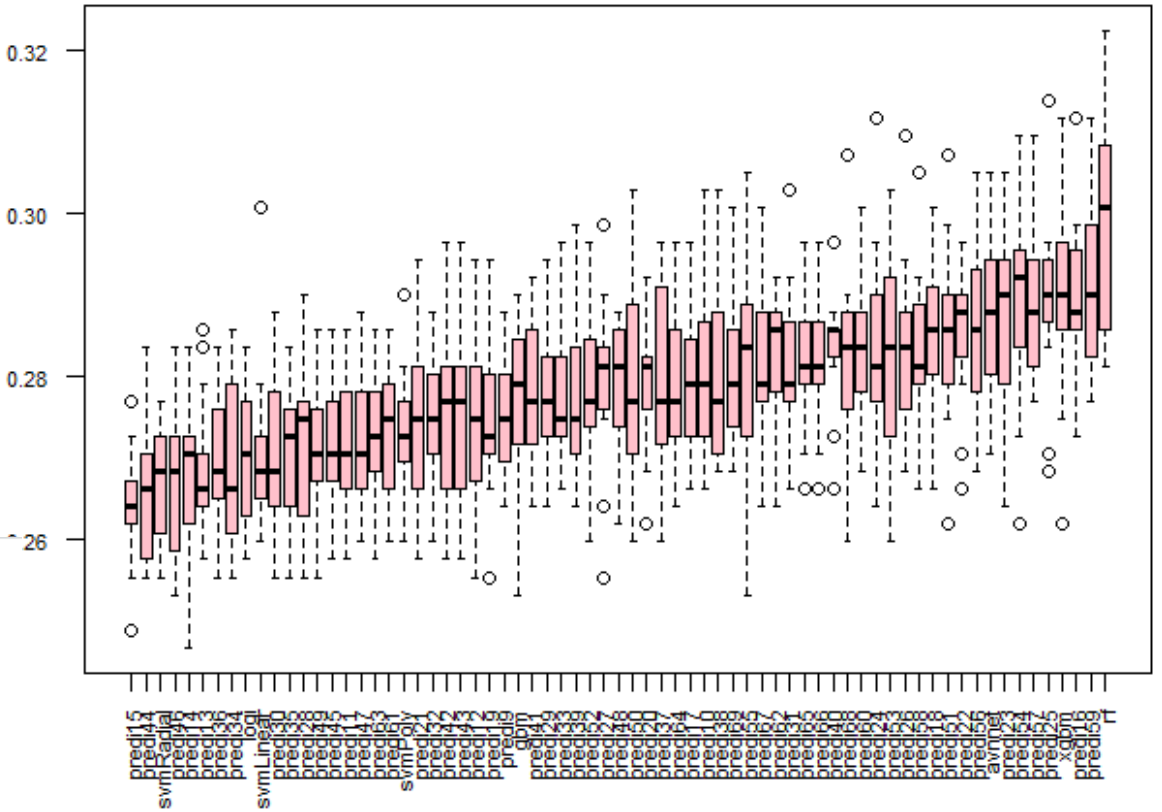
7) Tabla ordenada para observar

```
# PRESENTACION TABLA MEDIAS
tablamedias<-medias0 %>%
  group_by(modelo) %>%
  summarize(tasa=mean(tasa))

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

8) Boxplot ordenado

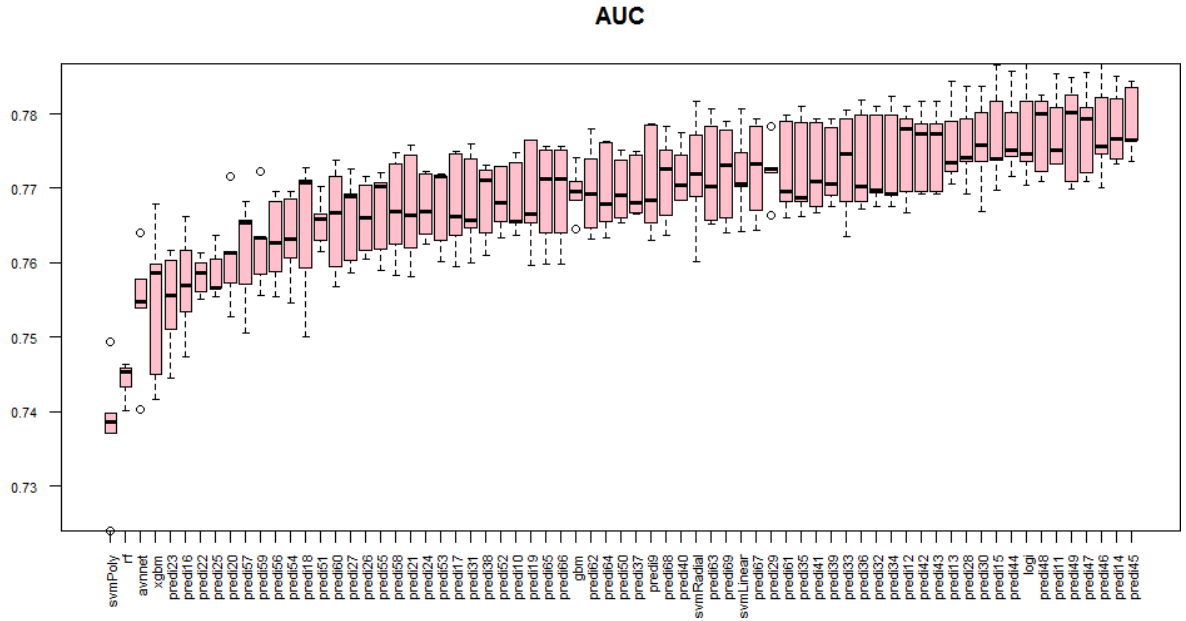
```
# ORDENACIÓN DEL FACTOR MODELO POR LAS MEDIAS EN TASA
# PARA EL GRAFICO

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

	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



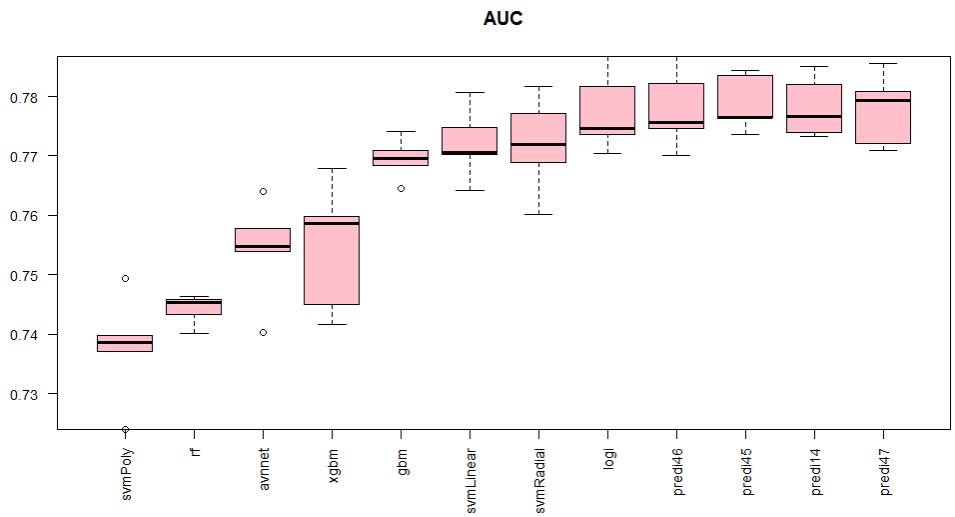
```
# ************************************
# PARA AUC
# ************************************

# PRESENTACION TABLA MEDIAS

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

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


# 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")
```

	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



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(para este ejemplo)

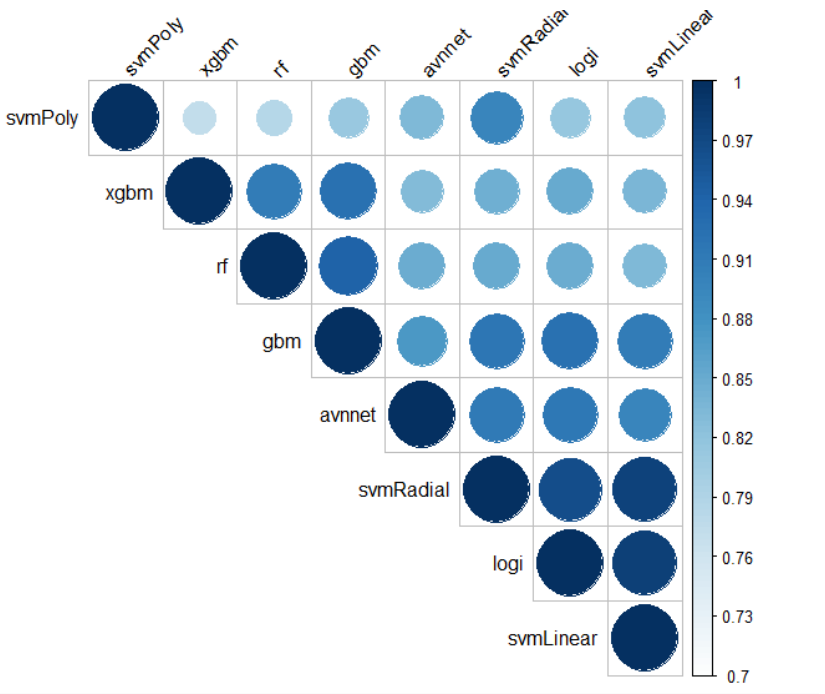
1) Los algoritmos simples que mejor funcionan son la regresión logística. SVM Radial y SVM Lineal.
2) Aparentemente ensamblar la logística, xgbm y svmLinear es el ensamblado con mejor AUC.
3) En este caso la diferencia con la regresión logística es pequeña y se preferiría por lo tanto la regresión logística.
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

```
unipredi<-cbind(predi1,predi2,predi3,predi4,predi5,predi6,predi7,predi8)
# Esto es para eliminar columnas duplicadas
unipredi<- unipredi[, !duplicated(colnames(unipredi))]

# Añadir ensamblados

unipredi$predi14<-(unipredi$logi+unipredi$svmPoly)/2
unipredi$predi15<-(unipredi$logi+unipredi$svmRadial)/2
unipredi$predi44<-(unipredi$logi+unipredi$gbm+unipredi$svmLinear)/3
unipredi$predi46<-(unipredi$logi+unipredi$gbm+unipredi$svmRadial)/3
```

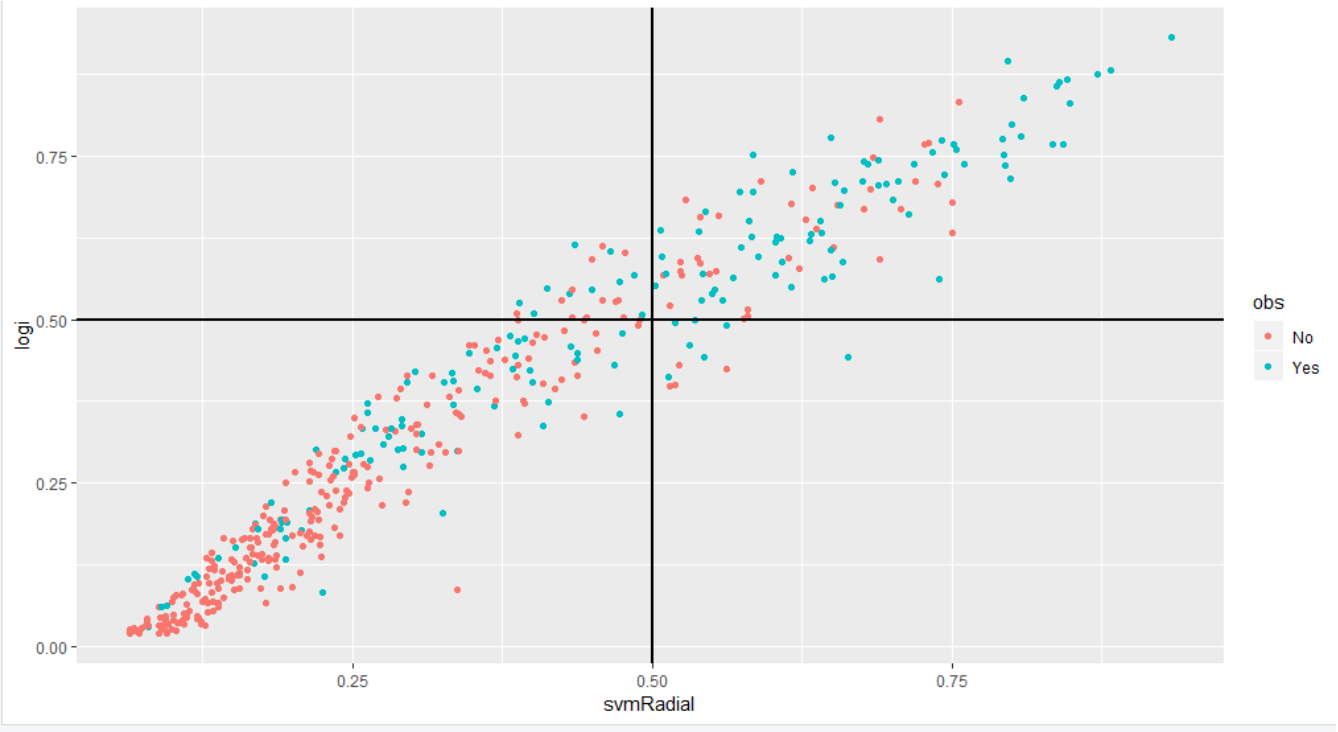
Me quedo con solo una repetición de validación cruzada para ver gráficos de puntos

```
# Me quedo con la primera repetición de validación cruzada para los análisis
unigraf<-unipredi[unipredi$Rep=="Rep001",]
# 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)
```



Comportamiento de algoritmos dos a dos mediante 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)
```



```
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)
```

```
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)
```