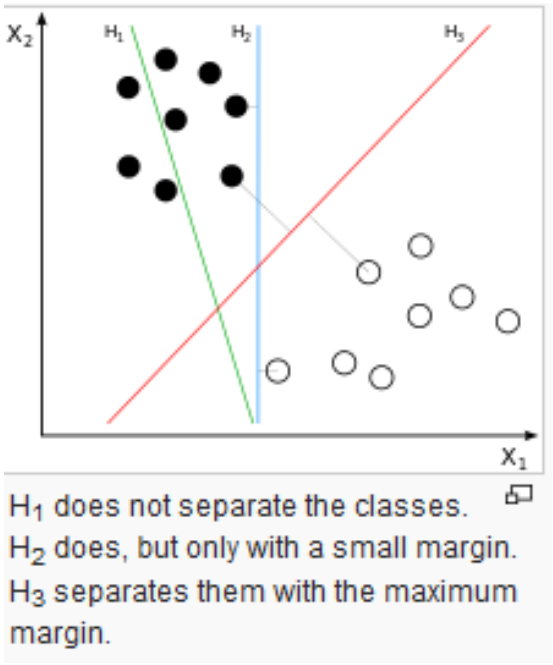
Support Vector Machines

Support Vector Machines

Se trata de plantear el problema de separación lineal de clases con métodos algebraicos (buscar el hiperplano de separación). Se basa en tres ideas importantes.

Primera idea importante: el "maximal margin" (Vapnik, 1963)

Se trata, no solamente de separar las clases por un hiperplano (función lineal), sino de incluir en la decisión de la construcción del separador, el concepto de separador con máximo margen. Esto a menudo mejora tanto el sesgo como la varianza de los resultados.



H_1 does not separate the classes.
H_2 does, but only with a small margin.
H_3 separates them with the maximum margin.

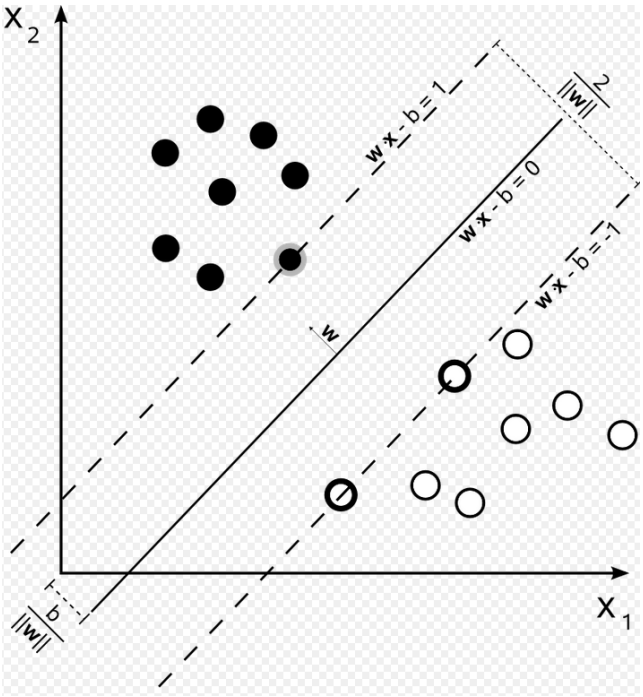
En el gráfico se intuye que será mejor el modelo rojo que el azul, ya que aunque ambos construyen separación perfecta, ante una nueva observación el modelo rojo está mejor preparado pues deja más espacio respecto de las dos clases. El modelo azul puede fácilmente errar en un nuevo punto negro que aparezca arriba un poquito a la derecha de los puntos existentes.

La regresión logística no tiene esto en cuenta, y aunque construye normalmente modelos con buen margen, no siempre es el caso y depende de la complejidad de los datos.

El planteamiento de este nuevo algoritmo será geométrico basado en esta idea de separación máxima.

Se trata de hallar el vector de parámetros **w** que maximice el margen. Las ecuaciones de los hiperplanos que delimitan el margen son wx=1 y wx=-1. Denotando **y** como (-1,1) en el problema de clasificación, hay que maximizar la distancia entre los dos hiperplanos de separación (2/norma de **w)**.

Se suele hacer con métodos clásicos de optimización (Lagrange, Kuhn Tucker).



$$\arg\min_{(\mathbf{w},b)} \frac{1}{2}\|\mathbf{w}\|^2$$

subject to (for any $i = 1, \ldots, n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

Gráficamente hay que encontrar la ecuación del hiperplano (en el gráfico una recta) que separe las dos clases, dejando máxima separación (máximo margen). El margen viene determinado por 2*la norma de w. La restricción en el planteamiento de optimización significa que los puntos de una clase (blancos) deben estar a un lado de la recta y los otros (negros) al otro lado.

Como se ve en el gráfico, hay puntos exactamente sobre el margen (dos puntos blancos y uno negro). Para clasificar una nueva observación, nos basta con tener en cuenta si la nueva observación está a un lado del hiperplano que pasa por los puntos blancos o del hiperplano que pasa por el punto negro. Por ello estos puntos (vectores, porque suelen tener varias coordenadas, en el ejemplo x1 y x2) se llaman puntos o **vectores de soporte**, lo que da nombre al algoritmo.

Hay que resaltar que en el planteamiento y=+1 o y=-1 según la observación pertenece a una clase o a otra. Una vez determinados los valores de w (w es un vector) y b (b es una constante), **para clasificar una nueva observación x_k simplemente se calcula $\boldsymbol{w \cdot x_k + b}$** . Si este valor es positivo , se clasifica la observación como y=1; si es negativo, como y=-1.

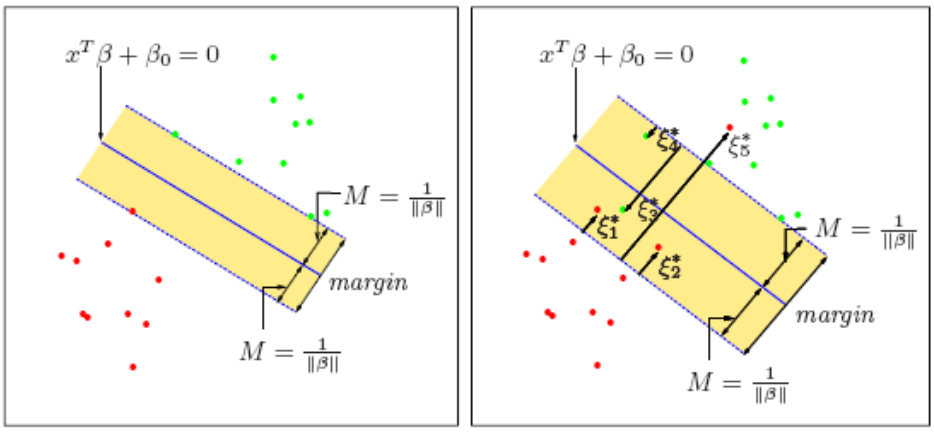
En este punto es necesario decir que **el resultado de SVM es una decisión binaria**, no da probabilidades de pertenencia a clases como otros métodos, solo como resultado 1 o -1.

La función $w \cdot x_k + b$ es la función de decisión asociada al SVM. Su signo es lo más importante. A menudo los paquetes dan también como resultado la distancia de los puntos al hiperplano (esta distancia siempre es positiva). Como esta distancia es una medida de confiabilidad de pertenencia a una clase, se usa como base para aproximar probabilidades de pertenencia a clase (a través de la regresión logística, según el método de Platt).

Muchos paquetes por ello pueden dar como resultado primario de SVM probabilidades de pertenencia a clase pero debe saberse que esto es consecuencia de un artificio adicional.

Segunda idea importante: el "soft margin" (Cortes, Vapnik, 1995)

La separación perfecta no suele existir, y en ese sentido el algoritmo no es práctico. Es necesario permitir observaciones mal clasificadas por los separadores para no incurrir en sobreajustes.



Esto implica añadir una variable ξ de "residuo" y constante C de regularización del margen que está relacionada inversamente con la anchura del margen y el "permiso para fallar" que vamos a permitir en la construcción del separador. A mayor C (lo que implica en el proceso de optimización menores "residuos" ξ_i), menor margen. A menor C (permitimos mayores residuos ξ_i), más permiso para fallar y más margen.

$$\arg\min_{\mathbf{w},\xi,b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right\}$$

subject to (for any $i = 1, \ldots n$)

$$y_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

Consecuencias prácticas del uso de la constante de regularización C

- Los valores de C no están escalados, en la práctica C puede ser 0.000000000001 y también C=1000000000000.

- En la práctica aumentar mucho C implica que la construcción de los hiperplanos es más fina: se crearán muchas separaciones estrechas; si hay muchas variables input y variables categóricas el algoritmo las exprimirá lo máximo posible para crear separaciones. Esto implica muchos puntos de soporte, creando un modelo que tiende a la sobreparametrización y por lo tanto al sobreajuste. Aumentar C implica por lo tanto intentar reducir el sesgo o error promedio, con el riesgo de sobreajuste.

- Disminuir C, por el contrario, lleva a crear márgenes más grandes. Esto suele implicar menos puntos de soporte y por lo tanto menos parámetros. El modelo es menos fino,

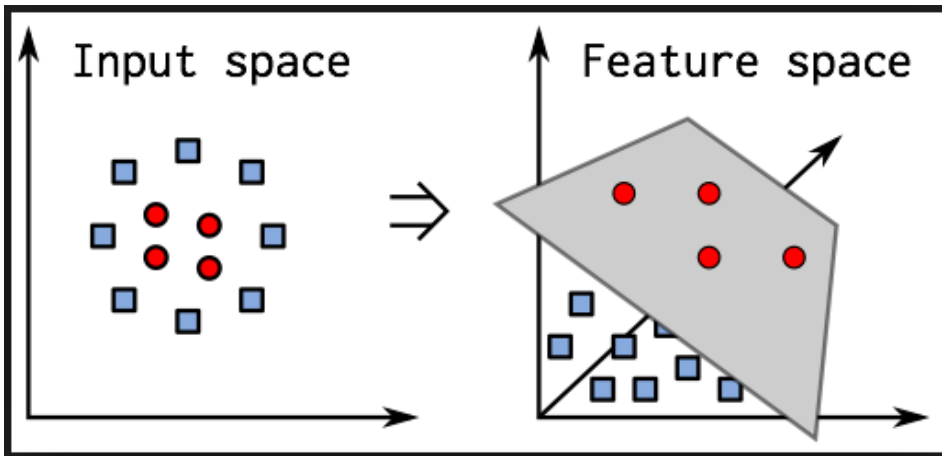
menos preciso, pero también más robusto y menos sobreajustado. Disminuir C implica por lo tanto intentar reducir la varianza del modelo, con el riesgo de alto sesgo o poca precisión promedio.

- Obviamente se jugará con la constante C como una constante a tunear, pero es conveniente conocer los anteriores conceptos.

Tercera idea importante: el Kernel (Boser, Guyon, Vapnik, 1992)

La separación entre clases en muchos problemas no es lineal. La utilización de SVM (Suppoprt Vector Machines) en datos con separabilidad no lineal, a través del truco kernel es un concepto que dio gran popularidad y uso a la técnica.

Una idea para aplicar a pesar de todo un algoritmo de separación lineal en datos no separables linealmente es trabajar en un **espacio de dimensión superior donde sí tenga sentido la separación lineal**. Simplemente extrapolar los datos con más dimensiones nos permite encontrar separadores lineales



y	x1	x2
-1	3	4
1	4	6
…	…	…

y	x1	x2	x2^2
-1	3	4	16
1	4	6	36
…	…	…	

Por ejemplo, supongamos que a la tabla de datos anterior (dibujados en la gráfica izquierda) añadimos una variable nueva, x_2^2, calculada directamente a partir de x2. Entonces en ese espacio de 3 dimensiones sí se podría separar linealmente como en el ejemplo de la figura, donde la nueva variable x_2^2 haría el papel de eje z o vertical.

Para ello, se pueden introducir nuevas funciones de las variables (x2, x3,x1x2, etc.) lo que aumenta la dimensión del vector de variables independientes. Entonces sería más fácil para el algoritmo encontrar separaciones lineales y buen tamaño de margen.

El problema es que este aumento de dimensión hace a menudo impracticables los cálculos, pero aquí interviene el "truco Kernel" ("the Kernel trick"): cualquier algoritmo que dependa solo de los productos escalares (como es el caso del SVM) permite trabajar computacionalmente en una dimensión controlada a través de una función llamada Kernel, que tiene que cumplir:

K(x,y)=< ϕ (x), ϕ (y)>.

Donde la función ϕ(**x**) representa una función que extrapola de la dimensión original a una superior. Por ejemplo, de 3 dimensiones pasamos a 9:

$\phi(x_1, x_2, x_3)$= (x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)

Ejemplo

En el caso anterior, el Kernel asociado a $\phi(x_1, x_2, x_3)$ es la función K(x,y)=(<x,y>)**2 pues se cumple la simetría K(x,y)=< ϕ (x), ϕ (y)>.

$\phi(x_1, x_2, x_3)$= (x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)

Supongamos x = (1, 2, 3) y = (4, 5, 6).

Si queremos calcular < ϕ (x), ϕ (y)>:

ϕ (x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)
ϕ (y) = (16, 20, 24, 20, 25, 36, 24, 30, 36)

< ϕ (x), ϕ (y)> = 16 + 40 + 72 + 40 + 100+ 180 + 72 + 180 + 324 = 1024

Ahora si en lugar de calcular ese producto haciendo todas esas operaciones utilizamos el Kernel, todo sería más sencillo (3 productos y 3 sumas y un exponente):

K(x, y) = (<x,y>)**2 =(4 + 10 + 18) **2 = 32**2 = 1024

Por lo tanto no ha sido necesario crear nuevas variables, ni utilizarlas como tal , ahorrando problemas de todo tipo.

El problema de optimización puede plantearse en su forma dual, como función de productos escalares $\mathbf{x}_i^T \mathbf{x}_j$ (ver formulación más abajo).

Sustituyendo en el problema de optimización por su Kernel $k(x_i,x_j)$, implícitamente estamos aumentando la dimensión del espacio de variables de cara a la construcción de hiperplanos de separación**, sin pasar realmente por la creación de nuevas variables.**

Using the fact that $\|\mathbf{w}\|^2 = \mathbf{w}^T \cdot \mathbf{w}$ and substituting $\mathbf{w} = \sum_{i=1} \alpha_i y_i \mathbf{x_i}$, one can show that the dual of the SVM reduces to the following optimization problem:

Maximize (in α_i)

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

subject to (for any $i = 1, \ldots, n$)

$$\alpha_i \geq 0,$$

and to the constraint from the minimization in b

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

Here the kernel is defined by $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$.

W can be computed thanks to the α terms:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i.$$

Obviamente hay que utilizar Kernels estándar que cumplen la propiedad. El más frecuente de los no lineales es el RBF Gaussiano (necesita un parámetro sigma o gamma). El Kernel polinomial necesita un parámetro de grado del polinomio y el sigmoide suele necesitar un parámetro de posición y otro de escala.

Linear function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j$
Polynomial function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r\right)^d$
RBF function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ $= \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \gamma = \frac{1}{2\sigma^2}$
Sigmoid function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \tanh\left(\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r\right)$ $\tanh(x) = \frac{e^x}{e^x + 1}$

Parámetros a considerar en SVM

1) Parámetro C. Aumentar C implica menor sesgo y mayor sobreajuste. Su rango depende mucho de los datos.

2) La función Kernel (no siempre es necesaria) y su(s) parámetro(s) en cada caso.

RBF: Aumentar gamma en la función RBF implica menor sesgo y mayor sobreajuste.

Polinomial. Aumentar el grado del polinomio implica menor sesgo y mayor sobreajuste.

Hay interdependencia entre los parámetros C y los del kernel. Ver página web

http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

Ventajas de SVM

(ver http://www.svms.org/ para más detalles sobre SVM)

- Muy flexible, sobre todo por el truco Kernel. Hay versión para regresión y para clasificación multinomial.
- Puede competir en datos separables linealmente con la regresión logística.
- Buena performance en clasificación de imágenes.
- Pocos parámetros a tunear
- Se puede usar para detección de outliers o datos anómalos (One-Class SVM)

Desventajas de SVM

- Lento a veces en la optimización. A más variables y n , poco escalable (mucho más lento).
- Dificultad en seleccionar la función Kernel y sus parámetros asociados
- Valores missings, categorías poco representadas, variables irrelevantes, etc. son un problema como en los algoritmos clásicos. El preprocesado es importante.
- El resultado para una nueva observación en clasificación binaria viene dado en términos de 0 ó 1, no en probabilidades. Normalmente para asignar probabilidades a

cada observación se utiliza la regresión logística, lo que se conoce como el método de Platt.

- En principio solo sirve para clasificación binaria pero se puede extender a multi clase.

Support Vector Machines con caret

Archivo SVM tuneado 2.0.R

Tuneado SVM binaria

En caret hay diferentes paquetes para SVM con kernel lineal (el SVM básico), con kernel polinomial y con kernel gaussiano (también llamado Radial Basis Function o RBF). Se llaman respectivamente svmLinear, svmPoly y svmRadial.

- El kernel lineal solo necesita la constante de regularización C.
- El kernel polinomial necesita la constante de regularización C, el grado del polinomio y el parámetro de escala (el gamma que aparece en la tabla de kernels más arriba).
- El kernel gaussiano necesita la constante de regularización C, y el parámetro sigma de varianza-escala.

Al ser pocos parámetros a tunear (una gran ventaja del SVM) nos podemos permitir evaluarlos con ciertos métodos gráficos.

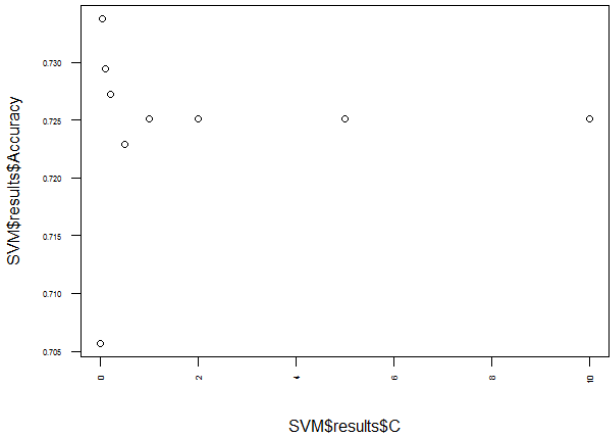
```
# *************************
# TUNEADO SVM BINARIA
# *************************

load ("c:/saheartbis.Rda")
```

SVM Lineal

```
#  SVM LINEAL: SOLO PARÁMETRO C

SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10))

control<-trainControl(method = "cv",number=4,savePredictions = "all")

SVM<- train(data=saheartbis,factor(chd)~sbp+tobacco+ldl+age+
  typea+famhist.Absent,
  method="svmLinear",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)
SVM
SVM$results
plot(SVM$results$C,SVM$results$Accuracy)
```

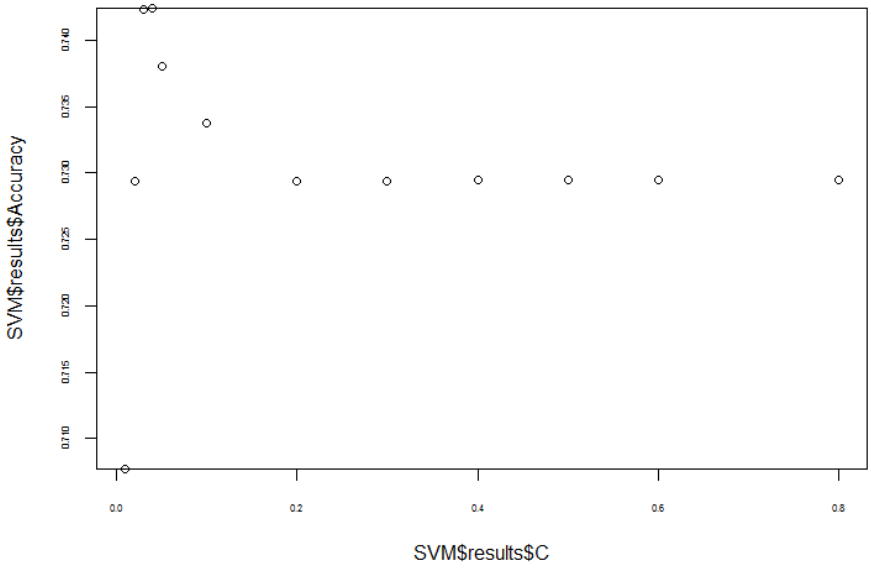
```
     C      Accuracy   Kappa
  0.01  0.7056784  0.2413527
  0.05  0.7337706  0.3773863
  0.10  0.7294228  0.3675741
  0.20  0.7272489  0.3730833
  0.50  0.7229198  0.3655104
  1.00  0.7250937  0.3712509
  2.00  0.7250937  0.3712509
  5.00  0.7250937  0.3712509
 10.00  0.7250937  0.3712509
```

```
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 0.05.
```



Es mejor siempre observar la tabla completa de resultados y si puede ser un gráfico, y no fiarse excesivamente de las recomendaciones automáticas de caret. En este caso parece que a partir de un cierto valor pequeño (pero no cero) la accuracy es alta. Caret recomienda 0.05 pero podríamos rehacer el grid en ese intervalo para ver bien si hay regularidad en el gráfico.

```
# Rehago el grid para observar mejor el intervalo de C entre 0 y 0.6
SVMgrid<-
expand.grid(C=c(0.01,0.02,0.03,0.04,0.05,0.8,0.1,0.2,0.3,0.4,0.5,0.6))

control<-trainControl(method = "cv",number=4,savePredictions = "all")

SVM<- train(data=saheartbis,factor(chd)~sbp+tobacco+ldl+age+
  typea+famhist.Absent,
  method="svmLinear",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)

SVM$results
plot(SVM$results$C,SVM$results$Accuracy)
```



Parece que C=0.03 puede dar buenos resultados.

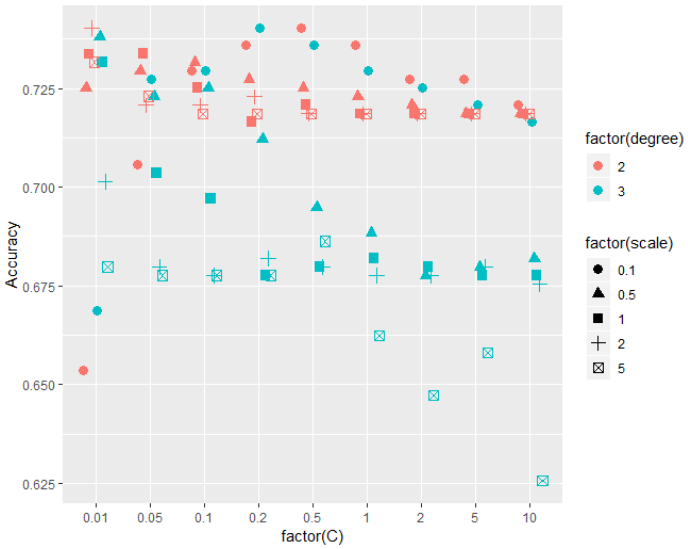
SVM Polinomial

Hay que recordar que el proceso de optimización con tantos parámetros puede ser largo, pero es necesario.

```
#  SVM Polinomial: PARÁMETROS C, degree, scale

SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10),
 degree=c(2,3),scale=c(0.1,0.5,1,2,5))

control<-trainControl(method = "cv",
 number=4,savePredictions = "all")

SVM<- train(data=saheartbis,factor(chd)~sbp+tobacco+ldl+age+
  typea+famhist.Absent,
  method="svmPoly",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)
SVM

Accuracy was used to select the optimal model using the
largest value.The final values used for the model were
degree = 2, scale = 2 and C = 0.01.

SVM$results
```

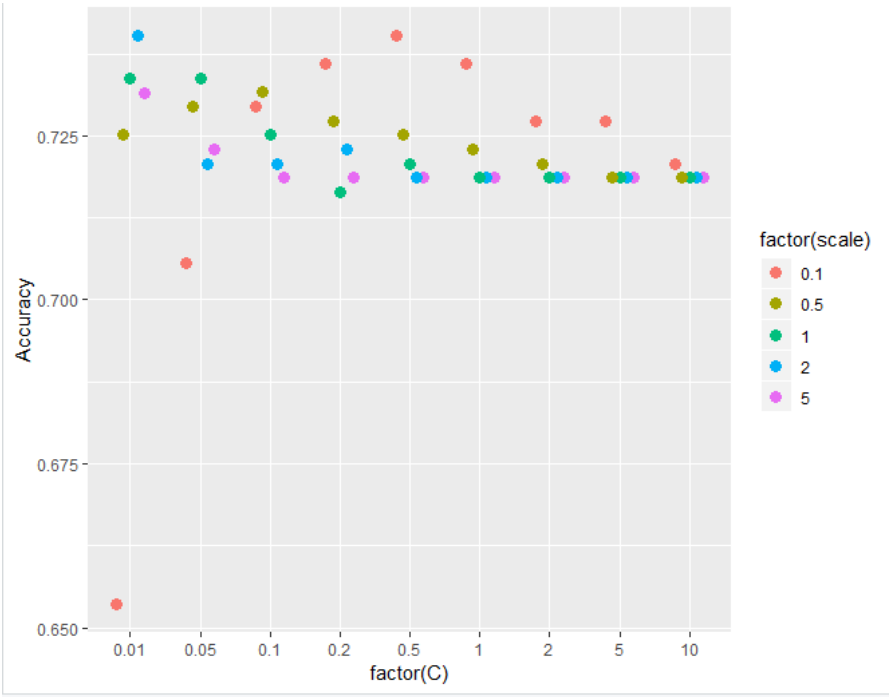
Al ser tres parámetros es difícil hacerse una idea solo con la tabla. Es mejor hacer gráficos en los que estén representados si puede ser todos los parámetros.

```
# LOS GRÁFICOS DOS A DOS NO SIRVEN
# plot(SVM$results$C,SVM$results$Accuracy)
# plot(SVM$results$degree,SVM$results$Accuracy)
# plot(SVM$results$scale,SVM$results$Accuracy)
dat<-as.data.frame(SVM$results)
library(ggplot2)
# PLOT DE DOS VARIABLES CATEGÓRICAS, UNA CONTINUA
ggplot(dat, aes(x=factor(C), y=Accuracy,
 color=factor(degree),pch=factor(scale))) +
  geom_point(position=position_dodge(width=0.5),size=3)
```



En el gráfico se puede apreciar que la Accuracy es más alta en general para grado 2 del polinomio (color rojo) así que reducimos el gráfico para solo representar esos puntos e intentar buscar patrones.

```
# SOLO DEGREE=2
dat2<-dat[dat$degree==2,]

ggplot(dat2, aes(x=factor(C), y=Accuracy,
 colour=factor(scale))) +
  geom_point(position=position_dodge(width=0.5),size=3)
```



Es importante observar patrones y no depender de valores puntuales porque eso podría llevar al sobreajuste. Por ejemplo, en este gráfico:

*La curva azul desciende desde C pequeño

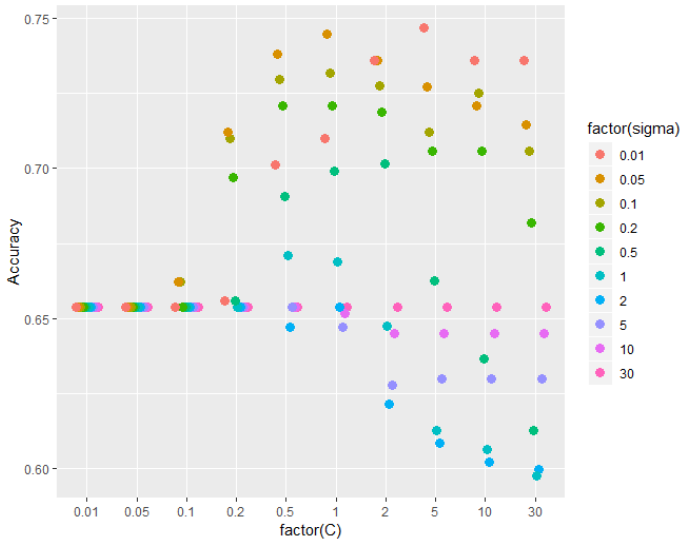
*La curva roja sigue una parábola con máximos en torno a C=0.5

*C grande, a partir de 1, no es bueno en general

Se observa que o bien se pone C pequeño (0.02) y escala 2 (punto azul) o bien, viendo la curva de puntos rojos (escala 0.1) C moderado, como C=0.5. Como la decisión está basada en patrones y no en valores puntuales, es de esperar que con otra estructura de datos training (otra semilla en set.seed por ejemplo), los valores óptimos de los parámetros serían parecidos, no iguales quizá, pero en un rango similar.

SVM RBF

```
#  SVM RBF: PARÁMETROS C, sigma

SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30),
 sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30))

control<-trainControl(method = "cv",
 number=4,savePredictions = "all")


SVM<- train(data=saheartbis,factor(chd)~sbp+tobacco+ldl+age+
  typea+famhist.Absent,
  method="svmRadial",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)

SVM

dat<-as.data.frame(SVM$results)

ggplot(dat, aes(x=factor(C), y=Accuracy,
 color=factor(sigma))) +
 geom_point(position=position_dodge(width=0.5),size=3)


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 = 5.
```



Se observa que la curva roja (sigma pequeño =0.01) sigue una curva con máximo cerca de C=5. En general los valores de sigma deben ser pequeños y C entre 0.5 y 5-6. Se podría rehacer el grid en esos intervalos pero aceptamos la recomendación de caret C=5 y sigma=0.01.

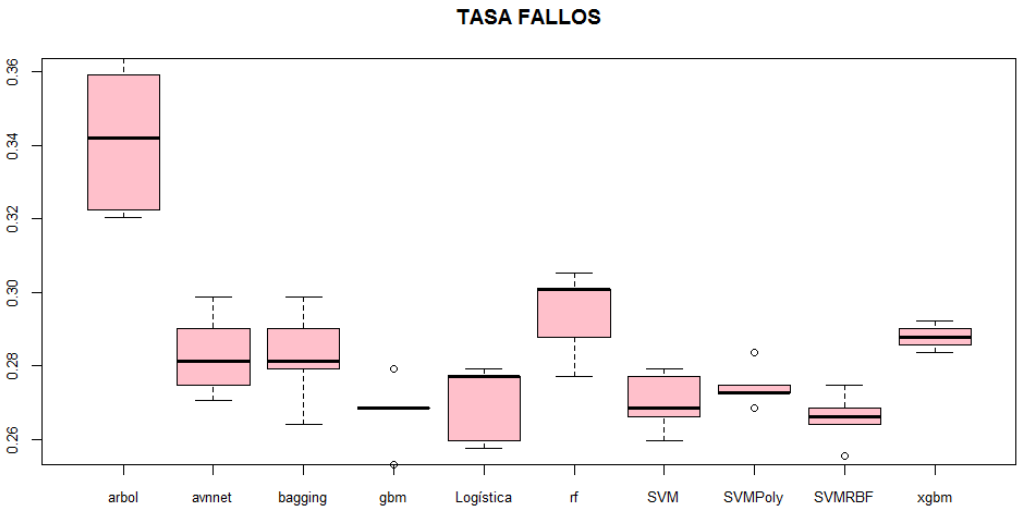
Comparación con otros modelos vía validación cruzada repetida

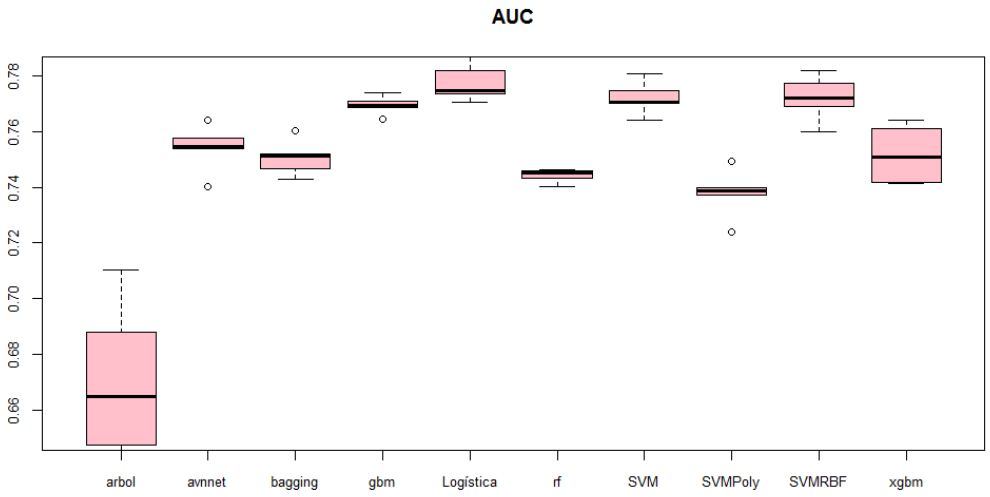
Aquí se utilizan las tres nuevas funciones para SVM lineal, polinomial y RBF

```
load ("saheartbis.Rda")
source ("cruzadas avnnet y log binaria.R")
source ("cruzada arbolbin.R")
source ("cruzada rf binaria.R")
source ("cruzada gbm binaria.R")
source ("cruzada xgboost binaria.R")
source ("cruzada SVM binaria lineal.R")
source ("cruzada SVM binaria polinomial.R")
source ("cruzada SVM binaria RBF.R")
```

…Omitimos parte del código

```
medias8<-cruzadaSVMbin(data=saheartbis, vardep="chd",
   listconti=c("sbp", "tobacco",
  "ldl","age", "typea","famhist.Absent"),
 listclass=c(""),
  grupos=4,sinicio=1234,repe=5,C=0.03)

medias8$modelo="SVM"


medias9<-cruzadaSVMbinPoly(data=saheartbis, vardep="chd",
   listconti=c("sbp", "tobacco",
  "ldl","age", "typea","famhist.Absent"),
 listclass=c(""),
  grupos=4,sinicio=1234,repe=5,
  C=0.02,degree=2,scale=2)

medias9$modelo="SVMPoly"


medias10<-cruzadaSVMbinRBF(data=saheartbis, vardep="chd",
   listconti=c("sbp", "tobacco",
  "ldl","age", "typea","famhist.Absent"),
 listclass=c(""),
  grupos=4,sinicio=1234,repe=5,
  C=5,sigma=0.01)

medias10$modelo="SVMRBF"


union1<-rbind(medias1,medias2,medias3,medias4,medias5,
 medias6,medias7,medias8,medias9,medias10)

par(cex.axis=0.8)
boxplot(data=union1,tasa~modelo,main="TASA FALLOS",col="pink")
      boxplot(data=union1,auc~modelo,main="AUC",col="pink")
```

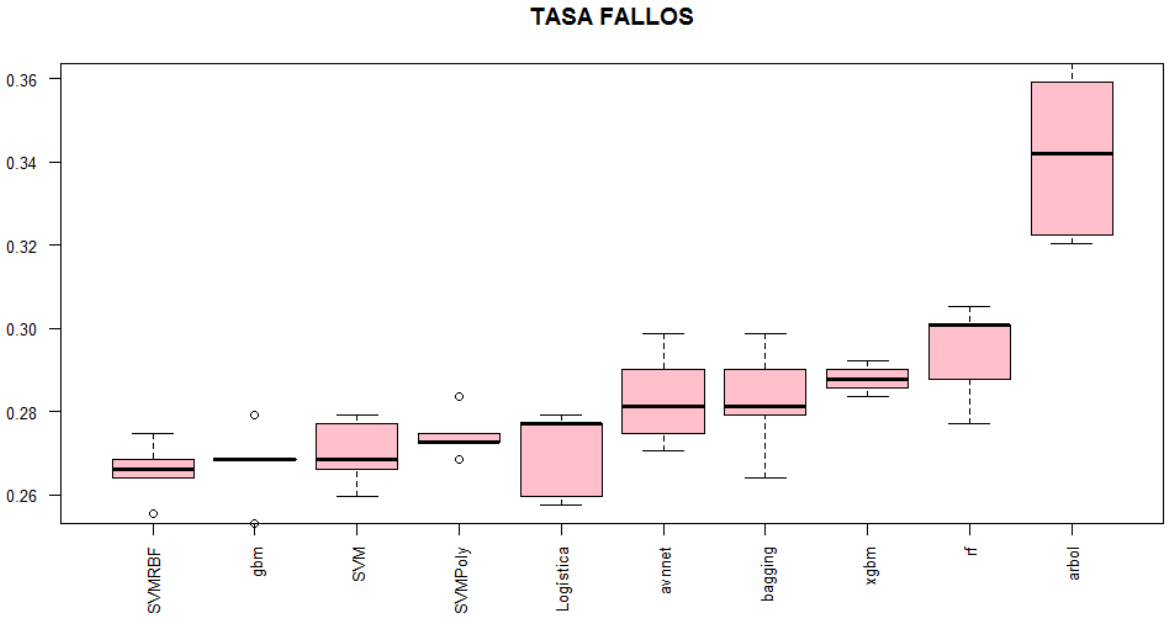


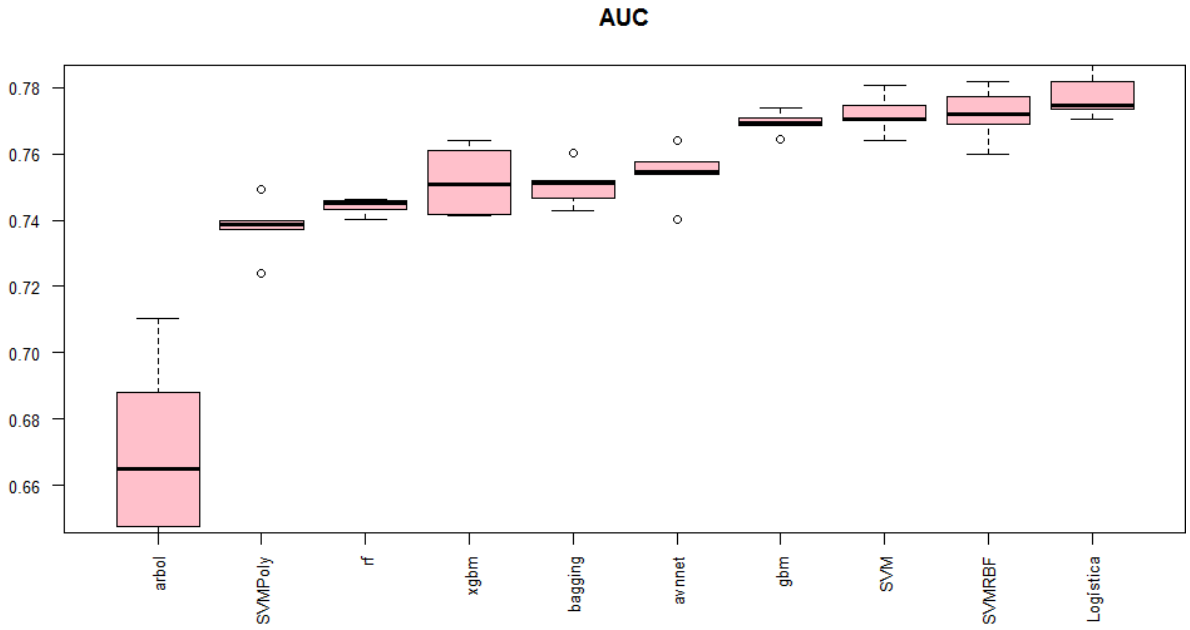


Como se ve SVM es un algoritmo competitivo, aún para datos de separabilidad lineal compite bien con la regresión logística. Aunque esta última es preferible por su mejor AUC y porque siempre preferiremos modelos clásicos ante la duda.

Estas dos funciones permiten ordenar los modelos por la mediana de tasa de fallos y de auc respectivamente.

```
uni<-union1
uni$modelo <- with(uni,
 reorder(modelo,tasa, median))
par(cex.axis=0.8,las=2)
boxplot(data=uni,tasa~modelo,col="pink",main="TASA FALLOS")

uni<-union1
uni$modelo <- with(uni,
 reorder(modelo,auc, median))
par(cex.axis=0.8,las=2)
boxplot(data=uni,auc~modelo,col="pink",main="AUC")
```





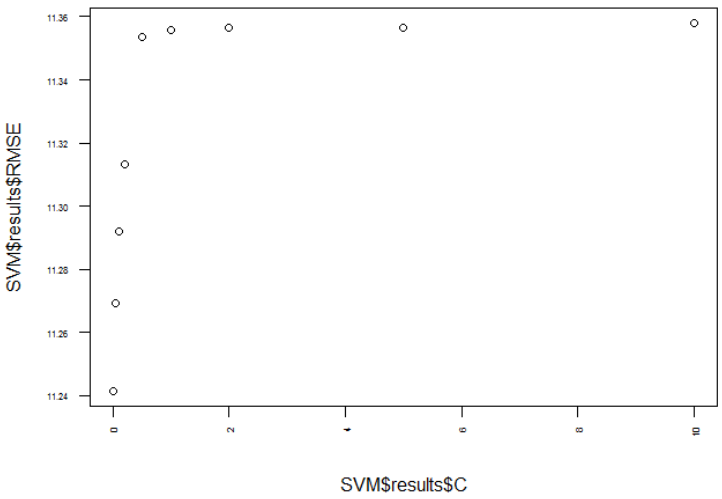
Como se ve, con el criterio mediana de la tasa de fallos el mejor modelo sería SVM-RBF, y con el criterio AUC la regresión logística. Es obvio que las diferencias son pequeñas en general en sesgo y varianza entre los modelos logística y SVM, SVMRBF, gbm siendo también competitivo aún en datos tan sencillos.

Ejemplo variable dependiente continua

Aunque no se ha visto la teoría, existe SVM para regresión, llamada Suport Vector Regression, par variable dependiente continua. Lo aplicamos en el ejemplo, tuneando los parámetros como en varible dependiente binaria.

(criterio: RMSE (cuanto más bajo mejor))

```
# *************************
# TUNEADO SVM CONTINUA
# *************************
load ("c:/compressbien.Rda")
#  SVM LINEAL: SOLO PARÁMETRO C
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10))
control<-trainControl(method = "cv",number=4,
 savePredictions = "all")
SVM<- train(data=compressbien,cstrength~age+water+cement+blast,
  method="svmLinear",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)

SVM$results
plot(SVM$results$C,SVM$results$RMSE)
```



Se observa que C debe ser pequeño, incluso se podría afinar más que C=0.01.

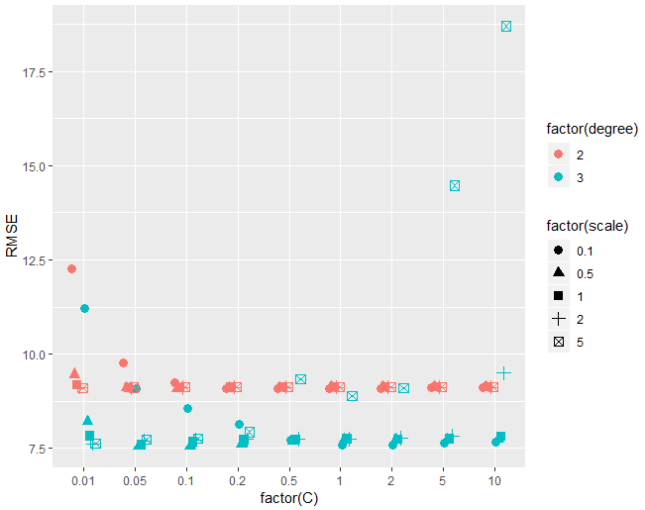
```
#  SVM Polinomial: PARÁMETROS C, degree, scale
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10),
 degree=c(2,3),scale=c(0.1,0.5,1,2,5))
control<-trainControl(method = "cv",
 number=4,savePredictions = "all")
```

(esto tarda mucho)

```
SVM<- train(data=compressbien,cstrength~age+water+cement+blast,
  method="svmPoly",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)
SVM
SVM$results
dat<-as.data.frame(SVM$results)
```

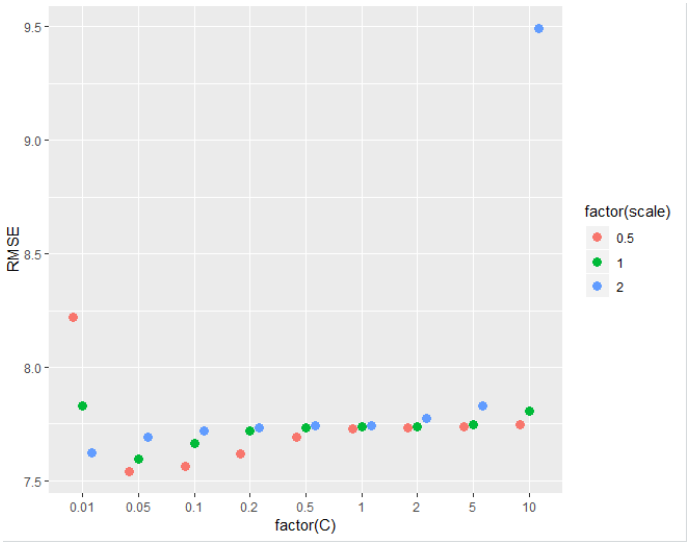
```
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were degree = 3, scale =
 0.5 and C = 0.05.
```

```
# PLOT DE DOS VARIABLES CATEGÓRICAS, UNA CONTINUA

ggplot(dat, aes(x=factor(C), y=RMSE,
 color=factor(degree),pch=factor(scale))) +
  geom_point(position=position_dodge(width=0.5),size=3)
```



Nos quedamos con grado 3, claramente mejor, y descartamos escala menor que 0.1 para ver mejor el gráfico.

```
# AFINO MÁS

dat2<-dat[dat$degree==3&dat$scale<5&dat$scale>0.1,]
ggplot(dat2, aes(x=factor(C), y=RMSE,
 colour=factor(scale))) +
  geom_point(position=position_dodge(width=0.5),size=3)
```

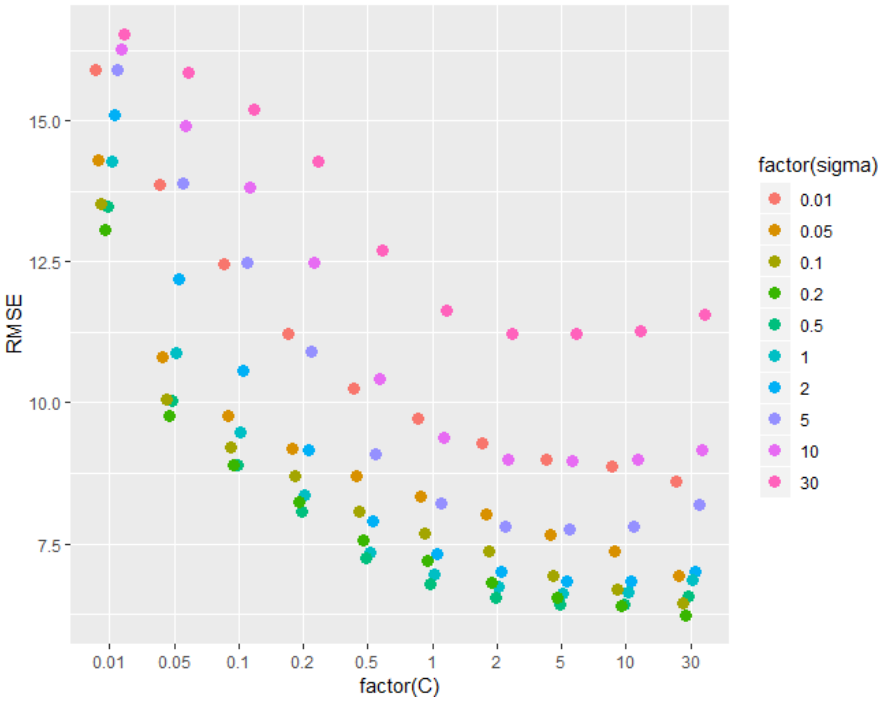


Parece que C debe estar en torno a 0.05 y escala 0.5.

```
#  SVM RBF: PARÁMETROS C, sigma

SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30),
 sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30))

control<-trainControl(method = "cv",
 number=4,savePredictions = "all")

SVM<- train(data=compressbien,cstrength~age+water+cement
+blast,  method="svmRadial",trControl=control,
 tuneGrid=SVMgrid,verbose=FALSE)

SVM
```

```
RMSE was used to select the optimal model
using the smallest value.
The final values used for the
model were sigma = 0.2 and C = 30.
```

```
dat<-as.data.frame(SVM$results)

ggplot(dat, aes(x=factor(C), y=RMSE,
 color=factor(sigma)))+
 geom_point(position=position_dodge(width=0.5),size=3)
```

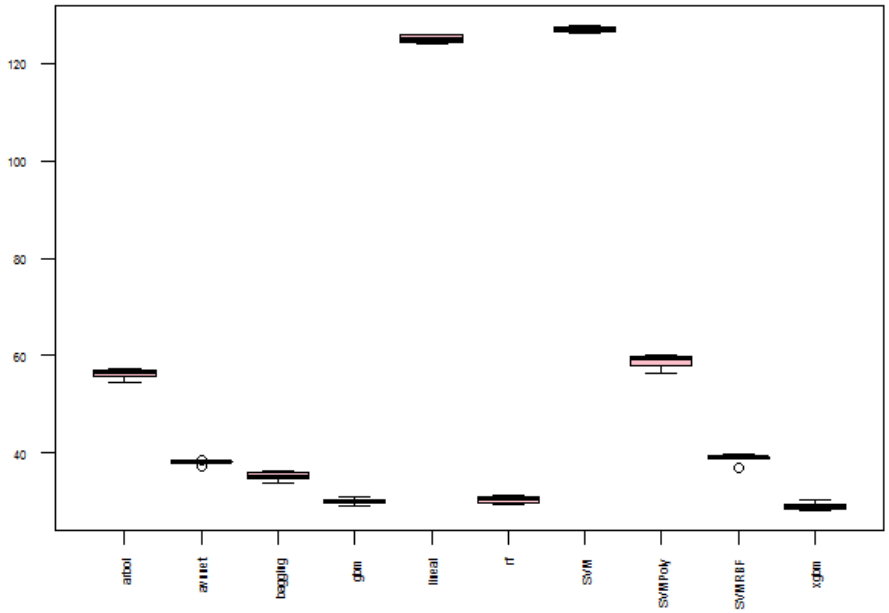


Aquí ha de ser C alto (podríamos afinar incluso con valores más altos que 30) y sigma 0.2.

Comparación con otros modelos vía validación cruzada repetida

Se omite código aquí…

```
medias8<-cruzadaSVM(data=data,
 vardep="cstrength",listconti=c("age","water","cement","blast"),
listclass=c(""),
  grupos=4,sinicio=1234,repe=5,C=0.01)

medias8$modelo="SVM"

medias9<-cruzadaSVMpoly(data=data,
 vardep="cstrength",listconti=c("age","water","cement","blast"),
listclass=c(""),
  grupos=4,sinicio=1234,repe=5,C=0.05,degree=3,scale=0.5)

medias9$modelo="SVMPoly"

medias10<-cruzadaSVMRBF(data=data,
 vardep="cstrength",listconti=c("age","water","cement","blast"),
listclass=c(""),
  grupos=4,sinicio=1234,repe=5,C=30,sigma=0.2)

medias10$modelo="SVMRBF"

union1<-rbind(medias1,medias2,medias3,medias4,medias5,medias6,
 medias7,medias8,medias9,medias10)

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



Se observa que en este caso SVM no llega al nivel predictivo de los algoritmos basados en árboles.