Máquinas de Vector Soporte (Support VectorMachines, SVMs)

Contenido

Máquinas de Vector de Soporte (Support Vector Machines, SVM)

1. Introducción

El método de clasificación. Vector Support Machines, fue desarrollado en la década de los 90, dentro de campo de la ciencia computacional. Si bien originariamente se desarrolló como un método de clasificación binaria, su aplicación se ha extendido a problemas de **clasificación múltiple y regresión.** SVMs ha resultado ser uno de los mejores clasificadores para un amplio abanico de situaciones, por lo que se considera uno de los referentes dentro del ámbito de aprendizaje estadístico y machine learning.

Las Máquinas de Vector Soporte se fundamentan en el Maximal Margin Classifier, que, a su vez, se basa en el concepto de hiperplano, comprender los fundamentos de las SVMs requiere de conocimientos sólidos en álgebra lineal. En este ensayo no se profundiza en el aspecto matemático, pero puede encontrarse una descripción detallada en el libro Support Vector Machines Succinctly by Alexandre Kowalczyk.

En R, las librerías e1071 y LiblineaR contienen los algoritmos necesarios para obtener modelos de clasificación simple, múltiple y regresión, basados en Support Vector Machines.

2. Hiperplano y Maximal Margin Classifier

En un espacio euclideo p-dimensional, un hiperplano se define como subespacio plano y afín (no tiene por qué pasar por el origen) de dimensiones $p - 1$ que divide el espacio en dos mitades. El término afín significa que el subespacio no tiene por qué pasar por el origen. En un espacio de dos dimensiones, el hiperplano es un subespacio de 1 dimensión, es decir, una recta. En un espacio tridimensional, un hiperplano es un subespacio de dos dimensiones, un plano convencional. Para dimensiones $p > 3$ no es intuitivo visualizar un hiperplano, pero el concepto de subespacio con $p - 1$ dimensiones se mantiene.

La definición matemática de un hiperplano es bastante simple. En el caso de dos dimensiones, el hiperplano se describe acorde a la ecuación de una recta (ecuación lineal).

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$$

Dados los parámetros β_0, β_1 y β_2, todos los pares de valores $\mathbf{x} = (x_1, x_2)$ para los que se cumple, la igualdad son puntos en el hiperplano. la ecuación es un punto en el hiperplano.

En un espacio de 3 dimensiones el hiperplano se corresponde con un sub-espacio de dos dimensiones, es decir, un plano. Si p>3 puede resultar difícil visualizar el hiperplano.

Esta ecuación puede generalizarse para p-dimensiones:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p = 0$$

Donde, los puntos definidos por el vector ($\mathbf{x} = x_1, x_2, \ldots, x_p$) que cumplen la ecuación pertenecen al hiperplano.

Cuando X no satisface la ecuación, se da los siguientes dos casos:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p < 0$$

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p > 0$$

el punto x estará situado a uno u otro lado del hiperplano no sobre él.

Así pues, se puede entender que un hiperplano divide un espacio p-dimensional en dos mitades. Para saber en qué lado del hiperplano se encuentra un determinado punto x, solo hay que calcular el signo de la ecuación.

La siguiente imagen muestra el hiperplano de un espacio bidimensional. La ecuación que describe el hiperplano (una recta) es $1 + 2x_1 + 3x_2 = 0$. La región azul representa el espacio en el que se encuentran todos los puntos para los que $1 + 2x_1 + 3x_2 > 0$ y la región roja el de los puntos para los que $1 + 2x_1 + 3x_2 < 0$.

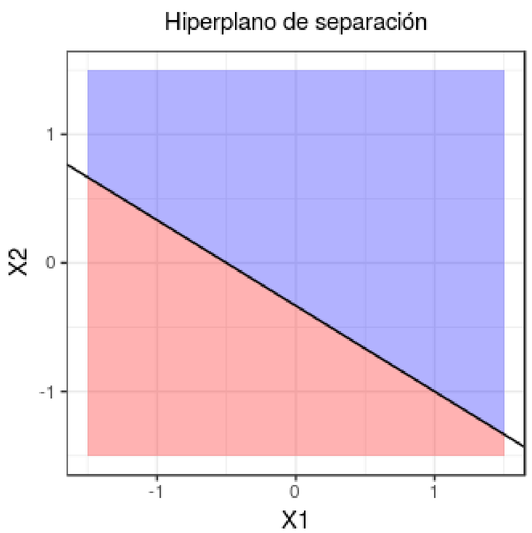


Figura 1. Hiperplano de un espacio bidimensional.

3. Clasificación binaria empleando un hiperplano

Cuando se dispone de n observaciones, cada una con p predictores y cuya **variable respuesta tiene dos niveles** (de aquí en adelante identificados como +1 y −1), se pueden emplear hiperplanos para construir un clasificador que permita predecir a que grupo pertenece una observación en función de sus predictores. Este mismo problema puede abordarse también con otros métodos (regresión logística, LDA, árboles de clasificación...) cada uno con ventajas y desventajas.

Para facilitar la comprensión, las siguientes explicaciones se basan en un espacio de dos dimensiones, donde un hiperplano es una recta. Sin embargo, los mismos conceptos son aplicables a dimensiones superiores.

4. Casos perfectamente separables linealmente. Uso del hiperplano para clasificación binaria.

Si la distribución de las observaciones es tal que se pueden separar linealmente de forma perfecta en las dos clases (+1 y −1), entonces, un hiperplano de separación cumple que:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p > 0, si\ y_i = 1$$

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p < 0, si\ y_i - 1$$

Al identificar cada clase como +1 o −1, y dado que multiplicar dos valores negativos resultan en un valor positivo, las dos condiciones anteriores pueden simplificarse en una única:

$$y_i(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p) > 0, para\ i = 1 \ldots n$$

Bajo este escenario, el clasificador más sencillo consiste en asignar cada observación a una clase dependiendo del lado del hiperplano en el que se encuentre. Es decir, la observación $f(\hat{x}*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$. Si $f(x^*)$ es positiva, la observación se asigna a la clase +1, si es negativa, a la clase −1. Además, la magnitud de $f(x^*)$ permite saber cómo de lejos está la observación del hiperplano y con ello la confianza de la clasificación

La definición de hiperplano para casos perfectamente separables linealmente resulta en un número infinito de posibles hiperplanos, lo que hace necesario un método que permita seleccionar uno de ellos como clasificador óptimo.

Clasificación binaria para casos separables linealmente

Si nuestros datos son perfectamente separables mediante un hiperplano, entonces existirá un número infinito de tales hiperplanos (debido a la posibilidad de moverlos o rotarlos sin entrar en contacto con las observaciones). Esto hace que necesitemos un modo de decidir cuál de todos los hiperplanos posibles utilizar.

Generando datos bajo la distribución normal, 10 datos con media 10 y sd=1.

```
# generando un grafico de MSV,  caso separable linealmente
set.seed(25)
X1 <- rnorm(n = 10, mean = 2, sd = 1)
X2 <- rnorm(n = 10, mean = 2, sd = 1)
datai <- data.frame(x1,x2)
datai
```

```
          x1        x2
1  2.9189774 3.3586796
2  2.7821363 1.8972123
3  2.0745650 2.3876716
4  0.0106483 1.9461950
5  2.6198257 0.6229404
6  1.9438713 1.5850054
7  1.8442045 1.6057100
8  0.5292476 1.9406866
9  1.5218499 3.1000254
10 2.4179416 2.7631757
```

Ampliando la data a 20 observaciones y generando la variable dependiente Y.

```
observaciones <- data.frame(X1 = c(X1, X1 + 2), X2 = c(X2, X2 + 2) ,
                            clase = rep(c(1, -1), each = 10))
```

```
observaciones
```

```
##           X1         X2 clase
## 1  3.435938  1.9103892     1
## 2  1.752886  1.1900574     1
## 3  1.621777  1.1790036     1
## 4  2.575575  2.9992595     1
## 5  1.568690  1.8647223     1
## 6  2.008283  2.0358829     1
## 7  3.150476 -0.2111556     1
## 8  2.412125  2.4591216     1
## 9  2.886648  1.4581008     1
## 10 2.832673  3.1661034     1
## 11 5.435938  3.9103892    -1
## 12 3.752886  3.1900574    -1
## 13 3.621777  3.1790036    -1
## 14 4.575575  4.9992595    -1
## 15 3.568690  3.8647223    -1
## 16 4.008283  4.0358829    -1
## 17 5.150476  1.7888444    -1
## 18 4.412125  4.4591216    -1
## 19 4.886648  3.4581008    -1
## 20 4.832673  5.1661034    -1
```

Convirtiendo la variable Y en factor y graficando.

```
observaciones$clase <- as.factor(observaciones$clase)
str(observaciones)
```

```
'data.frame':	20 obs. of  3 variables:
 $ X1   : num  1.788 0.958 0.847 2.322 0.5 ...
 $ X2   : num  0.257 0.675 1.452 0.544 2.083 ...
 $ clase: Factor w/ 2 levels "-1","1": 2 2 2 2 2 2 2 2 2 2 ...
```

```
library(ggplot2)
ggplot() +
  geom_point(data = observaciones, aes(x = X1, y = X2, color = clase), size = 4) +
  geom_abline(intercept = 9, slope = -2) +
  geom_abline(intercept = 8.5, slope = -1.7) +
  geom_abline(intercept = 8, slope = -1.5) +
```

```
geom_abline(intercept = 6.5, slope = -1) +
geom_abline(intercept = 5.4, slope = -0.75) +
 theme_bw() +
labs(title = "5 Posibles hiperplanos de separacion") +
theme( legend.position = "none",
      plot.title = element_text(hjust = 0.5, size = 11))
```

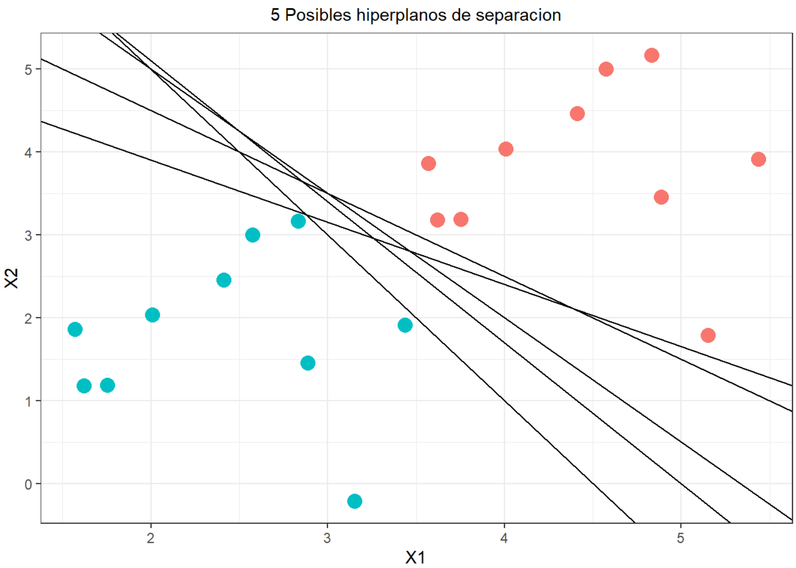


Figura. Linealmente separables, número infinito de hiperplanos

La solución a este problema consiste en seleccionar como clasificador óptimo al que se conoce como *maximal margin hyperplane* o hiperplano óptimo de separación, que se corresponde con el hiperplano que se encuentra más alejado de todas las observaciones de entrenamiento. Para obtenerlo, se tiene que calcular la distancia perpendicular de cada observación a un determinado hiperplano. La menor de estas distancias (conocida como margen) determina como de alejado está el hiperplano de las observaciones de entrenamiento. El maximal margin hyperplane se define como el hiperplano que consigue un mayor margen, es decir, que la distancia mínima entre el hiperplano y las observaciones es lo más grande posible. Aunque esta idea suena razonable, no es posible aplicarla, ya que habría infinitos hiperplanos contra los que medir las distancias. En su lugar, se recurre a métodos de optimización. Para encontrar una descripción más detallada de la solución por optimización consultar (Support Vector Machines Succinctly by Alexandre Kowalczyk).

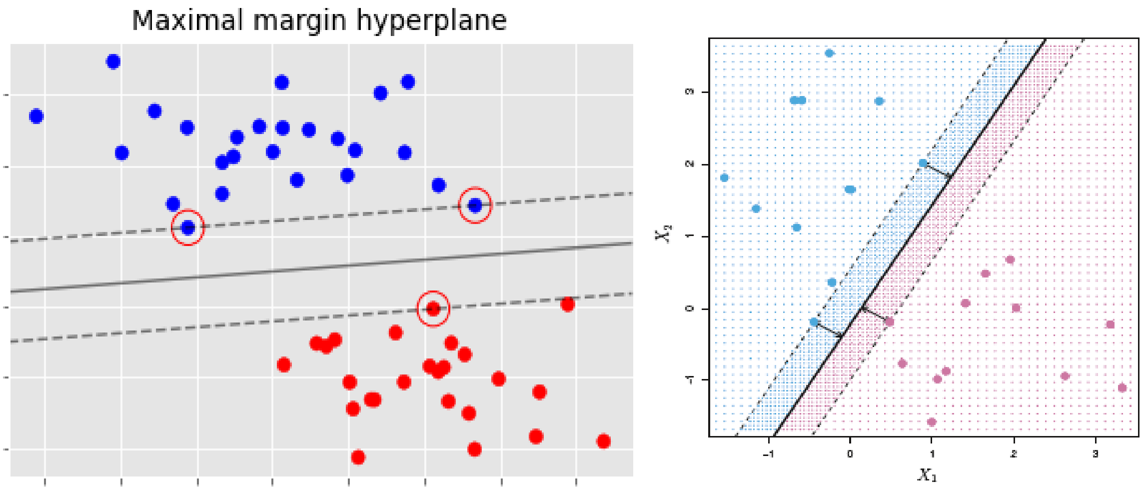


Figura. Imagen maximal margin hyperplane

La imagen anterior muestra el maximal margin hyperplane para un conjunto de datos de entrenamiento. Las tres observaciones equidistantes respecto al maximal margin hyperplane se encuentran a lo largo de las líneas discontinuas que indican la anchura del margen. A estas observaciones se les conoce como vectores soporte, ya que son vectores en un espacio p-dimensional y soportan (definen) el maximal margin hyperplane. Cualquier modificación en estas observaciones (vectores soporte) conlleva cambios en el maximal margin hyperplane. Sin embargo, modificaciones en observaciones que no son vector soporte no tienen impacto alguno en el hiperplano.

5. Clasificación binaria para casos cuasi-separables linealmente

El maximal margin hyperplane descrito en el apartado anterior es una forma muy simple y natural de clasificación siempre y cuando exista un hiperplano de separación. En la gran mayoría de casos reales, los datos no se pueden separar linealmente de forma perfecta, por lo que no existe un hiperplano de separación y no puede obtenerse un maximal margin hyperplane.

```
set.seed(101)
coordenadas <- matrix(rnorm(40), 20, 2)
coordenadas
```

```
##              [,1]        [,2]
##  [1,] -0.3260365 -0.1637557
##  [2,]  0.5524619  0.7085221
##  [3,] -0.6749438 -0.2679805
##  [4,]  0.2143595 -1.4639218
##  [5,]  0.3107692  0.7444358
##  [6,]  1.1739663 -1.4103902
##  [7,]  0.6187899  0.4670676
##  [8,] -0.1127343 -0.1193201
##  [9,]  0.9170283  0.4672390
## [10,] -0.2232594  0.4981356
## [11,]  0.5264481  0.8949372
```

```
## [12,] -0.7948444  0.2791520
## [13,]  1.4277555  1.0078658
## [14,] -1.4668197 -2.0731065
## [15,] -0.2366834  1.1898534
## [16,] -0.1933380 -0.7243742
## [17,] -0.8497547  0.1679838
## [18,]  0.0584655  0.9203352
## [19,] -0.8176704 -1.6716048
## [20,] -2.0503078  0.4484691
```

Generamos la variable Y

```
colnames(coordenadas) <- c("X1","X2")
y <- c(rep(-1,10), rep(1,10))
y
```

```
## [1] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1  1  1  1  1  1  1
```

```
# Y como factor
y <- as.factor(y)

coordenadas[y == 1, ] <- coordenadas[y == 1, ] + 1
datos <- data.frame(coordenadas, y)
datos

ggplot(data = datos, aes(x = X1, y = X2, color = as.factor(y))) +
  geom_point(size = 4) +
  theme_bw() +
  labs(title = "Clases no separables linealmente") +
  theme( legend.position = "none",
         plot.title = element_text(hjust = 0.5, size = 11))
```

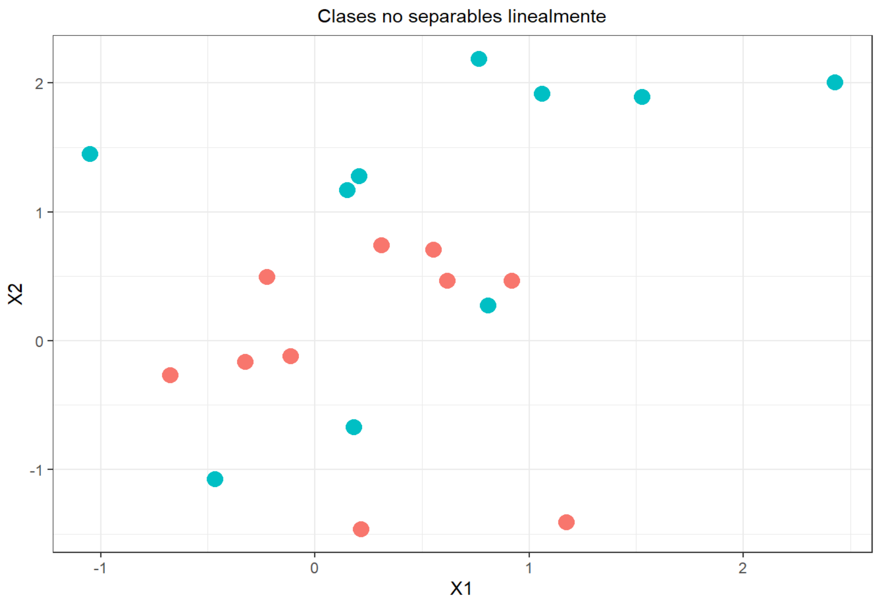


Figura. No clasificable linealmente

Para solucionar estas situaciones, se puede extender el concepto de maximal margin hyperplane para obtener un hiperplano que casi separe las clases, pero permitiendo que cometa unos pocos errores. A este tipo de hiperplano se le conoce como Support Vector Classifier o Soft Margin.

Support Vector Classifier o Soft Margin SVM

El Maximal Margin Classifier descrito en la sección anterior tiene poca aplicación práctica, ya que rara vez se encuentran casos en los que las clases sean perfecta y linealmente separables. De hecho, incluso cumpliéndose estas condiciones ideales, en las que exista un hiperplano capaz de separar perfectamente las observaciones en dos clases, esta aproximación sigue presentando dos inconvenientes:

- Dado que el hiperplano tiene que separar perfectamente las observaciones, es muy sensible a variaciones en los datos. Incluir una nueva observación puede suponer cambios muy grandes en el hiperplano de separación (poca robustez).
- Que el maximal margin hyperplane se ajuste perfectamente a las observaciones de entrenamiento para separarlas todas correctamente suele conllevar problemas de overfitting.

Por estas razones, es preferible crear un clasificador basado en un hiperplano que, aunque no separe perfectamente las dos clases, sea más robusto y tenga mayor capacidad predictiva al aplicarlo a nuevas observaciones (menos problemas de overfitting). Esto es exactamente lo que consiguen los clasificadores de vector soporte, también conocidos como soft margin classifiers o Support Vector Classifiers. Para lograrlo, en lugar de buscar el margen de clasificación más ancho posible que consigue que las observaciones estén en el lado correcto del margen; se permite que ciertas observaciones estén en el lado incorrecto del margen o incluso del hiperplano.

La siguiente imagen muestra un clasificador de vector soporte ajustado a un pequeño set de observaciones. La línea continua representa el hiperplano y las líneas discontinuas el margen a cada lado. Las observaciones 2, 3, 4, 5, 6, 7 y 10 se encuentran en el lado correcto del margen (también del hiperplano) por lo que están bien clasificadas. Las observaciones 1 y 8, a pesar de que se encuentran dentro del margen, están en el lado correcto del hiperplano, por lo que también están bien clasificadas. Las observaciones 11 y 12, se encuentran en el lado erróneo del hiperplano, su clasificación es incorrecta. Todas aquellas observaciones que, estando dentro o fuera del margen, se encuentren en el lado incorrecto del hiperplano, se corresponden con observaciones de entrenamiento mal clasificadas.

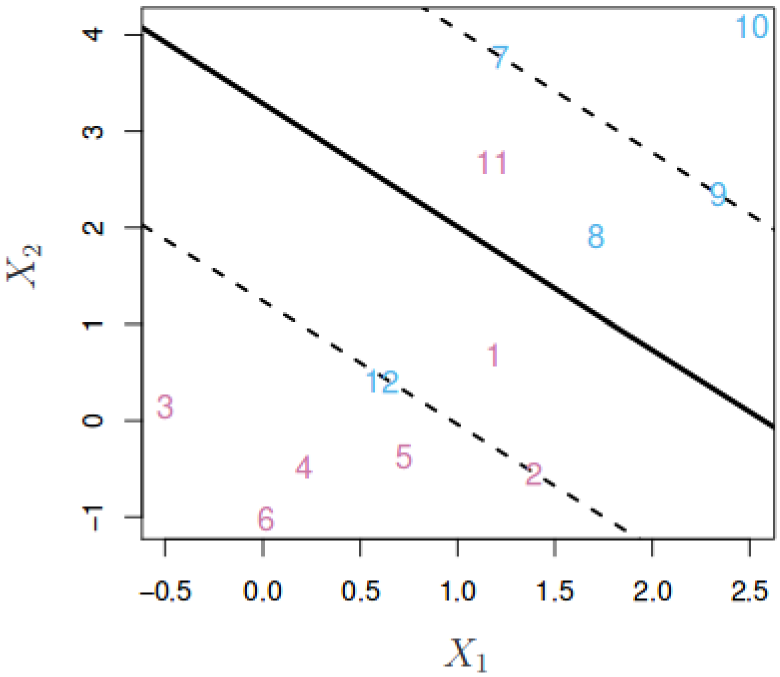


Figura .. Imagen clasificador vector soporte.

La identificación del hiperplano de un clasificador de vector soporte, que clasifique correctamente la mayoría de las observaciones a excepción de unas pocas, es un problema de optimización convexa. Si bien la demostración matemática queda fuera del objetivo de esta introducción, es importante mencionar que el proceso incluye un hiperparámetro de tuning C. C controla el número y severidad de las violaciones del margen (y del hiperplano) que se toleran en el proceso de ajuste. Si C=∞, no se permite ninguna violación del margen y, por lo tanto, el resultado es equivalente al Maximal Margin Classifier (teniendo en cuenta que esta solución solo es posible si las clases son perfectamente separables). Cuando más se aproxima C a cero, menos se penalizan los errores y más observaciones pueden estar en el lado incorrecto del margen o incluso del hiperplano. C es a fin de cuentas el hiperparámetro encargado de controlar el balance entre bias y varianza del modelo. En la práctica, su valor óptimo se identifica mediante cross-validation.

El proceso de optimización tiene la peculiaridad de que solo las observaciones que se encuentran justo en el margen o que lo violan influyen sobre el hiperplano. A estas observaciones se les conoce como vectores soporte y son las que definen el clasificador obtenido. Esta es la razón por la que el parámetro C controla el balance entre *bias* y varianza. Cuando el valor de C es pequeño, el margen es más ancho, y más observaciones violan el margen, convirtiéndose en vectores soporte. El hiperplano está, por lo tanto, sustentado por más observaciones, lo que aumenta el *bias* pero reduce la varianza. Cuando mayor es el valor de C, menor el margen, menos observaciones serán vectores soporte y el clasificador resultante tendrá menor *bias* pero mayor varianza.

Otra propiedad importante que deriva de que el hiperplano dependa únicamente de una pequeña proporción de observaciones (vectores soporte), es su robustez frente a observaciones muy alejadas del hiperplano. Esto hace al método de clasificación vector soporte distinto a otros métodos tales como Linear Discrimiant Analysis (LDA), donde la regla de clasificación depende de la media de todas las observaciones.

Nota: en el libro Introduction to Statistical Learning se emplea un término C que equivale a la inversa del descrito en este documento.

```
# Las máquinas de vector soporte son otro tipo de algoritmo de
# machine learning supervisado aplicable a problemas de regresión
# y clasificación, aunque se usa más comúnmente como modelo de clasificac
ión.
# Las máquinas de vector soporte suponen una generalización de un
# clasificador simple denominado maximal margin classifier.
# Sin embargo, este clasificador no puede aplicarse a sets de datos
# donde las clases de la variable respuesta no son separables
# mediante un límite lineal.
# Una extensión del mismo, el support vector classifier es aplicable
# en un mayor rango de casos. El support vector machine supone una
# extensión más del support vector classifier para casos con límites
# no lineales entre clases (clasificación binaria o de más clases).

# Support vector classifier/machine: library(e1071)
# svm() -> Ajuste de modelo support vector classifier
# (kernel = "linear","polinomial", "radial"…....).
# Si la variable respuesta contiene más de dos niveles,
# la función lleva a cabo la clasificación usando el método one-vs-one

# tune() -> Función genérica para optimización de hiperparámetros
# mediante validación cruzada
```

```
library(e1071)
# Estructura de datos
```

```r
datos <- read.csv("clases de repaso.csv", head=T, sep=",")
head(datos)
```

```
##   clases_repaso sexo examen_lectura examen_letras
## 1             0    1             91            87
## 2             0    1             60            78
## 3             0    0             70            60
## 4             0    0             54            54
## 5             0    0             50            60
## 6             0    1             62            58
```

```r
# etiquetando las categorias par efectos de obtener tablas
datos$clases_repaso <- factor(datos$clases_repaso,
                              levels = c("0","1"),labels=c("no.asista", "
asista"))
datos$sexo <- factor(datos$sexo,levels = c("0","1"),labels=c("Mujer", "Ho
mbre"))
str(datos)
```

```
## 'data.frame':    189 obs. of  4 variables:
##  $ clases_repaso : Factor w/ 2 levels "no.asista","asista": 1 1 1 1 1
1 1 1 1 1 ...
##  $ sexo          : Factor w/ 2 levels "Mujer","Hombre": 2 2 1 1 1 2 1
2 1 1 ...
##  $ examen_lectura: int  91 60 70 54 50 62 59 60 45 60 ...
##  $ examen_letras : int  87 78 60 54 60 58 55 49 65 60 ...
```

```r
head(datos)
```

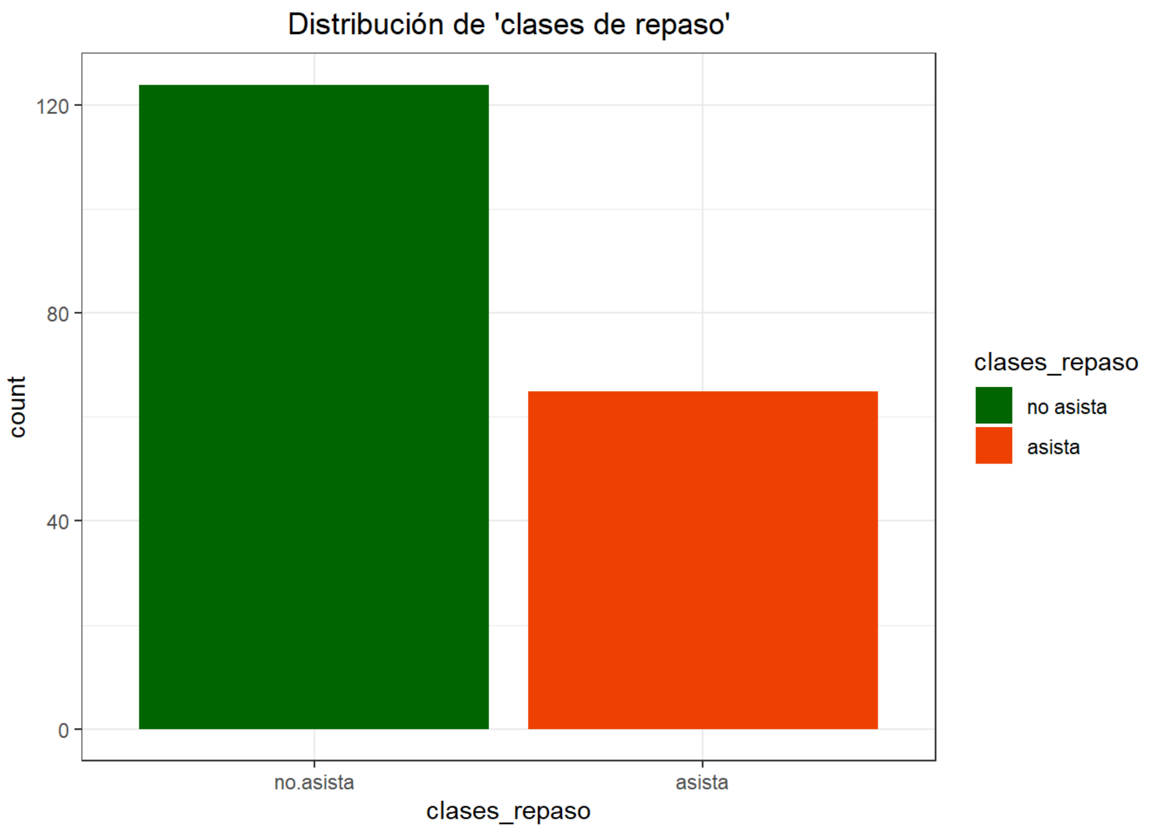
```
##   clases_repaso   sexo examen_lectura examen_letras
## 1     no.asista Hombre             91            87
## 2     no.asista Hombre             60            78
## 3     no.asista  Mujer             70            60
## 4     no.asista  Mujer             54            54
```

```
## 5      no.asista  Mujer                50            60
## 6      no.asista Hombre                62            58
```

```r
# Comprobamos valores faltantes variable respuesta
sum(is.na(datos$ clases_repaso))
```

```
## [1] 0
```

```r
# Distribución variable respuesta
library(ggplot2)
ggplot(data = datos, aes(x = clases_repaso, y = ..count.., fill = clases_
repaso)) +
  geom_bar() +
  labs(title = "Distribución de 'clases de repaso'") +
  scale_fill_manual(values = c("darkgreen", "orangered2"),
                    labels = c("no asista", "asista")) +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5))
```



```r
# Tabla frecuencias variable respuesta
table(datos$clases_repaso)
```

```
##
## no.asista    asista
##       124        65
```

```r
# Tabla proporciones variable respuesta
library(dplyr)
```

```r
prop.table(table(datos$clases_repaso)) %>% round(digits = 2)
```

```
## no.asista    asista
##      0.66      0.34
```

```r
# creacion de datos de entrenamiento y validacion
ind <- sample(2,nrow(datos), replace=TRUE, prob=c(0.8, 0.2))
train <-datos[ind==1,]
test <-datos[ind==2,]
str(train)
```

```
## 'data.frame':    159 obs. of  4 variables:
##  $ clases_repaso : Factor w/ 2 levels "no.asista","asista": 1 1 1 1 1
1 1 1 1 1 ...
##  $ sexo          : Factor w/ 2 levels "Mujer","Hombre": 2 2 1 2 1 2 1
1 2 2 ...
##  $ examen_lectura: int  91 60 50 62 59 60 45 60 40 75 ...
##  $ examen_letras : int  87 78 60 58 55 49 65 60 50 80 ...
```

```r
str(test)
```

```
## 'data.frame':    30 obs. of  4 variables:
##  $ clases_repaso : Factor w/ 2 levels "no.asista","asista": 1 1 1 1 1
1 1 1 1 2 ...
##  $ sexo          : Factor w/ 2 levels "Mujer","Hombre": 1 1 2 1 2 1 2
2 1 2 ...
##  $ examen_lectura: int  70 54 60 75 84 69 65 66 65 55 ...
##  $ examen_letras : int  60 54 55 80 70 50 70 70 60 45 ...
```

```
# Support vector classifier
# Indicar la funcion kernel."linear", polynomial, radial,sigmoid"


modelo <- svm(clases_repaso ~ sexo + examen_letras + examen_lectura, data
= train,
              kernel = "linear",
              #cost = 10,
              scale = TRUE)
summary(modelo)
```

```
## Call:
## svm(formula = clases_repaso ~ sexo + examen_letras + examen_lectura,
##     data = train, kernel = "linear", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  120
##
##  ( 63 57 )
##
##
## Number of Classes:  2
##
## Levels:
```

```
##  no.asista asista
```

```
# coeficientes del modelo
coef(modelo)
```

```
##    (Intercept)           sexo  examen_letras examen_lectura
<NA>
##   9.999557e-01   6.749117e-05  -6.749117e-05  -4.590282e-04   2.790986
e-04
```

```
# Indice de las observaciones que actuan como vector soporte
modelo$index
```

```
##   [1]   1   2   3   6   7   9  10  11  12  13  14  15  16  18  19  20
21  22
##  [19]  23  25  26  27  28  33  34  35  36  38  39  41  42  44  46  48
49  52
##  [37]  53  54  56  58  59  61  65  67  68  69  70  71  72  74  75  81
82  83
##  [55]  84  85  86  89  96  98 100 103 105  47  50  95 106 107 108 109
110 111
##  [73] 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
128 129
##  [91] 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145
146 147
## [109] 148 149 150 151 152 153 154 155 156 157 158 159
```

```
# Prediccion segun el modelo
# prediction <-predict(modelo,newdata=test[-1])
# head(prediction)
prediction <-predict(modelo,test)
prediction
```

```
##         3         4        19        28        29        37        43
48
```

```
## no.asista no.asista no.asista no.asista no.asista no.asista no.asista
no.asista
##        53        57        58        59        62        64        71
81
## no.asista no.asista no.asista no.asista no.asista no.asista no.asista
no.asista
##        89        91        94       103       104       112       113
117
## no.asista no.asista no.asista no.asista no.asista no.asista no.asista
no.asista
##       130       141       149       182       185       187
## no.asista no.asista no.asista no.asista no.asista no.asista
## Levels: no.asista asista
```

```
# Evaluar la precisión del modelo
precision <- mean(prediction == test$clases_repaso)
print(precision)
```

```
## [1] 0.7333333
```

```
# optimizando con la funcion tune
# A la hora de ajustar un support vector classifier,
# es importante tener en cuenta que el hiperparámetro C
# (cost) controla el equilibrio bias-varianza y la capacidad
# predictiva del modelo, ya que determina la severidad
# permitida respecto a las violaciones sobre el margen.
# En otras palabras, necesitamos fijar un margen de separación
# entre observaciones a priori. Por ello es recomendable
# evaluar distintos valores del mismo mediante validación
# cruzada y escoger el valor óptimo.

# Estandarizar los predictores cuando no estén medidos
# en la misma escala, para que los de mayor magnitud no
# tengan mayor influencia que el resto. Un argumento disponible
# en la función svm() para ello es scale = TRUE).

# Para ajustar un support vector classifier, el kernel
```

```r
# indicado en la función svm() ha de ser lineal.
# Obtendremos un valor de coste óptimo mediante
# validación cruzada utilizando la función tune() del paquete e1071:
```

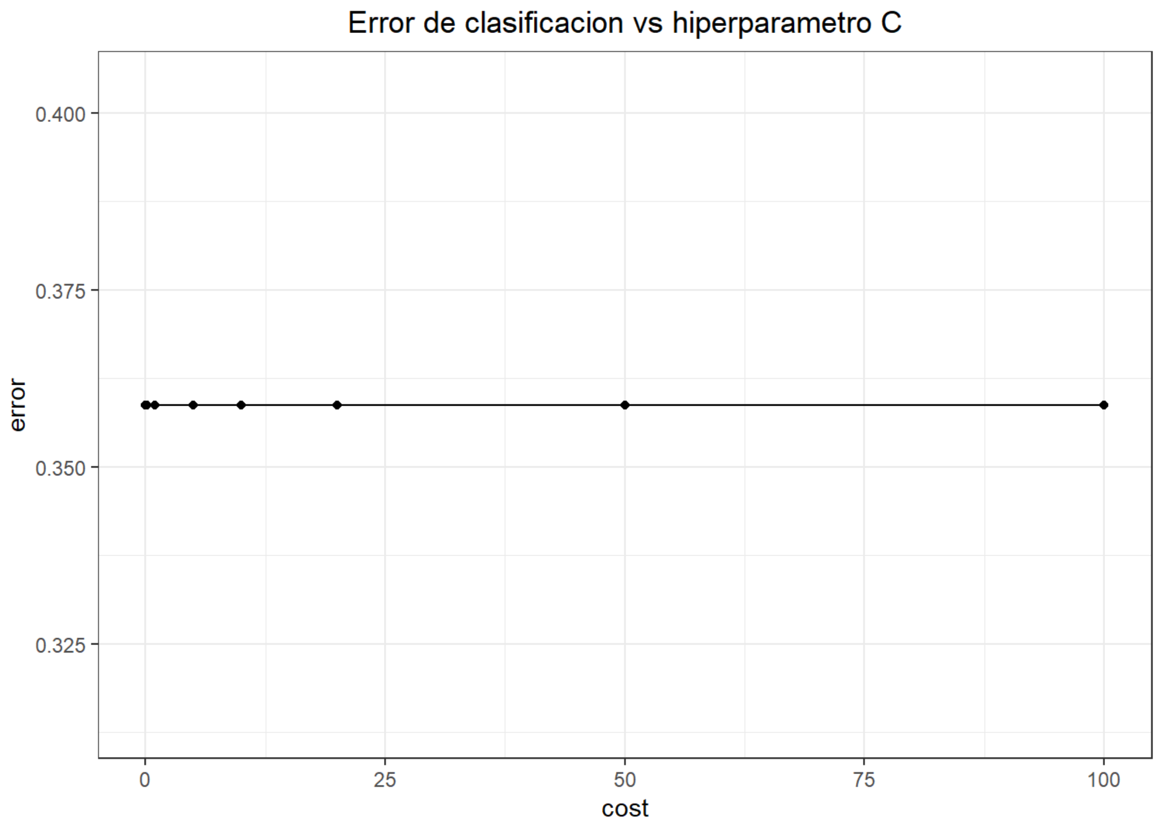
```r
set.seed(1)

# Optimización de hiperparámetros mediante validación cruzada 10-fold
tuning <- tune(svm, clases_repaso ~ sexo + examen_letras + examen_lectura
,
               data = train, kernel = "linear",
               ranges = list(cost = c(0.001,0.01, 0.1, 1, 5, 10, 20, 50,
100)),
               scale=TRUE)
summary(tuning)
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.001
##
## - best performance: 0.35875
##
## - Detailed performance results:
##    cost   error dispersion
## 1 1e-03 0.35875  0.1030861
## 2 1e-02 0.35875  0.1030861
## 3 1e-01 0.35875  0.1030861
## 4 1e+00 0.35875  0.1030861
## 5 5e+00 0.35875  0.1030861
## 6 1e+01 0.35875  0.1030861
## 7 2e+01 0.35875  0.1030861
## 8 5e+01 0.35875  0.1030861
## 9 1e+02 0.35875  0.1030861
```

```
names(tuning)
```

```
## [1] "best.parameters"  "best.performance" "method"           "nparcomb
"
## [5] "train.ind"        "sampling"         "performances"     "best.mod
el"
```

```
ggplot(data = tuning$performances, aes(x = cost, y = error)) +
  geom_line() +
  geom_point() +
  labs(title = "Error de clasificacion vs hiperparametro C") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```




```
mejor_modelo <- tuning$best.model
summary(mejor_modelo)
```

```
## Call:
## best.tune(METHOD = svm, train.x = clases_repaso ~ sexo + examen_letras 
+ 
##     examen_lectura, data = train, ranges = list(cost = c(0.001, 0.01, 
##     0.1, 1, 5, 10, 20, 50, 100)), kernel = "linear", scale = TRUE)
## 
## 
## Parameters:
##    SVM-Type:  C-classification 
##  SVM-Kernel:  linear 
##        cost:  0.001 
## 
## Number of Support Vectors:  118
## 
##  ( 61 57 )
## 
## 
## Number of Classes:  2 
## 
## Levels: 
##  no.asista asista
```

```
# modelo
coef(mejor_modelo)
```

```
##    (Intercept)           sexo  examen_letras examen_lectura           
<NA> 
##   0.9999426827   0.0001434212  -0.0001434212  -0.0003634296   0.000260
5330
```

```
# El mejor modelo obtenido sería equivalente a ajustar:


modelo_svc <- svm(clases_repaso ~ ., data = train,
                  kernel = "linear",
```

```r
                  cost = 0.001,
                  scale = TRUE)
summary(modelo_svc)
```

```
## Call:
## svm(formula = clases_repaso ~ ., data = train, kernel = "linear",
##     cost = 0.001, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
##
## Number of Support Vectors:  118
##
##  ( 61 57 )
##
##
## Number of Classes:  2
##
## Levels:
##  no.asista asista
```

```r
# Predicciones
predicciones <- predict(object = modelo_svc, test)

# observaciones mal clasificadas
paste("Error de test:", 100*mean(test$clases_repaso != predicciones),"%")
```

```
## [1] "Error de test: 26.6666666666667 %"
```

```r
# matriz de confusion
```

```r
table(predicción = predicciones, valor_real = test$clases_repaso)
```

```
##             valor_real
## predicción no.asista asista
##   no.asista        22      8
##   asista            0      0
```

```r
######table(pred, y)
resultado <- cbind(test,predicciones)
head(resultado)
```

```
##      clases_repaso   sexo examen_lectura examen_letras predicciones
## 3        no.asista  Mujer             70            60    no.asista
## 4        no.asista  Mujer             54            54    no.asista
## 19       no.asista Hombre             60            55    no.asista
## 28       no.asista  Mujer             75            80    no.asista
## 29       no.asista Hombre             84            70    no.asista
## 37       no.asista  Mujer             69            50    no.asista
```

```r
# UTILIZANDO EL PAQUETE CARET
# AJUSTE DEL MODELO
# Configuración del proceso de selección del modelo
library(caret)
```

```r
fitControl <- trainControl(method = "cv",
                           number = 10,
                           classProbs = TRUE,
                           search = "grid")

# Parametros del modelo disponibles
getModelInfo(model = "svmLinear")[[2]]$parameters
```

```
##   parameter   class label
## 1         C numeric  Cost
```

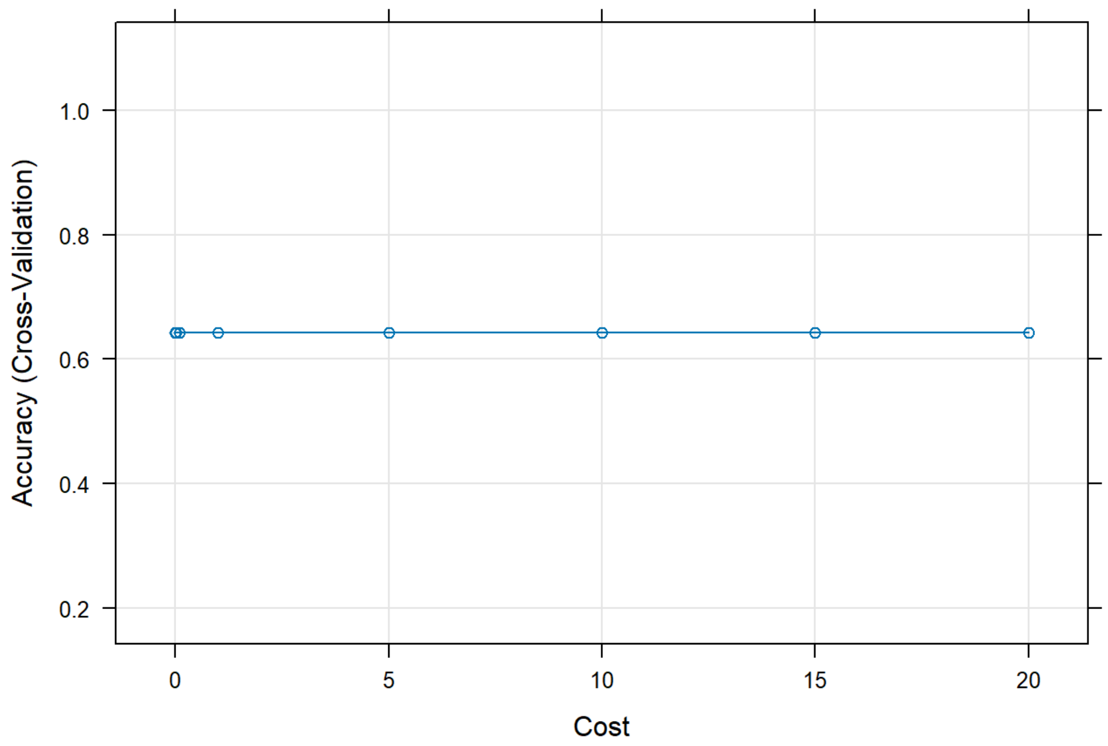
```r
# Valores del hiperparámetro C a evaluar
grid_C <- data.frame(C = c(0.001, 0.01, 0.1, 1, 5, 10, 15, 20))

# Entrenamiento del SVM con un kernel lineal y optimización del hiperpará
metro C
set.seed(325) # misma semilla que en el ejemplo con el paquete e1071
modelo_svc2 <- train(clases_repaso ~ ., data = train,
                     method = "svmLinear",
                     trControl = fitControl,
                     preProc = c("center", "scale"), #estandarizacion de l
os datos
                     tuneGrid = grid_C)
```

```r
# Resultado del entrenamiento
modelo_svc2
```

```
## Support Vector Machines with Linear Kernel
##
## 159 samples
##   3 predictor
##   2 classes: 'no.asista', 'asista'
##
## Pre-processing: centered (3), scaled (3)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 143, 142, 143, 144, 143, 144, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##    0.001  0.6419118  0
##    0.010  0.6419118  0
##    0.100  0.6419118  0
##    1.000  0.6419118  0
```

```
##     5.000  0.6419118  0
##    10.000  0.6419118  0
##    15.000  0.6419118  0
##    20.000  0.6419118  0
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.001.
```

```
# Evolución del accuracy en funcion del valor de coste en validacion cruzada
plot(modelo_svc2)
```




```
# EVALUACIÓN DEL MODELO
confusionMatrix(predict(modelo_svc2, test), test$clases_repaso)
```

```
# Confusion Matrix and Statistics
##
##              Reference
```

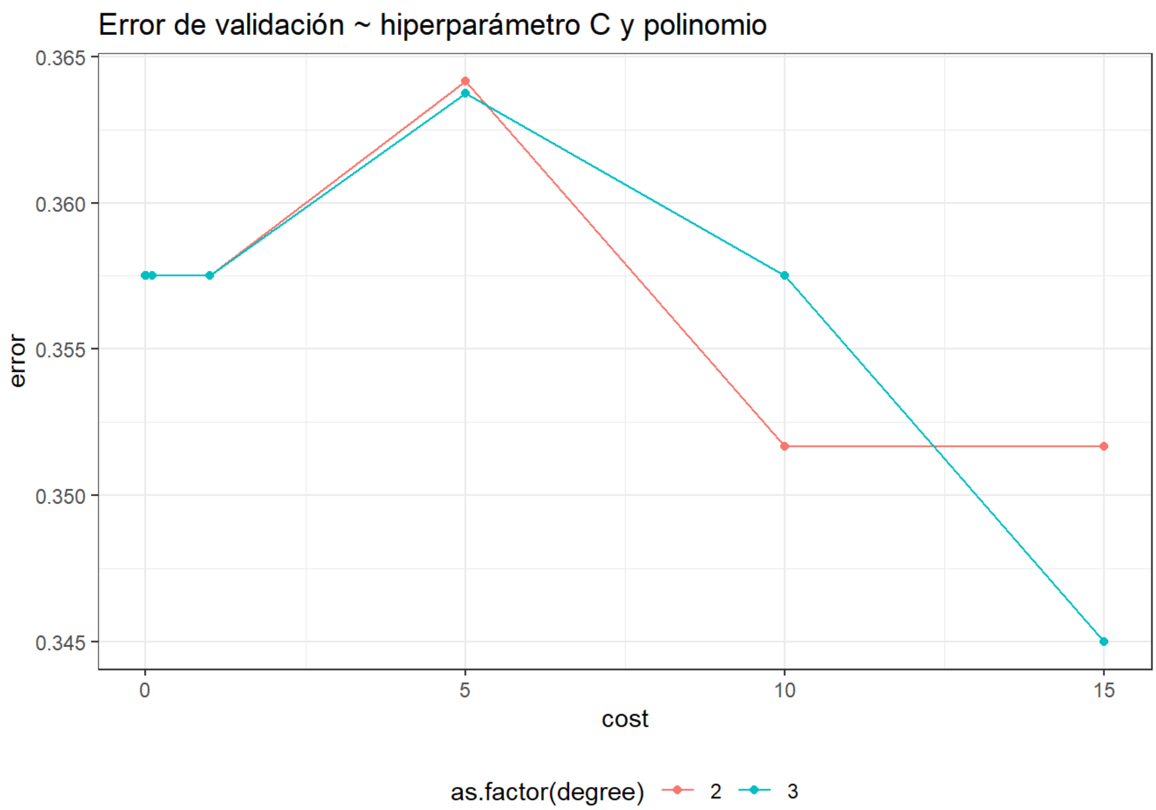
```
## Prediction  no.asista asista
##   no.asista        22      8
##   asista            0      0
##
##                Accuracy : 0.7333
##                  95% CI : (0.5411, 0.8772)
##     No Information Rate : 0.7333
##     P-Value [Acc > NIR] : 0.59367
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7333
##          Neg Pred Value :    NaN
##              Prevalence : 0.7333
##          Detection Rate : 0.7333
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : no.asista
```

```r
### Kernel polinomico
set.seed(325)
tuning <- tune(svm, clases_repaso ~ ., data = train,
               kernel = "polynomial",
               ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 15),
                             degree = c(2, 3)),
               scale = TRUE)


summary(tuning)
```

```
## Parameter tuning of 'svm':
##
```

```
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    15      3
##
## - best performance: 0.345
##
## - Detailed performance results:
##      cost degree     error dispersion
## 1   0.001      2 0.3575000  0.1000347
## 2   0.010      2 0.3575000  0.1000347
## 3   0.100      2 0.3575000  0.1000347
## 4   1.000      2 0.3575000  0.1000347
## 5   5.000      2 0.3641667  0.1154400
## 6  10.000      2 0.3516667  0.1133660
## 7  15.000      2 0.3516667  0.1133660
## 8   0.001      3 0.3575000  0.1000347
## 9   0.010      3 0.3575000  0.1000347
## 10  0.100      3 0.3575000  0.1000347
## 11  1.000      3 0.3575000  0.1233502
## 12  5.000      3 0.3637500  0.1258926
## 13 10.000      3 0.3575000  0.1334895
## 14 15.000      3 0.3450000  0.1342676
```

```
ggplot(data = tuning$performances, aes(x = cost, y = error, col = as.fact
or(degree))) +
  geom_line() +
  geom_point() +
  labs(title = "Error de validación ~ hiperparámetro C y polinomio") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme_bw() + theme(legend.position = "bottom")
```




```r
# Modelo SVM kernel polinómico
modelo_svmP <- svm(clases_repaso ~ ., data = train,
                   kernel = "polynomial",
                   cost = 15,
                   degree = 2,
                   scale = TRUE)

summary(modelo_svmP)
```

```
## Call:
## svm(formula = clases_repaso ~ ., data = train, kernel = "polynomial",
##     cost = 15, degree = 2, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  15
##      degree:  2
```

```
##      coef.0:  0
##
## Number of Support Vectors:  117
##
##  ( 61 56 )
##
##
## Number of Classes:  2
##
## Levels:
##  no.asista asista
```

```
# Evaluación del modelo
confusionMatrix(predict(modelo_svmP, test), test$clases_repaso)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction  no.asista asista
##   no.asista        21      8
##   asista            1      0
##
##                Accuracy : 0.7
##                  95% CI : (0.506, 0.8527)
##     No Information Rate : 0.7333
##     P-Value [Acc > NIR] : 0.7384
##
##                   Kappa : -0.063
##
##  Mcnemar's Test P-Value : 0.0455
##
##             Sensitivity : 0.9545
##             Specificity : 0.0000
##          Pos Pred Value : 0.7241
##          Neg Pred Value : 0.0000
##              Prevalence : 0.7333
```

```
##          Detection Rate : 0.7000
##    Detection Prevalence : 0.9667
##       Balanced Accuracy : 0.4773
##
##        'Positive' Class : no.asista
```

```
paste("Observaciones de test mal clasificadas:",
      100 * mean(test$clases_repaso != predict(modelo_svmP, test)) %>%
        round(digits = 4), "%")
```

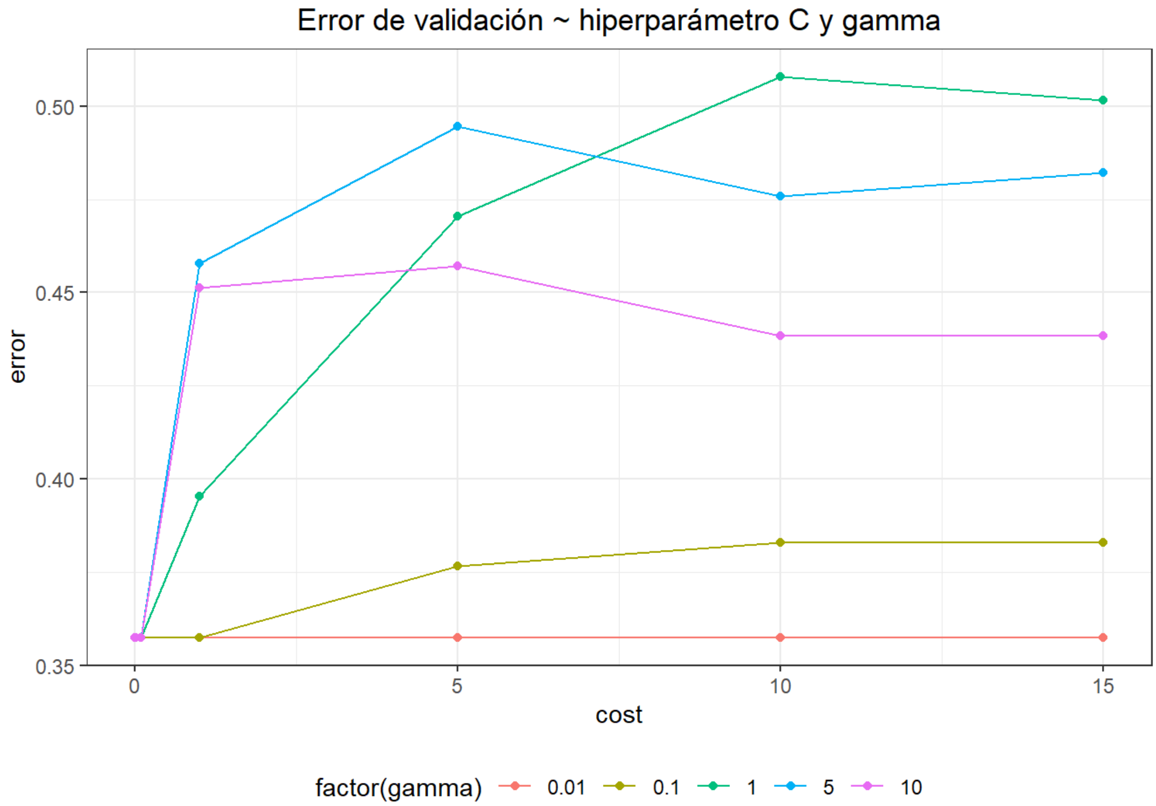
```
## [1] "Observaciones de test mal clasificadas: 30 %"
```

```
##Kernel radial
set.seed(325)
tuning <- tune(svm, clases_repaso ~ ., data = train,
               kernel = "radial",
               ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 15),
                             gamma = c(0.01, 0.1, 1, 5, 10)),
               scale = TRUE)

summary(tuning)
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost gamma
##  0.001  0.01
##
## - best performance: 0.3575
##
## - Detailed performance results:
##       cost gamma     error dispersion
```

```
## 1    0.001  0.01 0.3575000 0.10003472
## 2    0.010  0.01 0.3575000 0.10003472
## 3    0.100  0.01 0.3575000 0.10003472
## 4    1.000  0.01 0.3575000 0.10003472
## 5    5.000  0.01 0.3575000 0.10003472
## 6   10.000  0.01 0.3575000 0.10003472
## 7   15.000  0.01 0.3575000 0.10003472
## 8    0.001  0.10 0.3575000 0.10003472
## 9    0.010  0.10 0.3575000 0.10003472
## 10   0.100  0.10 0.3575000 0.10003472
## 11   1.000  0.10 0.3575000 0.10003472
## 12   5.000  0.10 0.3766667 0.09080402
## 13 10.000  0.10 0.3829167 0.11381180
## 14 15.000  0.10 0.3829167 0.11381180
## 15   0.001  1.00 0.3575000 0.10003472
## 16   0.010  1.00 0.3575000 0.10003472
## 17   0.100  1.00 0.3575000 0.10003472
## 18   1.000  1.00 0.3954167 0.11956470
## 19   5.000  1.00 0.4704167 0.12271759
## 20 10.000  1.00 0.5079167 0.11557776
## 21 15.000  1.00 0.5016667 0.12487957
## 22   0.001  5.00 0.3575000 0.10003472
## 23   0.010  5.00 0.3575000 0.10003472
## 24   0.100  5.00 0.3575000 0.10003472
## 25   1.000  5.00 0.4579167 0.09861013
## 26   5.000  5.00 0.4945833 0.16048790
## 27 10.000  5.00 0.4758333 0.17790577
## 28 15.000  5.00 0.4820833 0.17504464
## 29   0.001 10.00 0.3575000 0.10003472
## 30   0.010 10.00 0.3575000 0.10003472
## 31   0.100 10.00 0.3575000 0.10003472
## 32   1.000 10.00 0.4512500 0.12835092
## 33   5.000 10.00 0.4570833 0.17014938
## 34 10.000 10.00 0.4383333 0.16228975
## 35 15.000 10.00 0.4383333 0.15684977
```

```
ggplot(data = tuning$performances, aes(x = cost, y = error, color = facto
r(gamma))) +
  geom_line() +
  geom_point() +
  labs(title = "Error de validación ~ hiperparámetro C y gamma") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "bottom")
```




```
# Modelo SVM con kernel radial
modelo_svmR <- svm(clases_repaso ~ . , data = train,
                   kernel = "radial",
                   cost = 5,
                   gamma = 0.01,
                   scale = TRUE)

# Matriz de confusion y métricas en test
confusionMatrix(predict(modelo_svmR, test), test$clases_repaso)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  no.asista asista
##   no.asista        22      8
##   asista            0      0
##
##                Accuracy : 0.7333
##                  95% CI : (0.5411, 0.8772)
##     No Information Rate : 0.7333
##     P-Value [Acc > NIR] : 0.59367
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.01333
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7333
##          Neg Pred Value :    NaN
##              Prevalence : 0.7333
##          Detection Rate : 0.7333
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : no.asista
```

```
paste("Observaciones de test mal clasificadas:",
      100 * mean(test$clases_repaso != predict(modelo_svmR, test)) %>%
        round(digits = 4), "%")
```

```
## [1] "Observaciones de test mal clasificadas: 26.67 %"
```