Regresión Logística

Laura Zapata Prada

Teoria

Para el modelo de regresión logística se tiene una variable respuesta "Y" de tipo binaria (0,1).

"y"=1; ocurre el evento de interés (si) **"y"=0**; no ocurre el evento de interés (no)

Se usa una distribución Bernoulli con parámetro theta (θ) como modelo probabilístico, donde θ representa la probabilidad de que "y"=1.

Proceso General: Observo sí o no en la variable respuesta (y), tengo unas variables independientes asociadas a esta respuesta (x1…xk), ajusto un modelo de regresión logística, y con este puedo calcular la probabilidad de que "y"=1.

En general el modelo de regresión logística es fundamental para identificar variables que afectan o no la probabilidad de que ocurra un evento de interés.

Este modelo no permite la clasificación de individuos, simplemente permite hacer predicción.

Queremos saber a partir de que valor de θ podemos afirmar que "y"=1.

Para decir que el evento predicho es un éxito, se debe calcular un valor de "θ" que se llama "θ óptimo", lo cual se hace mediante el uso de diferentes puntos de corte para θ y se construye una matriz de confusión (contrasta lo que predice mi modelo vs lo que observe en la realidad) para cada punto.

En términos generales: Si la probabilidad de que ocurra el evento (θ) dado las variables explicativas (x1…xk) es **MAYOR O IGUAL** que el punto de corte óptimo (θ óptimo), entonces, ese evento es igual a 1 ("y"=1).

En caso de que lo anterior sea **MENOR**, el evento será igual a 0 ("y"=0).

	REAL	
MODELO	✔ (1)	✘ (0)
✔ (1)	*A*	*B*
✘ (0)	*C*	*D*

Matriz de Confusión

A = Número de personas que el modelo dice que "y"=1 (ejemplo: que si me van a comprar) y que en la realidad fue "y"=1 (ejemplo: que si me compraron), es decir, que acerté.

B = Número de personas que el modelo dice que "y"=1 (ejemplo: que si me van a comprar) y que en la realidad fue "y"=0 (ejemplo: que no me compraron), es decir, que es un falso positivo.

C = Número de personas que el modelo dice que "y"=0 (ejemplo: que no me van a comprar) y que en la realidad fue "y"=1 (ejemplo: que si me compraron), es decir, que es un falso negativo.

D = Número de personas que el modelo dice que "y"=0 (ejemplo: que no me van a comprar) y que en la realidad fue "y"=0 (ejemplo: que no me compraron), es decir, que acerté.

En la realidad el número de personas con y=1 fueron A+C, y el número de personas con y=0 fueron B+D.

La Sensibilidad (Se) es una medida que me resume si mi modelo es capaz de identificar correctamente las personas con y=1 entre todas aquellas que realmente tuvieron un y=1. Su fórmula sería:

$$Se = \frac{A}{A + C}$$

La Especificidad (Sp) es una medida que me resume si mi modelo es capaz de identificar correctamente las personas con y=0 entre todas aquellas que realmente tuvieron un y=0. Su fórmula sería:

$$Sp = \frac{D}{B + D}$$

El Accuracy (Ac) es una medida de que tan bueno es mi modelo para predecir lo que estoy viendo. Su fórmula sería:

$$Ac = \frac{A + D}{A + B + C + D}$$

Donde “n” es el tamaño de muestra y se toma como la suma de A+B+C+D.

Determinar el “θ óptimo”

La idea es evaluar muchos puntos de corte (θ), luego para cada punto calculo la sensibilidad, especificidad y el accuracy para verlos gráficamente. Finalmente, para elegir el θ óptimo se puede suponer que el mejor será en donde la sensibilidad se parece a la especificidad. Lo explicado se puede apreciar en la siguiente imagen:

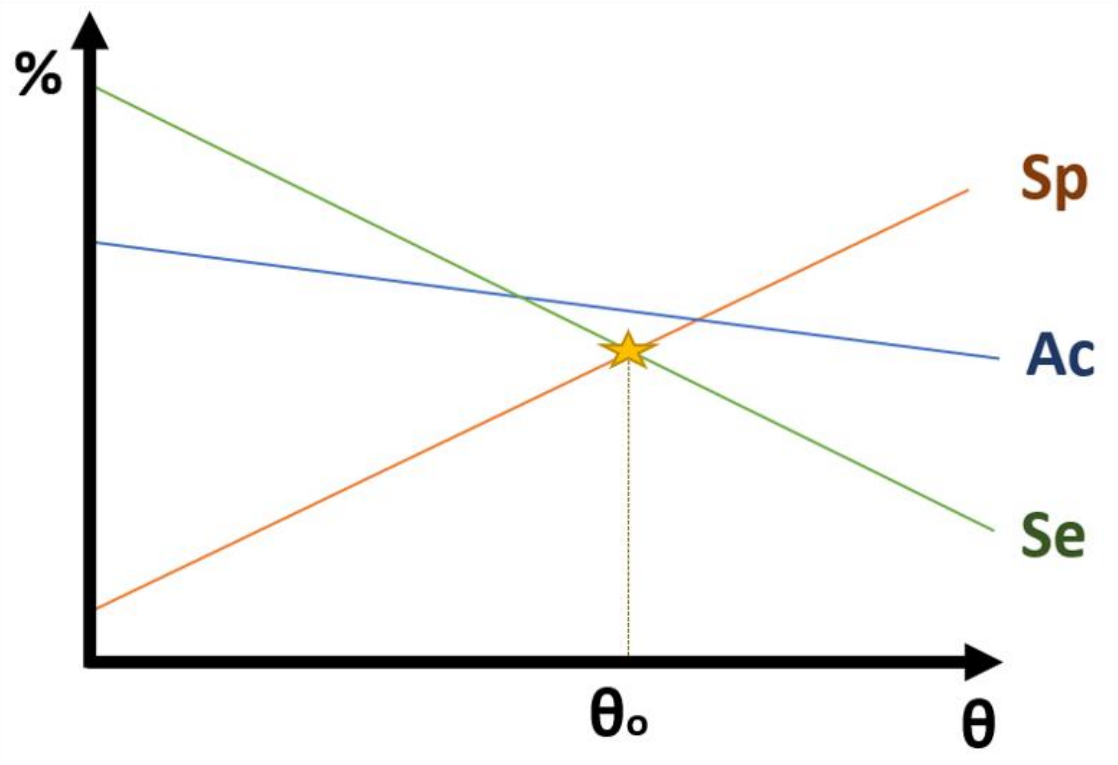


Gráfico para determinar θ óptimo

Otra forma para determinar el “θ óptimo” sería escoger aquel que tenga máximo accuracy, o podría elegir el que tenga máxima sensibilidad; el criterio para elegirlo varía dependiendo del contexto.

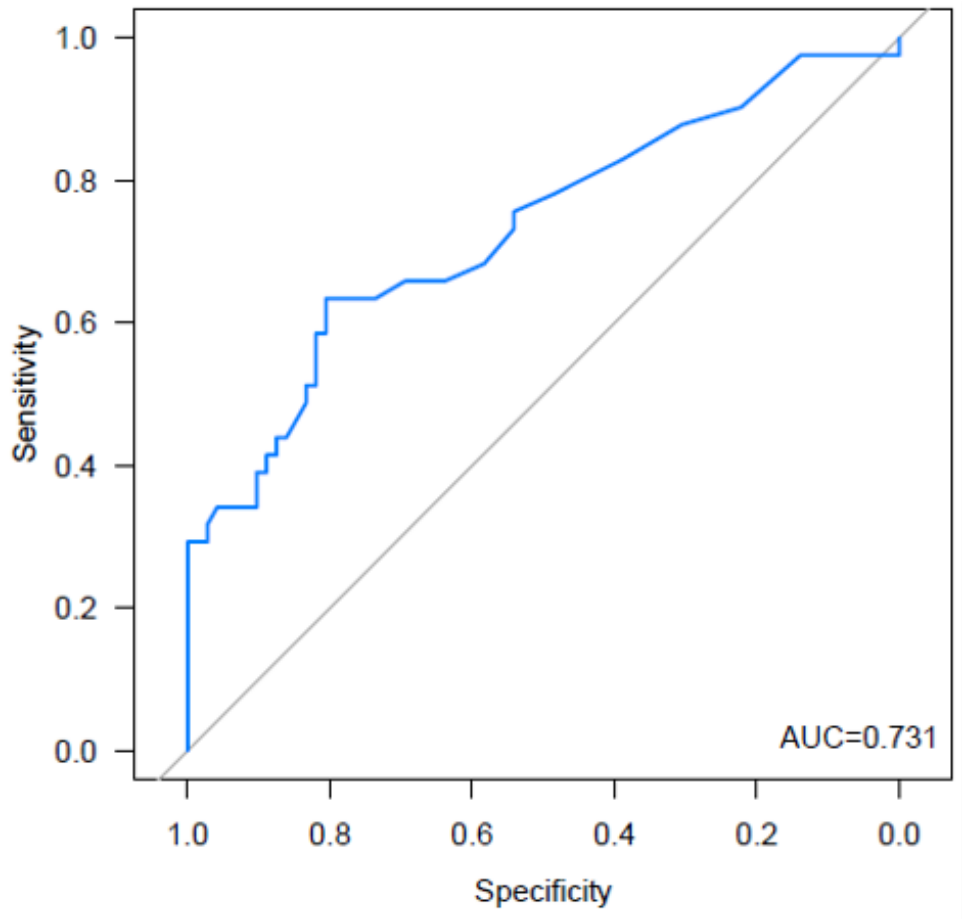
En caso de no querer elegir ninguno de los criterios anteriores se puede asumir que el “θ óptimo” es de 0.5

Para determinar que el modelo es bueno se hace a partir del análisis de una curva ROC o mediante la validación cruzada.

La validación cruzada es cuando se toma una muestra de los datos (aproximadamente el 70% de los datos), con esos se ajusta el modelo de regresión logística y luego se predice cual va a ser el evento de interés para el 30% restante, y

comparamos las predicciones que me dijo el modelo (entrenado con el 70%) con las predicciones que me da con el 30% que no use para el entrenamiento; al contrastar obtenemos una medida del desempeño del modelo ajustado para datos que nunca ha visto.

Cuando se realiza la **Curva ROC** y se obtiene el AUC (área bajo la curva), donde 0<=AUC<=1, mientras más cercano sea el AUC a 1, mejor será mi modelo. Con eso se podría decir que mi modelo es muy bueno para diferenciar personas con y=1 de las que son y=0, (ejemplo: diferenciar personas que me compran de personas que no me compran)



Ejemplo Curva ROC

Otra cosa a tener en cuenta es que se puede pasar de probabilidades predichas al **Odds Ratio** (en donde la cantidad de e^b1 es el cambio asociado en el odds de y=1 por cada unidad en z), es decir, si deseas saber cómo la variable controlable afecta a la probabilidad de que ocurra el evento (θ), se debe trabajar con el odds ratio. Por lo que, así como calculamos los betas para cada predictor, debemos calcular el Odds Ratio para cada uno.

*Si el Beta es cero, el Odds Ratio será 1.

*Si el Beta es negativo, el Odds Ratio será menor 1.

*Si el Beta es positivo, el Odds Ratio será mayor 1.

*El Odds Ratio nunca es cero.

Ejercicio en RStudio

Lectura de datos

Iniciamos con la lectura de datos, la cual será guardada con el nombre de "datos" y en este caso es extraído de una ruta en mi ordenador personal.

```
datos<- read.table("C:/Users/acer/Downloads/logistic.txt",header = TRUE)
```

Teniendo claro que la variable respuesta (y) es binaria (0,1), podemos hacer una tabla de la cantidad de veces que se repiten estos valores en los datos obtenidos y sus proporciones respectivas.

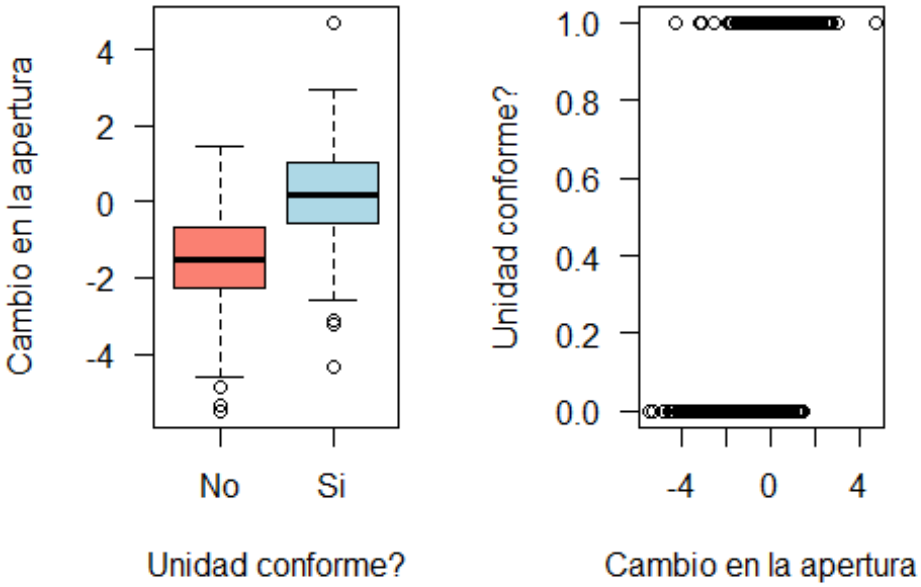
```
## número y proporciones de 1's y 0's
with(datos, table(y))

## y
##   0   1
## 340 160

prop.table(with(datos, table(y)))

## y
##    0    1
## 0.68 0.32
```

Asumiendo que los datos se tratan de piezas experimentales producidas, de acuerdo a lo anterior es posible afirmar que 160 piezas cumplen con la especificación y 340 no. Mismo análisis para proporciones en donde el 32% de las piezas producidas salen bien y el resto no.

Análisis inicial grafico

Ahora, de manera visual se pueden realizar los siguientes gráficos:

```
## boxplot
par(mfrow  = c(1,2))
with(datos, boxplot(z ~ y, las = 1, col = c("salmon", "lightblue"), xlab
= 'Unidad conforme?',
                names = c("No", "Si"), ylab = "Cambio en la apertura"))

## grafico de dispersión
with(datos, plot(z, y, las = 1, ylab = 'Unidad conforme?', xlab = 'Cambio
en la apertura'))
```



La idea es ver la distribución de la variable controlable (z) cuando la unidad cumple con la especificación y cuando no.

Es posible verificar visualmente si en promedio la variable (z) es la misma cuando el producto sale bien a cuando sale mal.

En el grafico es evidente que cuando las unidades salen bien, la apertura de válvula(z) en promedio es mayor.

Aproximadamente cuando el Z es mayor a 1.9 todas las unidades empiezan a salir bien.

Modelo de Regresión Logística

La función clave es "glm", y como la respuesta es de tipo binario, debemos seleccionar family = binomial.

```
modelo <- glm(y ~ z, data = datos, family = "binomial")

#ahora puedo proceder a hacer el summary del modelo:
summary(modelo)

##
## Call:
## glm(formula = y ~ z, family = "binomial", data = datos)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -1.9044  -0.6900  -0.3926   0.6858   3.1296
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.03018    0.12713  -0.237    0.812
## z            1.12437    0.11061  10.165   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 626.87  on 499  degrees of freedom
## Residual deviance: 453.13  on 498  degrees of freedom
## AIC: 457.13
##
## Number of Fisher Scoring iterations: 5
```

Con el summary del modelo se pueden hacer inferencias y saber que el modelo ajustado seria:

$$log(\frac{\theta}{1-\theta}) = B0 + B1 * X1 + B2 * X2 + \ldots + Bk * Xk + \in$$

$$log(\frac{\theta}{1-\theta}) = -0.03018 + 1.12437 * Z$$

Para saber si la variable Z me afecta el logit de la probabilidad, podría calcular los límites de confianza y agregarle el exponencial para calcular el Odds Ratio:

```
#Para el intervalo de confianza para el Odds Ratio de Z
exp(confint(modelo))

## Waiting for profiling to be done...

##                2.5 %   97.5 %
## (Intercept) 0.758129 1.248871
## z           2.502384 3.863711

#Solo para el valor del Odds Ratio de Z
exp(coefficients(modelo))

## (Intercept)           z
##   0.9702721   3.0782644
```

Con esto, se puede decir que, si yo controlo la variable Z, la probabilidad de piezas conformes va a ser 2.5 a 3.8 veces más que la probabilidad de no conformes.

El Odds Ratio de Z es 3.08, lo cual quiere decir que si yo aumento Z en una unidad es 3.08 veces más probable que la pieza cumpla con la especificación a que no lo haga. (Conclusión = aumente Z en este caso)

Probabilidades(θ) Estimadas

Se utiliza la función predict y se debe colocar type = 'response', para garantizar que me calcule la probabilidad (θ) y no el logit.

En caso de no utilizar type='response' y obtener el logit, para calcular la probabilidad se debería hacer 1/1+exp(-x), en este ejemplo x seria cada valor del logit.

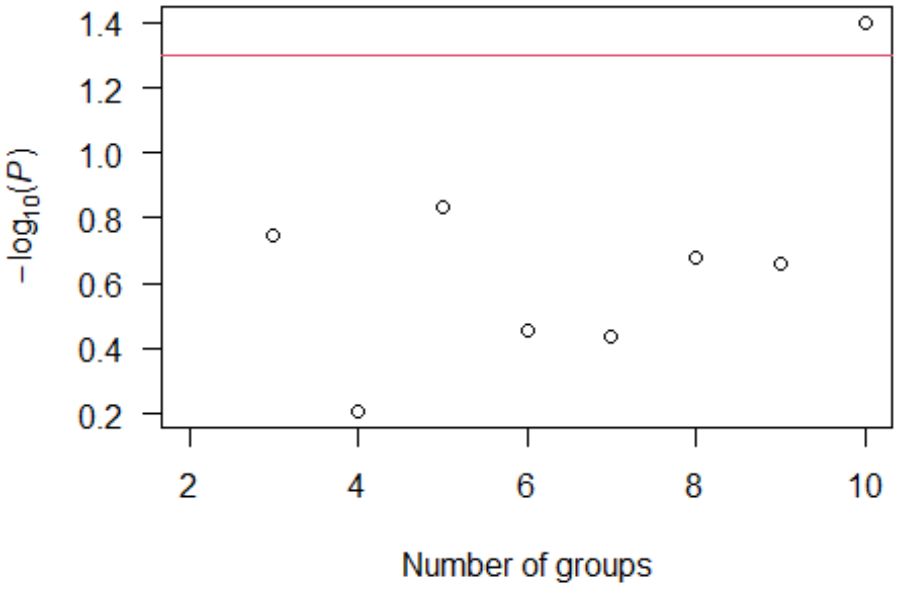
```
## probabilidades estimadas
thetahat <- predict(modelo, type = 'response')
## ahora incluyamos thetahat en los datos originales
datos <- data.frame(id = 1:NROW(datos), datos, thetahat)
```

Prueba de Hosmer & Lemeshow GoF

Ahora, p=número de predictores, en este modelo, solo se tiene un predictor, que es Z. g es el número de grupos que deber ser mayor a p+1.

En este caso g seria 3

```
## hosmer & lemeshow GoF test
if(!require(ResourceSelection)) install.packages('ResourceSelection')

## Loading required package: ResourceSelection

## ResourceSelection 0.3-5   2019-07-22

require(ResourceSelection)

## in theory, we use g > p + 1 ---> g > 2 in our case
p <- sapply(2:10,
            function(gi){
              hl <- hoslem.test(modelo$y, fitted(modelo), g = gi)
              hl$p.value
            })
## plot
par(mfrow = c(1, 1))
plot(2:10, -log10(p), las = 1, xlab = 'Number of groups', ylab =
expression(-log[10]*"("*italic(P)*")"))
abline(h = -log10(0.05), col = 2)
```




```
## check para g = 3 grupos
(hl <- hoslem.test(modelo$y, fitted(modelo), g = 3))

##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  modelo$y, fitted(modelo)
## X-squared = 1.8025, df = 1, p-value = 0.1794

cbind(hl$observed,hl$expected)

##                   y0  y1     yhat0     yhat1
## [0.00197,0.141] 159   8 155.83638  11.16362
## (0.141,0.404]   120  46 124.86771  41.13229
## (0.404,0.995]    61 106  59.29592 107.70408
```

Al realizar la prueba se obtiene un valor p y este indica si mi modelo de regresión logística ajusta bien a los datos; debido a esto quiero que no se rechace la H0, es decir, queremos que el valor p sea mayor a un Alpha de 0.05.

En el ejercicio aplicado se obtiene un valor p de 0.1794, lo cual indica que mi modelo se ajusta bien a los datos.

Curva ROC

Para saber si mi modelo es bueno para diferenciar las piezas buenas de las piezas malas debemos realizar la curva ROC y calcular su AUC.

```r
## curva ROC  -- necesitamos el paquete pROC
if(!require(pROC)) install.packages('pROC')
```

```
## Loading required package: pROC
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

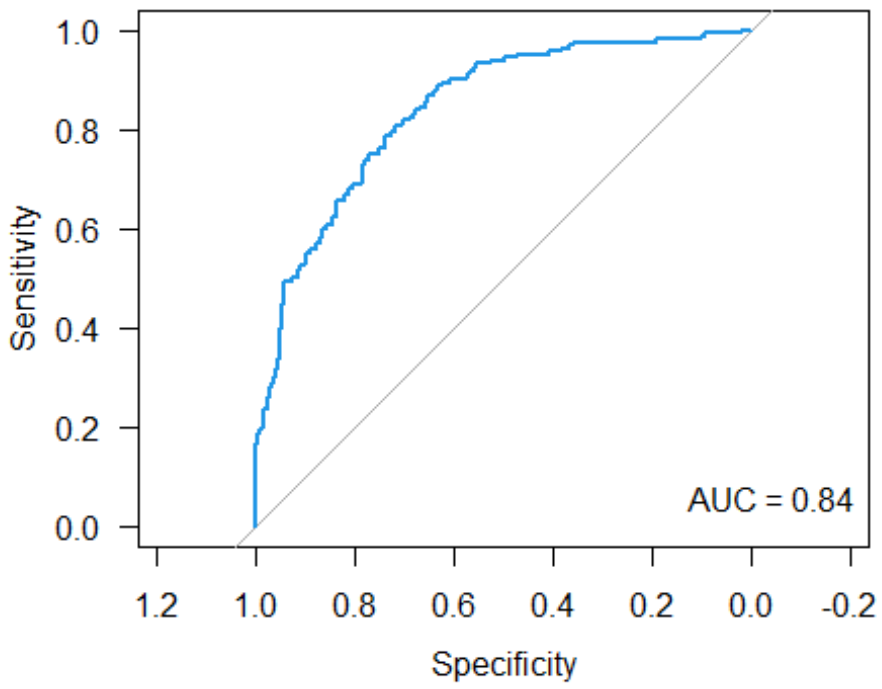
```r
require(pROC)

## useful when we have more than one predictor
# see http://web.expasy.org/pROC/ and ?roc for more details
par(mfrow = c(1,1))
(ROC <- roc(modelo$y , thetahat, ci = TRUE))
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
##
## Call:
## roc.default(response = modelo$y, predictor = thetahat, ci = TRUE)
##
## Data: thetahat in 340 controls (modelo$y 0) < 160 cases (modelo$y 1).
## Area under the curve: 0.8402
## 95% CI: 0.8037-0.8768 (DeLong)
```

```r
plot(ROC, las = 1, col = 4)
legend('bottomright', paste0('AUC = ', round(ROC$auc, 3)),
       bty = 'n')
```



Se obtiene un área bajo la curva de 0.8402, lo cual indica que mi modelo si es bueno para diferenciar piezas buenas de malas antes de producirlas, recordar que mientras más se acerque a 1 (gráficamente mientras más se acerque al borde del cuadrado), mejor diferencia el modelo.

Además, esta función nos presenta el intervalo de confianza del 95% el cual es de 0.8037-0.8768; con lo cual se podría decir que el área bajo la curva es mayor que 0.7.

Predicción

Al saber que el modelo ajusta bien y que sabe diferenciar las piezas buenas de las malas, podemos proceder a realizar predicciones.

```
# que pasa cuando z = 2?
predict(modelo, newdata = data.frame(z = 2), type = 'response', se.modelo
= TRUE)# thetahat + error
```

```
##         1
## 0.9019033
```

Se obtiene un θ de 0.9019, y si deseo calcular el intervalo de confianza para esta probabilidad solo debo hacer lo siguiente:

Límite inferior=θ-1.96*Error

Límite superior=θ+1.96*Error

Lo que concluirías es que si dejas trabajando el sistema con z=2, la probabilidad de que la pieza salga bien va a estar entre el límite inferior y el superior, con una confianza del 95%.

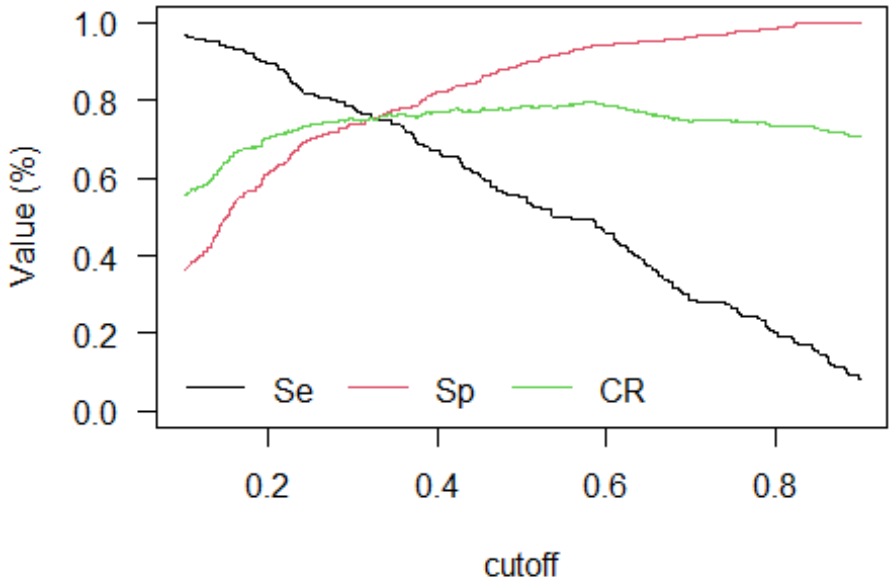
Cálculo del cutoff

Primero se calculan en este ejemplo 5000 puntos de corte.

Luego en donde dice ## cálculos, lo que se hace es obtener para cada cutoff los valores de A, B, C, D, sensitivity, specificity entre otros.

Para finalmente realizar el grafico con todos estos datos.

```
## funciones para calcular las medidas de desempeño para diferentes
puntos de corte
source('https://www.dropbox.com/s/gnwm7vuvhojg9q4/logistic-
functions.R?dl=1')

## Esta es una colecciÃ³n de funciones creadas
## por el profesor Jorge VÃ©lez <https://jivelez.github.io/>
## para el ajuste y validaciÃ³n de modelos de RegresiÃ³n LogÃstica
##
## En caso de presentar algÃºn inconveniente, por favor
## repÃ³rtelo a jvelezv@uninorte.edu.co
##

## definimos 5000 posibles valores para el cutoff
cutoff <- seq(.1, .9, length = 5000)

## cálculos
yreal <- datos$y
res <- data.frame(cutoff = cutoff, bycutoff2(modelo, real = yreal,
cutoff))

##  gráfico
with(res, matplot(cutoff, cbind(sensitivity, specificity, class_rate),
                  type = 'l', las = 1, ylim = c(0, 1), lty = 1, ylab =
'Value (%)'))
legend('bottomleft', c('Se', 'Sp', 'CR'), lty = 1, col = 1:3, bty = 'n',
ncol = 3)
```



Si se usara como criterio la tasa de clasificación correcta (CR), debería elegir el cutoff con un value(%) máximo.

A continuación, se verá cómo obtener los valores exactos de cutoff óptimo dependiendo del criterio.

```
#Criterios posibles
## Se \approx Sp
res[which.min(abs(res$sensitivity - res$specificity)), ]

##         cutoff   a  b  c   d sensitivity specificity       ppv
npv
## 1396 0.3232446 120 84 40 256        0.75   0.7529412 0.5882353
0.8648649
##            fdr       fpr class_rate     lift     delta
## 1396 0.4117647 0.2470588      0.752 1.838235 0.3514798

## maximizando CR
res[which.max(res$class_rate), ]

##         cutoff  a  b  c   d sensitivity specificity       ppv      npv
## 3014 0.5821764 79 20 81 320     0.49375   0.9411765 0.7979798 0.798005
##            fdr        fpr class_rate     lift    delta
## 3014 0.2020202 0.05882353      0.798 2.493687 0.509656

# maximizando lift
res[which.max(res$lift), ]
```

```
##         cutoff  a b   c   d sensitivity specificity ppv       npv fdr
fpr
## 4606 0.8369474 27 0 133 340     0.16875           1   1 0.7188161   0
0
##      class_rate  lift   delta
## 4606      0.734 3.125 0.83125
```

Dependiendo de la situación de elegirá el criterio para seleccionar el cutoff óptimo, recordando que cuando se obtengan predicciones de probabilidad (θ), si esta es mayor o igual al cutoff óptimo (θo), indicaría que la predicción seria y=1, en caso contrario obtendría y=0.