Comparación entre Regresión Logística, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA) y K-Nearest-Neighbors

Joaquín Amat Rodrigo *j.amatrodrigo@gmail-com*

Septiembre, 2016

Índice

Introducción

La Regresión Logística, el *Linear Discriminant Analysis (LDA)*, *Quadratic Discriminant Analysis (QDA)* y *K-nearest neighbors (K-NN)* son 4 métodos de clasificación empleados para predecir variables cualitativas. A continuación se describen los escenarios en los que mejor se adecúa cada uno de estos métodos. Para información detallada de cada uno consultar:

- Regresión logística simple y múltiple
- Análisis discriminante lineal (LDA) y Análisis discriminante cuadrático (QDA)

Linear Discriminant Analysis (LDA) asume que en cada una de las clases los predictores se distribuyen de forma normal y que la matriz de covarianza es común en todas ellas. Si estas condiciones se cumplen, el *LDA* es superior a la regresión logística. Por contra, si la condiciones de normalidad no se satisface, la regresión logística es más recomendable.

Tanto la regresión logística como el *LDA* suelen llegar a resultados similares debido a que están estrechamente conectados. Supóngase un escenario en el que la variable cualitativa tiene solo dos niveles y en el que solo hay un predictor. Considérese $p_1(x)$ y $p_2(x) = 1\text{-}p_1(x)$ como las probabilidades de que una observación $X = x$ pertenezca a la clase 1 y 2 respectivamente.

Acorde a la ecuación de *LDA*, la probabilidad de que una observación pertenezca a la clase *k* es:

$$log(p_k(x)) = x\ \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

Por lo tanto:

$$\frac{log(p_1(x))}{1 - log(p_1(x))} = log(\frac{p_1(x)}{p_2(x)}) = c_0 + c_1 x$$

Esta ecuación se asemeja en gran medida a la ecuación de la regresión logística:

$$log(\frac{p_1(x)}{p_2(x)}) = \beta_0 + \beta_1 x$$

Ambos métodos generan límites de decisión lineales. La única diferencia es que en regresión logística β_0 y β_1 se estiman mediante *maximum likelihood*, mientras que en *LDA* c_0 y c_1 se calculan a partir de estimaciones de la media y varianza de una distribución normal.

K-Nearest-neighbors (K-NN) consiste en identificar las *k* observaciones más cercanas a la nueva observación, calcular la proporción de estas que pertenece a cada clase y finalmente asignar la nueva observación a la clase más frecuente. Se trata de un método no paramétrico que no requiere de ninguna asunción sobre la forma (lineal o no lineal) de los límites de decisión ni del tipo de distribución de las observaciones en cada nivel. Por lo tanto, este método supera a los otros dos cuando el límite de decisión está alejado de la linealidad. Dos de la principales desventajas de *K-NN* son que se varía mucho dependiendo del nivel de flexibilidad *k* (observaciones cercanas) que se emplee y que, como la mayoría de métodos no paramétricos, empeora a medida que aumenta el número de predictores a no ser que se disponga de un gran número de observaciones. A este fenómeno se le conoce como *curse of dimensionality*.

Dado que este método requiere calcular la distancia entre observaciones, normalmente distancia euclídea, se ve influenciado por la escala en que se mide cada predictor. Si las escalas de los predictores empleados son distintas, una solución es estandarizar cada uno de ellos centrándolo en la media 0 y midiendo el número de desviaciones estándar de cada observación respecto de la media. En R esto se consigue con la función `scale()`.

Quadratic Discriminant Analysis (QDA) se encuentra en un punto medio entre el método no paramétrico *K-NN* y los métodos lineales *LDA* y regresión logística. Al generar límites de decisión cuadráticos, el método *QDA* dispone de cierta curvatura que le permite ajustarse mejor a escenarios que se alejan moderadamente de la linealidad. Si bien no es tan flexible como el *K-NN*, *QDA* es más adecuado cuando hay un número limitado de observaciones ya que al hacer ciertas asunciones sobre los límites de decisión no se corre el riesgo de *overfitting*.

Cuando la variable cualitativa estudiada tiene más de 2 niveles, a pesar de que es posible aplicar regresión logística multinivel, tal como se explica en Análisis discriminante lineal (LDA) y Análisis discriminante cuadrático (QDA) son más aconsejables los métodos *LDA*, *QDA* o *K-NN*.

Ejemplo 1. Comparación de modelos para predicciones en el mercado de valores.

Se dispone de un *data set* con información sobre los movimientos del mercado de valores de cada día entre 2001 a 2005. Para cada registro (día) se dispone de la siguiente información:

- Year: año al que pertenece el día.
- Lag1, Lag2, Lag3, Lag4 y Lag5: El valor de mercado para 1, 2, 3, 4 y 5 días previos al día del registro.
- Today: El valor de mercado del día.
- Volume: el número de movimientos totales de ese día.
- Direction: Si el valor de mercado ha subido o bajado.

```
library(ISLR)
library(ggplot2)
library(knitr)
data("Smarket")
kable(head(Smarket))
```

Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
2001	0.381	-0.192	-2.624	-1.055	5.010	1.1913	0.959	Up
2001	0.959	0.381	-0.192	-2.624	-1.055	1.2965	1.032	Up
2001	1.032	0.959	0.381	-0.192	-2.624	1.4112	-0.623	Down
2001	-0.623	1.032	0.959	0.381	-0.192	1.2760	0.614	Up
2001	0.614	-0.623	1.032	0.959	0.381	1.2057	0.213	Up
2001	0.213	0.614	-0.623	1.032	0.959	1.3491	1.392	Up

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2          
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000  
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500  
##  Median :2003   Median : 0.039000   Median : 0.039000  
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919  
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750  
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000  
```

```
##       Lag3                Lag4                Lag5          
##  Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200  
##  1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000  
##  Median : 0.038500   Median : 0.038500   Median : 0.03850  
##  Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561  
##  3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700  
##  Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300  
##      Volume           Today           Direction 
##  Min.   :0.3561   Min.   :-4.922000   Down:602  
##  1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648  
##  Median :1.4229   Median : 0.038500             
##  Mean   :1.4783   Mean   : 0.003138             
##  3rd Qu.:1.6417   3rd Qu.: 0.596750             
##  Max.   :3.1525   Max.   : 5.733000
```

El objetivo es crear un modelo que permita predecir si el mercado va a subir o bajar (*Direction*) empleando los predictores disponibles. Dado que se trata de una variable cualitativa con dos niveles, todos los métodos son *a priori* aplicables.

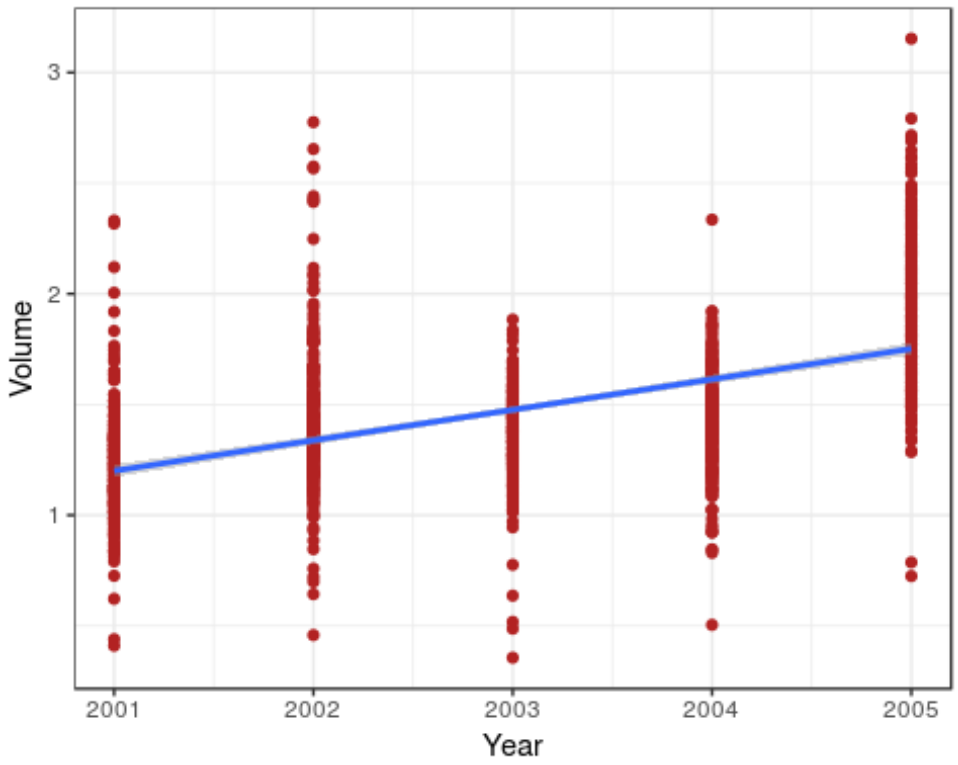
En primer lugar se estudia la correlación existente entre las variables continuas y su distribución:

```
cor_matrix <- round(cor(Smarket[, -9], method = "pearson"), digits = 4)
cor_matrix[upper.tri(cor_matrix, diag = TRUE)] <- ""
as.data.frame(cor_matrix)
```

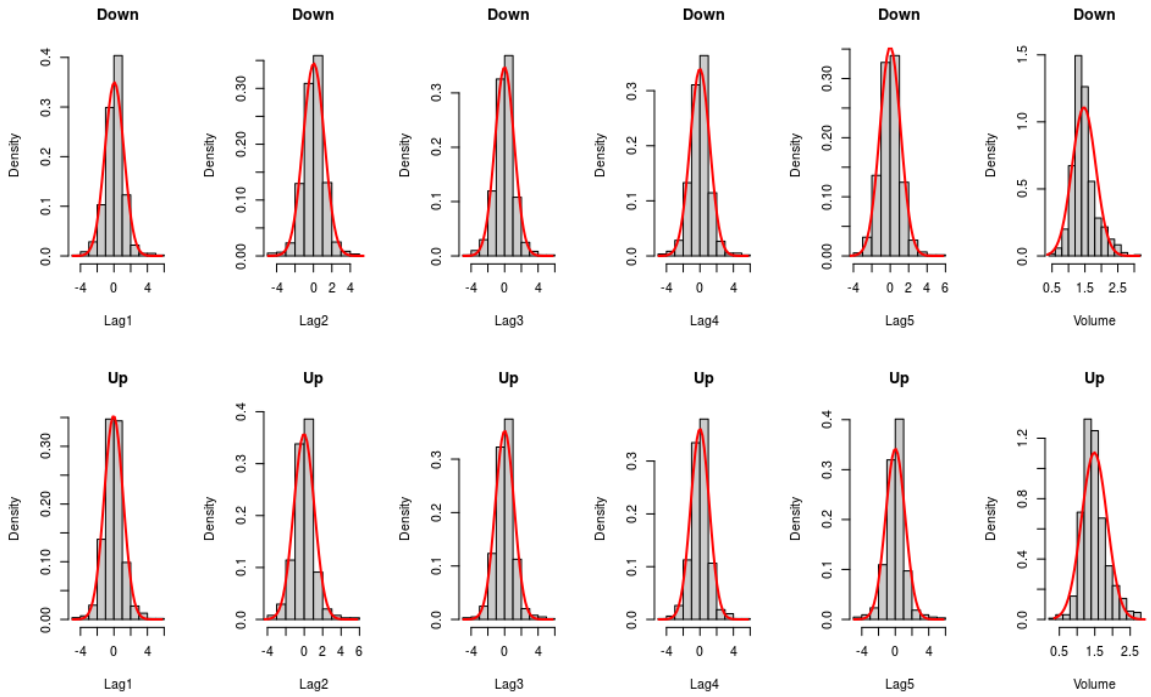
```
##          Year    Lag1    Lag2    Lag3    Lag4    Lag5 Volume Today
## Year                                                              
## Lag1   0.0297                                                     
## Lag2   0.0306 -0.0263                                            
## Lag3   0.0332 -0.0108 -0.0259                                    
## Lag4   0.0357  -0.003 -0.0109 -0.0241                            
## Lag5   0.0298 -0.0057 -0.0036 -0.0188 -0.0271                    
## Volume  0.539  0.0409 -0.0434 -0.0418 -0.0484  -0.022            
## Today  0.0301 -0.0262 -0.0103 -0.0024 -0.0069 -0.0349 0.0146
```

Apenas existe correlación entre las variables *lag* y *today*, lo que indica que no hay apenas correlación entre el valor de mercado de los días previos y el del día actual, algo de esperar ya que o es nada fácil predecir los movimientos del mercado. La única correlación intermedia que se aprecia es entre *Year* y *Volume*, lo que indica que el número de movimientos se ha ido incrementando a lo largo de los años.

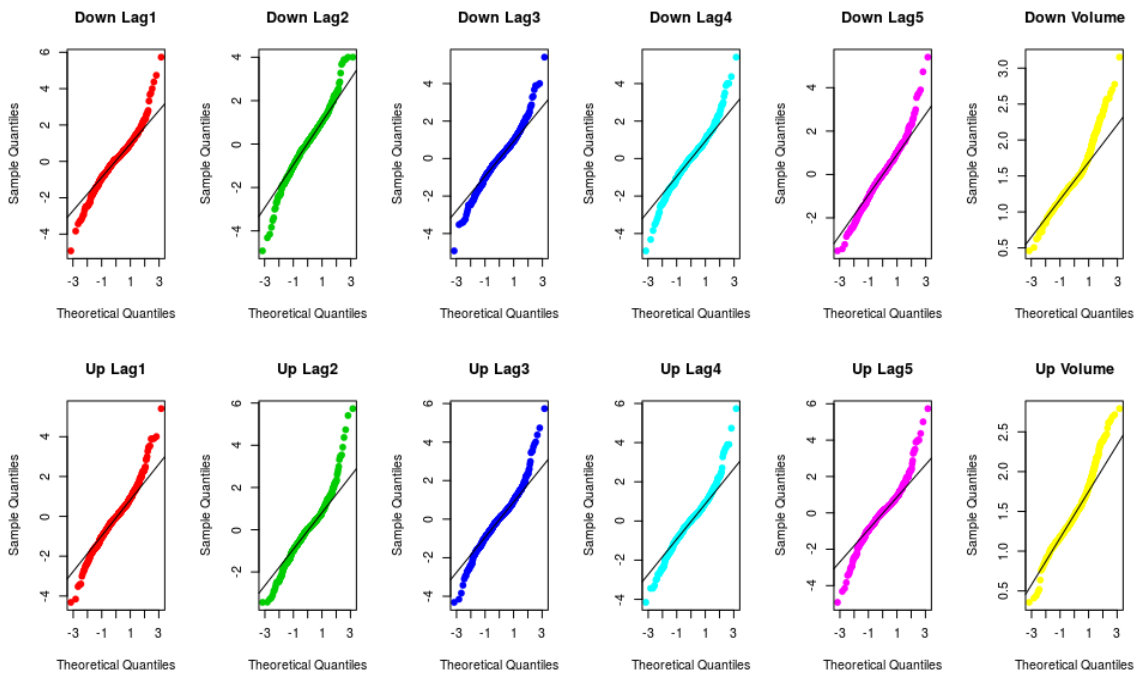
```
ggplot(data = Smarket, aes(x = Year, y = Volume)) +
 geom_point(color = "firebrick") +
 geom_smooth(method = "lm") + theme_bw()
```




```
# representación mediante Histograma de cada variable para cada especie
par(mfcol = c(2, 6))
for (i in 2:7) {
    variable <- names(Smarket)[i]
    rango <- seq(min(Smarket[, i]), max(Smarket[, i]), le = 50)
    for (k in 1:2) {
        grupo <- levels(Smarket$Direction)[k]
        x <- Smarket[Smarket$Direction == grupo, variable]
        hist(x, proba = T, col = grey(0.8), main = grupo, xlab = variable)
        lines(rango, dnorm(rango, mean(x), sd(x)), col = "red", lwd = 2)
    }
}
```




```
# representación de cuantiles normales de cada variable para cada especie
for (i in 2:7) {
    variable <- names(Smarket)[i]
    rango <- seq(min(Smarket[, i]), max(Smarket[, i]), le = 50)
    for (k in 1:2) {
        grupo <- levels(Smarket$Direction)[k]
        x <- Smarket[Smarket$Direction == grupo, variable]
        qqnorm(x, main = paste(grupo, variable), pch = 19, col = i)  # los colores
2 y 3 son el rojo y verde
        qqline(x)
    }
}
```




```
# Contraste de normalidad Shapiro-Wilk para cada variable en cada especie
library(reshape2)
library(knitr)
library(dplyr)
Smarket_tidy <- melt(Smarket[, -1], value.name = "valor")
kable(Smarket_tidy %>% group_by(Direction, variable) %>%
summarise(p_value_Shapiro.test = shapiro.test(valor)$p.value))
```

Direction	variable	p_value_Shapiro.test
Down	Lag1	0e+00
Down	Lag2	1e-07
Down	Lag3	1e-07
Down	Lag4	0e+00
Down	Lag5	8e-07
Down	Volume	0e+00
Down	Today	0e+00
Up	Lag1	0e+00
Up	Lag2	0e+00
Up	Lag3	0e+00

Up	Lag4	0e+00
Up	Lag5	0e+00
Up	Volume	0e+00
Up	Today	0e+00

Las variables no se distribuyen de forma normal.

Una vez estudiadas las variables se procede a crear los distintos modelos.

Logistic Regression

En R la regresión logística está incluida dentro de la función `glm()` que engloba los distintos tipos de *generalized linear models*. Para especificar que se quiere una regresión logística se tiene que indicar en el argumento *family=binomial*.

En primer lugar se prueba a incluir todas las variables disponibles como predictores excepto año.

```
modelo_logistico <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
    data = Smarket, family = "binomial")
summary(modelo_logistico)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = "binomial", data = Smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

Para conocer como se ha codificado la variable respuesta (*Direction*) se recurre a la función `contrast()`.

```
contrasts(Smarket$Direction)
```

```
##      Up
## Down  0
## Up    1
```

El modelo ha considerado como 1 el nivel *Up* y como 0 el nivel *Down*.

De entre los predictores introducidos, el que menor *p-value* tiene es *Lag1* (0.145). Dado que su coeficiente estimado es negativo, si el mercado subió el día anterior, es menos probable que suba hoy. Sin embargo, el *p-value* es considerablemente alto por lo que no hay una clara evidencia de que exista relación entre *Lag1* y *Direction*.

Una vez generado el modelo se tiene que evaluar como de bueno es prediciendo los resultados. Para ello se tienen en cuenta dos tipos de error:

- *Trainig error*: % de errores que comete el modelo al intentar predecir las observaciones que se han utilizado para generarlo.
- *Tested error*: % de errores que comete el modelo al intentar predecir nuevas observaciones que no se han empleado en la creación del modelo. Este es el % de error que más interesa y que por lo generar siempre es mayor que el *Training Error*.

La función `predict()` de R permite predecir la probabilidad de que la variable respuesta pertenezca al nivel de referencia (en este caso 1 = *Up*), dados unos valores del predictor o predictores. Por defecto devuelve el valor *Log of ODDs* ya que es esta la variable que se modela en la regresión logística. Si se emplea el argumento *type=response* se devuelve directamente el valor de la probabilidad. Si a la función `predict()` no se le pasa ningún *dataframe* como

argumento, devuelve directamente las predicciones para los datos empleados en su creación *training data*.

```
predicciones <- predict(object = modelo_logistico, type = "response")
head(predicciones)
```

```
##         1         2         3         4         5         6
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565
```

El siguiente paso una vez predichas las probabilidades es utilizarlas para determinar a qué clase de la variable cualitativa pertenece cada observación acorde a la predicción del modelo. En este caso se considera como límite de decisión p=0.5. Si p>0.5 la observación se clasifica en el nivel 1(*Up*) de lo contrario se clasifica en el nivel 0 (*Down*).

```
prediccion <- data.frame(probabilidad = predicciones, clase = rep(NA,
length(predicciones)))
prediccion[prediccion$probabilidad < 0.5, "clase"] <- "Down"
prediccion[prediccion$probabilidad > 0.5, "clase"] <- "Up"
head(prediccion)
```

```
##   probabilidad clase
## 1    0.5070841    Up
## 2    0.4814679  Down
## 3    0.4811388  Down
## 4    0.5152224    Up
## 5    0.5107812    Up
## 6    0.5069565    Up
```

Mediante la función *table()* se puede obtener una matriz que muestre los aciertos, falsos positivos y falsos negativos:

```
table(clase_predicha = prediccion$clase, clase_real = Smarket$Direction)
```

```
##               clase_real
## clase_predicha Down  Up
##           Down  145 141
##           Up    457 507
```

```
paste("% de acierto:", mean(prediccion$clase == Smarket$Direction))
```

```
## [1] "% de acierto: 0.5216"
```

```
paste("% de error:", mean(prediccion$clase != Smarket$Direction))
```

```
## [1] "% de error: 0.4784"
```

El modelo logístico ha sido capaz de predecir correctamente el 52.2% de las observaciones, solo un poco mejor de lo que cabría esperar por puro azar (50%). El *training error* es de 47.8%. A efectos prácticos, es mucho más interesante conocer estimar el *tested error*. Un modo de hacerlo consiste en emplear solo un subconjunto de los datos disponibles para crear el modelo y con el resto de datos se estudia la precisión de las predicciones. En este caso se van a emplear las observaciones pertenecientes al intervalo 2001-2004 a modo de *traning data* y se evaluará el error de predicción con las observaciones del 2005.

```
train_data <- Smarket[Smarket$Year < 2005, ]
test_data <- Smarket[!(Smarket$Year < 2005), ]

# Se crea el modelo de regresión logística
modelo <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data =
train_data, family = "binomial")

# Se realizan las predicciones para el set de datos no empleado en la
# creación del modelo
predicciones <- predict(object = modelo, newdata = test_data, type = "response")
head(predicciones)
```

```
##       999      1000      1001      1002      1003      1004
## 0.5282195 0.5156688 0.5226521 0.5138543 0.4983345 0.5010912
```

```
# Se considera como threshold de clasificación el 0.5
predicciones[predicciones > 0.5] <- "Up"
predicciones[predicciones != "Up"] <- "Down"
```

Finalmente se genera una tabla cruzada para identificar las predicciones correctas y erróneas.

```
table(clase_predicha = predicciones, clase_real = test_data$Direction)
```

```
##               clase_real
## clase_predicha Down Up
##           Down   77 97
##           Up     34 44
```

```
# % de aciertos
paste("% de acierto:", mean(predicciones == test_data$Direction))
```

```
## [1] "% de acierto: 0.48015873015873"
```

```
# % de errores
paste("% de error:", mean(predicciones != test_data$Direction))
```

```
## [1] "% de error: 0.51984126984127"
```

El % de aciertos empleando un nuevo conjunto de datos es del 47% y el *test error* del 52%. El modelo no se puede considerar útil ya que el error es superior al esperado por puro azar (50%). Esto no es de extrañar ya que el *p-value* de los predictores introducidos en el modelo es muy alto, indicando que no hay apenas relación entre ellos y la variable dependiente.

Una posible forma de mejorar el modelo es incluir únicamente los predictores que tienen menor *p-value*: *Lag1, Lag2*

```
# Se crea el modelo de regresión logística
modelo <- glm(Direction ~ Lag1 + Lag2, data = train_data, family = "binomial")

# Se realizan las predicciones para el set de datos no empleado en la
# creación del modelo
predicciones <- predict(object = modelo, newdata = test_data, type = "response")
head(predicciones)
```

```
##       999      1000      1001      1002      1003      1004
## 0.5098275 0.5208237 0.5332635 0.5260574 0.5072103 0.5061388
```

```
# Se considera como threshold de clasificación el 0.5
predicciones[predicciones > 0.5] <- "Up"
predicciones[predicciones != "Up"] <- "Down"
table(clase_predicha = predicciones, clase_real = test_data$Direction)
```

```
##                clase_real
## clase_predicha Down  Up
##           Down   35  35
##           Up     76 106
```

```
# % de aciertos
paste("% de acierto:", mean(predicciones == test_data$Direction))
```

```
## [1] "% de acierto: 0.55952380952381"
```

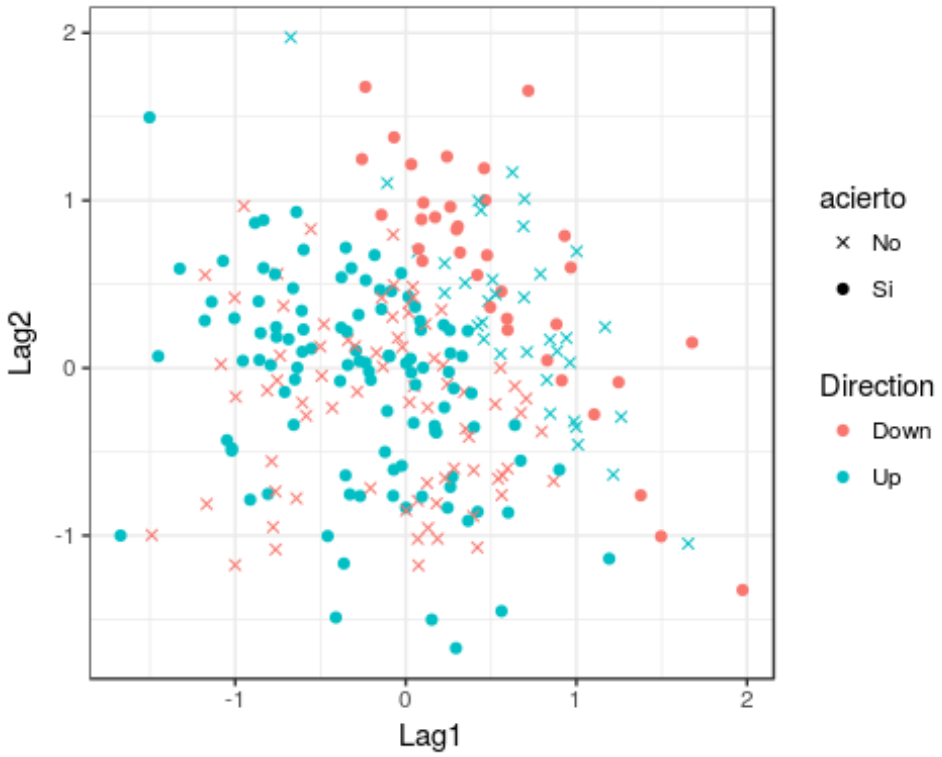
```
# % de errores
paste("% de error:", mean(predicciones != test_data$Direction))
```

```
## [1] "% de error: 0.44047619047619"
```

Se ha conseguido una pequeña mejora, esta vez el modelo es capaz de predecir correctamente la dirección del mercado en un 56% de las ocasiones, pasando a ser ligeramente mejor que lo esperado por azar. Aun así lo recomendable sería probarlo con nuevos datos para confirmarlo.

Además de estudiar el *test error* global, es interesante estudiar cómo se reparte este error entre falsos negativos y falsos positivos. Puede ocurrir que un modelo sea mucho mejor prediciendo en una dirección que en otra. Para el modelo aquí generado, la de predicciones de subida (*Up*) tiene un porcentaje de acierto de *106/(106+76)= 58%* y la de bajada de *35/(35+35)= 50%*. Por lo tanto es más seguro apostar por los días en los que el modelo indica subida que por los que indica bajada.

```
test_data$prediccion <- (predicciones)
test_data$acierto <- ifelse(test = test_data$Direction == test_data$prediccion,
    yes = "Si", no = "No")
ggplot(data = test_data, aes(x = Lag1, y = Lag2, color = Direction, shape =
acierto)) +
 geom_point() +
 scale_shape_manual(values = c(No = 4, Si = 19)) +
 theme_bw()
```



Linear Discriminant Analysis (LDA)

El método LDA se puede emplear para variables cualitativas con dos o más niveles. En R se crean los modelos LDA con la función `lda()` del paquete *MASS*.

```
library(MASS)

modelo_lda <- lda(Direction ~ Lag1 + Lag2, data = train_data)
modelo_lda
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = train_data)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##              Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

```
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

Las *prior probabilities of groups* se estiman a partir de la proporción de observaciones de cada clase presentes en el set de datos empleado para crear el modelo. La media de los grupos se corresponden con la media de cada predictor dentro de cada una de las clases (en este caso *Up* y *Down*) y se emplean como estimación de μ_k del modelo. Se observa una ligera tendencia a que en promedio ambos predictores son negativos cuando el mercado sube (*Up*) y positivo cuando baja (*Down*).

Se calculan las predicciones para los datos del año 2005:

```
predicciones_lda <- predict(object = modelo_lda, test_data)
# La función predict() aplicada a un modelo LDA devuelve una lista con 3
# objetos. class: contiene la clasificación predicha por el modelo.
# posterior: las probabilidades posteriores de que la observación pertenezca
# a cada clase. Se asigna la observación a la clase con mayor probabilidad
# posterior.
head(predicciones_lda$class, n = 3)
```

```
## [1] Up Up Up
## Levels: Down Up
```

```
head(predicciones_lda$posterior, n = 3)
```

```
##           Down        Up
## 999  0.4901792 0.5098208
## 1000 0.4792185 0.5207815
## 1001 0.4668185 0.5331815
```

Evaluación del modelo:

```
table(clase_predicha = predicciones_lda$class, clase_real = test_data$Direction)
```

```
##               clase_real
## clase_predicha Down  Up
##           Down   35  35
##           Up     76 106
```

```
# % de aciertos
paste("% de acierto:", mean(predicciones_lda$class == test_data$Direction))
```

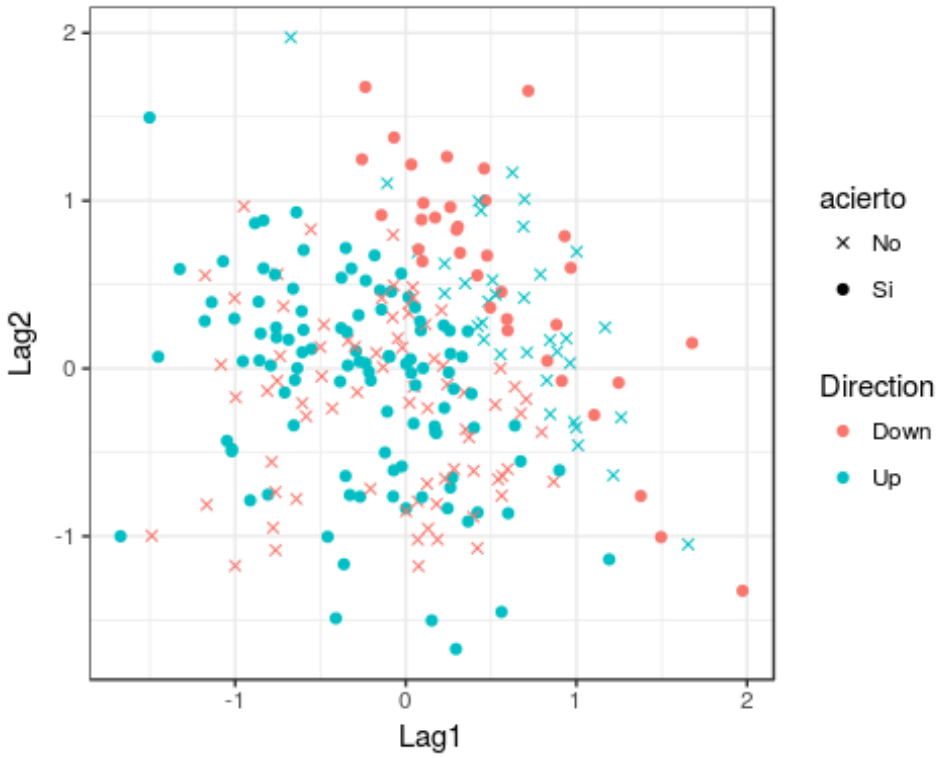
```
## [1] "% de acierto: 0.55952380952381"
```

```
# % de errores
paste("% de error:", mean(predicciones_lda$class != test_data$Direction))
```

```
## [1] "% de error: 0.44047619047619"
```

El modelo tiene un *test error* del 44%, es decir que acierta en un 56% de las ocasiones. Esta ligeramente por encima de los aciertos esperados por azar. Los resultados son muy similares a los obtenidos mediante regresión logística.

```
test_data$prediccion <- predicciones_lda$class
test_data$acierto <- ifelse(test = test_data$Direction == test_data$prediccion,
    yes = "Si", no = "No")
ggplot(data = test_data, aes(x = Lag1, y = Lag2, color = Direction, shape =
acierto)) + geom_point() + scale_shape_manual(values = c(No = 4, Si = 19)) +
theme_bw()
```



Quadratic Discriminant Analysis (QDA)

El método QDA se puede emplear para variables cualitativas con dos o más niveles. En R se crean los modelos LDA con la función `qda()` del paquete *MASS*. La sintaxis y el tipo de *output* es igual que el de la función `lda()` descrita anteriormente.

```
library(MASS)
modelo_qda <- qda(Direction ~ Lag1 + Lag2, data = train_data)
modelo_qda
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = train_data)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##              Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

Predicciones con nuevos datos (2005):

```
predicciones_qda <- predict(object = modelo_qda, test_data)
head(predicciones_qda$class)
```

```
## [1] Up Up Up Up Up Up
## Levels: Down Up
```

```
head(predicciones_qda$posterior)
```

```
##           Down        Up
## 999  0.4873243 0.5126757
## 1000 0.4759011 0.5240989
## 1001 0.4636911 0.5363089
## 1002 0.4739253 0.5260747
## 1003 0.4903426 0.5096574
## 1004 0.4913561 0.5086439
```

Evaluación del modelo:

```
table(clase_predicha = predicciones_qda$class, clase_real = test_data$Direction)
```

```
##                clase_real
## clase_predicha Down  Up
##           Down   30  20
##           Up     81 121
```

```
# % de aciertos
paste("% de acierto:", mean(predicciones_qda$class == test_data$Direction))
```

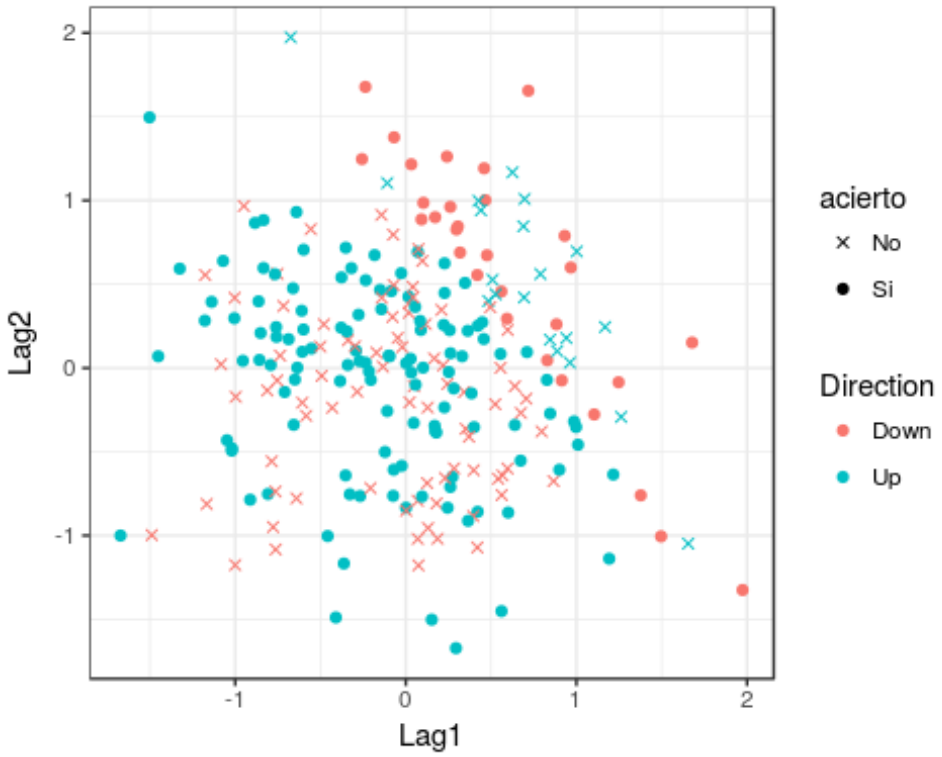
```
## [1] "% de acierto: 0.599206349206349"
```

```
# % de errores
paste("% de error:", mean(predicciones_qda$class != test_data$Direction))
```

```
## [1] "% de error: 0.400793650793651"
```

El modelo obtenido mediante *quadratic discriminant analysis* tiene un *test error* del 40%, es capaz de predecir correctamente casi un 60% de las veces. Este modelo es superior a los obtenidos por *LDA* y regresión logística, indicando que el modelo cuadrático captura mejor la verdadera relación entre predictores y variable dependiente que los modelos lineales.

```
test_data$prediccion <- predicciones_qda$class
test_data$acierto <- ifelse(test = test_data$Direction == test_data$prediccion,
    yes = "Si", no = "No")
ggplot(data= test_data, aes(x = Lag1, y = Lag2, color = Direction, shape=acierto))+
  geom_point() +
  scale_shape_manual(values = c(No = 4, Si = 19)) +
  theme_bw()
```



K-Nearest-Neighbors

En R, método de predicción no paramétrico *K-NN* se realiza mediante la función `knn()` del paquete *class*. Esta función tiene un funcionamiento distinto a las dos anteriores. En lugar de generar un modelo que después se interroga con nuevos valores de los predictores, en un único paso se evalúa los datos de *training*, se pasan los valores de *test* y se devuelven las predicciones. Requiere 4 argumentos obligatorios:

- train: Una matriz (que no *data.frame*) que contenga el valor de los predictores para las observaciones que se van a emplear como *trainnig data*.
- test: Una matriz (que no *data.frame*) que contenga el valor de los predictores para las nuevas observaciones cuya variable respuesta se quiere predecir *test data*.
- cl: un vector que contenga la verdadera clasificación de las observaciones empleadas en el *training data*.
- K: el número de observaciones vecinas empleadas por el modelo (este parámetro es el que determina la flexibilidad).

La función `knn()` requiere fraccionar el *data set* en 3 partes. Es importante que se mantenga el orden para que los valores del argumento *train* se correspondan con los del argumento *cl*.

Dado que la escala de ambos predictores empleados en este ejercicio es la misma, no es necesario estandarizarlos.

```
library(class)
# En caso de empate entre los vecinos, se elige uno aleatoriamente. Esto
# influye en la reproducibilidad.
set.seed(1)
prediccion_knn <- knn(train = train_data[, c("Lag1", "Lag2")], test =
 test_data[, c("Lag1", "Lag2")], cl = train_data[, "Direction"], k = 3)
head(prediccion_knn)
```

```
## [1] Down Down Down Up   Up   Down
## Levels: Down Up
```

Evaluación del modelo:

```
table(clase_predicha = prediccion_knn, clase_real = test_data$Direction)
```

```
##                clase_real
## clase_predicha Down Up
##           Down   48 55
##           Up     63 86
```

```
# % de aciertos
paste("% de acierto:", mean(prediccion_knn == test_data$Direction))
```

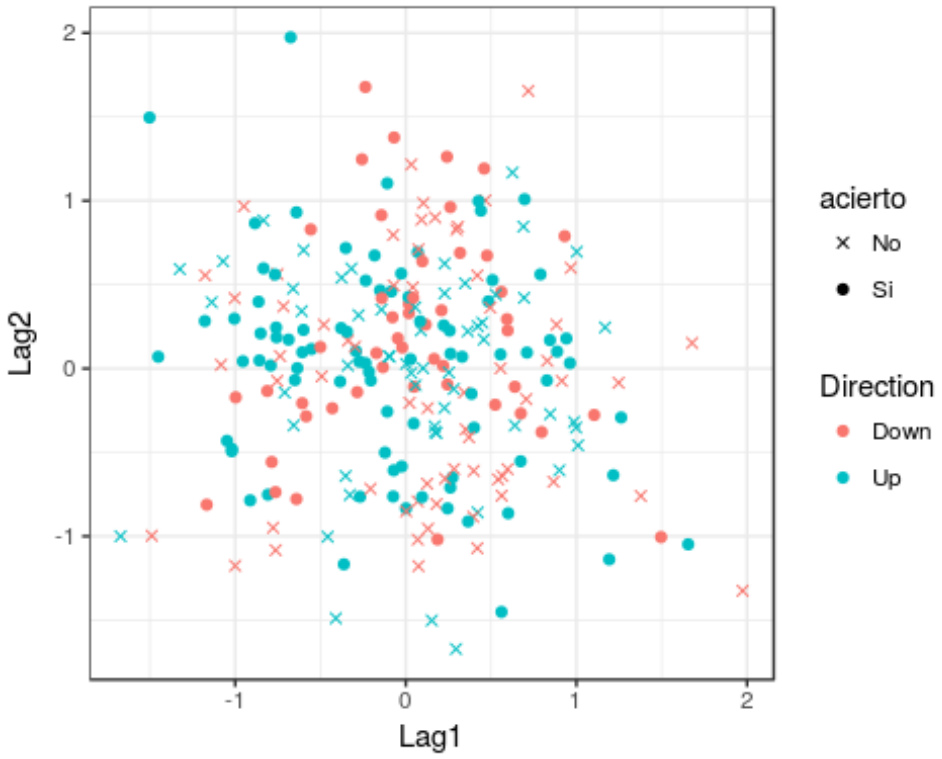
```
## [1] "% de acierto: 0.531746031746032"
```

```
# % de errores
paste("% de error:", mean(prediccion_knn != test_data$Direction))
```

```
## [1] "% de error: 0.468253968253968"
```

El modelo por *K-NN* empleando k=3 es capaz de predecir correctamente el 53% de las veces. *Test error* de 47%.

```
test_data$prediccion <- prediccion_knn
test_data$acierto <- ifelse(test = test_data$Direction == test_data$prediccion,
    yes = "Si", no = "No")
ggplot(data=test_data, aes(x = Lag1, y = Lag2, color = Direction, shape= acierto))+
  geom_point() +
  scale_shape_manual(values = c(No = 4, Si = 19)) +
  theme_bw()
```



Conclusión

De entre todos los métodos empleados con este set de datos, el *QDA* es el que menor *test error* ha mostrado y por lo tanto el que mejor recoge la relación entre los predictores *Lag1* y *Lag2*, y la variable respuesta *Direction*.

Ejemplo 2. K-NN Predicción potenciales compradores, muy interesante la interpretación de los resultados.

En el siguiente ejemplo se muestra la utilización del método de *K-NN* para intentar predecir si un determinado comprador contratará una póliza de seguro. Se dispone de un set de datos en el que para 5822 compradores se han registrado 85 variables. La variable respuesta que se quiere estudiar está identificada como *Purchase*.

```
data("Caravan")
summary(Caravan$Purchase)
```

```
##   No  Yes
## 5474  348
```

Aproximadamente solo un 6% de los compradores contrata el seguro. Es importante conocer el porcentaje de eventos de cada clase ya que esta información es clave para decidir si el modelo predictivo es útil.

Dado que el método *K-NN* se ve influenciado por la escala en la que se miden los predictores empleados, para que todos tengan el mismo peso, se procede a estandarizar todas las variables (excepto la variable respuesta).

```
Caravan_stand <- scale(Caravan[, -86])
```

Se divide el set de datos en dos partes. Los primeros 1000 registros se emplean como *test data* y los 4822 restantes como *trainnig data*.

```
library(vcd)

test <- 1:1000
set.seed(1)
prediccion_knn <- knn(train = Caravan_stand[-test, ], test =Caravan_stand[test,  ],
cl = Caravan$Purchase[-test], k = 1)
```

```
matriz_confusion <- table(clase_predicha = prediccion_knn, clase_real =
Caravan[test, "Purchase"])
matriz_confusion
```

```
##               clase_real
## clase_predicha  No Yes
##            No  873  50
##            Yes  68   9
```

```
mosaic(matriz_confusion, shade = T, colorize = T, gp = gpar(fill =
matrix(c("green3", "red2", "red2", "green3"), 2, 2)))
```


```
# % de aciertos
paste("% de acierto:", mean(prediccion_knn == Caravan[test, "Purchase"]))
```

```
## [1] "% de acierto: 0.882"
```

```
# % de errores
paste("% de error:", mean(prediccion_knn != Caravan[test, "Purchase"]))
```

```
## [1] "% de error: 0.118"
```

El *KNN test error rate* obtenido es del 11.8% y el porcentaje de aciertos 88.2%. A primera vista podría parecer un error bajo, sin embargo, teniendo en cuenta que solo el 6% de los consumidores suscribe el seguro, es de esperar un error del 6% solo con predecir "NO" a todos los consumidores sin tener en cuenta el valor de los predictores.

El presente ejemplo es una clara situación en las que el % de casos favorables (en este ejemplo que "SI" paguen el seguro) es muy bajo. La compañía podría intentar contactar con todos los consumidores (algo normalmente inviable) o con una muestra aleatoria de ellos y en promedio solo conseguiría un 6% de contrataciones del seguro. Esto supone un gasto muy alto para un beneficio muy pequeño. Sería mucho más competitivo invertir recursos en intentar venderle el seguro solo a aquellos consumidores que tienen mayor probabilidad de aceptarlo. Por lo tanto, es este tipo de escenarios, el % de error total no es de mucho interés, sino el porcentaje de consumidores que han sido predichos como compradores del seguro y que realmente lo son.

Habiendo empleado *K-NN* con *k=1*, de los *68+9=77* consumidores que el modelo predice como compradores, 9 sí lo son. Por lo tanto, el % de aciertos que la compañía lograría focalizándose en estos consumidores sería del *9/77=11.7%*. Casi duplicando la cantidad que se obtendría seleccionado consumidores de forma aleatoria.

Empleando *k=5*, el % de acierto entre los individuos predichos por el modelo como compradores es de *4/(11+4) = 26.7%*. Cuatro veces más que si se eligiesen consumidores de forma aleatoria.

```
set.seed(1)
prediccion_knn <- knn(train = Caravan_stand[-test, ], test=Caravan_stand[test, ],
cl = Caravan$Purchase[-test], k = 5)

table(prediccion = prediccion_knn, valor_real = Caravan[test, "Purchase"])
```

```
##           valor_real
## prediccion  No Yes
##        No  930  55
##        Yes  11   4
```

Ejemplo 3. Regresión Logística. Predicción de potenciales compradores, importancia del *threshold* de decisión.

Este ejemplo muestra cómo crear un modelo de predicción con el mismo fin que el ejemplo 2 pero esta vez, dado que la variable cualitativa tiene dos niveles, utilizando regresión logística.

```
modelo_glm <- glm(Purchase ~ ., data = Caravan[-test, ], family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
prediccion <- predict(object = modelo_glm, newdata = Caravan[test, ], type =
"response")
contrasts(Caravan$Purchase)
```

```
##     Yes
## No    0
## Yes   1
```

Para estudiar el impacto que tiene el *threshold* que se emplea para asignar cada nueva observación a una de las dos clases en función de la probabilidad predicha por el modelo, se procede a calcular el *test error* empleando como *threshold* $p=0.5$ y $p=0.25$.

```
# threshold de p=0.5. Si la probabilidad predicha es mayor de 0.5 se asigna
# a la clase Yes, si es menor a la clase No
prediccion[prediccion > 0.5] <- "Yes"
prediccion[prediccion != "Yes"] <- "No"
table(prediccion = prediccion, valor_real = Caravan[test, "Purchase"])
```

```
##           valor_real
## prediccion  No Yes
##        No  934  59
##        Yes   7   0
```

Al emplear un *threshold* de $p=0.5$, se está siendo demasiado estricto, el modelo no ha generado ninguna predicción con suficiente probabilidad para ser considerada como comprador del seguro.

```r
# threshold de p=0.25. Si la probabilidad predicha es mayor de 0.5 se asigna
# a la clase Yes, si es menor a la clase No
prediccion[prediccion > 0.25] <- "Yes"
prediccion[prediccion != "Yes"] <- "No"
table(prediccion = prediccion, valor_real = Caravan[test, "Purchase"])
```

```
##           valor_real
## prediccion  No Yes
##        Yes 941  59
```

Al disminuir el *threshold* se está rebajando la condición para poder considerar a un consumidor como posible comprador del seguro. Empleando un *threshold* de 0.25 se consigue una proporción de aciertos dentro de los identificados por el modelo como compradores de *11/(22+11)= 33.3%*, más de 5 veces lo esperado por selección aleatoria.

Conclusión

Tanto el modelo *K-NN* como la regresión logística, escogiendo bien los parámetros (*k* en *K-NN* y el *threshold* en log.regresión) permiten identificar de forma eficiente posibles compradores, lo que se traduce en una muy buena optimización de los recursos de una compañía. De entre todos los modelos, el que tiene mayor ratio de aciertos es el de regresión logística con un *threshold* de *p=0.25*.

Bibliografía

An Introduction to Statistical Learning