

Práctica 2: Regresión

Grupo 4

Iria Lago Portela
Mario Picáns Rey
Javier Kniffki
David Bamio Martínez

En primer lugar vamos a cargar el conjunto de datos para la realización de esta práctica:

```
load("College4.RData")  
datos<-College4[,-1]
```

Nuestro conjunto de datos contiene estadísticas de 500 universidades públicas y privadas de EE.UU., para las cuales se observaron 16 variables. A continuación mostramos las tres primeras filas del conjunto de datos:

```
head(datos,n=3)
```

```
##               Apps   Accept   Enroll Top10perc Top25perc  
## University of Southern Colorado 7.244942 7.122060 6.405228      10      34  
## University of San Francisco 7.743270 7.450661 6.287859      23      48  
## Clarkson University 7.684324 7.577122 6.322565      35      68  
##  
## P.Undergrad Outstate Room.Board Books Personal  
## University of Southern Colorado 6.514713 7.100 4.380 5.4 2.948  
## University of San Francisco 6.308098 13.226 6.452 7.5 2.450  
## Clarkson University 3.970292 15.960 5.580 7.0 1.300  
##  
## PhD Terminal S.F.Ratio perc.alumni Expend  
## University of Southern Colorado 63 88 19.4 0 5.389  
## University of San Francisco 86 86 13.6 8 10.074  
## Clarkson University 95 95 15.8 32 11.659  
##  
## Grad.Rate  
## University of Southern Colorado 36  
## University of San Francisco 62  
## Clarkson University 77
```

Consideraremos como variable respuesta la variable `Accept`, número de solicitudes aceptadas (en escala logarítmica), y como predictores el resto de variables numéricas del conjunto de datos.

Ejercicios

1. Ajustar un modelo lineal con penalización *lasso* a los datos de entrenamiento

a. Seleccionar el parámetro λ de regularización por validación cruzada empleando el criterio de un error estándar.

Comenzaremos utilizando el 80% de los datos como muestra de entrenamiento y el 20% restante como muestra de test. Establecemos como semilla el número del grupo multiplicado por 10, utilizando la función `set.seed` de R:

```
set.seed(40)
nobs <- nrow(datos)
itrain <- sample(nobs, 0.8 * nobs)
train <- datos[itrain, ]
test <- datos[-itrain, ]
```

Para este ejercicio utilizaremos el paquete `glmnet`. Este paquete no emplea formulación de modelos, sino que hay que establecer la respuesta `y` y una matriz o `data.frame` con las variables explicativas `x`. Como ya hemos comentado, la variable respuesta será `Accept` y las variables explicativas serán el resto de variables numéricas del conjunto de datos.

```
x<-as.matrix(train[,-2])
y<-train$Accept
```

A continuación ajustamos el modelo lineal con penalización *lasso*:

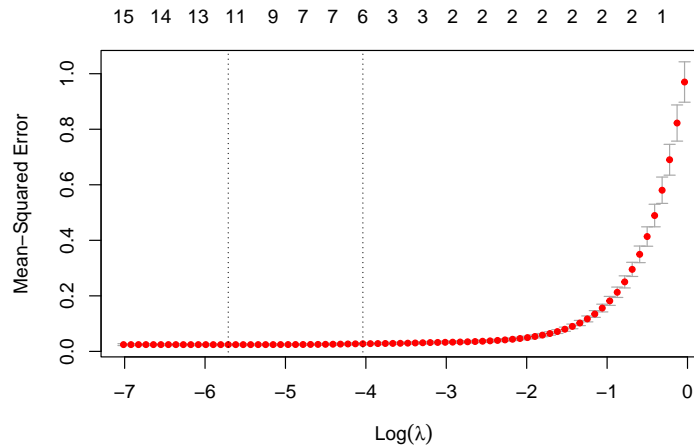
```
library(glmnet)

set.seed(40)
cv.lasso<-cv.glmnet(x,y,alpha=1)
```

Para seleccionar el parámetro de penalización por validación cruzada hemos utilizado la función `cv.glmnet()` y especificando `alpha=1` (opción por defecto) se utiliza la penalización *lasso*.

Podemos representar el error cuadrático medio con respecto a los valores del logaritmo de λ :

```
plot(cv.lasso)
```



Este gráfico representa de izquierda a derecha los valores del error cuadrático medio desde un modelo más complejo hasta el modelo más simple. Además, aparecen dos rectas verticales correspondientes a los valores del $\log(\lambda)$ que minimizan el error de validación cruzada y la regla del error estándar respectivamente. Por último, en la parte superior se indica el número de coeficientes no nulos de los modelos correspondientes a cada valor del logaritmo de λ .

El parámetro óptimo según la regla de un error estándar es:

```
cv.lasso$lambda.1se
```

```
## [1] 0.01764946
```

b. Obtener los coeficientes del modelo y evaluar las predicciones en la muestra de test (gráfico y medidas de error).

Para obtener los coeficientes del modelo con el parámetro de penalización calculado en el apartado anterior podemos usar la función `coef()` de R:

```
coef(cv.lasso,s="lambda.1se")
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) 0.5260302287
## Apps        0.6629303971
## Enroll      0.2743080806
## Top10perc   -0.0022815161
## Top25perc    .
## P.Undergrad .
## Outstate    .
## Room.Board  .
## Books       -0.0029141385
## Personal    .
## PhD         .
## Terminal    0.0007474212
## S.F.Ratio   .
```

```
## perc.alumni .
## Expend      -0.0009544681
## Grad.Rate   .
```

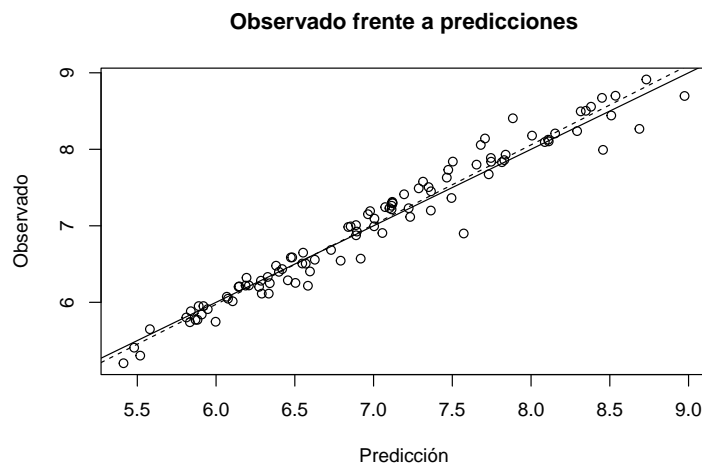
La penalización lasso fuerza a que algunos de los parámetros sean cero. En nuestro caso los parámetros asociados a las variables Apps, Enroll, Top10perc, Books, Terminal y Expend son distintos de cero, por lo que estas variables serán las más influyentes en el modelo. Además, la estimación del intercepto es $\hat{\beta}_0 = 0.5260$.

Por último evaluaremos las predicciones en la muestra de test:

```
newx<-as.matrix(test[,-2])
pred_lasso<-predict(cv.lasso,newx=newx,s="lambda.1se")
```

Podemos representar las predicciones frente a los valores observados:

```
obs<-test$Accept
plot(pred_lasso, obs, main = "Observado frente a predicciones",
     xlab = "Predicción", ylab = "Observado")
abline(a=0,b=1)
res <- lm(obs ~ pred_lasso)
abline(res, lty = 2)
```



Como podemos observar los valores observados frente a las predicciones se disponen en la diagonal, lo que indica que el modelo ajusta bien los datos. Podemos comprobarlo calculando las medidas de error:

```
accuracy <- function(pred, obs, na.rm = FALSE,
                     tol = sqrt(.Machine$double.eps)) {
  err <- obs - pred # Errores
  if(na.rm) {
    is.a <- !is.na(err)
    err <- err[is.a]
    obs <- obs[is.a]
  }
  perr <- 100*err/pmax(obs, tol) # Errores porcentuales
```

```

return(c(
  me = mean(err), # Error medio
  rmse = sqrt(mean(err^2)), # Raíz del error cuadrático medio
  mae = mean(abs(err)), # Error absoluto medio
  mpe = mean(perr), # Error porcentual medio
  mape = mean(abs(perr)), # Error porcentual absoluto medio
  r.squared = 1 - sum(err^2)/sum((obs - mean(obs))^2) # Pseudo R-cuadrado
))
}

obs<-test$Accept
accuracy(pred_lasso,obs)

```

```

##           me           rmse           mae           mpe           mape  r.squared
## 0.01577432 0.18590434 0.14066577 0.09065397 1.97939610 0.95832058

```

Para ello hemos definido la función `accuracy`, donde calculamos el error medio, la raíz del error cuadrático medio, el error absoluto medio, el error porcentual medio, el error porcentual medio absoluto y el pseudo R-cuadrado. Obtuvimos que el 95.8% de la variabilidad se encuentra explicada con estos datos.

c. ¿Cuál sería el número de coeficientes distintos de cero si se selecciona λ de forma que minimice el error de validación cruzada?

El valor de λ que minimiza el error de validación cruzada viene dado por:

```
cv.lasso$lambda.min
```

```
## [1] 0.003307187
```

Luego si consideramos este parámetro en el modelo lineal con penalización lasso obtenemos los siguientes coeficientes:

```
coef(cv.lasso,s="lambda.min")
```

```

## 16 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) 0.3305471323
## Apps        0.6650541136
## Enroll      0.3048456195
## Top10perc   -0.0040941138
## Top25perc   .
## P.Undergrad -0.0020981289
## Outstate    0.0137904027
## Room.Board  .
## Books       -0.0112841212
## Personal    .
## PhD         0.0001411949
## Terminal    0.0017854324
## S.F.Ratio   -0.0017354895
## perc.alumni .
## Expend      -0.0096960052
## Grad.Rate   -0.0004681128

```

Es decir, pasaríamos de 6 coeficientes distintos de cero a 11 coeficientes. En este caso, además de las variables anteriormente mencionadas, los coeficientes asociados a las variables P.Undergrad, Outstate, PhD, S.F.Ratio y Grad.Rate serían distintos de cero.

2. Ajustar un modelo mediante regresión spline adaptativa multivariante (MARS) empleando el método "earth" del paquete caret.

a. Utilizar validación cruzada con 5 grupos para seleccionar los valores “óptimos” de los hiperparámetros considerando `degree = 1` y `nprune = c(5, 10, 15, 20)`, y fijar `nk = 30`.

Para la resolución de este apartado emplearemos la librería `caret`.

```
library(caret)
```

En primer lugar, para seleccionar la combinación óptima de hiperparámetros, creamos el objeto `tune.grid`. En él guardamos la combinación de grado máximo de interacción (argumento `degree`) y el número máximo de términos en el modelo final (argumento `nprune`).

```
tune.grid<-expand.grid(degree=1,  
                       nprune=c(5,10,15,20))
```

En este apartado ajustamos el modelo con la función `train` del paquete `caret`. Le indicamos que use el conjunto de datos de entrenamiento, el método `earth` (Enhanced Adaptive Regression Through Hinges) y el número máximo de términos en el crecimiento modelo (`nk`). El argumento `trControl` sirve para indicar, entre otras cosas, que use validación cruzada (`method="cv"`) para la selección óptima de los hiperparámetros. Finalmente en `tuneGrid` le indicamos la rejilla que definimos anteriormente.

```
set.seed(40)  
caret.mars<-train(Accept~.,data=train,method="earth",nk=30,  
                 trControl=trainControl(method="cv",number=5),  
                 tuneGrid=tune.grid)
```

Podemos consultar el modelo que ha calculado en el objeto `finalModel`, y los hiperparámetros escogidos en `bestTune`.

```
final.model<-caret.mars$finalModel  
caret.mars$bestTune
```

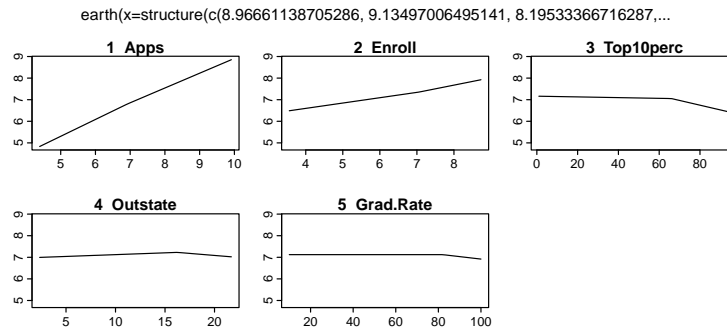
```
##  nprune degree  
## 2      10      1
```

b. Estudiar el efecto de los predictores incluidos en el modelo final y obtener medidas de su importancia.

En este apartado, comenzamos representando los efectos de los predictores incluidos en el modelo final para poder interpretarlos. Para ello, nos ayudamos de la librería `plotmo`:

```
library(plotmo)  
plotmo(final.model)
```

```
## plotmo grid:   Apps   Enroll Top10perc Top25perc P.Undergrad Outstate
##               7.388328 6.124681      24      55      6.002633   9.995
## Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni Expend Grad.Rate
##           4.178    5      1.2  76      84      13.6           20  8.435      65
```



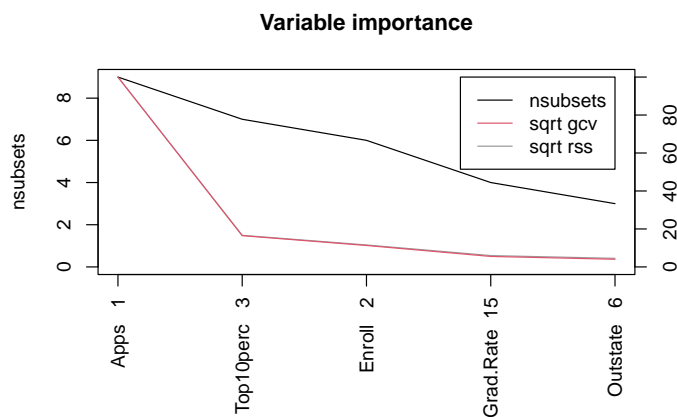
En primer lugar, vemos que el efecto de **Apps** (Número de solicitudes recibidas) crece de forma prácticamente lineal a medida que aumenta su número; en segundo lugar, el efecto de **Enroll** (Número de nuevos estudiantes matriculados) podríamos decir que crece de forma casi lineal, aunque con una pendiente menor que en el caso anterior; en tercer lugar, el efecto de **Top10perc** va decreciendo de forma gradual hasta pasado el valor 60, donde decrece de forma drástica; en cuarto lugar, el efecto **Outstate** (Número de estudiantes de otro estado) crece ligeramente hasta pasado el valor 15, donde empieza a decrecer; por último, **Grad.Rate** (Tasa de graduación) tiene un efecto casi plano hasta el valor 80, donde comienza a decrecer.

Continuamos estudiando la importancia de las variables con la función `evimp` del paquete `earth`:

```
varimp<-evimp(final.model)
varimp
```

```
##           nsubsets   gcv    rss
## Apps           9 100.0  100.0
## Top10perc       7  16.5   16.7
## Enroll          6  11.3   11.6
## Grad.Rate       4   5.5    6.0
## Outstate        3   4.0    4.5
```

```
plot(varimp)
```



Tanto en la salida de la función, como en el gráfico, podemos ver cuáles son las variables más importantes. Observamos que el número de solicitudes recibidas es la variable que tiene mayor importancia, después tendríamos la variable **Top10perc**, luego el número de nuevos estudiantes matriculados, y por último la tasa de graduación y el número de estudiantes de otro estado.

c. Evaluar las predicciones en la muestra de test.

Para hacer las predicciones emplearemos la función `predict` del paquete base de R, como viene siendo habitual. Empleamos el 20% restante de los datos que se guardaron como muestra de test:

```
obs<-test$Accept
pred_mars<-predict(caret.mars,newdata=test)
```

Para poder evaluar estas predicciones, nos ayudaremos de la función `accuracy` que definimos en el primer ejercicio:

```
accuracy(pred_mars,obs)
```

```
##          me          rmse          mae          mpe          mape r.squared
## 0.0209189 0.1630819 0.1203098 0.2206453 1.6919972 0.9679259
```

En este caso obtenemos un $R^2 = 0.9679$, es decir, este modelo explica un 96.79% de la variabilidad de los datos. En el apartado anterior obtuvimos un $R^2 = 0.9583$, por lo que este modelo podría explicar un 1% mejor la variabilidad.

3. Volver a ajustar el modelo aditivo del ejercicio anterior empleando la función `gam()` del paquete `mcgv`.

a. Incluir los efectos no paramétricos de los predictores seleccionados por el método MARS.

Las variables predictoras seleccionadas por el método MARS en el ejercicio anterior eran **Apps**, **Enroll**, **Top10perc**, **Outstate** y **Grad.Rate**, donde no se consideraron las posibles interacciones. Así pues, ajustamos el modelo GAM con dichas variables predictoras:


```
library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-38. For overview type 'help("mgcv-package")'.

set.seed(40)
gam<-gam(Accept~s(Apps)+s(Enroll)+s(Top10perc)+s(Outstate)+s(Grad.Rate),data=train,select=T)
summary(gam)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Accept ~ s(Apps) + s(Enroll) + s(Top10perc) + s(Outstate) + s(Grad.Rate)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.175763   0.006178   1161   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df      F  p-value
## s(Apps)      3.044     9 158.079 < 2e-16 ***
## s(Enroll)    3.712     9  17.935 < 2e-16 ***
## s(Top10perc) 5.351     9  13.954 < 2e-16 ***
## s(Outstate)  4.784     9   4.144 4.56e-07 ***
## s(Grad.Rate) 4.642     9   5.263 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.984   Deviance explained = 98.5%
## GCV = 0.016179   Scale est. = 0.015268   n = 400
```

b. Evaluar las predicciones en la muestra de test y comparar los resultados con los métodos anteriores.

```
obs<-test$Accept
pred_gam<-predict(gam,newdata=test)
accuracy(pred_gam,obs)

##           me           rmse           mae           mpe           mape  r.squared
## 0.02268146 0.16399568 0.11981018 0.25344546 1.69932702 0.96756549
```

Recordemos la evaluación de los errores con los métodos anteriores:

```
accuracy(pred_lasso,obs)
```

```
##           me           rmse           mae           mpe           mape  r.squared  
## 0.01577432 0.18590434 0.14066577 0.09065397 1.97939610 0.95832058
```

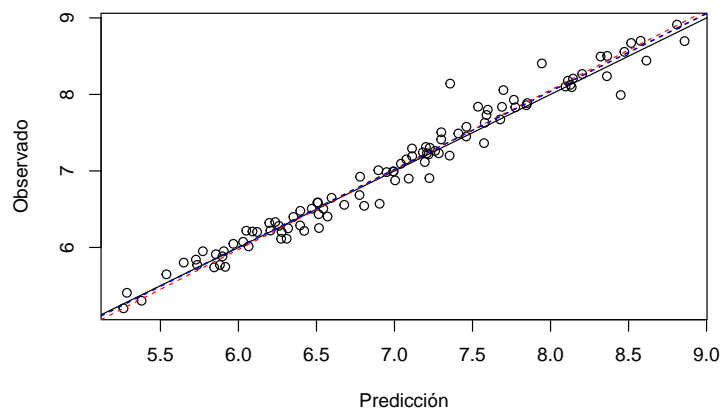
```
accuracy(pred_mars,obs)
```

```
##           me           rmse           mae           mpe           mape r.squared  
## 0.0209189 0.1630819 0.1203098 0.2206453 1.6919972 0.9679259
```

Todas las predicciones arrojan un pseudo-R-cuadrado elevado, superior al 95%, por lo que los modelos ajustados explicarían casi toda la variabilidad de la respuesta. El modelo GAM parece explicar ligeramente mejor la respuesta que el modelo lasso y prácticamente no hay diferencias con el modelo MARS.

Podemos observar el gráfico de observaciones frente a las predicciones obtenidas por el último modelo, en el que los puntos deberían situarse en torno a la recta $y = x$. Se representan también con línea discontinua los ajustes lineales obtenidos por los tres modelos.

```
plot(pred_gam, obs, xlab = "Predicción", ylab = "Observado")  
abline(a = 0, b = 1)  
res_gam <- lm(obs ~ pred_gam)  
abline(res_gam, lty = 2)  
res_lasso <- lm(obs ~ pred_lasso)  
abline(res_lasso, lty = 2,col="red")  
res_mars <- lm(obs ~ pred_mars)  
abline(res_mars, lty = 2,col="blue")
```



Como ya habíamos visto, todos los ajustes son muy próximos a la identidad y también muy similares entre sí. También se puede observar que la línea azul del ajuste MARS (que está prácticamente superpuesto con el GAM) está ligeramente más próxima a la identidad que la línea roja del ajuste lasso.