EXAMEN Cómputo estadístico

J. Antonio García Ramirez

Octobre 11, 2018

Ejercicio 7

a) Comenzaré estandarizando el conjunto de datos para poder comparar los modelos de mejor subconjunto, Lasso, Ridge y PCR

En vista de que se nos requiere trabajar todas las variables como continuamos transformamos las nominales y ordinales en flotantes.

El conjunto de datos no presenta valores faltantes dividire el conjunto original en 320 observaciones de entrenamiento y el resto de prueba. Los errores los reporto en términos del MSE.

Primero la regresión Ridge y lasso sobre el conjunto de entrenamiento y reportó lo MCE pertinentes obtenidos con 10-fold cv

comenzamos con los resultados de Ridge

```
set.seed(0)
library(ISLR)
datos <- Carseats
class(datos)
```

```
## [1] "data.frame"
```

```
sapply(datos, class)
```

```
##       Sales   CompPrice      Income Advertising  Population       Price
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##   ShelveLoc         Age   Education       Urban          US
##    "factor"   "numeric"   "numeric"    "factor"    "factor"
```

```
datos$ShelveLoc <- as.numeric(datos$ShelveLoc)
datos$Urban <- as.numeric(datos$Urban)
datos$US <- as.numeric(datos$US)
n <- dim(datos)[1]
class(datos)
```

```
## [1] "data.frame"
```

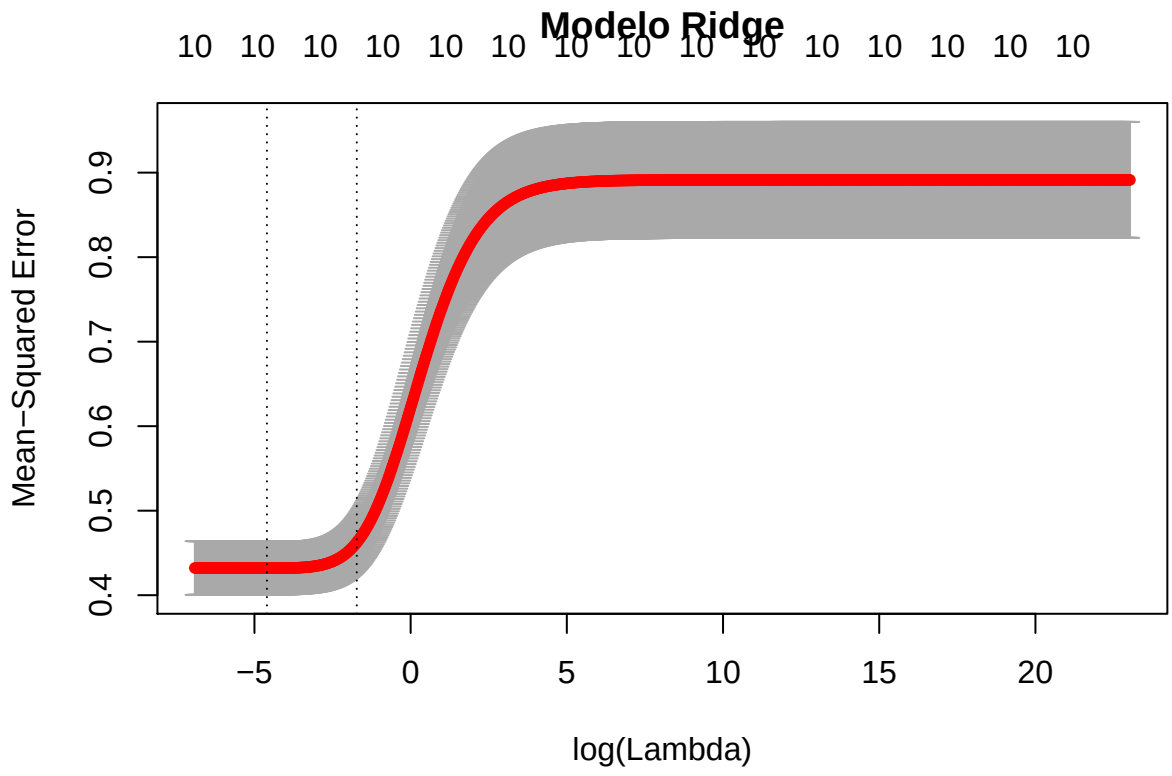
```
datos <- as.matrix(datos)
datos <- scale((datos))
index <- sample(1:dim(datos)[1], n*.8 )
datos <- as.data.frame(datos)
y <- datos$Sales
y.test <- y[-index]
y.train <- y[index]
train <- datos[index, ]
test <- datos[-index, ]
###############
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
train2 <- model.matrix (Sales~., train )
grid <- 10^seq(-3,10,length =1000)
set.seed(0)
modelo.Ridge <- cv.glmnet(train2, train$Sales, alpha =0,
                          lambda =grid, nfolds = 10)
plot(modelo.Ridge, main='Modelo Ridge')
```




```
(l <- modelo.Ridge$lambda.min)
```

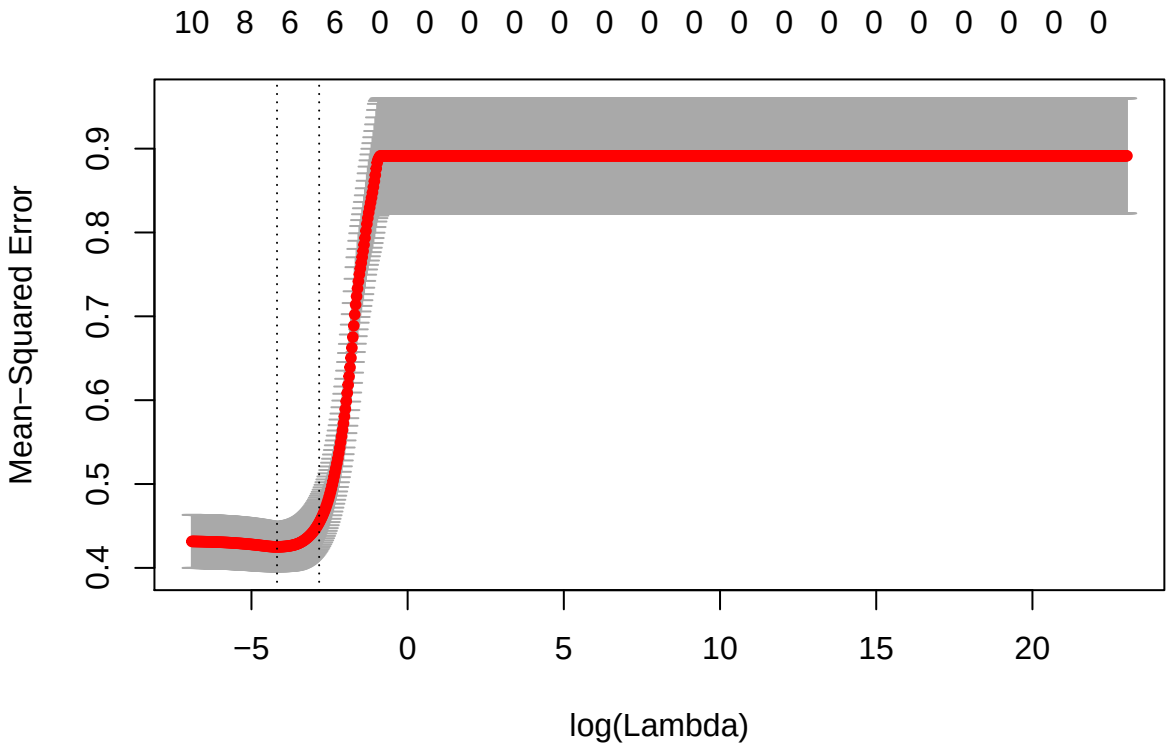
```
## [1] 0.0100462
```

```
test2 <- model.matrix(Sales ~. , test)
y.hat.rige <- predict(modelo.Ridge, test2)
MSE(as.numeric(y.hat.rige), y.test)
```

```
## [1] 0.6448648
```

Continuamos con el modelo de Lasso

```
set.seed(0)
modelo.lasso <- cv.glmnet(train2, train$Sales, alpha =1,
                          lambda =grid, nfolds = 10)
plot(modelo.lasso)
```




```
(l <- modelo.lasso$lambda.min)
```

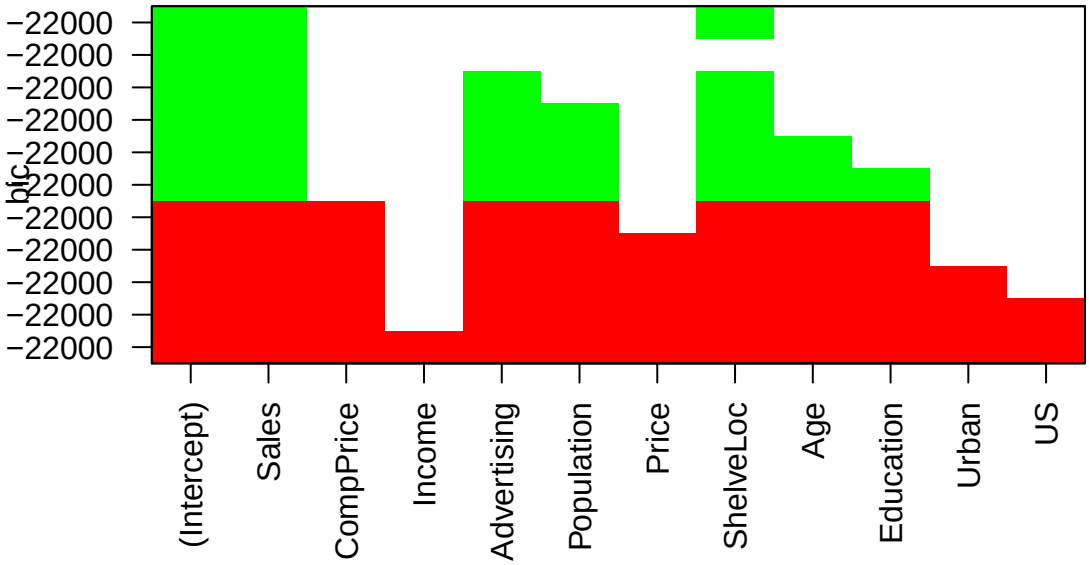
```
## [1] 0.01528214
```

```
test2 <- model.matrix(Sales ~. , test)
y.hat.lasso <- predict(modelo.lasso, test2)
MSE(y.hat.lasso, y.test)
```

```
## [1] 0.6441137
```

Por brevedad sólo mostraremos resultados del modelo de selección de variables exhaustivo

```
library(leaps)
modelos <- regsubsets(y.train ~ ., data =train, method = 'exhaustive', nvmax = 20)
plot(modelos, scale="bic", col=c('green', 'red'))
```



Finalmente probamos con los modelos PCR y PLS

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

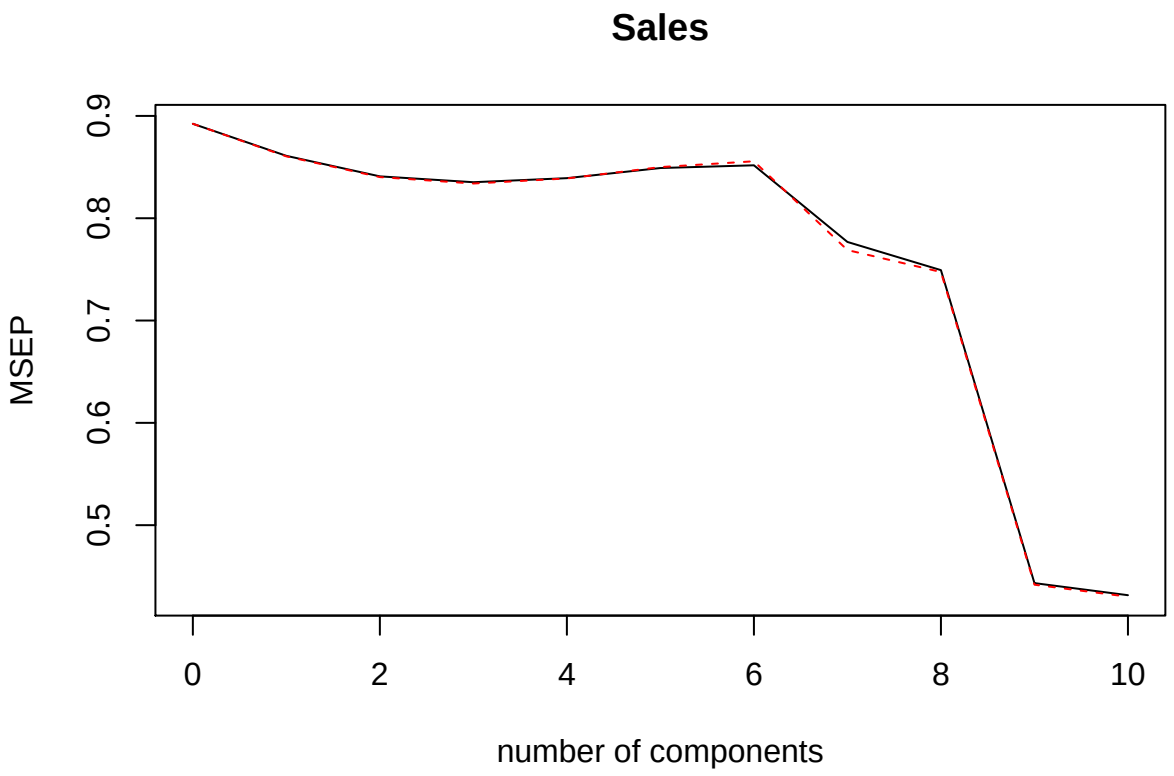
```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
set.seed(0)
modelo.pcr <- pcr(Sales ~ ., data=train,
                  validation ="CV")
summary(modelo.pcr)
```

```
## Data:    X dimension: 320 10
##  Y dimension: 320 1
## Fit method: svdpc
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.9446   0.9279   0.9170   0.9139   0.9161   0.9215   0.9229
## adjCV       0.9446   0.9276   0.9166   0.9132   0.9159   0.9219   0.9250
##        7 comps  8 comps  9 comps  10 comps
## CV      0.8814   0.8656   0.6658    0.6569
```

```
## adjCV   0.8769   0.8646   0.6647    0.6558
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        18.534   34.296   45.951   56.485   66.687   76.306    85.27
## Sales     4.242    7.278    8.054    8.183    8.336    9.318    17.61
##         8 comps  9 comps  10 comps
## X         93.45    97.39    100.00
## Sales     20.92    52.82     54.37
```

```
validationplot(modelo.pcr, val.type="MSEP")
```




```
y.hat.pcr <- predict(modelo.pcr, test, ncomp = 7)
MSE(as.numeric(y.hat.pcr), y.test)
```

```
## [1] 1.150588
```

e) *Ajusta un modelo **PLS** en el conjunto de entrenamiento, con **M** elegido por la validación cruzada. Informe el error de prueba obtenido, junto con el valor de **M** seleccionado mediante validación cruzada*

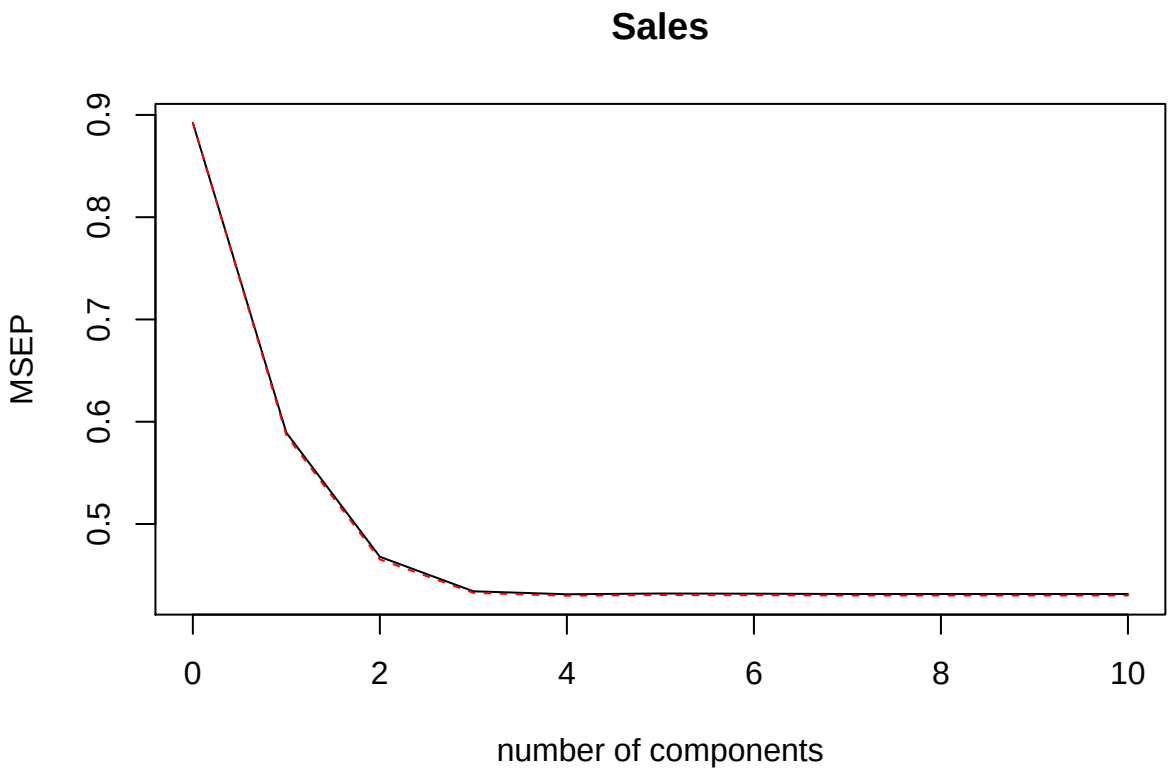
Utilizando validación cruzada, PLS como método de regresión y 7 componentes principales (en vista de que en el conjunto de entrenamiento la octava componente de ppls es de poca significancia) se obtiene un error de prueba de 0.06016777 sobre los datos escalados.

```
library(pls)
set.seed(0)
modelo.pls <- plsr(Sales ~ ., data=train, validation ="CV")
#summary(modelo.pls)
#validationplot(modelo.pls ,val.type="MSEP")
```

```r
y.hat.pls <- predict(modelo.pls, test, ncomp = 7)
MSE(as.numeric(y.hat.pls), y.test)
```

```
## [1] 0.5249173
```

```r
validationplot(modelo.pls, val.type="MSEP")
```



del análisis final observamos que el modelo de regresión con PLS mejora el MSE además de proporcionar un modelo más parsimonioso, por que al final del dia nunca sabemos en qué dimensión vive nuestro problema