CP002 - Regularización de Variables

Mayra Goicochea Neyra

17/10/2019

Caso NBA

Introducción

Se solicita realizar un modelo predictivo sobre la asignación de los salarios de los jugadores de la NBA. Para este trabajo se entregó la información historica de 485 casos con 28 variables. Los salarios se han asignado de la siguiente manera:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
library(kknn)
library(rsample)
```

```
## Loading required package: tidyr
```

```
library(glmnet)
```

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

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

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

```
## Loaded glmnet 2.0-18
```

```
library(ggplot2)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

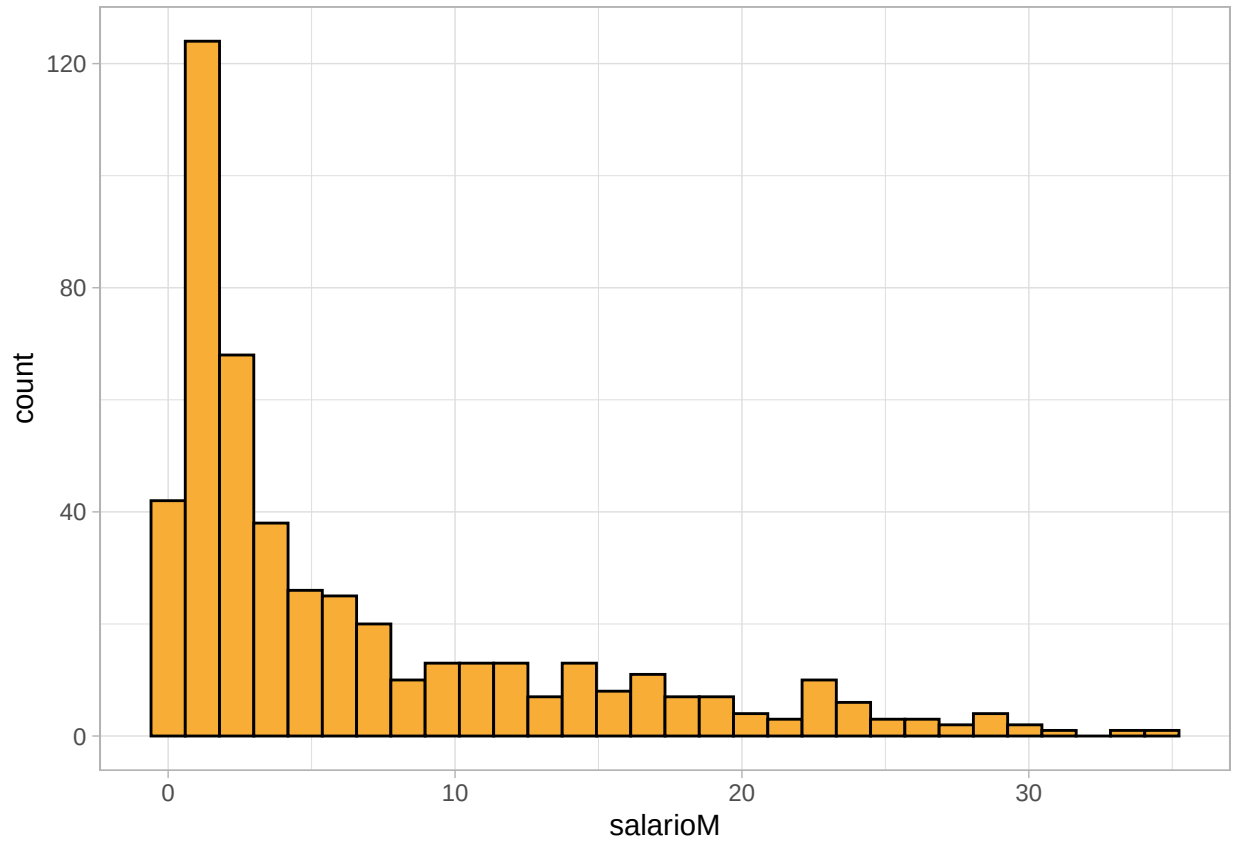
```r
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```r
setwd("C:/Users/Goicochea/Desktop/CUNEF/Cursos/Prediccion/Tarea")
mNba  <- read.csv("./nba.csv")
mNba %>% mutate(salarioM = (Salary/1000000)) %>%
  ggplot(aes(x = salarioM)) +
  geom_histogram(fill = "#f7ad36", colour="black") +
  theme_light()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

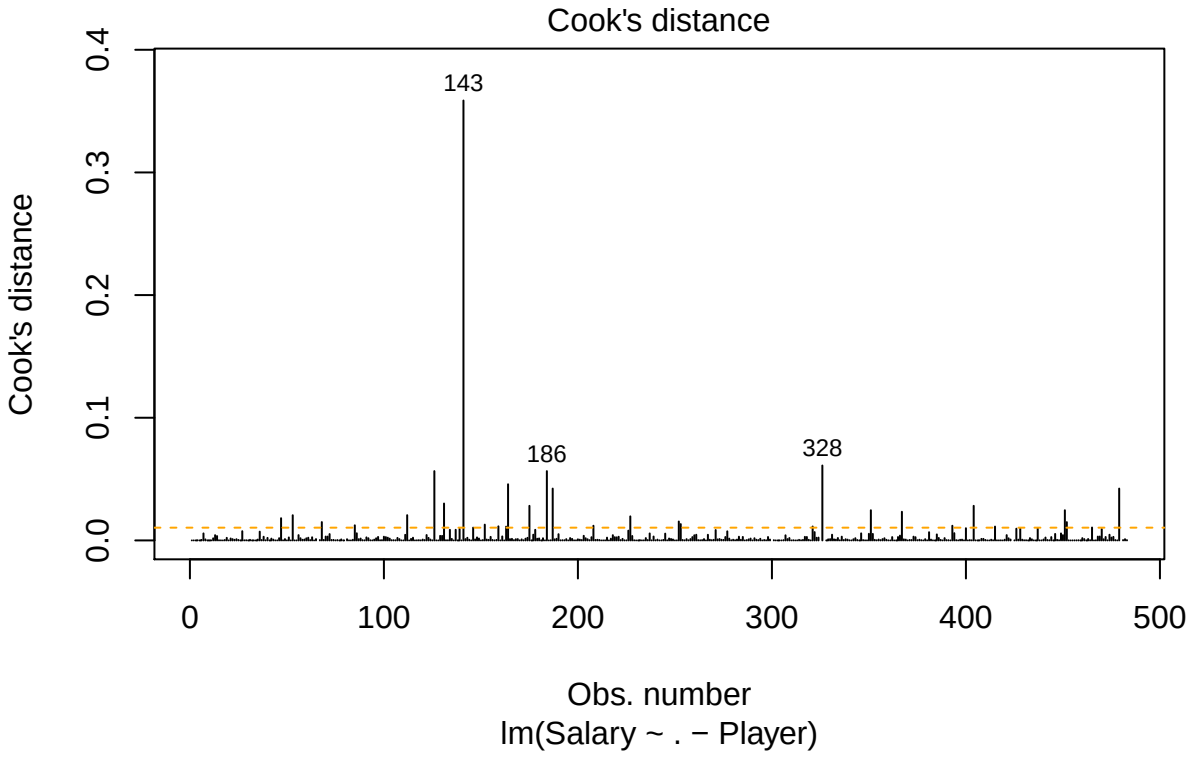


Se tiene la mediana de 3.2 millones de dólares y de salario medio de 6.6 millones de dólares.

Data Cleaning

En la practica anterior se ajustaron los valores ausentes. Asi como también se extrajeron los valores influyentes que son los siguientes:

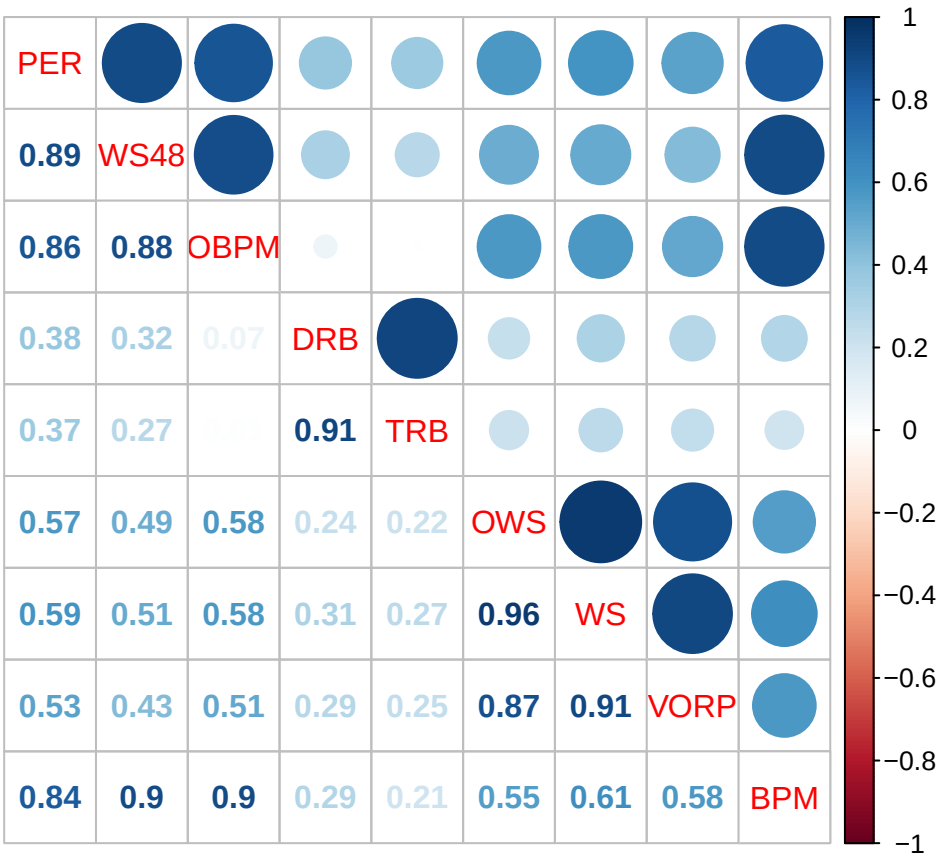
```r
reg0 <- lm(Salary~ .-Player, data = mNba)
cutoff <- 4/(nrow(mNba) - length(reg0$coefficients) - 2)
plot(reg0, which = 4, cook.levels = cutoff)
abline(h = cutoff, lty = 2, col = "orange")
```



Multicolinealidad

Sobre la correlación que puede haber entre las variables explicativas, se realiza una matriz de correlación. De la cual se obtuvo que las variables "Clasificación de eficiencia del jugador", "Ganar acciones por 48 minutos", "Box ofensivo Plus / Minus", "Porcentaje de rebote defensivo", "Porcentaje de rebote total", "Acciones ofensivas ganadoras", "Ganar acciones", "Valor sobre el jugador de reemplazo" y "Box Plus / Minus".

```
colnames(mNba) <- c("Player","Salary","NBA_Country","NBA_DraftNumber","Age","Tm","G","MP","PER","TS","T
mNba <- rminer::imputation("hotdeck",mNba,"TOV")
mNba <- rminer::imputation("hotdeck",mNba,"FTr")
mNba <- rminer::imputation("hotdeck",mNba,"TPAr")
mNba <- rminer::imputation("hotdeck",mNba,"TS")
mNba <- mNba[-c(2,49,114,143,328),]
corrplot::corrplot.mixed(cor(mNba[,c('PER','WS48','OBPM','DRB','TRB','OWS','WS','VORP','BPM')]))
```



El coeficiente de correlación entre éstas variables es muy alto, esto puede ocasionar inestabilidad en el modelo, a lo que denominamos Multicolinealidad, para verificar que se tiene este problema se calculó el Factor de inflación de la varianza y se determinó que 18 variables causan este problema:

```
reg1 <- lm(Salary~ .-Player-NBA_Country-Tm, data = mNba)
vif(reg1)
```

```
## NBA_DraftNumber             Age               G              MP
##        1.356804        1.079917        7.119223       14.844924
##             PER              TS            TPAr              FTr
##       77.451970        5.250164        5.185203        1.427709
##             ORB             DRB             TRB             AST
##      328.274963      710.063069     1498.266958        3.206107
##             STL             BLK             TOV             USG
##        3.198483        5.414707        2.099878        8.327910
##             OWS             DWS              WS            WS48
##     1321.744245      403.959547     2677.241239       53.141614
##            OBPM            DBPM             BPM            VORP
##     6254.874773     2105.037294    10231.608579       12.152298
```

```
rev.mc <- sqrt(vif(reg1)) > 2
sum(rev.mc)
```

```
## [1] 18
```

Modelo Anterior

En la practica CP001 se obtuvo un modelo con menor error, generado por la tecnica de Stepwise en ambas direcciones. Con un error de:

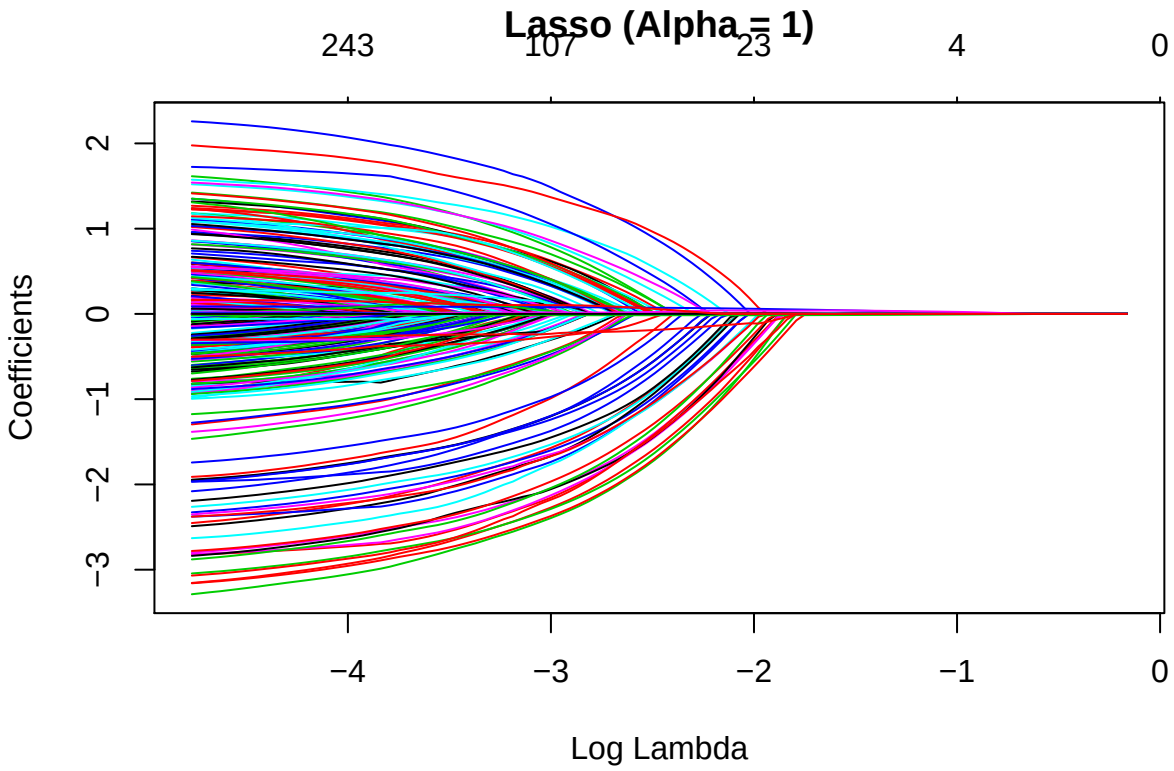
```
glm.fit4 = glm(Salary ~ NBA_DraftNumber + Age + G + MP + PER + TPAr + ORB + TRB + USG + WS + OBPM,mNba[
cv.err = cv.glm(mNba[,-c(1,3,6)],glm.fit4)
cv.err$delta[1]
```

```
## [1] 2.431333e+13
```

Regularización de las variables

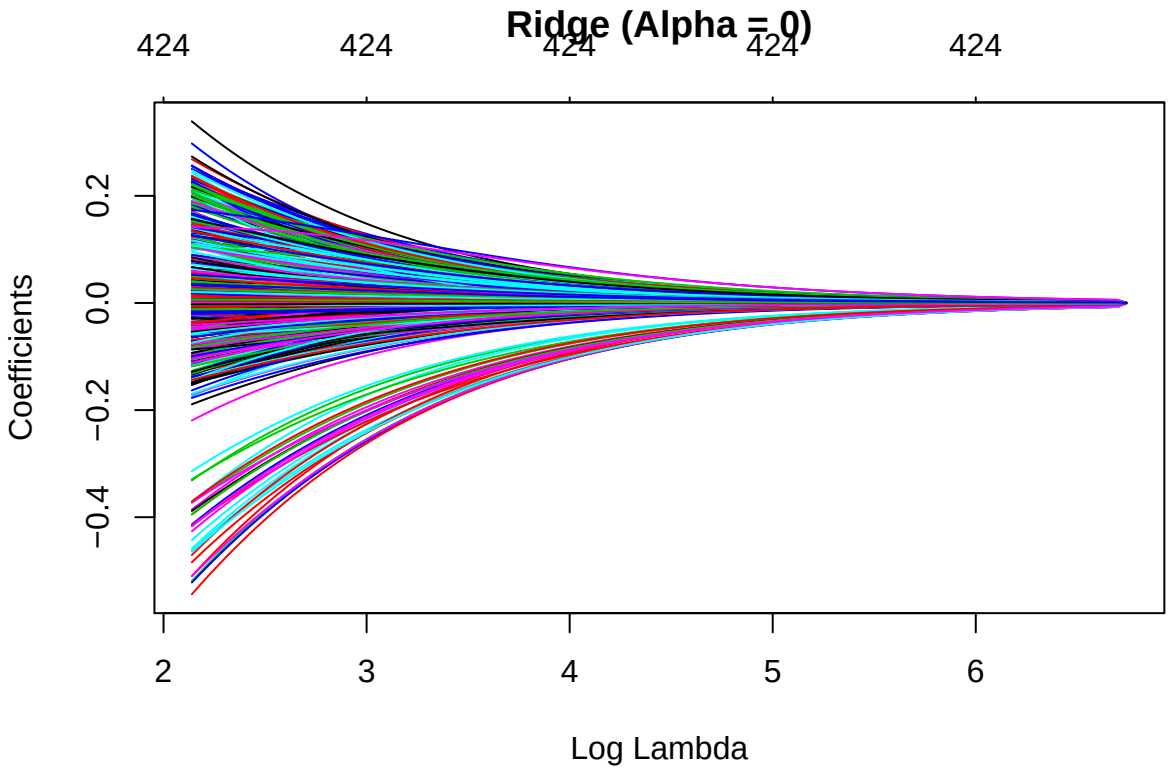
Dado que hay multicolinealidad en las variables, lo óptimo sería crear un modelo mediante la regularización para reducir este problema.

```
set.seed(123)
nba_split <- initial_split(mNba, prop = .7, strata = "Salary")
nba_train <- training(nba_split)
nba_test  <- testing(nba_split)
nba_train_x <- model.matrix(Salary ~ ., nba_train)[, -1]
nba_train_y <- log(nba_train$Salary)
nba_test_x <- model.matrix(Salary ~ ., nba_test)[, -1]
nba_test_y <- log(nba_test$Salary)
lasso    <- glmnet(nba_train_x, nba_train_y, alpha = 1.0, standardize = TRUE)
elastic1 <- glmnet(nba_train_x, nba_train_y, alpha = 0.25, standardize = TRUE)
elastic2 <- glmnet(nba_train_x, nba_train_y, alpha = 0.75, standardize = TRUE)
ridge    <- glmnet(nba_train_x, nba_train_y, alpha = 0.0, standardize = TRUE)
plot(lasso, xvar = "lambda", main = "Lasso (Alpha = 1)")
```



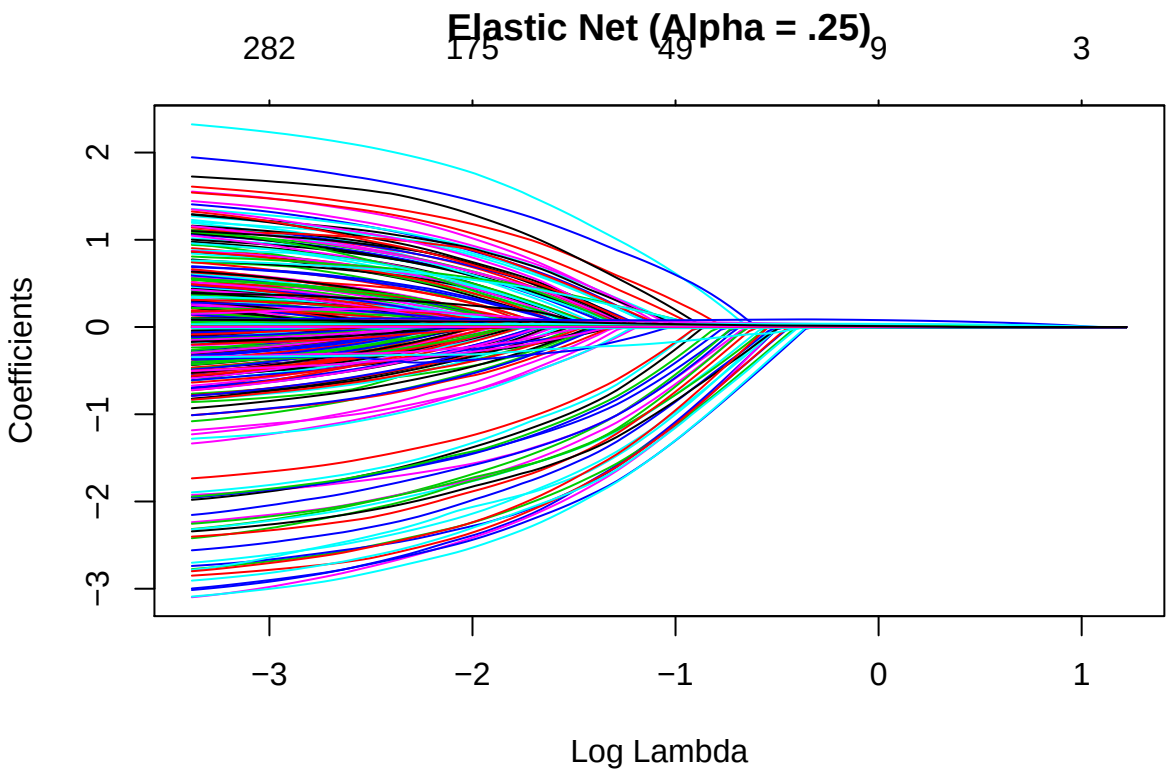
Siendo alpha de valor 1, se observa que muchas variables pierden valor cuando lambda va en aumento. La mayoria de coeficientes se convierten en valor 0.

```
plot(ridge, xvar = "lambda", main = "Ridge (Alpha = 0)")
```



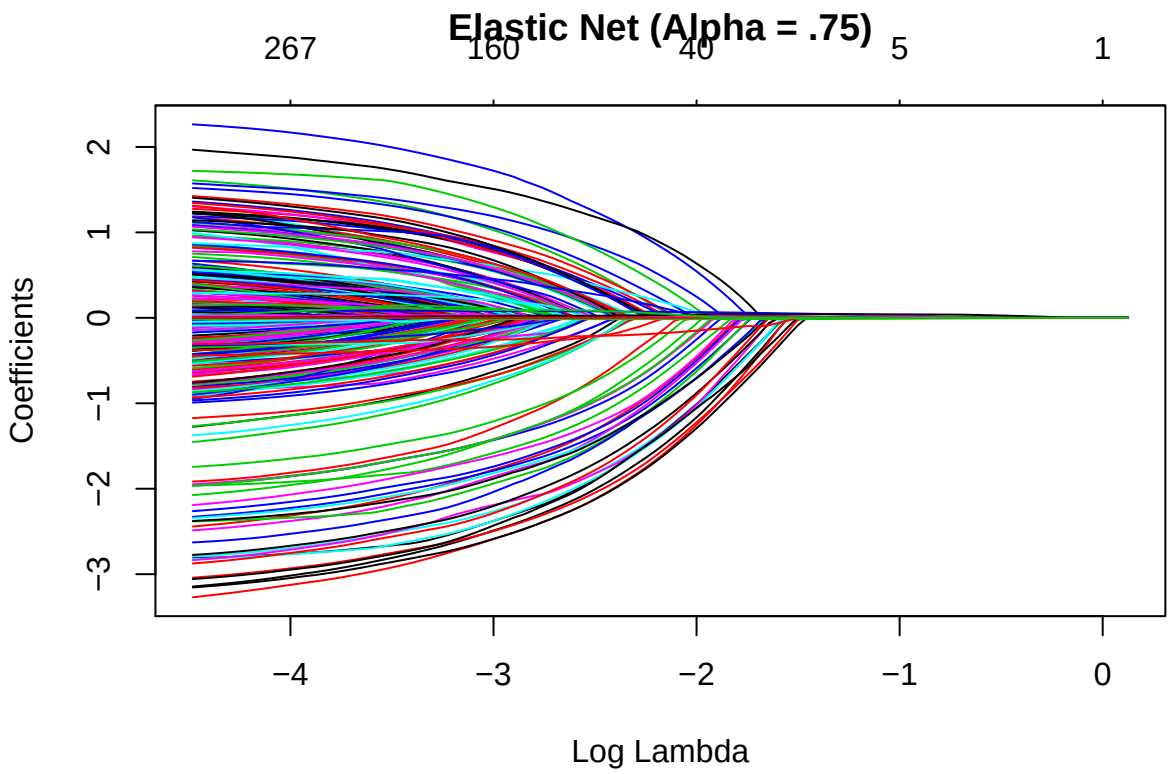
En cuanto a cuando alpha es 0, los coeficientes se mantiene con valores pequeños pero hasta lambda sea 6 cuando ya son casi 0.

```
plot(elastic1, xvar = "lambda", main = "Elastic Net (Alpha = .25)")
```



Cuando alpha es 0.25, los coeficientes pierden valor al llegar a un lambda de valor 0.

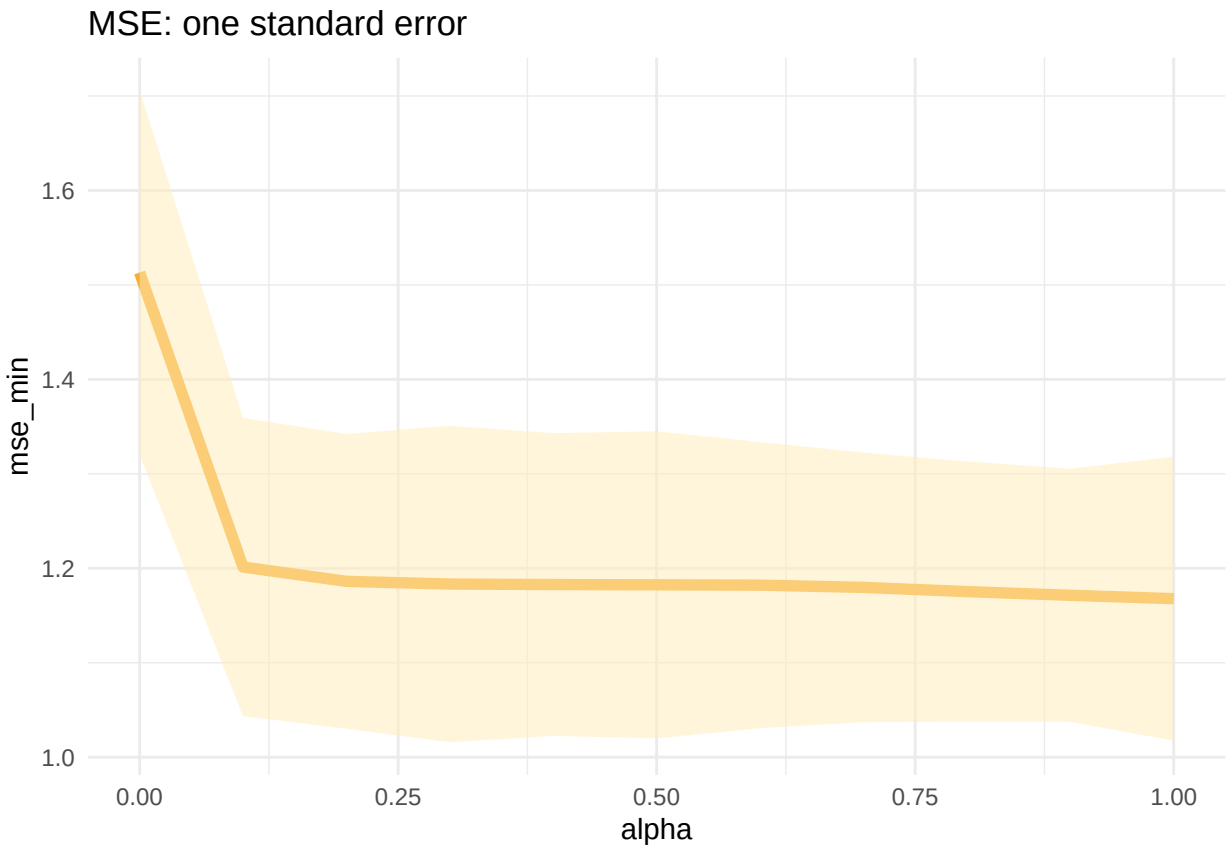
```
plot(elastic2, xvar = "lambda", main = "Elastic Net (Alpha = .75)")
```



A diferencia del caso anterior, con alpha igual a 0.75, los coeficienttes se van reduciendo a 0 cuando lambda es -1.

```
fold_id <- sample(1:10, size = length(nba_train_y), replace = TRUE)

tuning_grid <- tibble::tibble(
  alpha      = seq(0, 1, by = .1),
  mse_min    = NA,
  mse_1se    = NA,
  lambda_min = NA,
  lambda_1se = NA
)
for (i in seq_along(tuning_grid$alpha)) {
  fit <- cv.glmnet(nba_train_x, nba_train_y, alpha = tuning_grid$alpha[i], foldid = fold_id)
  tuning_grid$mse_min[i]    <- fit$cvm[fit$lambda == fit$lambda.min]
  tuning_grid$mse_1se[i]    <- fit$cvm[fit$lambda == fit$lambda.1se]
  tuning_grid$lambda_min[i] <- fit$lambda.min
  tuning_grid$lambda_1se[i] <- fit$lambda.1se
}
tuning_grid %>%
  mutate(se = mse_1se - mse_min) %>%
  ggplot(aes(alpha, mse_min)) +
  geom_line(size = 2, colour = "#f7ad36") +
  geom_ribbon(aes(ymax = mse_min + se, ymin = mse_min - se),fill = "#ffedb8", alpha = .5) +
  ggtitle("MSE: one standard error") +
  theme_minimal()
```



El modelo cuando alpha es 1 tiene el menor error. Es el caso del metodo lasso (alpha=1)

Conclusiones

- El modelo de lasso es el mejor modelo resultante, según el MSE obtenido.

```
cv_lasso   <- cv.glmnet(nba_train_x, nba_train_y, alpha = 1.0)
min(cv_lasso$cvm)
```

```
## [1] 1.133798
```

- Según Cross Validation, se tiene un minimo error de 1.15718 en la data de la muestra train.

```
pred <- predict(cv_lasso, s = cv_lasso$lambda.min, nba_test_x)
mean((nba_test_y - pred)^2)
```

```
## [1] 1.166295
```

- En cuanto se evalua con la data de la muestra test, se consigue un valor medio de error de 1.16.