UNIVERSIDAD ORT
Uruguay

O B L I G A T O R I O

Analítica de Negocios 2

Lic. En Gerencia y Administración
Docente: Lic. Guillermo Magnou

Pioli Gastón – N° 166371
Machado Cecilia – N° 213640
Silveyra Andrés – N° 207336

Fecha de entrega: 21 de junio de 2021

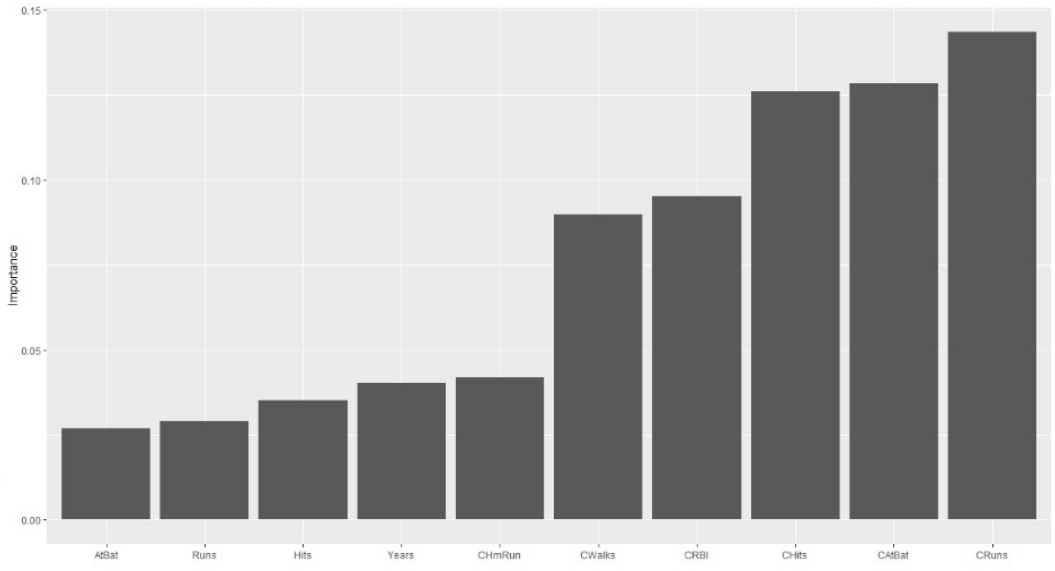
Índice

1. Introducción

En el siguiente informe se analizará la base de datos "*Hitters*" del paquete ISLR, la misma contiene 322 observaciones y 20 variables de jugadores de la MLB (Major League Baseball) para los años 1986 y 1987.
El objetivo es crear un modelo que permita, en base a las estadísticas históricas y del año anterior, estimar el salario para el próximo año.
Para elaborar este modelo se seguirá las fases de un proyecto analítico siguiendo la metodología CRISP-DM, es decir, definir el problema, entender los datos, modelar distintos modelos y evaluarlos, luego se determina el modelo a implementar y se comunica la decisión.

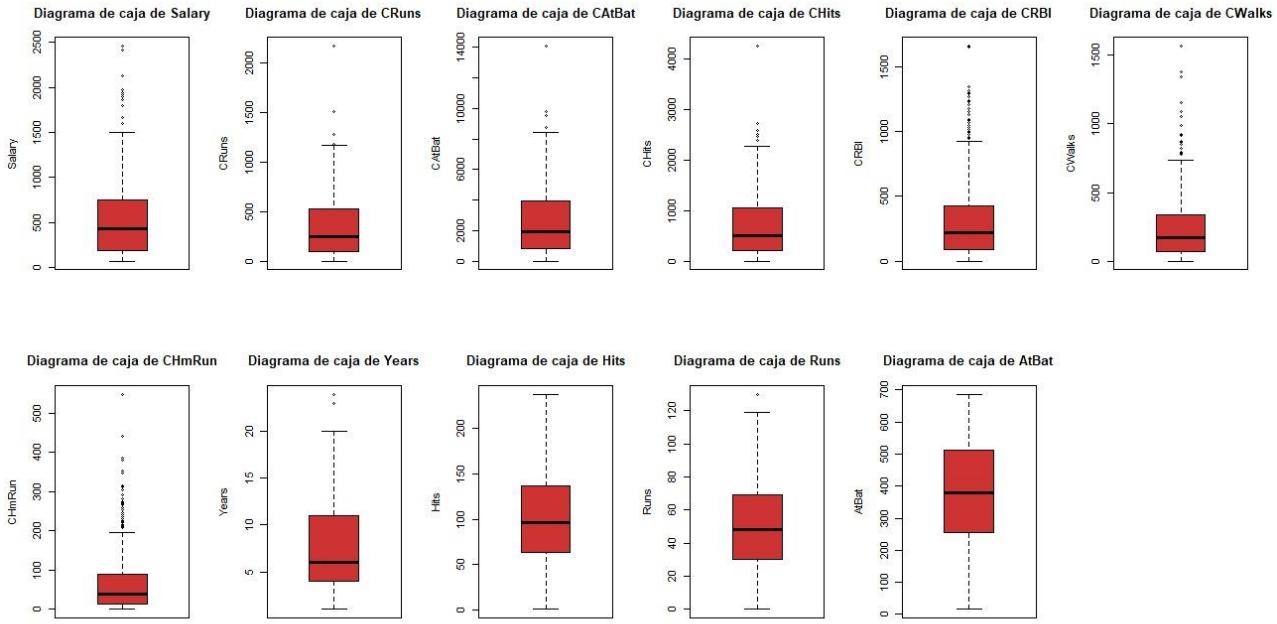
2. Análisis descriptivo

De los modelos realizados, seleccionamos el de Random Forest, es por esto que analizaremos las variables más relevantes del mismo. Para ello utilizamos la función `Vip,` incluida en la librería del mismo nombre.
En el siguiente gráfico, se visualizan dichas variables:



- Salary: Sueldo anual al inicio de la temporada de 1987 en miles de dólares.
- CRuns: Número de carreras durante su carrera.
- CAtBat: Número de veces “al bate” durante su carrera.
- CHits: Número de bolas bateadas durante su carrera.
- CRBI: Número de carreras bateadas durante su carrera.
- CWalks: Número de caminatas durante su carrera.
- CHmRun: Número de Home Runs durante su carrera.
- Years: Número de años en las ligas mayores.
- Hits: Número de bolas bateadas durante el año 1986.
- Runs: Número de carreras durante el año 1986.
- AtBat: Número de veces al bate durante el año 1986.

A continuación, se grafican Diagramas de Caja para cada una de las variables mencionadas.



El Salario promedio de las observaciones del dataset es de $535. En cuanto a la mayor diferencia de Salario la misma es de $2.392,5 , asimismo, el 50% central de los datos tiene una diferencia máxima de $560.

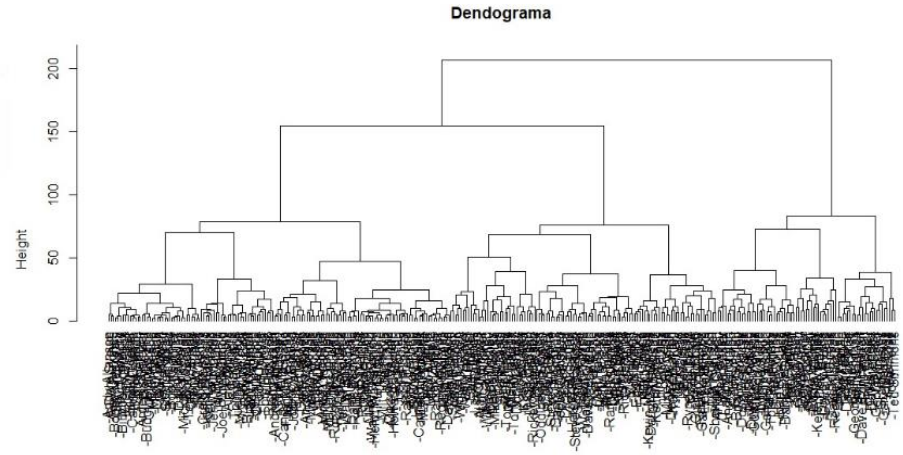
Se visualiza que las variables con mayor cantidad de outliers son: Salary, CRBI, CWalks y CHmRun. Además, la gran mayoría de las variables presentan un sesgo hacia la derecha debido a la presencia de los outliers mencionados.

3. Análisis de clustering

Debido a la existencia de 59 observaciones faltantes en la muestra, se nos plantea el problema de cómo proceder con el análisis de datos. Las opciones presentadas son:

- Eliminar las observaciones faltantes, lo que conllevaría a una pérdida aproximada de un 20% de la muestra.
- Imputar los valores faltantes mediante la clasificación de las observaciones utilizando técnicas de clustering y realizando la estimación de dichas observaciones como el promedio de sus compañeras de clúster.

Tomamos la decisión de ir por el segundo camino, generando clústeres mediante la función `Agnes` y distintos métodos y métricas. Dado a que nos encontramos con un outlier en los datos descartamos la métrica "`manhattan`" y decidimos utilizar la métrica "`euclídea`". A su vez, y con el fin de mejorar la clusterización, utilizamos el método "`ward`" con el cual obtuvimos el mejor resultado de Akaike, pasando de un valor inicial de `0.8413` a uno final de `0.9717`.
Con éste modelo obtuvimos el siguiente Dendograma:



Tras haber intentado realizar cortes a distintas alturas, pudimos formar distintos clústeres, pero nos fue imposible imputar las observaciones faltantes de forma de mantener una coherencia en los siguientes aspectos:

- Tamaño adecuado de clústeres sin que ello implique generalizar el valor imputado a los NA. De lo contrario, estaríamos imputando a un gran número de NA en un mismo valor.
- No deben existir clústeres conformados con todas las observaciones NA. De lo contrario, no podríamos imputar valores a las mismas.
- Similar cantidad de NA por clúster que garanticen una uniformidad en los valores a imputar.

Es por esto que no pudimos estimar los valores promedios de `Salary` para los datos faltantes. A raíz de esto, procedimos a eliminar éstas observaciones de la base de datos principal para poder seguir trabajando en los siguientes análisis.

4. Análisis de componentes principales

El objetivo del método de Análisis de Componentes Principales (ACP) es el de simplificar la complejidad de espacios muéstrales con muchas dimensiones a la vez que conserva su

información. Sin embargo, es de esperar que al reducir el número de componentes del modelo esto conlleve a una pérdida de información y por consiguiente a que el modelo resultante tenga un peor ajuste.
Esto queda evidenciado en nuestro trabajo al aplicar un método de ACP con 7 componentes al modelo de Random Forest previo:

	Train	CV	Test
Random Forest	2.93	7.67	7.31
Random Forest con ACP	4.28	8.92	8.57

5. Comparación y elección de modelos

Para el análisis de los datos fueron seleccionados 4 modelos, en el cual se utilizaron el 70% de ellos para Train y el restante 30% para Test. Dichos modelos son:

a. Random Forest

Para la creación de nuestro modelo de Random Forest tomamos la totalidad de las variables disponibles en la base de datos, tras omitir los NA y tomando como variable dependiente a "Salary". Se corre el modelo inicial utilizando la función "ranger" con los parámetros por defecto. Trabajamos con distinta cantidad de árboles ("num.Trees"), observaciones por nodo terminal ("min.node.size") y número de variables en que se puede dividir cada nodo ("mtry"). Inicialmente seleccionamos la combinación de las variables que arrojan el mejor error mediante una iteración. En base a ésta selección ampliamos los márgenes de dichos parámetros y repetimos el proceso donde finamente se determinó la siguiente combinación:

Num.Trees: 490
Mtry: 3
Min.node.size: 2

El resultado del error en Train arrojó un valor de 2.61, en Crossvalidation 6.86 y en Test 7.81.

b. Random Forest con análisis de componentes principales

En cuanto a este modelo, inicialmente se crearon dummies para las variables "League", "Division" y "New League". Ejecutamos el algoritmo y en base a los resultados observados en Summary tomamos como referencia para determinar el número de componentes, los

```
> summary(acp)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8    PC9    PC10
Standard deviation     2.7606 1.9582 1.4418 1.24186 1.00177 0.91067 0.81650 0.71990 0.5084 0.42283
Proportion of Variance 0.4011 0.2018 0.1094 0.08117 0.05282 0.04365 0.03509 0.02728 0.0136 0.00941
Cumulative Proportion  0.4011 0.6029 0.7124 0.79352 0.84634 0.88998 0.92507 0.95235 0.9659 0.97536
                          PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19
Standard deviation     0.36550 0.32674 0.27051 0.24746 0.22340 0.15375 0.11629 0.07232 0.03273
Proportion of Variance 0.00703 0.00562 0.00385 0.00322 0.00263 0.00124 0.00071 0.00028 0.00006
Cumulative Proportion  0.98239 0.98801 0.99186 0.99509 0.99771 0.99896 0.99967 0.99994 1.00000
```

parámetros de “Stándar deviation” y “Cumulative Proportion” como se muestran a continuación:
Es así como determinamos una cantidad de 7 componentes, los cuales logran explicar un 92.5% de la varianza lo cual consideramos que es aceptable. En consecuencia, obtenemos los siguientes valores en el modelo:

```
Call:
 ranger(formula = log(Salary) ~ ., data = datos, num.trees = correcto[1,
     1], mtry = correcto[1, 2], min.node.size = correcto[1, 3],      impo
rtance = "permutation", seed = 123451)

Type:                             Regression
Number of trees:                  800
Sample size:                      184
Number of independent variables:  7
Mtry:                             4
Target node size:                 8
Variable importance mode:         permutation
Splitrule:                        variance
OOB prediction error (MSE):       0.2797413
R squared (OOB):                  0.6407468
```

Nuevamente mediante una iteración, probamos distintas combinaciones de los parámetros que arrojaron el mejor error, tal cual como lo hicimos en el modelo anterior.

Num.Trees: 800
Mtry: 4
Min.node.size: 8

El resultado del error en Train arrojó un valor de 4.28, en Crossvalidation 8.92y en Test 8.57.

c. Ridge

En lo que respecta a este modelo, también convertimos las variables categóricas en dummies y aplicamos la función glmnet obteniendo la siguiente salida:

```
Call:  glmnet(x = as.matrix(datosNNa[train, -19]), y = log(datosNNa
[train,      19]), alpha = 0, lambda = cv_error$lambda.min, standar
dize = TRUE)

  Df  %Dev Lambda
1 19 58.07 0.1618
```

El resultado del error en Train arrojó un valor de 9.61, en Crossvalidation 10.36 y en Test 11.62.

d. Regresión Lineal

En este modelo inicialmente realizamos una regresión con la totalidad de las variables y estableciendo como variable dependiente a `Salary.`

Luego, aplicando un método de selección automática de variables, se descartan aquellas que empeoran el modelo, y almacenando únicamente las de mayor significancia.

Posteriormente, verificamos la existencia de multicolinealidad, para seleccionar las variables finales que pasarán a formar el modelo de regresión lineal.

```
Call:
lm(formula = log(Salary) ~ Hits + HmRun + Walks + Years + League +
    Division + PutOuts + Assists + Errors, data = datosNNa, subset = train)

Residuals:
     Min       1Q   Median       3Q      Max
-1.48601 -0.43169  0.06552  0.42167  2.34058

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.2741155  0.1514514  28.221  < 2e-16 ***
Hits         0.0070844  0.0014219   4.982 1.51e-06 ***
HmRun        0.0053625  0.0067045   0.800   0.4249
Walks        0.0053761  0.0025728   2.090   0.0381 *
Years        0.0930762  0.0098167   9.481  < 2e-16 ***
League      -0.1740410  0.0915569  -1.901   0.0590 .
Division     0.1258443  0.0900733   1.397   0.1642
PutOuts      0.0002888  0.0001683   1.716   0.0880 .
Assists      0.0005197  0.0005304   0.980   0.3285
Errors      -0.0198396  0.0104813  -1.893   0.0600 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5996 on 174 degrees of freedom
Multiple R-squared:  0.561,     Adjusted R-squared:  0.5383
F-statistic:  24.7 on 9 and 174 DF,  p-value: < 2.2e-16
```

En primer lugar, vemos en el F Estadístico la significancia del modelo, que en este caso es ampliamente significativo con un p valor de 2.2e-16, demostrando que se rechaza H0, por lo que al menos una de las variables es explicativa de la variable dependiente.

Los valores que nos indican la significancia de cada una en la tabla de coeficientes en la última columna de “`Pr(>|t|)`” donde se esperan valores menores a 0.10 en nuestro caso.

Debido a esto eliminamos las variables `Assists,` `Division` y `HmRun` con el objetivo de obtener un nivel de confianza del 90%.

Finalmente obtenemos el siguiente modelo:

```
Call:
lm(formula = log(Salary) ~ Hits + Walks + Years + League + PutOuts +
    Errors, data = datosNNa, subset = train)

Residuals:
     Min       1Q   Median       3Q      Max
-1.54060 -0.45602  0.08596  0.44878  2.31708

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.2754191  0.1471542  29.054   <2e-16 ***
Hits         0.0077317  0.0012571   6.151    5e-09 ***
Walks        0.0059976  0.0025397   2.362   0.0193 *
Years        0.0948273  0.0096867   9.789   <2e-16 ***
League      -0.1590368  0.0904229  -1.759   0.0803 .
PutOuts      0.0002879  0.0001648   1.747   0.0824 .
Errors      -0.0123868  0.0069276  -1.788   0.0755 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5996 on 177 degrees of freedom
Multiple R-squared:  0.5535,    Adjusted R-squared:  0.5384
F-statistic: 36.57 on 6 and 177 DF,  p-value: < 2.2e-16
```

El resultado del error en Train arrojó un valor de 9.92, en Crossvalidation no aplica y en Test 11.56.

Tabla comparativa

En base a la explicación de los modelos anteriormente mencionados, representamos en una tabla comparativa los errores en Train, Test y Crossvalidation

	Train	CV	Test
Random Forest	2.93	7.67	7.31
Random Forest con ACP	4.28	8.92	8.57
Ridge	9.61	10.36	11.62
Regresión Lineal	9.92	NA	11.56

Elección del modelo

En base a los resultados obtenidos en la tabla anterior, concluimos que el mejor modelo a utilizar para predecir los valores de Salary es el de **`Random Forest,`** ya que **muestra el menor error en Test**. Cabe destacar sin embargo, que si buscaramos que la diferencia entre los resultados prometidos y la predicción real fuera mínima, en ese caso optaríamos por el modelo de Regresión Lineal, ya que es el que presenta menor overfitting.

6. Anexo

Script:

```
################################################################################
############----------Universidad ORT Uruguay----------##########
############--------Obligatorio Analítica de Negocios 2---------##########
############----------Machado | Pioli | Silveyra----------##########
############----------Prof. Guillermo Magnou----------##########
################################################################################

#===============================================================================
#############
# Preámbulo #
#############

# Borramos datos de la memoria
rm(list = ls())

# Establecemos directorio de trabajo
#setwd("C:/Users/Cecilia Machado/Desktop/Obligatorio Analítica 2")
```

```
setwd("C:/Users/el_to/OneDrive/Escritorio/Drive/ORT/8vo Sem/Analitica de Negocios
2/Obligatorio/")

# Cargamos libreria a utilizar
library(ISLR)
library(cluster)
library(ranger)

# Cargamos los datos
datos = Hitters

## Visualizamos los datos
View(datos)

#
summary(datos)

# Fin del preambulo
#================================================================

#**********
# Parte 1
#**********

#********************
#Análisis descriptivo
#********************

summary(datos)

# Diagrama de caja de las principales variables
par(mfrow = c(3, 4))
boxplot(datos$Salary, col = "brown3", main = "Diagrama de caja de Salary", ylab="Salary")
boxplot(datos$CRuns, col = "brown3", main = "Diagrama de caja de CRuns", ylab="CRuns")
boxplot(datos$CAtBat, col = "brown3", main = "Diagrama de caja de CAtBat", ylab="CAtBat")
boxplot(datos$CHits, col = "brown3", main = "Diagrama de caja de CHits", ylab="CHits")
boxplot(datos$CRBI, col = "brown3", main = "Diagrama de caja de CRBI", ylab="CRBI")
boxplot(datos$CWalks, col = "brown3", main = "Diagrama de caja de CWalks", ylab="CWalks")
boxplot(datos$CHmRun, col = "brown3", main = "Diagrama de caja de CHmRun",
ylab="CHmRun")
boxplot(datos$Years, col = "brown3", main = "Diagrama de caja de Years", ylab="Years")
boxplot(datos$Hits, col = "brown3", main = "Diagrama de caja de Hits", ylab="Hits")
boxplot(datos$Runs, col = "brown3", main = "Diagrama de caja de Runs", ylab="Runs")
boxplot(datos$AtBat, col = "brown3", main = "Diagrama de caja de AtBat", ylab="AtBat")
```

```
#**********
# Parte 2
#**********

#********************
#Análisis de cluster
#********************

# Creamos copia de base de datos para convertir variables dicotómicas en dummies

datos_dummies <- datos

# Convertimos variables dicotómicas en dummies

datos_dummies$League <- ifelse(datos_dummies$League == 'A', 1, 0)
datos_dummies$Division <- ifelse(datos_dummies$Division == 'E', 1, 0)
datos_dummies$NewLeague <- ifelse(datos_dummies$NewLeague == 'A', 1, 0)

View(datos_dummies)

# Escalamos base de datos
datos_dummies_scaled <- scale(datos_dummies)

# Verificamos que los datos están escalados
colMeans(datos_dummies_scaled)
apply(datos_dummies_scaled, 2, var)

#Verificamos la posibilidad de imputar los NA para no perder observaciones

# Omitimos datos NA
datos_dummies_scaled_NO_NA <- na.omit(datos_dummies_scaled)

#Clusterizamos sin tomar la variable de interés (salary)
#clustereu = agnes (datos_dummies_scaled_NO_NA[,-19], metric = "euclidean")
#clustereu$ac

#!#Clusterizamos sin tomar la variable de interés (salary)
clustereu = agnes (datos_dummies_scaled[,-19], metric = "euclidean")
clustereu$ac

#Se visualiza que la observación correspondiente a "Peter Rose" es un outlier, por lo que
descartamos
#la métrica Euclidea y pasamos a Manhattan.
```

```
#Se realiza dendograma para visualizar y seleccionar cantidad de clusters.
dendo1 = pltree(clustereu, main = "Dendograma")

# Se visualiza que el dendograma inicial es poco claro, se hace dificil de interpretar
dendo1 = pltree(clustereu, main = "Dendograma", hang = -1)

#Encontramos un outlier por lo que debemos proceder con cambios en la metrica

# Se realiza clusterización con otras métricas para mejorar el ac utlizando "manhattan"
clusterman = agnes (datos_dummies_scaled[,-19], metric = "manhattan" )
clusterman$ac

# Se realiza clusterización con otros metodos para mejorar el ac utlizando "complete"
clusterman2 = agnes (datos_dummies_scaled[,-19], metric = "manhattan", method =
"complete")
clusterman2$ac

# Mejora la clusterización

# Se realiza clusterización con otros metodos para mejorar el ac utlizando "ward"
clusterman3 = agnes (datos_dummies_scaled[,-19], metric = "manhattan", method = "ward")
clusterman3$ac

# Se visualiza dendograma con la mejor custerización
dendo2 = pltree(clusterman3, main = "Dendograma", hang = -1)

# Comenzamos el proceso de poda del dendograma buscando clusters en los que:
  #ni todas sus observaciones sean NA
  #ni los clusters nos queden demasiado grandes, ya que sino estaremos imputando el mismo
  #valor para todos los NA lo cual no tendría mucho sentido
  #similar cantidad de NA por cluster

arbol_cortado1 = cutree(as.hclust(clusterman3), h = 50)
table(arbol_cortado1)

arbol_cortado2 = cutree(as.hclust(clusterman3), h = 40)
table(arbol_cortado2)

arbol_cortado3 = cutree(as.hclust(clusterman3), h = 35)
table(arbol_cortado3)

arbol_cortado4 = cutree(as.hclust(clusterman3), h = 30)
table(arbol_cortado4)

arbol_cortado5 = cutree(as.hclust(clusterman3), h = 25)
```

```
table(arbol_cortado5)


# Se imputa la culsterización a la base de datos
#datos_dummies = as.data.frame(datos_dummies)
datos_dummies$clusters1 = arbol_cortado1
datos_dummies$clusters2 = arbol_cortado2
datos_dummies$clusters3 = arbol_cortado3
datos_dummies$clusters4 = arbol_cortado4
datos_dummies$clusters5 = arbol_cortado5

View(datos_dummies)
summary(datos_dummies)

############################################################

# Una vez descartado el cálculo de los NA,
# procedemos a omitir la observaciones con datos faltantes

############################################################

#Razones por las que omitimos los datos faltantes:
 # 1) No nos fue posible inputar los NA
 # 2) La muestra es lo suficientemente grande para permitirnos perder el 20% de las obs
 # 3) Al inputar la variable de interés ("salary") estamos, mejorando la eficacia de los modelos
 # porque inpute la variable de interés.

#=================================================================
# Reiniciamos el ambiente

# Borramos datos de la memoria
rm(list = ls())

# Establecemos directorio de trabajo
#setwd("C:/Users/Cecilia Machado/Desktop/Obligatorio Analítica 2")
setwd("C:/Users/el_to/OneDrive/Escritorio/Drive/ORT/8vo Sem/Analitica de Negocios
2/Obligatorio/")

# Cargamos libreria a utilizar
library(ISLR)
library(cluster)
library(ranger)
# Cargamos los datos
datos = Hitters
```

```
# Visualizamos los datos
View(datos)
#====================================================================

#**********
# Parte 3
#**********

#****************************
#3.1) MODELO 1: RANDOM FOREST
#****************************

# Omitimos datos NA
datos_nNA = na.omit(datos)

# Separamos base en test y train
set.seed(123451)
train <- sample(nrow(datos_nNA), nrow(datos_nNA)*0.70)
test <- (-train)

# Modelo Random Forest
RF=ranger(log(Salary)~., data=datos_nNA[train,],seed=123451)
RF

# Predicción
pred_ranger = predictions(predict(RF,data=datos_nNA))
pred_ranger

# Errores relativo en train y test


round(100*sqrt(mean((log(datos_nNA$Salary[train])-
pred_ranger[train])^2))/mean(log(datos_nNA$Salary)),2) #train

round(100*sqrt(RF$prediction.error)/mean(log(datos_nNA$Salary)),2)  #verdadero train

round(100*sqrt(mean((log(datos_nNA$Salary[test])-
pred_ranger[test])^2))/mean(log(datos_nNA$Salary)),2) #test


#      Train  (CV)  / Test /
# RFi    3,28 (7.73) / 7,21 /

# Optimizacion de parametros de nuestro Random Forest inicial
```

```
opt = expand.grid(
 'num_trees' = c(300, 425, 490, 550, 700),
 'mtry'      = c(2, 3, 4, 7),  #Verificar con otros valores (mayores y menores al inicial)
 'min.node.size' =c(2,4,6))

#opt

oob_error = rep(NA, nrow(opt))

min_i = 0
min_error = 99999
correcto = 0
for(i in 1:nrow(opt)){

 modelo <- ranger(
  formula   = log(Salary) ~ .,
  data      = datos_nNA[train,],
  num.trees = opt$num_trees[i],
  mtry      = opt$mtry[i],
  min.node.size = opt$min.node.size[i],
  seed      = 123451)

 oob_error[i] <- round(100*sqrt(modelo$prediction.error)/mean(log(datos_nNA$Salary)),2)

 if(oob_error[i] < min_error){
  min_error = oob_error[i]
  min_i = i
  correcto = cbind(modelo$num.trees, modelo$mtry, modelo$min.node.size)
 }
}

print(correcto)


# Modelo final con los datos que mejoran el error

RF <- ranger(
 formula   = log(Salary) ~ .,
 data      = datos_nNA[train,],
 num.trees = correcto[1,1],
 mtry      = correcto[1,2],
 min.node.size = correcto[1,3],
 importance='permutation',
 seed      = 123451)
```

```
RF



oob_errorFinal <- round(100*sqrt(RF$prediction.error)/mean(log(datos_nNA$Salary)),2)
oob_errorFinal

# predicion
pred_ranger=predictions(predict(RF,data=datos_nNA))

# errores relativo en train y test

round(100*sqrt(mean((log(datos_nNA$Salary[train])-
pred_ranger[train])^2))/mean(log(datos_nNA$Salary)),2) #train

round(100*sqrt(RF$prediction.error)/mean(log(datos_nNA$Salary)),2)  #verdadero train

round(100*sqrt(mean((log(datos_nNA$Salary[test])-
pred_ranger[test])^2))/mean(log(datos_nNA$Salary)),2) #test


#      Train  (CV)  / Test /
# RFi    3,28 (7.73) / 7,21 /
# RFf    2.93 (7.67) / 7.31 /

# Importancia de las variables
sort(RF$variable.importance)     #importancia de las variables ordenadas

# Gráfico de las variables de mayor importancia
library(vip)
vip(RF, horizontal = FALSE)


#Reiniciamos ambiente
#=================================================================
# Reiniciamos el ambiente

rm(list = ls())

# Establecemos directorio de trabajo
#setwd("C:/Users/Cecilia Machado/Desktop/Obligatorio Analítica 2")
setwd("C:/Users/el_to/OneDrive/Escritorio/Drive/ORT/8vo Sem/Analitica de Negocios
2/Obligatorio/")

# Cargamos libreria a utilizar
```

```
library(ISLR)
library(cluster)
library(ranger)
library(glmnet)

# Cargamos los datos
datos = Hitters

# Visualizamos los datos
View(datos)
#====================================================================

#**********************************
#3.2) MODELO 2: RANDOM FOREST con ACP
#**********************************

# Omitimos datos Na

datosNNa=na.omit(datos)

# Crear las dummys para ACP

datosNNa$League <- ifelse(datosNNa$League == 'A', 1, 0)
datosNNa$Division <- ifelse(datosNNa$Division == 'E', 1, 0)
datosNNa$NewLeague <- ifelse(datosNNa$NewLeague == 'A', 1, 0)


# Separamos los datos en train y test
set.seed(123451)
train <- sample(nrow(datosNNa), nrow(datosNNa)*0.7)
test <- (-train)

# realizamos el analisis de componentes principales
acp <- prcomp(datosNNa[train,-19], scale = TRUE)

summary(acp)

p=7

datos=as.data.frame(cbind(acp$x[,1:p],datosNNa[train,19]))
colnames(datos)=c(paste('PC',1:p,sep = ''),'Salary')

# Modelo
RF=ranger(log(Salary)~., data=datos,seed=123451)
RF
```

```
# Optimizacion de parametros

opt = expand.grid(
 'num_trees' = c(300, 400, 500,550, 600,625, 650,700, 800, 850),
 'mtry'      = c(1,2, 3, 4),
 'min.node.size' =c(3,5,8))

#opt

oob_error = rep(NA, nrow(opt))
min_i = 0
min_error = 99999
correcto = 0
for(i in 1:nrow(opt)){

 modelo <- ranger(
  formula   = log(Salary) ~ .,
  data      = datos,
  num.trees = opt$num_trees[i],
  mtry      = opt$mtry[i],
  min.node.size = opt$min.node.size[i],
  seed      = 123451)

 oob_error[i] <- round(100*sqrt(modelo$prediction.error)/mean(log(datosNNa$Salary)),2)
 if(oob_error[i] < min_error){
  min_error = oob_error[i]
  min_i = i
  correcto = cbind(modelo$num.trees, modelo$mtry, modelo$min.node.size)
 }

}
print(correcto)

# Modelo final

RF <- ranger(
 formula   = log(Salary) ~ .,
 data      = datos,
 num.trees = correcto[1,1],
 mtry      = correcto[1,2],
 min.node.size = correcto[1,3],
 importance='permutation',
 seed      = 123451)
```

```
RF

# escalando los resultados
base_scale=scale(datosNNa[,-19],center=acp$center,scale=acp$scale)

# creando los componentes para todas las observaciones
base_ACP=as.data.frame(base_scale%*%acp$rotation[,1:p])
colnames(base_ACP)=paste('PC',1:p,sep = '')

# predicion
pred_rangerACP=predictions(predict(RF,data=base_ACP))

# errores relativo en train y test

round(100*sqrt(mean((log(datosNNa$Salary[train])-
pred_rangerACP[train])^2))/mean(log(datosNNa$Salary)),2) #train

round(100*sqrt(RF$prediction.error)/mean(log(datosNNa$Salary)),2)  #verdadero train

round(100*sqrt(mean((log(datosNNa$Salary[test])-
pred_rangerACP[test])^2))/mean(log(datosNNa$Salary)),2) #test

# Importancia de las variables
sort(RF$variable.importance)     #importancia de las variables ordenadas

barplot(RF$variable.importance)


#         Train  (CV)  / Test /
# RFi      3,28 (7.73) / 7,21 /
# RFf      2.93 (7.67) / 7.31 /
# RFacp(5)  3.57 (8.32) / 7.63 /
# RFacp(6)  3.67 (8.67) / 8.17 /
# RFacp(7)  4.28 (8.92) / 8.57 /


#Reiniciamos el ambiente
#======================================================================

# Borramos datos de la memoria
rm(list = ls())

# Establecemos directorio de trabajo
#setwd("C:/Users/Cecilia Machado/Desktop/Obligatorio Analítica 2")
```

```
setwd("C:/Users/el_to/OneDrive/Escritorio/Drive/ORT/8vo Sem/Analitica de Negocios
2/Obligatorio/")

# Cargamos libreria a utilizar
library(ISLR)
library(cluster)
library(ranger)
library(glmnet)

# Cargamos los datos
datos = Hitters

# Visualizamos los datos
View(datos)
#====================================================================

#**********************************
#3.3) MODELO 3: RIDGE
#**********************************

# Omitimos datos Na

datosNNa=na.omit(datos)

# Crear las dummys

datosNNa$League <- ifelse(datosNNa$League == 'A', 1, 0)
datosNNa$Division <- ifelse(datosNNa$Division == 'E', 1, 0)
datosNNa$NewLeague <- ifelse(datosNNa$NewLeague == 'A', 1, 0)

# Separamos los datos en train y test
set.seed(123451)
train <- sample(nrow(datosNNa), nrow(datosNNa)*0.7)
test <- (-train)

# Modelo rigde

reg_ridge <- glmnet(
  x         = datosNNa[train,-19],
  y         = log(datosNNa[train,19]),
  alpha     = 0,
  nlambda    = 100,
  standardize = TRUE
)
```

```
# cross -validation
set.seed(123451)
cv_error <- cv.glmnet(
  x     = as.matrix(datosNNa[train,-19]),
  y     = log(datosNNa[train,19]),
  alpha  = 0,
  nfolds = 10,
  type.measure = "mse",
  standardize  = TRUE
)
cv_error

reg_ridge.min <- glmnet(
  x         = as.matrix(datosNNa[train,-19]),
  y         = log(datosNNa[train,19]),
  alpha      = 0,
  lambda     = cv_error$lambda.min,
  standardize = TRUE
)
# beta0
reg_ridge.min$a0

# distintos betas
reg_ridge.min$beta

# predicciones

pred_ridge.min <- predict(reg_ridge.min,as.matrix(datosNNa[,-19]))

# errores relativo en train y test

round(100*sqrt(mean((log(datosNNa$Salary[train])-
pred_ridge.min[train])^2))/mean(log(datosNNa$Salary)),2) #train

round(100*sqrt(cv_error$cvm[cv_error$index[1]])/mean(log(datosNNa$Salary)),2)
#verdadero train

round(100*sqrt(mean((log(datosNNa$Salary[test])-
pred_ridge.min[test])^2))/mean(log(datosNNa$Salary)),2) #test



#         Train  (CV)  / Test /
# RFi      3,28 (7.73) / 7,21 /
# RFf      2.93 (7.67) / 7.31 /
```

```
# RFacp(5)  3.57 (8.32) / 7.63 /
# RFacp(6)  3.67 (8.67) / 8.17 /
# RFacp(7)  4.28 (8.92) / 8.57 /

# Reiniciamos el ambiente
#=====================================================================
# Borramos datos de la memoria
rm(list = ls())

# Establecemos directorio de trabajo
#setwd("C:/Users/Cecilia Machado/Desktop/Obligatorio Analítica 2")
setwd("C:/Users/el_to/OneDrive/Escritorio/Drive/ORT/8vo Sem/Analitica de Negocios
2/Obligatorio/")

# Cargamos libreria a utilizar
library(ISLR)
library(cluster)
library(ranger)
library(glmnet)
library(car)

# Cargamos los datos
datos = Hitters

# Visualizamos los datos
View(datos)
#=====================================================================

#*****************************
#3.4) MODELO 4: REGRESIÓN LINEAL
#*****************************

# Omitimos datos Na

datosNNa=na.omit(datos)

# Crear las dummys

datosNNa$League <- ifelse(datosNNa$League == 'A', 1, 0)
datosNNa$Division <- ifelse(datosNNa$Division == 'E', 1, 0)
datosNNa$NewLeague <- ifelse(datosNNa$NewLeague == 'A', 1, 0)

# Separamos los datos en train y test
set.seed(123451)
train <- sample(nrow(datosNNa), nrow(datosNNa)*0.7)
```

```
test <- (-train)

# Regresión con todos los parámetros
reg <- lm(log(Salary)~., data= datosNNa, subset=train)
summary(reg)

# Algoritmo de selección de variables
reg_step=step(reg, direction = "both", trace = 1)

# Control de multicolinealidad

vif(reg_step)

# Regresión lineal con selección de variables independientes

reg2=lm(formula = log(Salary) ~  Hits + HmRun + Walks + Years +
      League + Division + PutOuts + Assists +
      Errors, data = datosNNa, subset = train)
vif(reg2)
summary(reg2)

# Regresión lineal FINAL con selección de variables independientes

reg3=lm(formula = log(Salary) ~  Hits + Walks + Years +
      League + PutOuts + Errors, data = datosNNa, subset = train)
vif(reg3)
summary(reg3)

# predicciones

pred_lm <- predict(reg3,datosNNa[,-19])

# errores relativo en train y test

RL_error_train = round(100*sqrt(mean((log(datosNNa$Salary[train])-
pred_lm[train])^2))/mean(log(datosNNa$Salary)),2) #train

RL_error_test =round(100*sqrt(mean((log(datosNNa$Salary[test])-
pred_lm[test])^2))/mean(log(datosNNa$Salary)),2) #test

#************************************
# Parte 4: Tabla comparativa de errores
#************************************

#Errores de cada modelo
```

```
RF_error = c(2.93, 7.67, 7.31)
RFacp_error = c(4.28, 8.92, 8.57)
Ridge_error = c(9.61, 10.36, 11.62)
RegLin_error = c(9.92, NA, 11.56)

# Combinación de errores
tabla_errores = cbind(RF_error,RFacp_error, Ridge_error, RegLin_error)

# Creación de tabla con todos los modelos
tabla_final <- matrix(tabla_errores, ncol=3, byrow=TRUE)
colnames(tabla_final) <- c('Train','CV','Test')
rownames(tabla_final) <- c('Random Forest','Random Forest con ACP','Ridge', 'Regresión
Lineal')
tabla_final

# Creación de tabla comparando Random Fores y Random Forest ACP
tabla_errores2 = cbind(RF_error, RFacp_error)
tabla_final <- matrix(tabla_errores2, ncol=3, byrow=TRUE)
colnames(tabla_final) <- c('Train','CV','Test')
rownames(tabla_final) <- c('Random Forest','Random Forest con ACP')
tabla_final

# ELEGIMOS EL MODELO DE RANDOM FOREST

####################
# FIN DEL ANALISIS#
####################
```