Práctica 1 - Técnicas de Machine Learning

Roberto Bonilla Ibarra

22 marzo, 2023

UNIVERSIDAD
COMPLUTENSE
MADRID

Índice

1 Redes neuronales para regresión

En este trabajo desarrollaremos un trabajo de regresión para predecir el **costo del seguro de gastos médicos** de acuerdo a las siguientes variables.

1. Edad
2. Peso
3. IMC
4. Número de dependientes
5. Fumador
6. Región

Este dataset contiene *1338* y se puede encontrar en el siguiente link: https://www.kaggle.com/datasets/mirichoi0218/insurance?datasetId=13720&searchQuery=data+cleaning

2 Solución con SAS Enterprise Miner

En este capítulo del trabajo detallaremos el trabajo implementado con el software SAS Enterprise Miner.

2.1 Lectura de datos

El primer paso es leer los datos por medio de la acción de *importar archivo* en el menú *muestreo*.

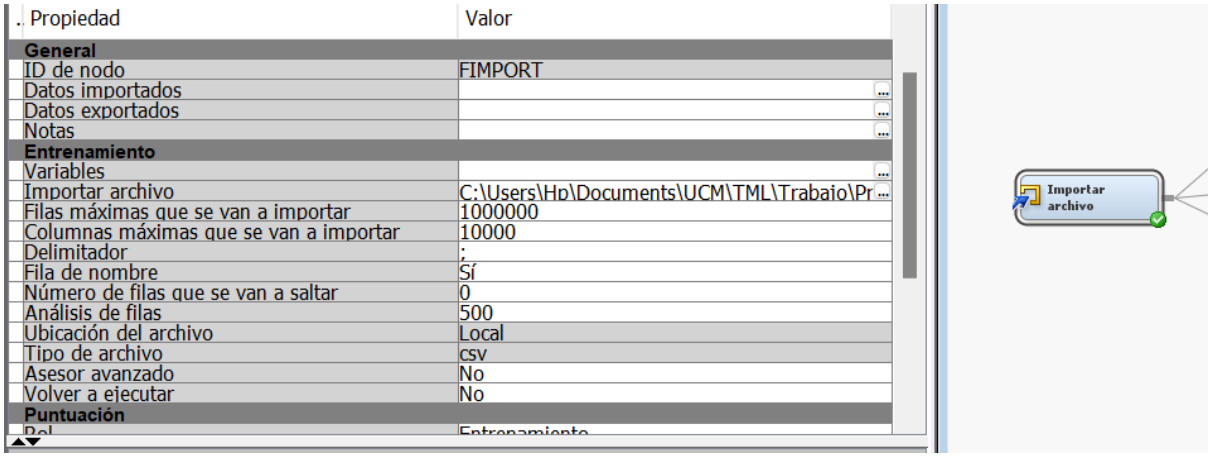


Figura 1: Lectura de datos

2.2 Análisis exploratorio

Una vez importado el archivo, procederemos a crear diferentes análisis de la distribución de los datos y su comportamiento.

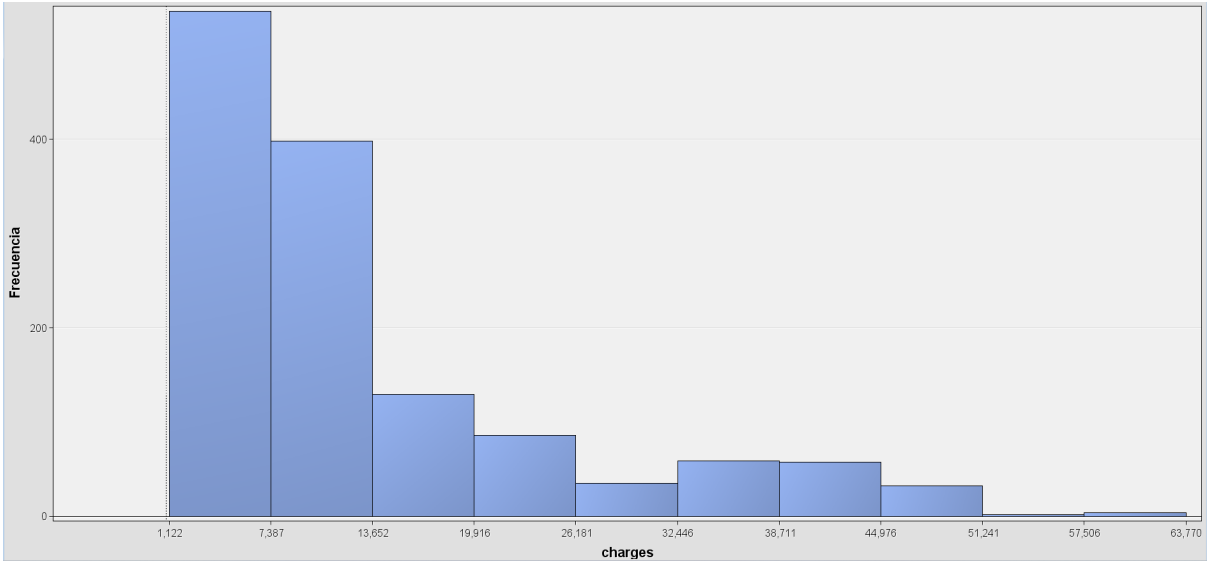


Figura 2: Distribución variable objetivo

Basándonos en la distribución de la variable objetivo podemos detectar la presencia de datos atípicos.

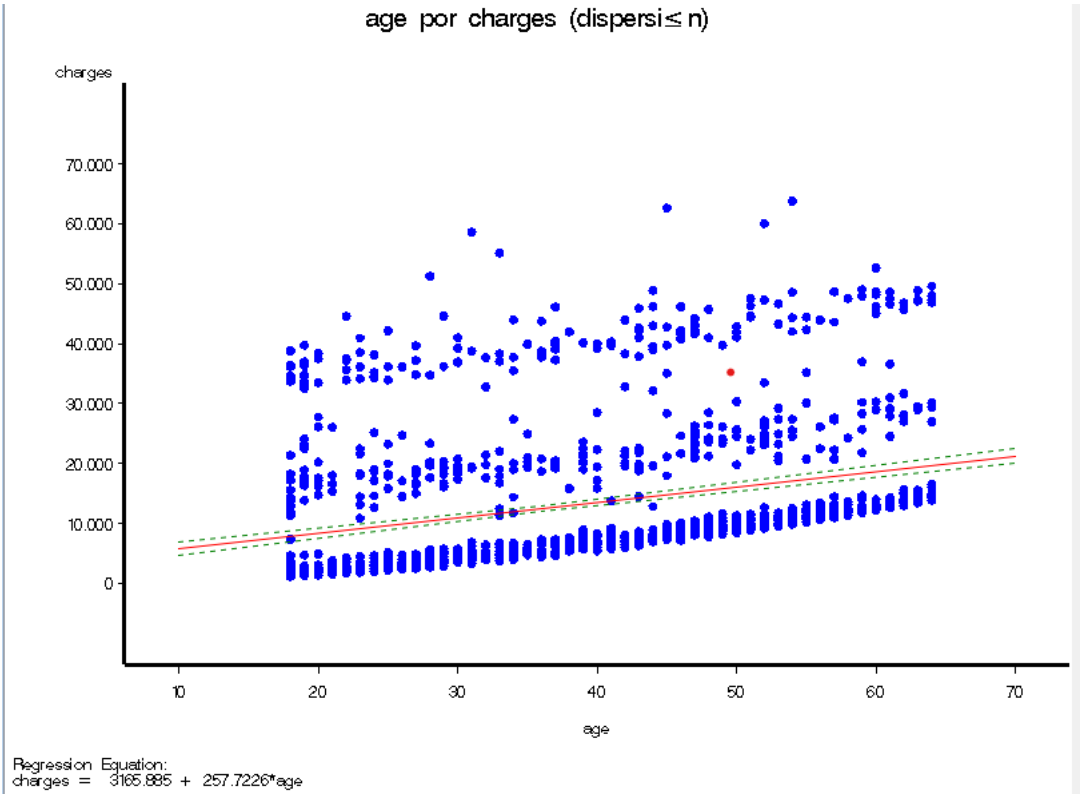


Figura 3: Gráfico de dispersión y regresión lineal en variable edad

En la figura anterior podemos concluir que existe una relación entre la edad del asegurado y el costo de su seguro médico.

2.3 Selección de variables

El proceso de selección de variables en Sas Miner utiliza una regresión de mínimos cuadrados *forward stepwise* que maximiza el valor de R-cuadrado del modelo, este proceso se puede explicar en 3 sencillos

pasos:

Paso 1: Se ajusta un modelo de regresión lineal utilizando cada variable de entrada individualmente como predictora de la variable objetivo binaria. Se calcula el valor de R-cuadrado correspondiente a cada modelo y se selecciona la variable de entrada que produce el mayor valor de R-cuadrado.

Paso 2: Se ajustan modelos de regresión lineal utilizando cada variable de entrada individualmente, junto con la variable de entrada seleccionada en el paso anterior, como predictores de la variable objetivo binaria. Se calcula el valor de R-cuadrado correspondiente a cada modelo y se selecciona la variable de entrada que produce el mayor incremento en el valor de R-cuadrado.

Paso 3: Se repite el paso 2 hasta que no se observen mejoras significativas *(threshold)* en el valor de R-cuadrado.

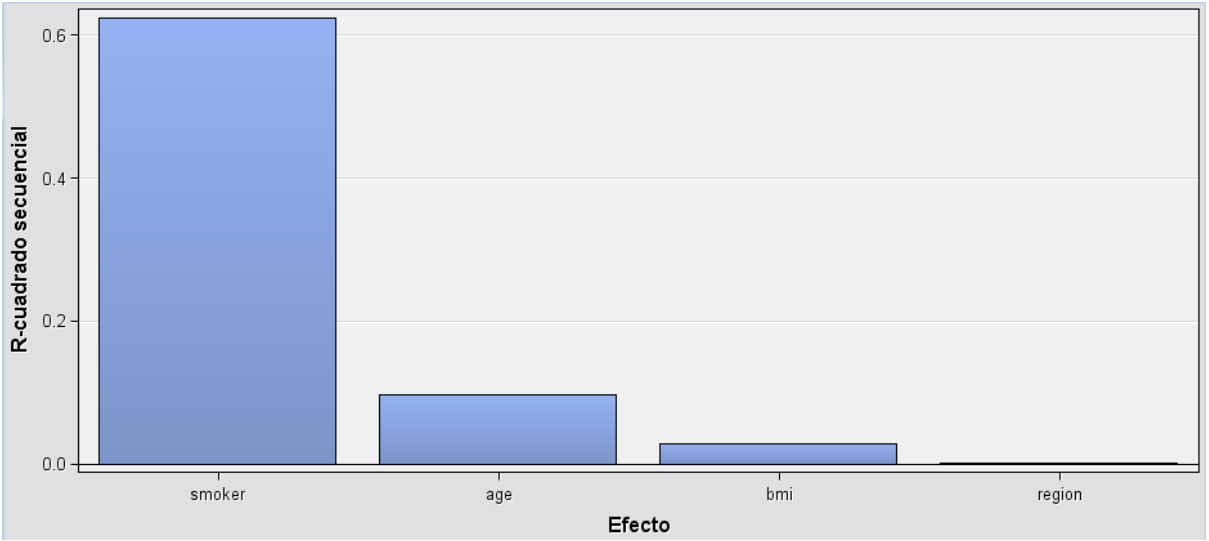


Figura 4: Selección de variables por r cuadrado secuencial.

En la Figura anterior se puede visualizar las 4 variables seleccionadas por el modelo, siendo *smoker (fumador)* la variable más determinante; es importante añadir que la variable región se descompone en 3 variables más siendo un total de 6 variables.

2.4 Túneo

A continuación se especificará el detalle en el túneo de parámetros.

2.4.1 Configuración de nodos.

En la selección final de variables contamos con 6 variables independientes, 1338 obs. con 30 obs. por parámetro, usando la siguiente fórmula:

$Nmero\ de\ parmetros = h(k+1) + h + 1$

donde: * h es el número de nodos ocultos * k es el número de nodos input.

Una red con 6 variables y 5 nodos necesitaría al menos 1,230 obs. y con 6 nodos 1,470 registros.

Al realizar pruebas en el SAS Enterprise Miner, podemos ver que los resultados con 6 nodos son mejores en sus resultados de VMSE (Validation Mean Squared Error) por aproximadamente 25 dólares en comparación de los modelos de 5 nodos, por lo tanto, procederemos con esa selección.

2.4.2 Configuración de número de iteraciones

Uno de los parámetros más importantes a configurar en una red neuronal es el número de iteraciones, el cual nos permite poder cuidar que la red sobreajuste.

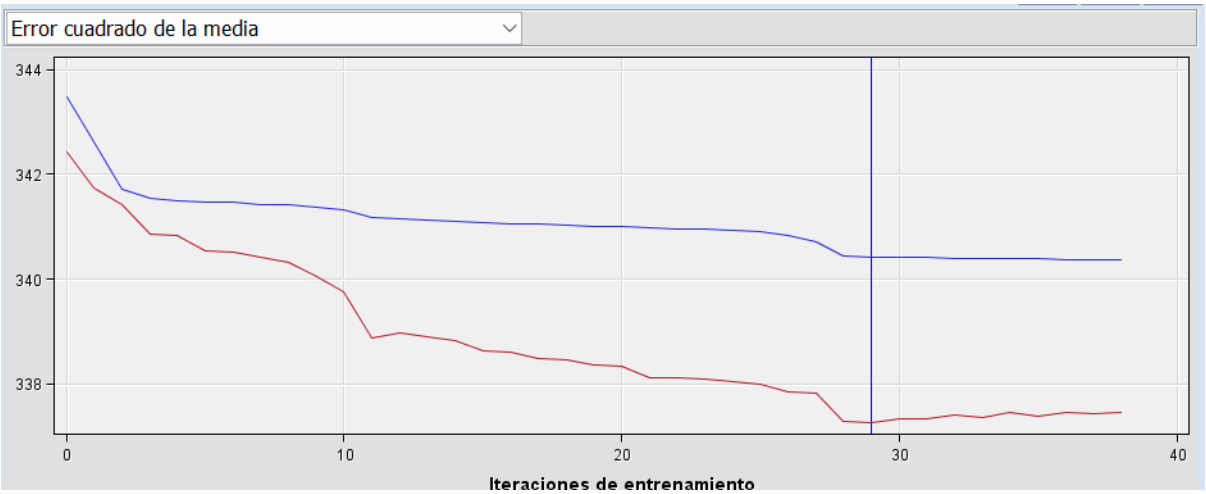


Figura 5: Iteraciones adecuadas en red neuronal

En la imagen anterior podemos observar que el número correcto de iteraciones es 30, ya que a partir de este número el modelo solo comienza a sobreajustar y la disminución del error es nulo.

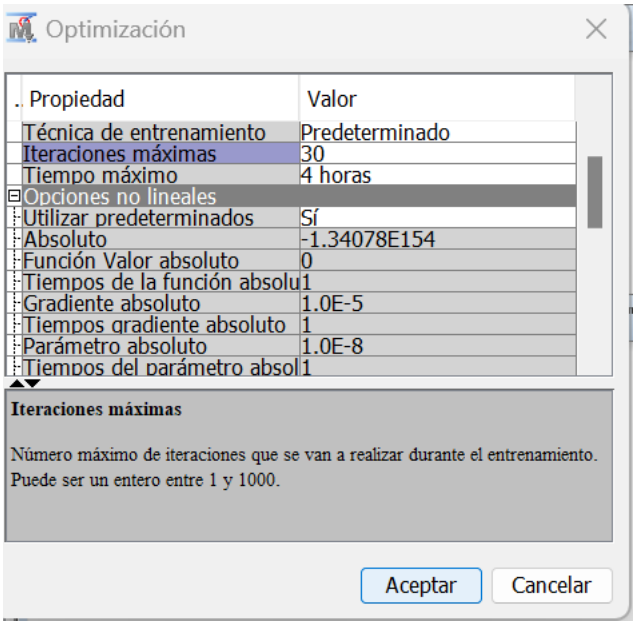


Figura 6: Cambio en el número de iteraciones en Sas

2.4.3 Configuración de semillas de inicialización

El uso de diferentes semillas de inicialización en una red neuronal puede ayudar a explorar diferentes regiones del espacio de pesos y a encontrar un mínimo local más óptimo.

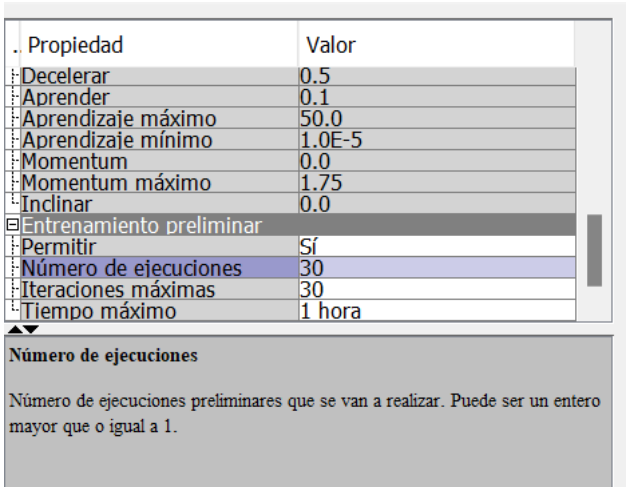


Figura 7: Cambio en el número de semillas de inicialización en Sas

Inicialmente en Sas Enterprise Miner el número predeterminado es 5 diferentes semillas, sin embargo, para este ejercicio usaremos 30 semillas, intentando en todo momento encontrar el mínimo local que reduzca en mayor proporción el error.

2.5 Flujo completo de solución

A continuación se presentará el flujo completo de la solución al problema en la predicción del costo del seguro médico de un individuo basado en sus características.

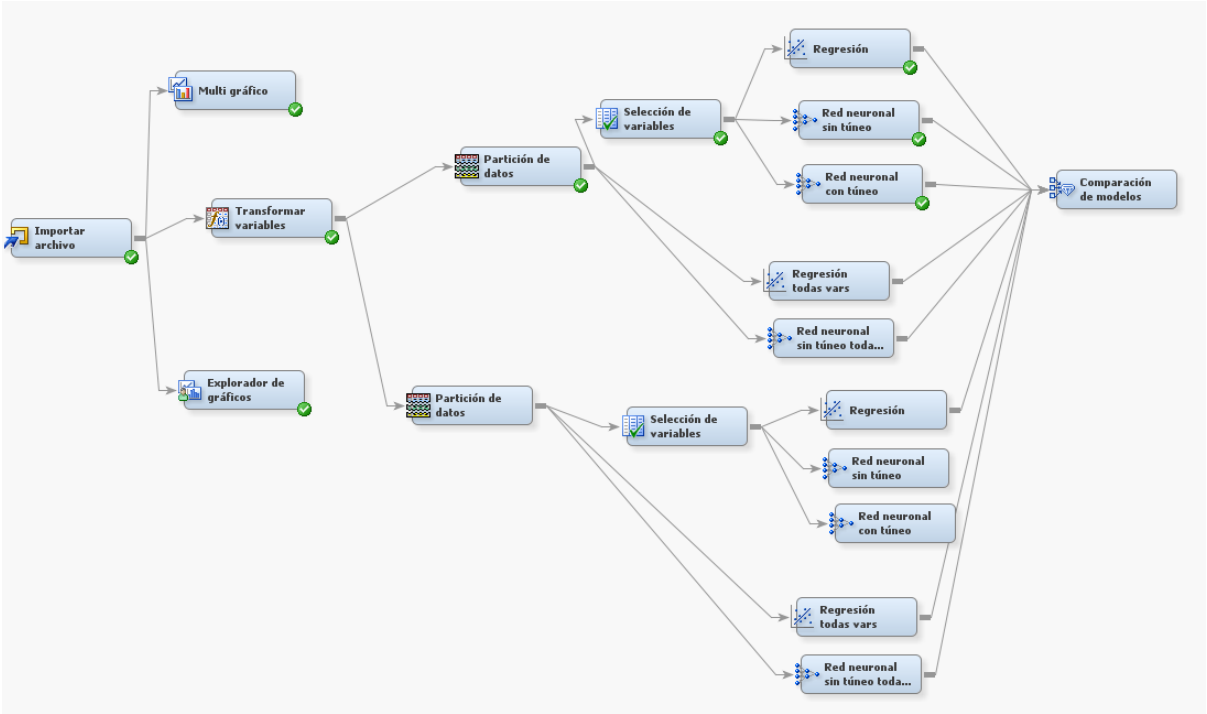


Figura 8: Flujo completo de solución

2.5.1 Transformación variable dependiente

Se ha transformado la variable dependiente usando su raíz cuadrada, intentando disminuir la influencia de los datos atípicos en la distribución de la variable dependiente.

2.5.2 Configuración de semilla aleatoria.

Se ha cambiado la semilla en la separación del set de datos y en la semilla inicial del algoritmo de redes neuronales, generando así diferentes resultados de algoritmos con los mismos parámetros lo cuál nos puede ayudar a elegir el modelo óptimo.

2.6 Resultados

La siguiente tabla representa los resultados obtenidos:

	Red neuronal con tuneo	Red neuronal sin tuneo	Red neuronal todas vars.	Regresión todas vars.	Regresión con selección vars.
Media MSE	352.07	345.62	353.74	539.58	456.33
Desv. Estándar MSE	2.65	12.90	14.12	13.70	1.44

Se ha seleccionado la medición del error por MSE, ya que la variable dependiente se encuentra transformada con su raíz cuadrada, haciendo que esta medida sea el fiel reflejo a los dólares de error en los distintos modelos.

Podemos observar que la red con tuneo cuenta con la mejor varianza entre las redes neuronales, sin embargo, el sesgo no es el mejor entre las redes neuronales, esto es debido a que en SAS Enterprise Miner no cuenta con un método repetitivo y con validación cruzada para poder seleccionar los mejores parámetros de la red neuronal.

3 Solución con R

A continuación se especificará la solución hecha con el lenguaje de programación R.

3.1 Paquetes necesarios

Necesitaremos los siguientes paquetes

```
# Borramos
rm(list = ls())

# Paquetes
shhh <- suppressPackageStartupMessages
shhh(library(skimr)) # resumen numérico
shhh(library(dplyr)) # depuración datos
shhh(library(tidyverse)) # depuración datos
shhh(library(ggthemes)) # tema para graficar
shhh(library(corrplot)) # tema para graficar matriz de correlaciones
shhh(library(fastDummies)) # Creación de Dummies
shhh(library(corrr)) # Crear correlaciones
shhh(library(Hmisc)) # Creación de histogramas
shhh(library(parallel)) # Librerías de Cómputo en Paralelo
shhh(library(doParallel)) # Librerías de Cómputo en Paralelo
```

3.2 Análisis exploratorio de datos.

Primero comenzaremos explorando los estadísticos básicos del set de datos y su distribución:

```
insurance<- read_csv("C:\\Users\\Hp\\Documents\\UCM\\TML\\Trabajo\\Práctica 1\\insurance.csv",show_c
insurance
```

```
## # A tibble: 1,338 x 7
##      age sex      bmi children smoker region    charges
##    <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl>
##  1    19 female  27.9        0 yes    southwest  16885.
##  2    18 male    33.8        1 no     southeast   1726.
##  3    28 male    33          3 no     southeast   4449.
##  4    33 male    22.7        0 no     northwest  21984.
##  5    32 male    28.9        0 no     northwest   3867.
##  6    31 female  25.7        0 no     southeast   3757.
##  7    46 female  33.4        1 no     southeast   8241.
##  8    37 female  27.7        3 no     northwest   7282.
##  9    37 male    29.8        2 no     northeast   6406.
## 10    60 female  25.8        0 no     northwest  28923.
## # ... with 1,328 more rows
```

Usando la función `skim()`

```
insurance |> skim()
```

Cuadro 2: Data summary

Name	insurance
Number of rows	1338
Number of columns	7
Column type frequency:	
character	3
numeric	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
sex	0	1	4	6	0	2	0
smoker	0	1	2	3	0	2	0
region	0	1	9	9	0	4	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	39.21	14.05	18.00	27.00	39.00	51.00	64.00	
bmi	0	1	30.66	6.10	15.96	26.30	30.40	34.69	53.13	
children	0	1	1.09	1.21	0.00	0.00	1.00	2.00	5.00	
charges	0	1	13270.42	12110.01	1121.87	4740.29	9382.03	16639.91	63770.43	

Podemos ver que en nuestro set de datos cibtanis con una variable objetivo denominado *charges*, tres variables numéricas y 3 variables categóricas.

Ahora veamos la distribución de las variables categóricas:

```
# Revisando variable sex
insurance |>   count(sex, sort = TRUE) |>
  mutate(porc = 100*n/sum(n), cumsum(porc))
```

```
## # A tibble: 2 x 4
##   sex        n  porc `cumsum(porc)`
##   <chr>  <int> <dbl>          <dbl>
## 1 male     676  50.5           50.5
## 2 female   662  49.5          100
```

Podemos ver que la variable sexo se encuentra balanceada entre femenino y masculino.

```
# Revisando variable smoker
insurance |>   count(smoker, sort = TRUE) |>
  mutate(porc = 100*n/sum(n), cumsum(porc))
```

```
## # A tibble: 2 x 4
##   smoker     n  porc `cumsum(porc)`
##   <chr>  <int> <dbl>          <dbl>
## 1 no      1064  79.5           79.5
## 2 yes      274  20.5          100
```

Podemos ver que la mayor parte de las personas aseguradas no son fumadores.

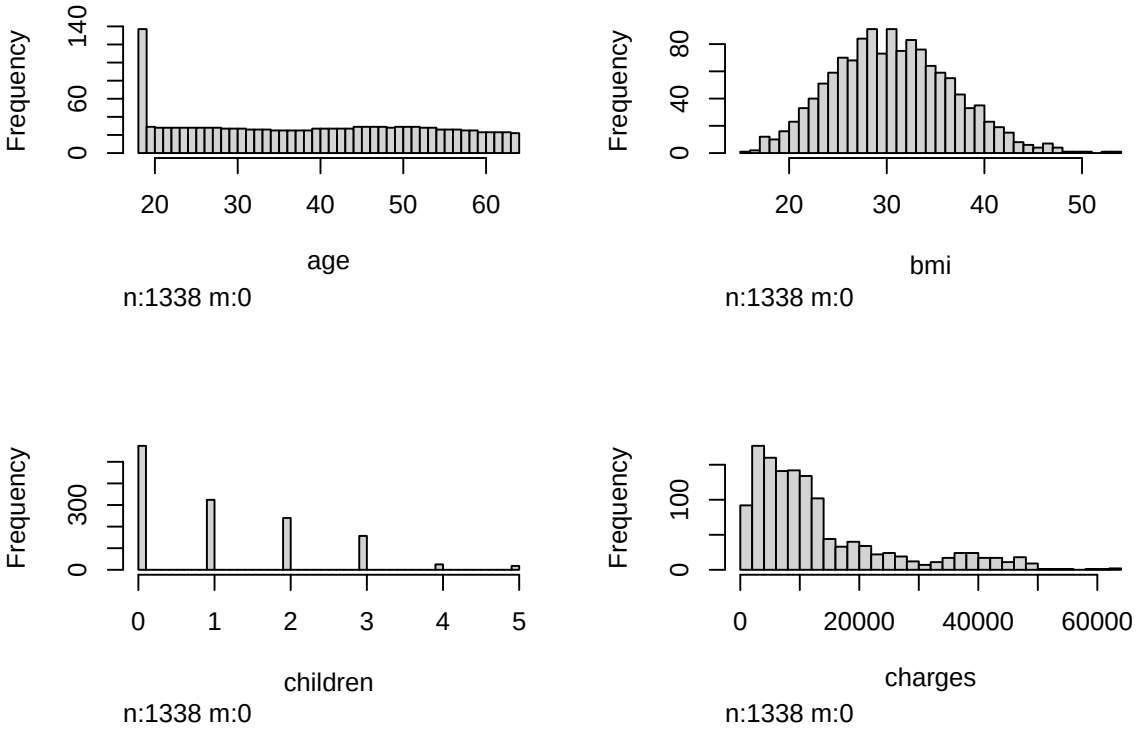
```
# Revisando variable region
insurance |>   count(region, sort = TRUE) |>
  mutate(porc = 100*n/sum(n), cumsum(porc))
```

```
## # A tibble: 4 x 4
##   region        n  porc `cumsum(porc)`
##   <chr>     <int> <dbl>          <dbl>
## 1 southeast   364  27.2           27.2
## 2 northwest   325  24.3           51.5
## 3 southwest   325  24.3           75.8
## 4 northeast   324  24.2          100
```

Podemos ver que la variable región prácticamente se encuentra balanceada, sin necesidad de alguna agrupación en sus clases.

Ahora veamos la distribución de las variables numéricas:

```
hist.data.frame(insurance %>% dplyr::select(where(is.numeric)))
```



Podemos ver que en la variable objetivo existe un sesgo hacia la derecha por la presencia de datos atípicos, las otras variables parecen tener un comportamiento normal.

También procederemos a transformar la edad a *buckets* de 10 años.

```
insurance <- insurance %>%
  mutate(age_bucket = cut(age, breaks = seq(15, 65, by = 10),
      labels = gsub(" ", "", paste(seq(15, 55, by = 10),
                "_", seq(25, 65, by = 10))), include.lowest = TRUE))
```

Ahora analizemos su distribución:

```
# Revisando variable region
insurance |>   count(age_bucket, sort = TRUE) |>
  mutate(porc = 100*n/sum(n), cumsum(porc))
```

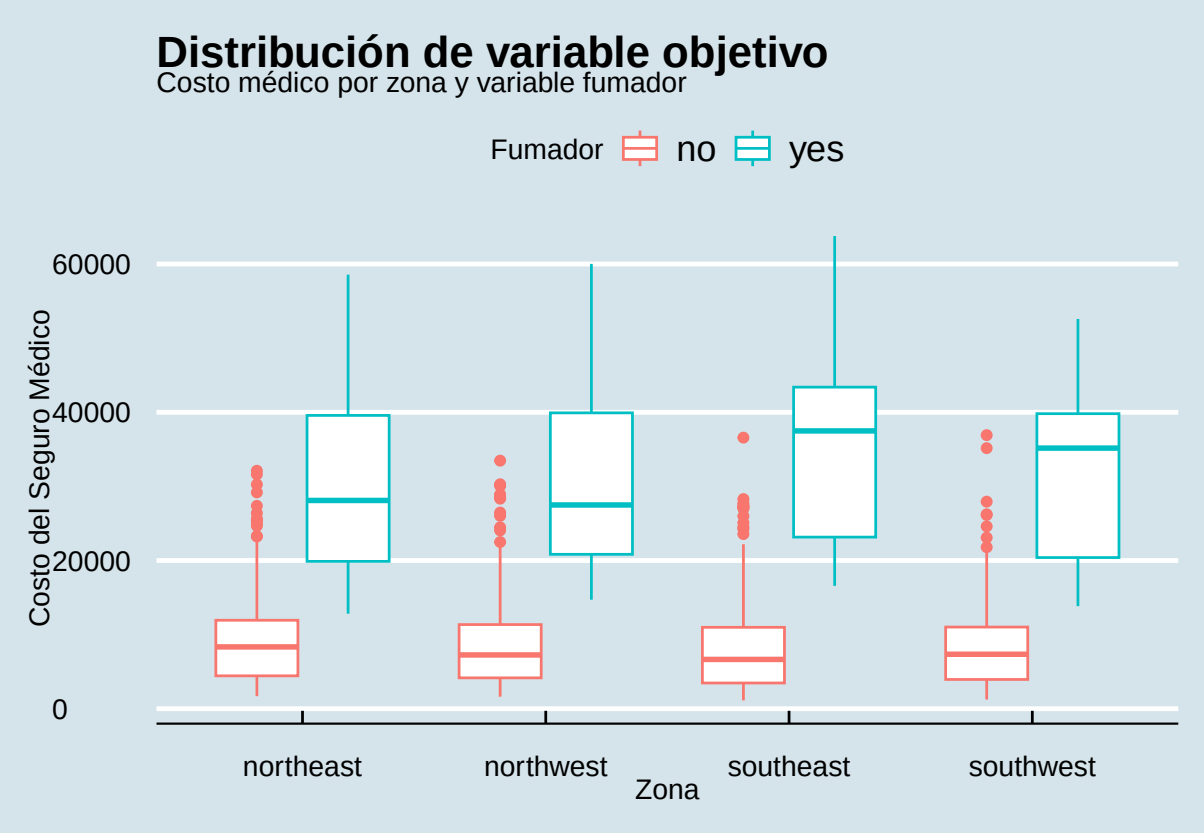
```
## # A tibble: 5 x 4
##   age_bucket     n  porc `cumsum(porc)`
##   <fct>      <int> <dbl>          <dbl>
## 1 15_25        306  22.9           22.9
## 2 45_55        284  21.2           44.1
## 3 25_35        268  20.0           64.1
## 4 35_45        264  19.7           83.9
## 5 55_65        216  16.1          100
```

Podemos ver que la muestra cuenta con edades bastante balanceadas, siendo el menor grupo el de la gente mayor.

3.2.1 Visualización de datos

Procederemos a analizar el factor de los fumadores y las regiones donde viven en el costo del seguro de gastos médicos:

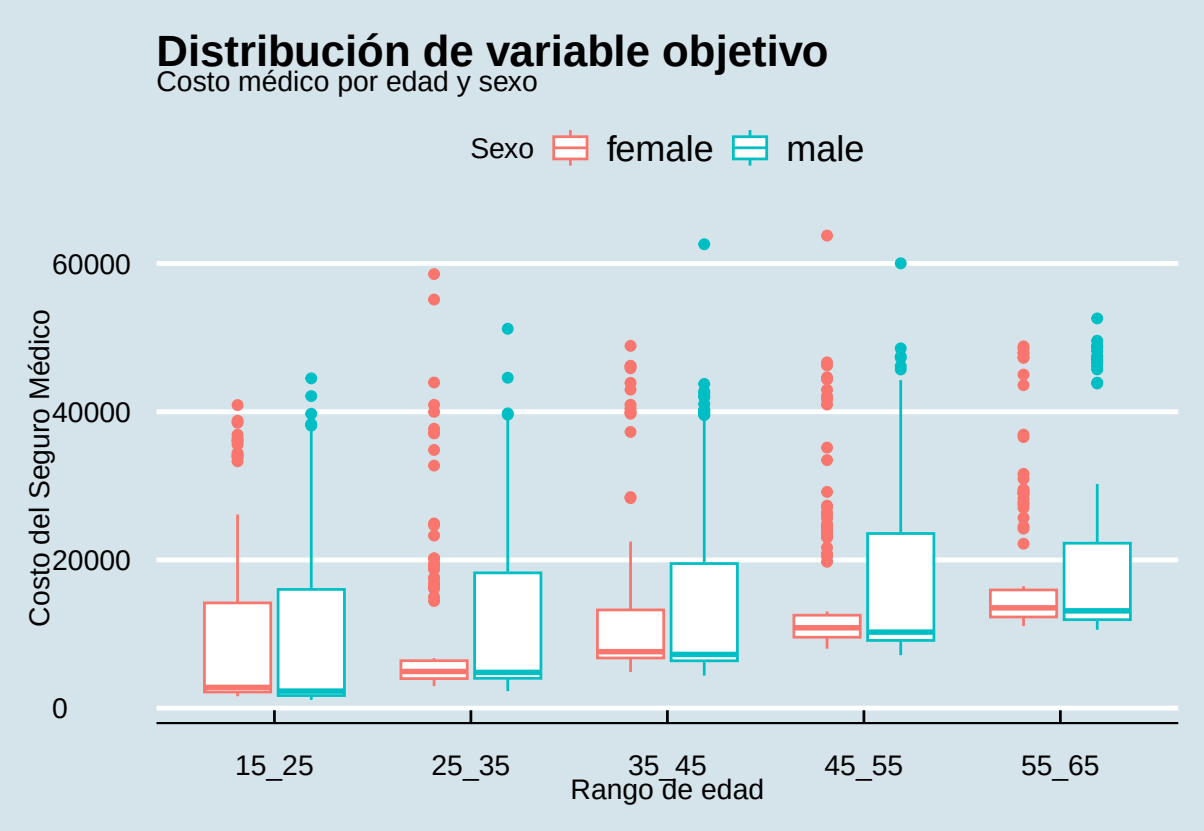
```
ggplot(insurance, aes(x = factor(region), y = charges, color=factor(smoker))) +
  geom_boxplot() +
   labs(title = "Distribución de variable objetivo",
        subtitle = "Costo médico por zona y variable fumador",
        x = "Zona", y = "Costo del Seguro Médico", color= "Fumador") +
  theme_economist()
```



Podemos ver que la zona realmente no es significativa en comparación de la variable que nos permite identificar si el asegurado es o no fumador.

Ahora analizemos el impacto de la edad del asegurado y el sexo con respecto al costo del seguro.

```
ggplot(insurance, aes(x = factor(age_bucket), y = charges,color=factor(sex))) +
  geom_boxplot() +
   labs(title = "Distribución de variable objetivo",
        subtitle = "Costo médico por edad y sexo",
        x = "Rango de edad", y = "Costo del Seguro Médico", color= "Sexo") +
  theme_economist()
```

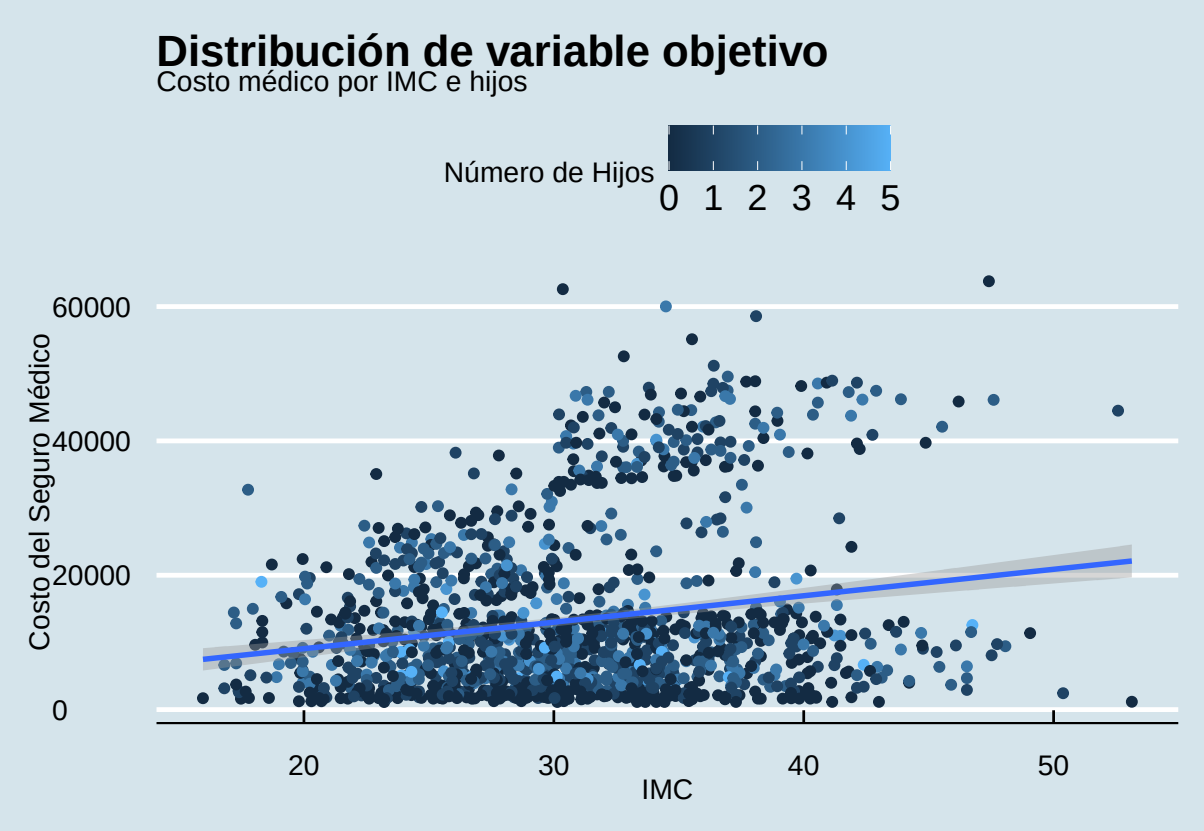


Podemos ver que existe un incremento de costo paulatino conforme cada rango de edades y que el sexo masculino tiende a tener costos mayores.

Ahora procederemos a medir el impacto del Índice de Masa Corporal, recordando lo siguiente:

- Si el IMC es menor a 18.4, el rango se denomina como peso insuficiente.
- Si el IMC se encuentra entre 18.5 y 24.9, se encuentra dentro del rango de peso normal.
- Si el IMC es entre 25.0 y 29.9, se encuentra dentro del rango de sobrepeso.
- Si el IMC es mayor o igual a 30 se considera obesidad.

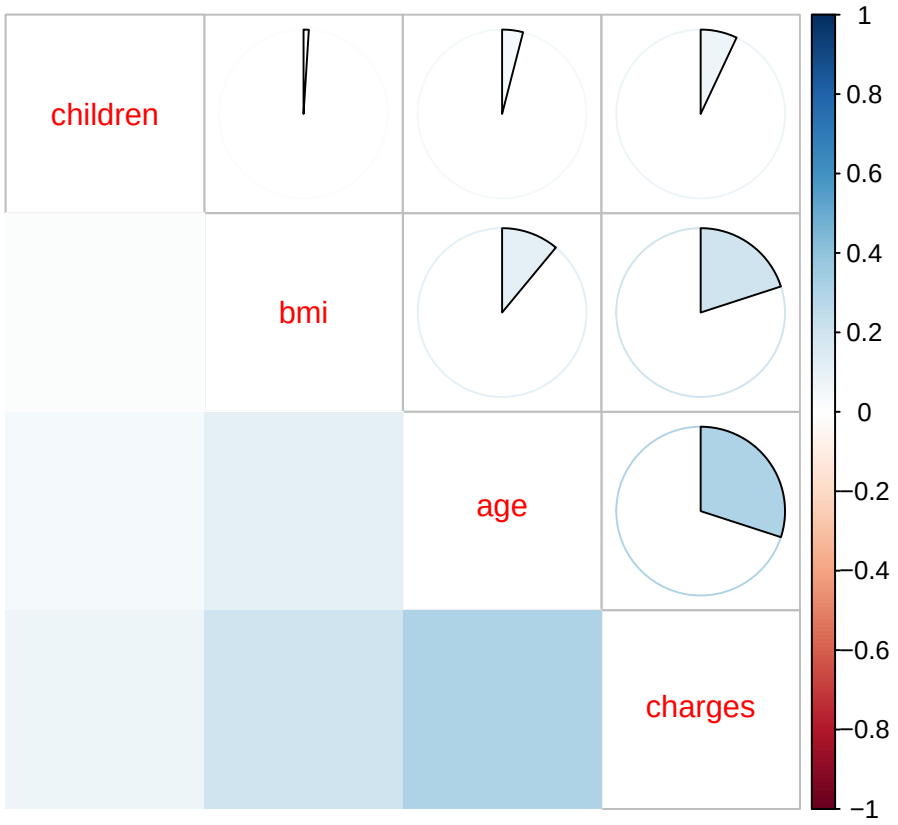
```
ggplot(insurance, aes(x = bmi, y = charges)) +
  geom_point(aes(color=children))+
  geom_smooth(formula = y ~ x, method = "lm")+
   labs(title = "Distribución de variable objetivo",
        subtitle = "Costo médico por IMC e hijos",
        x = "IMC", y = "Costo del Seguro Médico", color= "Número de Hijos") +
  theme_economist()
```



Podemos ver entre más alto sea el IMC, mayor es el costo del seguro médico.

Por último procederemos con la matriz de correlación:

```
cor_matrix <- insurance |> dplyr::select(where(is.numeric)) |> cor() |> round(2)

cor_matrix |>
  corrplot.mixed( lower = 'shade', upper = 'pie', order = 'hclust')
```



Donde la edad es la variable numérica más relacionada con el costo del seguro médico.

3.3 Depuración de datos

Antes de proceder con la transformación de variables es importante analizar con más profundidad la variable objetivo.

El primer paso vamos a crear el Zscore de la variable charges y filtrar los registros que se encuentren con valores mayores a 3:

```
insurance %>% mutate(zscore = (charges - mean(charges))/sd(charges)) %>% filter(zscore>3)
```

```
## # A tibble: 7 x 9
##     age sex      bmi children smoker region    charges age_bucket zscore
##   <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl> <fct>       <dbl>
## 1    28 male    36.4        1 yes    southwest  51195. 25_35        3.13
## 2    54 female  47.4        0 yes    southeast  63770. 45_55        4.17
## 3    31 female  38.1        1 yes    northeast  58571. 25_35        3.74
## 4    33 female  35.5        0 yes    northwest  55135. 25_35        3.46
## 5    60 male    32.8        0 yes    southwest  52591. 55_65        3.25
## 6    52 male    34.5        3 yes    northwest  60021. 45_55        3.86
## 7    45 male    30.4        0 yes    southeast  62593. 35_45        4.07
```

Podemos ver que tenemos 7 registros con cargos excesivos de seguro de gastos médicos. Para corregir este fenómeno procederemos a aplicar la raíz cuadrada de la variable:

```
insurance <- insurance %>% mutate(charges_sqr = sqrt(charges))
```

Y procederemos a volver a revisar el Z-score de la variable:

```
insurance %>%
  mutate(zscore = (charges_sqr -
mean(charges_sqr))/ sd(charges_sqr))%>%
  filter(zscore>3)
```

```
## # A tibble: 2 x 10
##     age sex      bmi children smoker region    charges age_bucket charg~1 zscore
##   <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl> <fct>        <dbl>  <dbl>
## 1    54 female  47.4        0 yes    southeast  63770. 45_55         253.   3.09
## 2    45 male    30.4        0 yes    southeast  62593. 35_45         250.   3.04
## # ... with abbreviated variable name 1: charges_sqr
```

Podemos ver que todavía existen dos registros, sin embargo, se encuentran muy cerca de 3 unidades, por lo tanto, podemos estar seguros que la variable ha sido corregida.

3.4 Codificación y normalizació de variables

En este apartado del trabajo procederemos a dummyficar las variables categóricas, cuidando la multicolinealidad de las variables (removiendo la primer columna que se genera) con la librería *fastDummies*.

```
dummyfied_data <-
  fastDummies::dummy_cols(
    insurance, remove_first_dummy = TRUE,
    select_columns = c( "sex", "smoker", "region", "age_bucket"))
```

Una vez transformado el set de datos, procederemos a reducir las variables a las que serán usadas en los procesos siguientes:

```
model_data <-
  dummyfied_data %>%
  dplyr::select(-c("sex", "smoker","region",
                   "age", "charges", "age_bucket"))
```

Una vez que se encuentren dummyficadas las variables, procederemos a normalizarlas:

```
model_data  <- model_data  %>%
  dplyr::mutate(across(c(bmi, children), ~ (. - mean(.)) / sd(.)))
```

3.5 Selección de variables

Primero procederemos identificando las variables dependientes e independientes:

```
dput(names(model_data))
```

```
## c("bmi", "children", "charges_sqr", "sex_male", "smoker_yes",
## "region_northwest", "region_southeast", "region_southwest", "age_bucket_25_35",
## "age_bucket_35_45", "age_bucket_45_55", "age_bucket_55_65")
```

```
variable_dep = c("charges_sqr")
variable_indepe = c("bmi", "children", "sex_male", "smoker_yes",
"region_northwest", "region_southeast", "region_southwest", "age_bucket_25_35",
"age_bucket_35_45", "age_bucket_45_55", "age_bucket_55_65")
```

Usaremos el cómputo en paralelo del computador:

```
GS_T0 <- Sys.time()
cluster <- makeCluster(detectCores() - 1) # number of cores
registerDoParallel(cluster) # register the parallel processing
```

Cargaremos el archivo de R especializado en selección de variables con step repetido y el archivo que nos permite realizar avnnet:

```
source("cruzadas avnnet y lin.R")
source("funcion steprepetido.R")
shhh(library(MASS)) # Librería para el step AIC y BIC
shhh(library(MXM)) # Librería para selección de variables con modelo Wrapper.
shhh(library(Boruta)) # Librería para selección de variables con modelo Wrapper.
shhh(library(caret)) # Librería de Machine Learning en R
```

Procederemos a seleccionar las variables más importantes usando el criterio AIC.

```
lista<-steprepetido(data=model_data,vardep=variable_dep,
                    listconti=variable_indepe,
                    sinicio=12345,sfinal=12385,porcen=0.8,criterio="AIC")

tabla<-lista[[1]]
dput(lista[[2]][[1]])

# Podemos ver que las variables seleccionadas por el Step repetido AIC son:
var_AIC <- c("smoker_yes", "age_bucket_55_65", "age_bucket_45_55", "age_bucket_35_45",
"bmi", "children", "age_bucket_25_35", "region_southeast", "region_southwest"
)
```

Ahora procederemos usando el criterio BIC:

```
lista<-steprepetido(data=model_data,vardep=variable_dep,
                    listconti=variable_indepe,
                    sinicio=12345,sfinal=12385,porcen=0.8,criterio="BIC")

tabla<-lista[[1]]
dput(lista[[2]][[1]])

# Podemos ver que las variables seleccionadas por el Step repetido BIC son:
var_BIC <-
c("smoker_yes", "age_bucket_55_65", "age_bucket_45_55", "age_bucket_35_45",
"bmi", "children", "age_bucket_25_35", "region_southeast")
```

Ahora procederemos con el Wrapper Boruta:

```
out.boruta <- Boruta(charges_sqr~., data = model_data)

print(out.boruta)

summary(out.boruta)

sal<-data.frame(out.boruta$finalDecision)

sal2<-sal[which(sal$out.boruta.finalDecision=="Confirmed"),,drop=FALSE]

dput(row.names(sal2))

length(dput(row.names(sal2)))

var_boruta <- c("bmi", "children", "smoker_yes", "age_bucket_25_35", "age_bucket_35_45",
"age_bucket_45_55", "age_bucket_55_65")
```

Ahora procederemos con el Wrapper MXM:

```
targetVariable <-as.vector( model_data$charges_sqr)
str(targetVariable,2)

model_data_matrix <- as.matrix(model_data[ ,!(colnames(model_data) == "charges_sqr")])
  model_data
dim(model_data_matrix)

mmpc1 <- MMPC( target  = targetVariable,
               dataset = model_data_matrix,
               max_k = 3, hash = TRUE, test = "testIndFisher")

mmpc1@selectedVars
```

```
a<-dput(names(model_data[ ,!(colnames(model_data) ==
              "charges_sqr")][,c(mmpc1@selectedVars)]))
length(a)
a

var_MXM <- c("bmi", "children", "smoker_yes", "age_bucket_45_55", "age_bucket_55_65")
```

Y por último utilizaremos el Wrapper RFE:

```
y<-model_data[,variable_dep]
x<-model_data[,variable_indepe]


control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(x, model_data[[3]], sizes=c(1:8), rfeControl=control)

selecrfe<-results$optVariables
length(selecrfe)
dput(selecrfe)

var_RFE <-c("smoker_yes", "age_bucket_55_65", "age_bucket_45_55", "bmi",
"age_bucket_35_45", "children", "region_southwest",
"region_southeast")
```

Veamos las variables seleccionadas por cada algoritmo:

```
var_AIC <- c("smoker_yes", "age_bucket_55_65", "age_bucket_45_55", "age_bucket_35_45",
"bmi", "children", "age_bucket_25_35", "region_southeast", "region_southwest"
)
length(var_AIC)
```

```
## [1] 9
```

```
var_BIC <-
c("smoker_yes", "age_bucket_55_65", "age_bucket_45_55", "age_bucket_35_45",
"bmi", "children", "age_bucket_25_35", "region_southeast")
length(var_BIC)
```

```
## [1] 8
```

```
var_boruta <- c("bmi", "children", "smoker_yes", "age_bucket_25_35", "age_bucket_35_45",
"age_bucket_45_55", "age_bucket_55_65")
length(var_boruta)
```

```
## [1] 7
```

```
var_MXM <- c("bmi", "children", "smoker_yes", "age_bucket_45_55", "age_bucket_55_65")
length(var_MXM)
```

```
## [1] 5
```

```
var_RFE <-c("smoker_yes", "age_bucket_55_65", "age_bucket_45_55", "bmi",
"age_bucket_35_45", "children", "region_southwest",
"region_southeast")
length(var_RFE)
```

```
## [1] 8
```

Una vez seleccionadas las variables de cada modelo, procederemos a evaluarlas con una regresión lineal y el uso de validación cruzada repetida:

```
medias1<-cruzadalin(data=model_data,
                    vardep="charges_sqr", listconti=var_AIC,
                    listclass=c(""), grupos=4,
                    sinicio=1234, repe=25)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1      TRUE 23.09582 0.7664525 15.68911 1.045924 0.02568053 0.5887543
```

```
medias1$modelo="STEPAIC_9"

medias2<-cruzadalin(data=model_data,
                    vardep="charges_sqr",listconti=var_BIC,
                    listclass=c(""), grupos=4,
                    sinicio=1234, repe=25)
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1      TRUE 23.12523 0.7658407 15.67569 1.057883 0.02598635 0.577777
```

```
medias2$modelo="STEPBIC_8"

medias3<-cruzadalin(data=model_data,
                    vardep="charges_sqr", listconti=var_boruta,
                    listclass=c(""), grupos=4,
                    sinicio=1234, repe=25)
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1      TRUE 23.15094 0.7653276 15.66406 1.066646 0.02620139 0.5894288
```

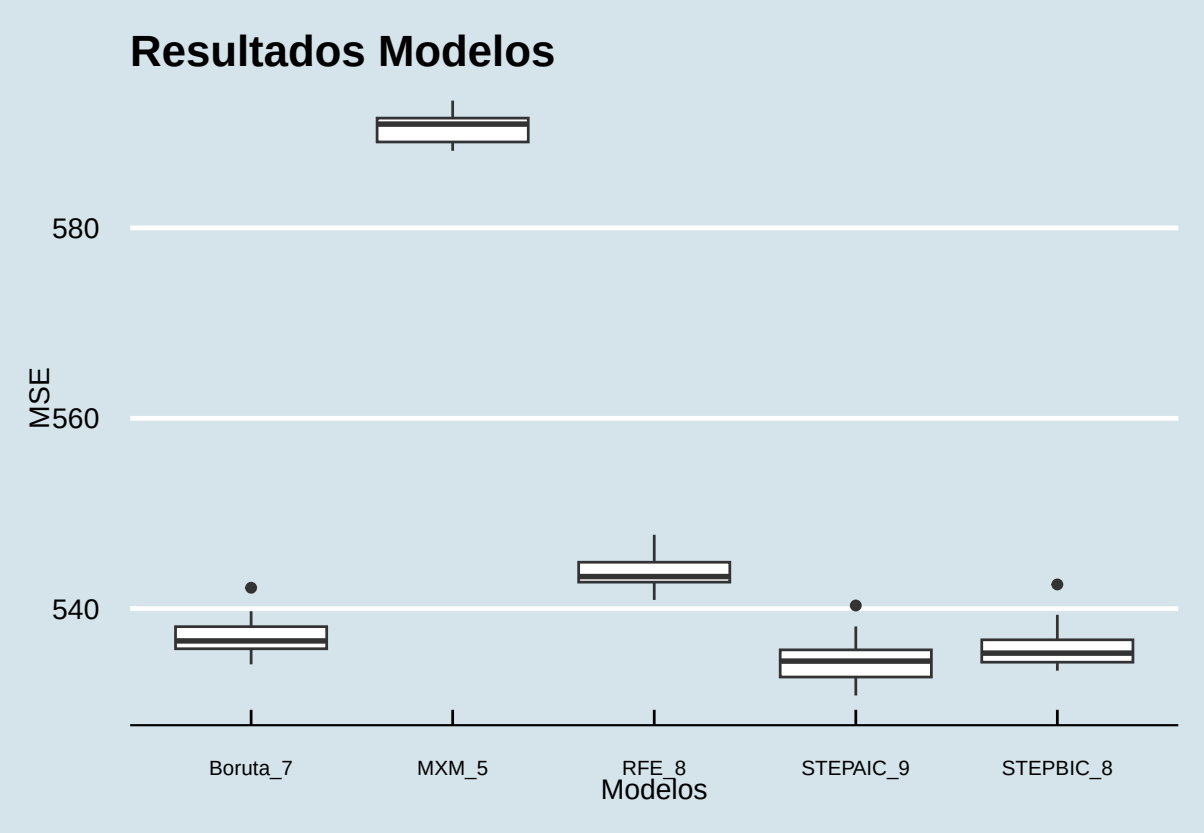
```
medias3$modelo="Boruta_7"

medias4<-cruzadalin(data=model_data,
                    vardep="charges_sqr", listconti=var_MXM,
                    listclass=c(""), grupos=4,
                    sinicio=1234, repe=25)
```

```
##   intercept     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1      TRUE 24.27967 0.7419401 17.69893 1.019206 0.02691089 0.6215507
```

```
medias4$modelo="MXM_5"

medias5<-cruzadalin(data=model_data,
                    vardep="charges_sqr", listconti=var_RFE,
                    listclass=c(""), grupos=4,
                    sinicio=1234, repe=25)
```

```
##   intercept     RMSE  Rsquared      MAE  RMSESD RsquaredSD     MAESD
## 1      TRUE 23.29719 0.7623194 15.98512 1.05791 0.02634972 0.6282285
```

```
medias5$modelo="RFE_8"

union1<-rbind(medias1, medias2, medias3, medias4, medias5)

ggplot(union1, aes(x = modelo, y = error)) +
  geom_boxplot() +
  xlab("Modelos") +
  ylab("MSE") +
  ggtitle("Resultados Modelos")+
  theme_economist(base_size = 12, base_family = "sans") +
  theme(axis.text.x = element_text(size = 8))
```



Con base a los resultados anteriores podemos decidir que las 7 variables del Wrapper Boruta son las más adecuadas para construir el modelo ya que sú MSE el cuál prácticamente es su error original (ya que la variable dependiente se encuentra transformada con su raíz cuadrada) es uno de los más bajos, con una varianza baja y con el menor número de variables, lo cuál nos ayuda a combatir el sobre ajuste.

3.6 Construcción red neuronal

Una vez elegidas las variables que participarán en la construcción del modelo, procederemos a encontrar los mejores parámetros de la red neuronal, primero sin early stopping.

3.6.1 Calculo de nodos.

7 variables, 1338 obs, con 30 obs. por parámetro, usando la siguiente fórmula:

$Nmero\ de\ parmetros = h(k+1)+h+1$

donde: * h es el número de nodos ocultos * k es el número de nodos input.

Una red con 7 variables independientes y 4 nodos ocultos necesitaría al menos 1,110 obs. y con 5 nodos ocultos 1,440 registros.

3.6.2 Tuneo de parámetros

Una red neuronal cuenta con 3 básicos parámetros en su construcción, el primero es el número de **nodos ocultos** en la red, el segundo es el **learning rate** el cuál es la velocidad a la que el algoritmo converge a las ponderaciones óptimas de sus pesos y por último el **número de interacciones** el cuál nos permite cuidar el sobreajuste del modelo.

Analizaremos los resultados del modelo de acuerdo a los siguientes rangos de parámetros:

- Nodos ocultos:

```
data2<-model_data[,c(var_boruta,"charges_sqr")]

control<-trainControl(method = "cv",
```

```
                      number=4,savePredictions = "all")

set.seed(123)
nnetgrid <-  expand.grid(size=c(4,5,8,10), decay=c(0.1, 0.01, 0.001, .0001),bag=F)
completo<-data.frame()
listaiter<-c(10,20,50,100,200,300,500)

for (iter in listaiter)
{
  rednnet<- train(charges_sqr~.,
                  data=data2,
                  method="avNNet", linout = TRUE, maxit=iter,
                  trControl=control, repeats=5, tuneGrid=nnetgrid,
                  trace=F)
  # Añado la columna del parametro de iteraciones
  rednnet$results$itera<-iter
  # Voy incorporando los resultados a completo
  completo<-rbind(completo,rednnet$results)
}

completo$MSE  <-(completo$RMSE)^2
completo<-completo[order(completo$MSE),]

ggplot(completo, aes(x=factor(itera), y=MSE,
                     color=factor(decay), pch=factor(size))) +
  geom_point(position=position_dodge(width=0.5),size=3)
```

Podemos ver que el error va en función de las iteraciones y el learning rate más que por los nodos, por lo tanto, nos quedaremos con 4 y 5 nodos y profundizaremos entre el factor de aprendizaje .01 y .1 y entre las 150 y 300 iteraciones del modelo.

```
set.seed(123)
nnetgrid <-  expand.grid(size=c(4,5),decay=c(0.2, 0.1, 0.05, 0.01),bag=F)
completo<-data.frame()
listaiter<-c(150, 200,250,300)

for (iter in listaiter)
{
  rednnet<- train(charges_sqr~.,
                  data=data2,
                  method="avNNet",linout = TRUE,maxit=iter,
                  trControl=control, repeats=5, tuneGrid=nnetgrid,
                  trace=F)
  # Añado la columna del parametro de iteraciones
  rednnet$results$itera<-iter
  # Voy incorporando los resultados a completo
  completo<-rbind(completo,rednnet$results)
}

completo$MSE  <-(completo$RMSE)^2
completo<-completo[order(completo$MSE),]

ggplot(completo, aes(x=factor(itera), y=MSE, color=factor(decay), pch=factor(size))) +
  geom_point(position=position_dodge(width=0.5), size=3)
```

Podemos ver que la mejor selección es el modelo con 4 nodos, 250 iteraciones y un learning rate de .2

3.7 Resultados

Ahora procederemos a comparar los resultados de cada método previo de regresión lineal con una red sin tuneo de hiperparámetros y otra con su debido tuneo.

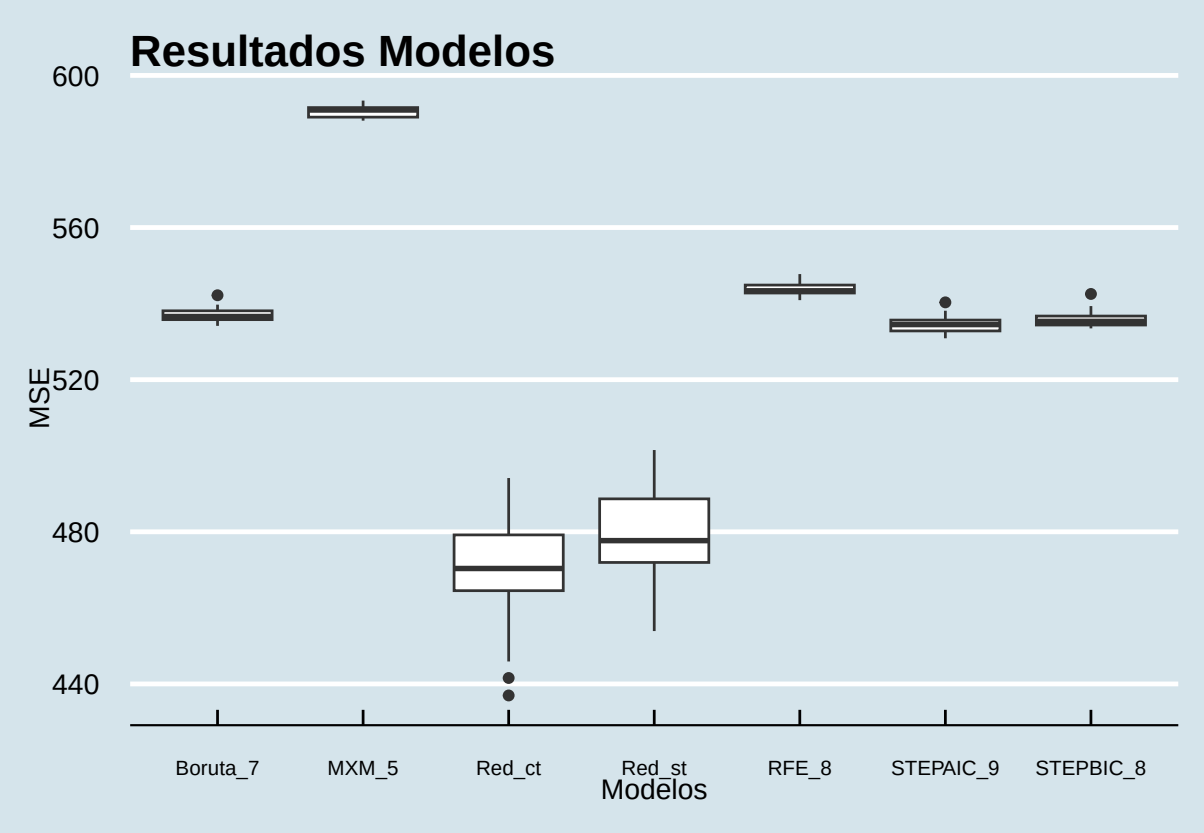
```
data2<-model_data[,c(var_boruta, "charges_sqr")]
medias6<-cruzadaavnnet(data=data2, vardep="charges_sqr",
                       listconti=var_boruta, listclass=c(""),
                       grupos=4, sinicio=1234, repe=25,
                       repeticiones=5, itera=50, size=c(4),
                       decay=c(0.2))
```

```
##   size decay   bag     RMSE Rsquared      MAE   RMSESD RsquaredSD     MAESD
## 1    4   0.2 FALSE 21.64706 0.797773 14.43601 1.226014 0.02865521 0.9686378
```

```
medias6$modelo="Red_ct"

medias7<-cruzadaavnnet(data=data2,  vardep="charges_sqr",
                       listconti=var_boruta, listclass=c(""),
                       grupos=4, sinicio=1234, repe=25,
                       repeticiones=5, itera=50, size=c(4),
                       decay=c(0.01))
```

```
##   size decay   bag     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1    4  0.01 FALSE 21.82915 0.7944698 14.66414 1.314505 0.02971302 1.092603
```

```
medias7$modelo="Red_st"

union1<-rbind(medias1,medias2,medias3,medias4,medias5,medias6, medias7)

ggplot(union1, aes(x = modelo, y = error)) +
  geom_boxplot() +
  xlab("Modelos") +
  ylab("MSE") +
  ggtitle("Resultados Modelos")+
  theme_economist(base_size = 12, base_family = "sans") +
  theme(axis.text.x = element_text(size = 8))
```



Podemos ver que el modelo **Red_ct** el cual hace alusión al modelo de redes neuronales con tuneo en parámetros tiene un mejor resultado que todos los anteriores, incluyendo al mismo modelo de red (variables iguales al modelo tuneado) con diferentes parámetros.

Podemos apreciar que el resultado obtenido a través de una red neuronal con un proceso correcto en selección de variables y tuneo de parámetros ha sido mejor en al menos 80 dólares que el siguiente modelo y más de 200 dólares del peor modelo. El resultado del trabajo es una red neuronal no solamente con el mejor sesgo de todos los modelos anteriores, si no también cuenta con una varianza controlada al poder hacer predicciones precisas y controladas evitando el sobreajuste.

4 Conclusiones generales

DD

	Red neuronal con tuneo SAS	Red neuronal con tuneo R
MSE	352.07506	392.492748

Variable Age que en SAS se quedo como continua y en R se dividió en rangos de 10 años y después se transformó en dummyes y se filtraron ciertos rangos por lo cual perdió explicabilidad en el modelo final.