# Redes neuronales Tarea

María Isabel Chuya-Nataly Quintanilla

## Tarea REDES NEURONALES

1. Descripción de los mismos numérica y gráficamente
2. Entender las fortalezas y debilidades del ANN perceptrón
4. Saber qué son las funciones de activación
5. Realizar un modelo preliminar de una capa sobre la clasificacion begnigno o maligno
6. Mejorar el modelo
7. Comparación de resultados mediante una matriz de confusión

### Fortalezas y debilidades del ANN perceptrón

**Fortalezas:**

1. Capacidad de aprendizaje no lineal: un perceptrón de ANN puede aprender y representar funciones no lineales, a diferencia de un perceptrón simple. Un perceptrón en un ANN puede capturar relaciones más complejas entre las características de entrada y las salidas deseadas al agregar capas ocultas con funciones de activación no lineales, como la función sigmoide o ReLU.
2. Mayor capacidad de representación: En comparación con un perceptrón simple, un perceptrón en un ANN tiene una mayor capacidad de representación debido a su estructura multicapa y las conexiones entre capas. Puede aprender las características abstractas y las representaciones jerárquicas de los datos, lo que lo hace adecuado para abordar problemas más complejos y tareas de aprendizaje más difíciles.
3. Aprendizaje con retroalimentación: la retropropagación del error, un algoritmo efectivo para ajustar los pesos y sesgos de la red neuronal, permite que los perceptrones de una red neuronal aprendan. La retroalimentación del error ayuda a la red a aprender de sus errores y mejorar su rendimiento con el tiempo.

**Debilidades:**

1. La complejidad computacional aumenta significativamente a medida que se agregan capas y conexiones en un ANN. Esto implica que, en comparación con un perceptrón simple, el entrenamiento y la predicción pueden requerir más tiempo y recursos computacionales.
2. Posibles sobreajustes: Cuando los ANN se entrenan con conjuntos de datos pequeños o ruidosos, es posible que se sobreajusten. En lugar de generalizar patrones más amplios, la red puede memorizar los datos de entrenamiento. Se necesitan métodos de regularización, como reducir la tasa de aprendizaje o incluir términos de regularización.
3. Dificultades de interpretación: La complejidad de ANN puede dificultar la comprensión del proceso de toma de decisiones de la red. La interpretación de los resultados de un ANN puede ser más difícil en comparación con modelos más simples y lineales debido a la naturaleza de la representación distribuida y las conexiones no lineales.
4. Mayor necesidad de datos de entrenamiento: Para que las ANNs funcionen correctamente, especialmente las con múltiples capas, pueden requerir una cantidad significativa de datos de entrenamiento. El ANN puede no poder comprender adecuadamente la complejidad del problema y sufrir de baja generalización si tiene pocos datos de entrenamiento.

## Funciones de Activación

Las funciones de activación de las redes neuronales se aplican a la salida de una neurona o unidad de procesamiento de una red neuronal. Estas características permiten que las redes neuronales capturen patrones y relaciones más complejos en los datos y introducen la no linealidad en la red.

Las funciones de activación se aplican a todas las entradas de una neurona, incluido el sesgo, y producen una salida que se transmite a las neuronas de la siguiente capa. Dado que permiten que la salida de una neurona no sea simplemente una combinación lineal de entradas, estas funciones introducen no linealidad en la red.

## Cargar Librerias

Se cargan las librerias al inicio, puesto que ayuda a que todo el codigo se desarrolle de manera continua.

```
library(ggplot2)
library(e1071)
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(reshape2)
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
library(caret)
```

```
Loading required package: lattice
```

```
library(kernlab)
```

```

Attaching package: 'kernlab'

The following object is masked from 'package:ggplot2':

    alpha
```

```
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```
library(grid)
library(ggfortify)
library(purrr)
```

```
Attaching package: 'purrr'

The following object is masked from 'package:kernlab':

    cross

The following object is masked from 'package:caret':

    lift
```

```
library(nnet)
library(ggstatsplot)
```

```
You can cite this package as:
     Patil, I. (2021). Visualizations with statistical details: The 'ggstatsplot' approach.
     Journal of Open Source Software, 6(61), 3167, doi:10.21105/joss.03167
```

```
library(knitr)
library(lavaan)
```

```
This is lavaan 0.6-15
lavaan is FREE software! Please report any bugs.
```

```r
library(doParallel) # parallel processing
```

```
Loading required package: foreach
```

```
Attaching package: 'foreach'
```

```
The following objects are masked from 'package:purrr':

    accumulate, when
```

```
Loading required package: iterators
```

```
Loading required package: parallel
```

```r
registerDoParallel()
require(foreach)
require(iterators)
require(parallel)
library(tidymodels)
```

```
Registered S3 method overwritten by 'parsnip':
  method          from
  autoplot.glmnet ggfortify
```

```
-- Attaching packages ---------------------------------------- tidymodels 1.1.0 --
```

```
v broom        1.0.4     v tibble       3.2.1
v dials        1.2.0     v tidyr        1.3.0
v infer        1.0.4     v tune         1.1.1
v modeldata    1.1.0     v workflows    1.1.3
v parsnip      1.1.0     v workflowsets 1.0.1
v recipes      1.0.6     v yardstick    1.2.0
v rsample      1.1.1
```

```
-- Conflicts ---------------------------------------- tidymodels_conflicts() --
x foreach::accumulate()    masks purrr::accumulate()
x scales::alpha()          masks kernlab::alpha(), ggplot2::alpha()
x gridExtra::combine()     masks dplyr::combine()
x purrr::cross()           masks kernlab::cross()
x scales::discard()        masks purrr::discard()
x dplyr::filter()          masks stats::filter()
x dplyr::lag()             masks stats::lag()
x purrr::lift()            masks caret::lift()
x rsample::permutations()  masks e1071::permutations()
x yardstick::precision()   masks caret::precision()
x yardstick::recall()      masks caret::recall()
x yardstick::sensitivity() masks caret::sensitivity()
x yardstick::specificity() masks caret::specificity()
x recipes::step()          masks stats::step()
x tune::tune()             masks parsnip::tune(), e1071::tune()
x recipes::update()        masks lavaan::update(), stats::update()
x foreach::when()          masks purrr::when()
* Learn how to get started at https://www.tidymodels.org/start/
```

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v readr     2.1.4
v lubridate 1.9.2     v stringr   1.5.0

-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x foreach::accumulate() masks purrr::accumulate()
x scales::alpha()       masks kernlab::alpha(), ggplot2::alpha()
x readr::col_factor()   masks scales::col_factor()
x gridExtra::combine()  masks dplyr::combine()
x purrr::cross()        masks kernlab::cross()
x scales::discard()     masks purrr::discard()
x dplyr::filter()       masks stats::filter()
x stringr::fixed()      masks recipes::fixed()
x dplyr::lag()          masks stats::lag()
x purrr::lift()         masks caret::lift()
x readr::spec()         masks yardstick::spec()
x foreach::when()       masks purrr::when()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
library(ggthemes)
library(skimr)
library(DataExplorer)
library(ggpubr)
library(mosaicData)
library(h2o)
```

```
----------------------------------------------------------------------

Your next step is to start H2O:
    > h2o.init()

For H2O package documentation, ask for help:
    > ??h2o

After starting H2O, you can use the Web UI at http://localhost:54321
For more information visit https://docs.h2o.ai

----------------------------------------------------------------------


Attaching package: 'h2o'

The following objects are masked from 'package:lubridate':

    day, hour, month, week, year

The following object is masked from 'package:pROC':

    var

The following objects are masked from 'package:stats':

    cor, sd, var

The following objects are masked from 'package:base':

    &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
    colnames<-, ifelse, is.character, is.factor, is.numeric, log,
    log10, log1p, log2, round, signif, trunc
```

```r
library(neuralnet)
```

```
Attaching package: 'neuralnet'

The following object is masked from 'package:dplyr':

    compute
```

```r
library(NeuralNetTools)
```

## Cargar Datos

- Para cargar los datos se establece una variable y se lee los datos cargados en la misma carpeta "read.csv("./wdbc.data"), header=T)"
- Se usa el comando "head()" para visualizar las primeras filas de un conjunto de datos o un objeto en forma de tabla.
- Se usa el comando "summary(datos)" para obtener las estadisticas descriptivas de las variables
- Se usa el comando "str(datos)" para realizar el analisis estadistico de los datos

```r
# Cargar los datos
datos <- read.csv("./wdbc.data",header = T)
datos.numericos <- datos[, which(unlist(lapply(datos, is.numeric)))]
colnames(datos.numericos) <- paste0("Var", rep(1:10))
datos<-datos[,1:10]
colnames(datos) <- c("ID_number", "Diagnosis", "radius1", "texture1",
                     "perimeter1", "area1","smoothness1", "compactness1",
                     "concavity1", "concave_points1")

# Explorar los datos
head(datos)  # Ver las primeras filas del conjunto de datos
```

```
  ID_number Diagnosis radius1 texture1 perimeter1  area1 smoothness1
1    842517         M   20.57    17.77     132.90 1326.0     0.08474
2  84300903         M   19.69    21.25     130.00 1203.0     0.10960
3  84348301         M   11.42    20.38      77.58  386.1     0.14250
```

```
4  84358402         M   20.29    14.34     135.10 1297.0     0.10030
5    843786         M   12.45    15.70      82.57  477.1     0.12780
6    844359         M   18.25    19.98     119.60 1040.0     0.09463
  compactness1 concavity1 concave_points1
1      0.07864     0.0869         0.07017
2      0.15990     0.1974         0.12790
3      0.28390     0.2414         0.10520
4      0.13280     0.1980         0.10430
5      0.17000     0.1578         0.08089
6      0.10900     0.1127         0.07400
```

```r
summary(datos)  # Obtener estadísticas descriptivas de las variables
```

```
   ID_number          Diagnosis            radius1          texture1
 Min.   :     8670   Length:568         Min.   : 6.981   Min.   : 9.71
 1st Qu.:   869222   Class :character   1st Qu.:11.697   1st Qu.:16.18
 Median :   906157   Mode  :character   Median :13.355   Median :18.86
 Mean   : 30423820                      Mean   :14.120   Mean   :19.31
 3rd Qu.:  8825022                      3rd Qu.:15.780   3rd Qu.:21.80
 Max.   :911320502                      Max.   :28.110   Max.   :39.28
   perimeter1         area1         smoothness1       compactness1
 Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
 1st Qu.: 75.14   1st Qu.: 420.2   1st Qu.:0.08629   1st Qu.:0.06481
 Median : 86.21   Median : 548.8   Median :0.09587   Median :0.09252
 Mean   : 91.91   Mean   : 654.3   Mean   :0.09632   Mean   :0.10404
 3rd Qu.:103.88   3rd Qu.: 782.6   3rd Qu.:0.10530   3rd Qu.:0.13040
 Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
   concavity1      concave_points1
 Min.   :0.00000   Min.   :0.00000
 1st Qu.:0.02954   1st Qu.:0.02031
 Median :0.06140   Median :0.03345
 Mean   :0.08843   Mean   :0.04875
 3rd Qu.:0.12965   3rd Qu.:0.07373
 Max.   :0.42680   Max.   :0.20120
```

```r
str(datos)
```

```
'data.frame':   568 obs. of  10 variables:
 $ ID_number      : int  842517 84300903 84348301 84358402 843786 844359 84458202 844981 845
 $ Diagnosis      : chr  "M" "M" "M" "M" ...
```

```
$ radius1         : num  20.6 19.7 11.4 20.3 12.4 ...
$ texture1        : num  17.8 21.2 20.4 14.3 15.7 ...
$ perimeter1      : num  132.9 130 77.6 135.1 82.6 ...
$ area1           : num  1326 1203 386 1297 477 ...
$ smoothness1     : num  0.0847 0.1096 0.1425 0.1003 0.1278 ...
$ compactness1    : num  0.0786 0.1599 0.2839 0.1328 0.17 ...
$ concavity1      : num  0.0869 0.1974 0.2414 0.198 0.1578 ...
$ concave_points1: num  0.0702 0.1279 0.1052 0.1043 0.0809 ...
```

### Descripcion numérica y gráfica

```
#Tabla de resusmen de datos
skim(datos)
```

Table 1: Data summary

| | |
|---|---|
| Name | datos |
| Number of rows | 568 |
| Number of columns | 10 |
| _____ | |
| Column type frequency: | |
| character | 1 |
| numeric | 9 |
| _____ | |
| Group variables | None |

**Variable type: character**

| skim_variable | n_missing | complete_rate | min | max | empty | n_unique | whitespace |
|---|---|---|---|---|---|---|---|
| Diagnosis | 0 | 1 | 1 | 1 | 0 | 2 | 0 |

**Variable type: numeric**

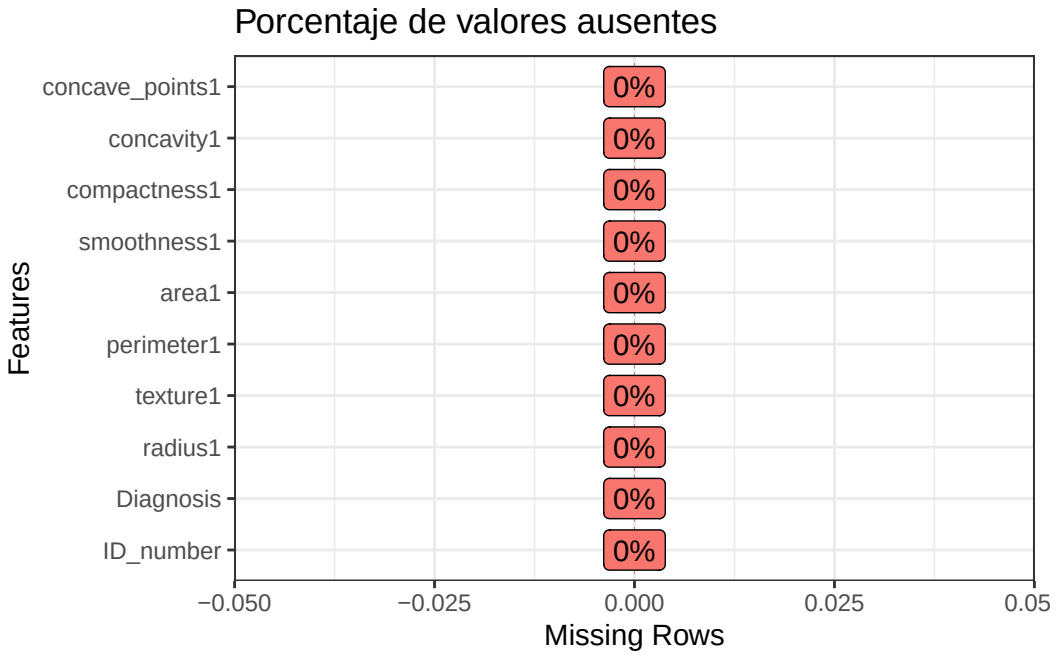
| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| ID_number | 0 | 1 | 30423820 | 125124628 | 8670.00 | 869222.50 | 906157.00 | 8825022.25 | 91320502.00 | |
| radius1 | 0 | 1 | 14.12 | 3.52 | 6.98 | 11.70 | 13.36 | 15.78 | 28.11 | |
| texture1 | 0 | 1 | 19.31 | 4.29 | 9.71 | 16.18 | 18.86 | 21.80 | 39.28 | |
| perimeter1 | 0 | 1 | 91.91 | 24.29 | 43.79 | 75.13 | 86.21 | 103.88 | 188.50 | |

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| area1 | 0 | 1 | 654.28 | 351.92 | 143.50 | 420.18 | 548.75 | 782.62 | 2501.00 | |
| smoothness1 | 0 | 1 | 0.10 | 0.01 | 0.05 | 0.09 | 0.10 | 0.11 | 0.16 | |
| compactness1 | 0 | 1 | 0.10 | 0.05 | 0.02 | 0.06 | 0.09 | 0.13 | 0.35 | |
| concavity1 | 0 | 1 | 0.09 | 0.08 | 0.00 | 0.03 | 0.06 | 0.13 | 0.43 | |
| concave_points1 | 0 | 1 | 0.05 | 0.04 | 0.00 | 0.02 | 0.03 | 0.07 | 0.20 | |

```
# Número de datos ausentes por variable
# ==============================================================================
datos %>% map_dbl(.f = function(x){sum(is.na(x))})
```

```
    ID_number     Diagnosis       radius1      texture1     perimeter1 
            0             0             0             0             0 
        area1    smoothness1   compactness1     concavity1 concave_points1 
            0             0             0             0             0 
```

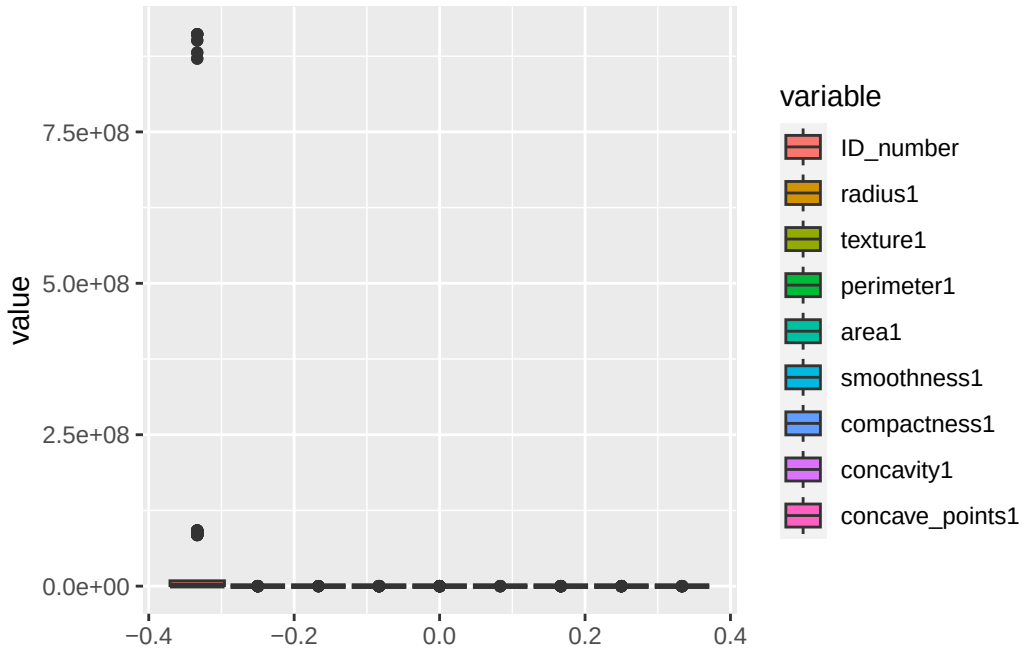
```
plot_missing(
  data    = datos,
  title   = "Porcentaje de valores ausentes",
  ggtheme = theme_bw(),
  theme_config = list(legend.position = "none")
)
```




```
#diagrama de cajas unido
datos.melt<-reshape2::melt((datos))
```

```
Using Diagnosis as id variables
```

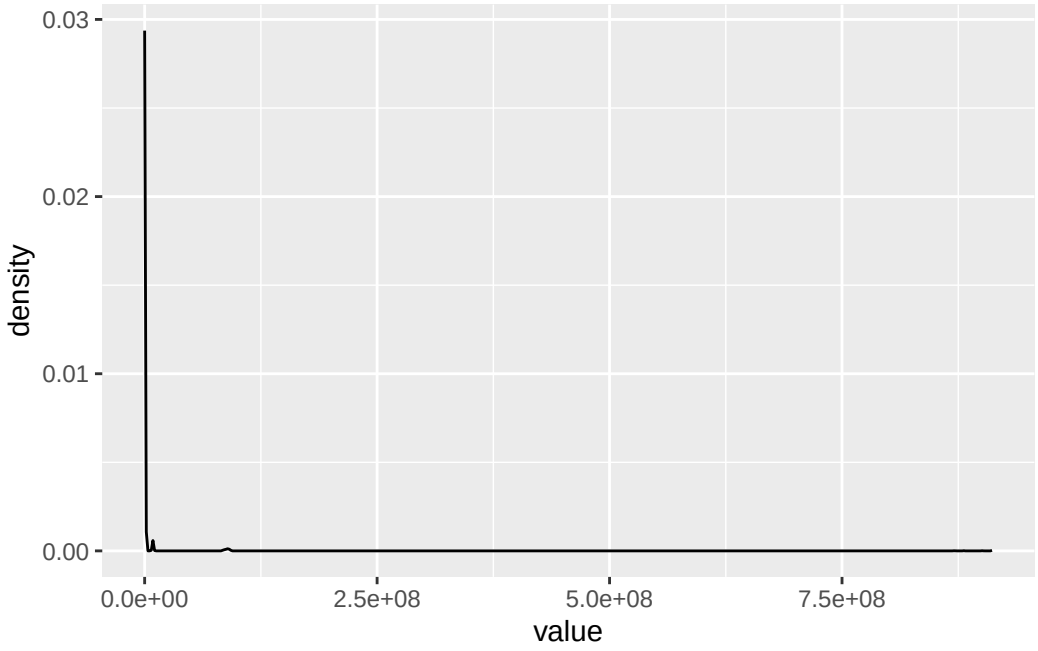
```
ggplot(datos.melt,aes(y=value,fill=variable))+geom_boxplot()
```




```
#grafica de densidad
datos.melt<-reshape2::melt((datos))
```

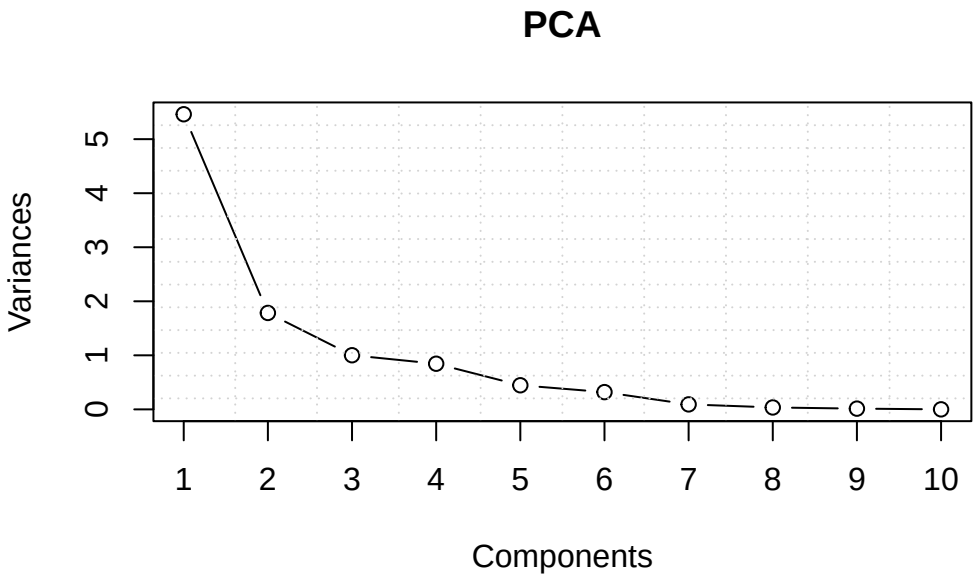
```
Using Diagnosis as id variables
```

```
ggplot(datos.melt,aes(x=value))+geom_density()
```

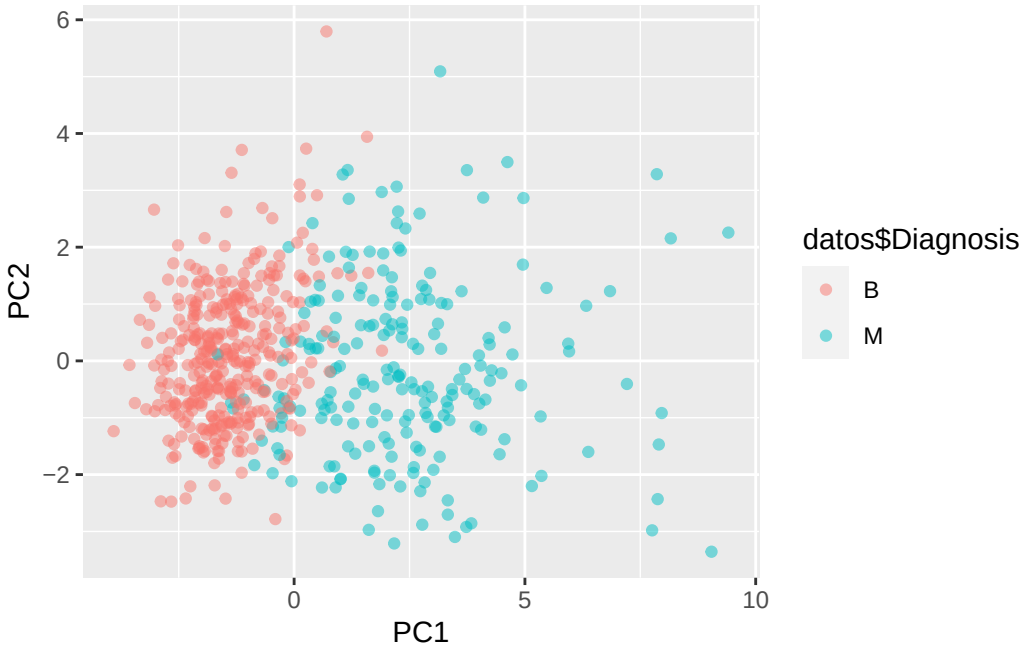



```
#csa<- as.data.frame(datos.test)
```

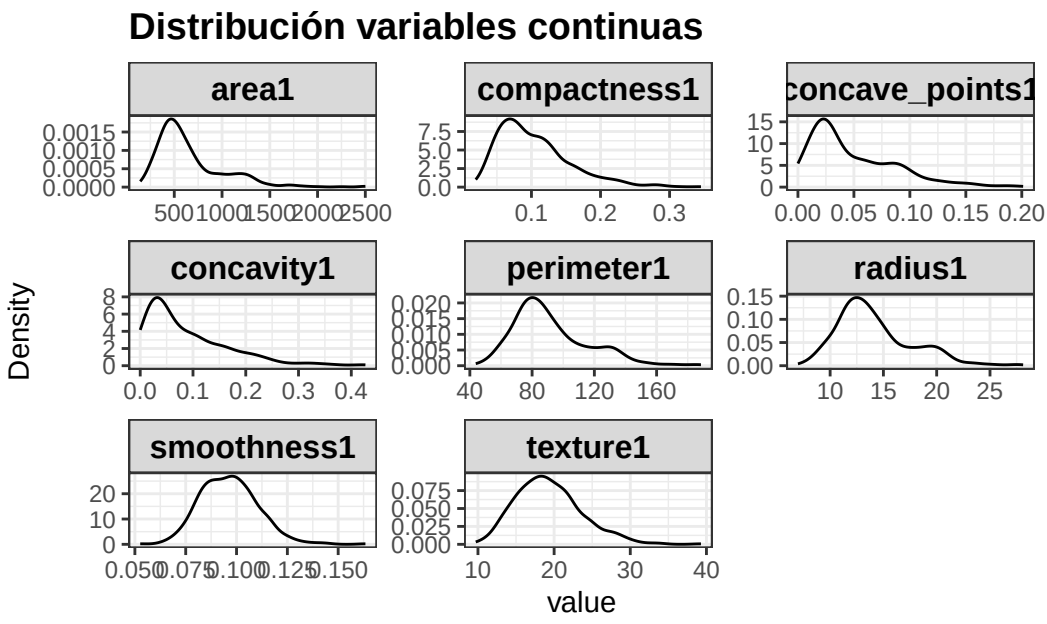
```
#PCA
cancer.pca <- prcomp(datos.numericos[, 1:10], center=TRUE, scale=TRUE)
plot(cancer.pca, type="l", main='')
grid(nx = 10, ny = 14)
title(main = "PCA", sub = NULL, xlab = "Components")
box()
```




```
pca_df <- as.data.frame(cancer.pca$x)
ggplot(pca_df, aes(x=PC1, y=PC2, col=datos$Diagnosis)) + geom_point(alpha=0.5)
```



```r
# Gráfico de distribución para cada variable numérica
# ==============================================================================
datos1<- datos[,-2]
plot_density(
  data     = datos %>% select(-ID_number),
  ncol     = 3,
  title    = "Distribución variables continuas",
  ggtheme = theme_bw(),
  theme_config = list(
                   plot.title = element_text(size = 14, face = "bold"),
                   strip.text = element_text(colour = "black", size = 12, face = 2)
                  )
  )
```




```r
# Reparto de datos en train y test
# ==============================================================================
n<-nrow(datos)
set.seed(123456)

datos$Diagnosis<-as.factor(datos$Diagnosis)
```

```r
train <- sample(n,floor(n*0.7))
datos.train <- datos[train,]
datos.test  <- datos[-train,]

csa<- as.data.frame(datos.test)
```

```r
# Se almacenan en un objeto `recipe` todos los pasos de preprocesado y, finalmente,
# se aplican a los datos.
transformer <- recipe(
                   formula = ID_number ~ .,
                   data =  datos.train
                ) %>%
                step_naomit(all_predictors()) %>%
                step_nzv(all_predictors()) %>%
                step_center(all_numeric(), -all_outcomes()) %>%
                step_scale(all_numeric(), -all_outcomes()) %>%
                step_dummy(all_nominal(), -all_outcomes())

transformer
```

```
-- Recipe ----------------------------------------------------------------------
```

```
-- Inputs

Number of variables by role

outcome:   1
predictor: 9
```

```
-- Operations

* Removing rows with NA values in: all_predictors()
```

```
* Sparse, unbalanced variable filter on: all_predictors()

* Centering for: all_numeric(), -all_outcomes()

* Scaling for: all_numeric(), -all_outcomes()

* Dummy variables from: all_nominal(), -all_outcomes()
```

```r
# Se entrena el objeto recipe
transformer_fit <- prep(transformer)

# Se aplican las transformaciones al conjunto de entrenamiento y de test
datos_train_prep <- bake(transformer_fit, new_data = datos.train)
datos_test_prep  <- bake(transformer_fit, new_data = datos.test)

glimpse(datos_train_prep)
```

```
Rows: 397
Columns: 10
$ radius1         <dbl> -1.13514497, -1.30682691, 1.39880068, -0.15573227, 1.8~
$ texture1        <dbl> -1.03994386, -0.79907142, 1.28225362, 1.25419081, 0.49~
$ perimeter1      <dbl> -1.1390348, -1.3177976, 1.4868003, -0.1830249, 1.65979~
$ area1           <dbl> -0.9821227, -1.0741236, 1.2740359, -0.2395643, 1.84939~
$ smoothness1     <dbl> 1.176845256, -0.810772101, -0.384360449, -0.262034903,~
$ compactness1    <dbl> -0.453929138, -1.193750788, 2.122415050, -0.490976070,~
$ concavity1      <dbl> -0.97825620, -0.90683930, 1.53466306, -0.45004776, 0.0~
$ concave_points1 <dbl> -0.94276622, -0.84467861, 1.31324874, -0.47574432, 0.2~
$ ID_number       <int> 858970, 882488, 855625, 866083, 883263, 873701, 881472~
$ Diagnosis_M     <dbl> 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, ~
```

```r
# Inicialización del cluster
# ==============================================================================
h2o.init(
  nthreads = -1,
  max_mem_size = "4g"
)
```

```
 Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         5 days 22 hours
    H2O cluster timezone:       America/Guayaquil
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.40.0.4
    H2O cluster version age:    2 months and 4 days
    H2O cluster name:           H2O_started_from_R_isach_wfo456
    H2O cluster total nodes:    1
    H2O cluster total memory:   3.23 GB
    H2O cluster total cores:    8
    H2O cluster allowed cores:  8
    H2O cluster healthy:        TRUE
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    R Version:                  R version 4.2.3 (2023-03-15)
```

```r
# Se eliminan los datos del cluster por si ya había sido iniciado.
h2o.removeAll()
h2o.no_progress()
```

```r
datos_train  <- as.h2o(datos_train_prep, key = "datos.train")
datos_test   <- as.h2o(datos_test_prep, key = "datos.test")
# Espacio de búsqueda de cada hiperparámetro
# ==============================================================================
hiperparametros <- list(
                      epochs = c(50, 100, 500),
                      hidden = list(5, 10, 25, 50, c(10, 10))
                    )


# Búsqueda por validación cruzada
# ==============================================================================
variable_respuesta <- 'radius1'
predictores <- setdiff(colnames(datos_train), variable_respuesta)

grid <- h2o.grid(
              algorithm    = "deeplearning",
```

```
                activation   = "Rectifier",
                x            = predictores,
                y            = variable_respuesta,
                training_frame = datos_train,
                nfolds       = 3, #validacion cruzada
                standardize  = FALSE,
                hyper_params = hiperparametros,
                search_criteria = list(strategy = "Cartesian"),
                seed         = 123456,
                grid_id      = "grid"
            )
```

```
Warning in h2o.getGrid(grid_id = grid_id): Some models were not built due to a
failure, for more details run `summary(grid_object, show_stack_traces = TRUE)`
```

```
# Resultados del grid
# ================================================================================
resultados_grid <- h2o.getGrid(
                     sort_by = 'rmse',
                     grid_id = "grid",
                     decreasing = FALSE
                   )
```

```
Warning in h2o.getGrid(sort_by = "rmse", grid_id = "grid", decreasing = FALSE):
Some models were not built due to a failure, for more details run
`summary(grid_object, show_stack_traces = TRUE)`
```

```
data.frame(resultados_grid@summary_table)
```

```
      epochs hidden     model_ids         rmse
1  104.30646      5  grid_model_2 2.122026e+00
2  503.77978     10  grid_model_6 6.237714e+03
3  503.77978      5  grid_model_3 1.155008e+04
4  104.30646     25  grid_model_8 1.926905e+04
5   51.05105      5  grid_model_1 2.694265e+04
6  104.30646     10  grid_model_5 3.780233e+04
7  104.30646     50 grid_model_11 3.878753e+04
8   51.05105     10  grid_model_4 6.031864e+04
9   51.05105     50 grid_model_10 6.441599e+04
```

```
10 503.77978     50 grid_model_12 8.638660e+04
11  51.05105     25  grid_model_7 2.813358e+05
12 503.77978     25  grid_model_9 3.734522e+05
```

```r
# Mejor modelo encontrado
# ================================================================================
modelo_final <- h2o.getModel(resultados_grid@model_ids[[1]])
```

```r
predicciones <- h2o.predict(
                  object  = modelo_final,
                  newdata = datos_test
                )

predicciones <- predicciones %>%
                as_tibble() %>%
                mutate(valor_real = as.vector(datos_test$Diagnosis))

predicciones %>% head(10)
```

```
# A tibble: 10 x 2
   predict valor_real
     <dbl>      <int>
 1    788.          1
 2  78889.          1
 3  79012.          1
 4    790.          1
 5   7929.          0
 6   7930.          1
 7    792.          1
 8    794.          1
 9    793.          1
10  79554.          1
```

```r
rmse(predicciones, truth = valor_real, estimate = predict, na_rm = TRUE)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard     144677.
```

### Figura percepción ANN

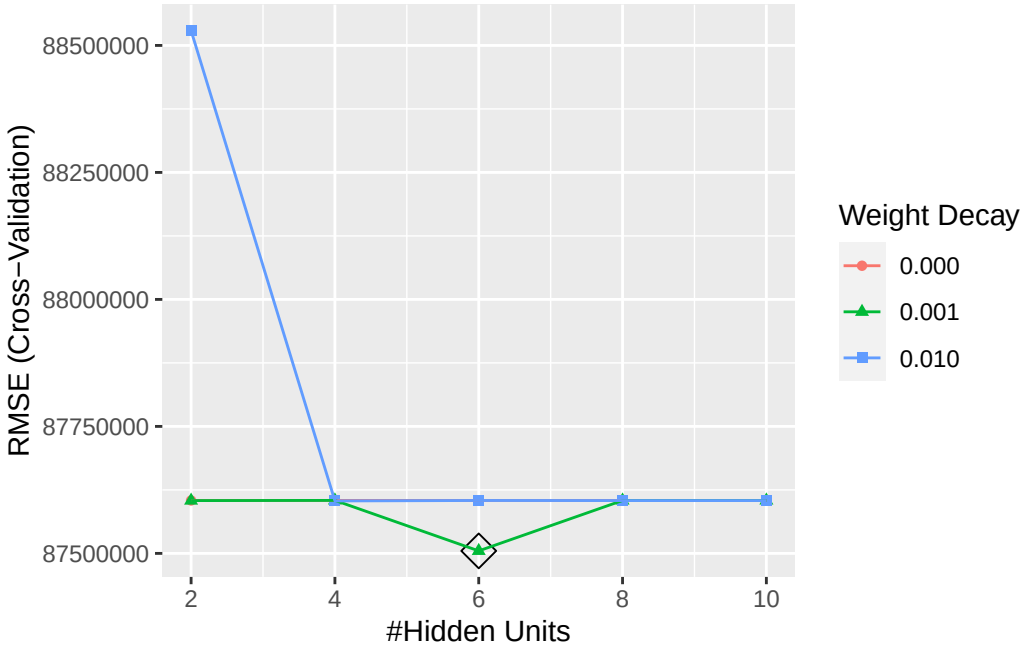
```
modelLookup("nnet")
```

```
  model parameter         label forReg forClass probModel
1  nnet      size #Hidden Units   TRUE     TRUE      TRUE
2  nnet     decay  Weight Decay   TRUE     TRUE      TRUE
```

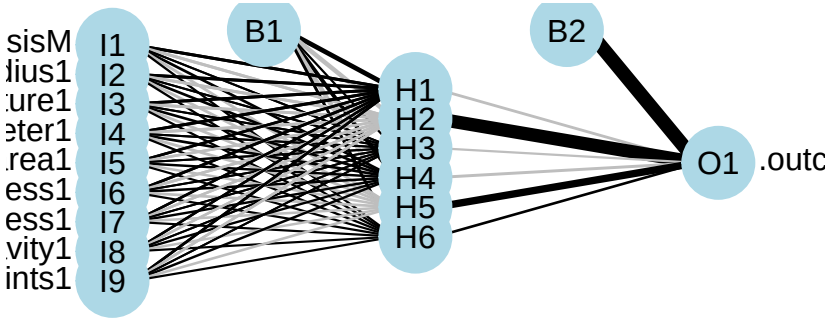
```
tuneGrid <- expand.grid(size = 2*1:5, decay = c(0, 0.001, 0.01))

set.seed(123456)
caret.nnet <- train(ID_number ~ ., data = datos.train, method = "nnet",
              preProc = c("range"), # Reescalado en [0,1]
              tuneGrid = tuneGrid,
              trControl = trainControl(method = "cv", number = 10),
              linout = TRUE, maxit = 200, trace = FALSE)
```

```
Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
: There were missing values in resampled performance measures.
```

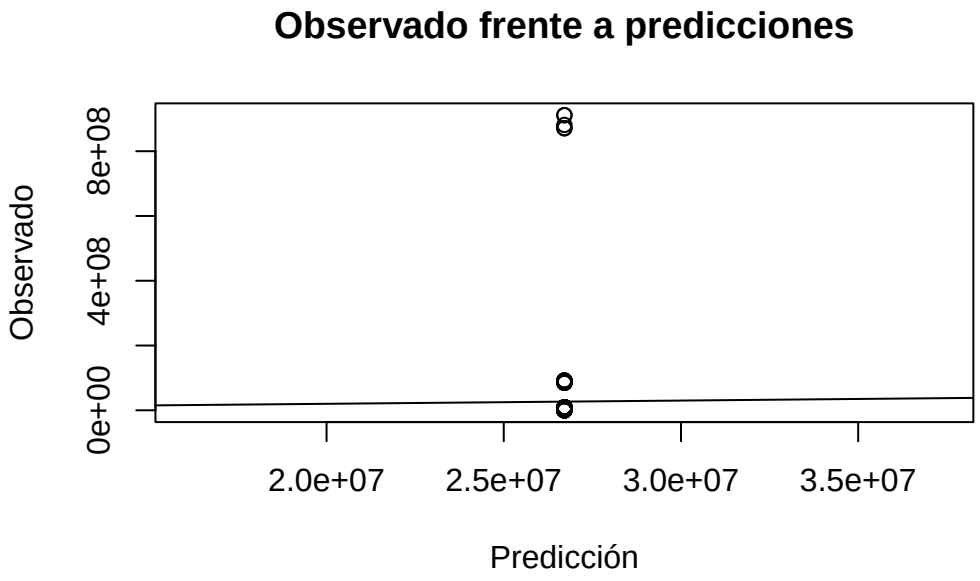
```
ggplot(caret.nnet, highlight = TRUE)
```




```
plotnet(caret.nnet$finalModel)
```



```r
pred <- predict(caret.nnet, newdata = datos.test)
obs <- datos.test$ID_number

plot(pred, obs, main = "Observado frente a predicciones",
     xlab = "Predicción", ylab = "Observado")
abline(a = 0, b = 1)
```




```r
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
```

```r
    mape = mean(abs(perr)),   # Error porcentual absoluto medio
    r.squared = 1 - sum(err^2)/sum((obs - mean(obs))^2) # Pseudo R-cuadrado
  ))
}

accuracy(pred, obs)
```

```
          me          rmse           mae           mpe          mape
 1.233360e+07  1.507906e+08  5.335515e+07 -7.131313e+03  7.154155e+03
   r.squared
-6.735132e-03
```