Curso : MACHINE LEARNING CON R

Tema : Preprocesamiento de Datos.

Docente : Victor Henostroza

**GUIA DE LABORATORIO 6**

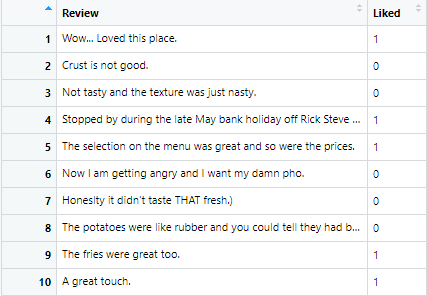
**Objetivos del laboratorio:**

1. **Desarrollar un Modelo de Minería de Texto.**
2. **Desarrollar un Análisis de Sentimientos en Twitter.**
3. **Desarrollar un Modelo de Clasificación con Keras (paquete Deep Learning).**
4. **Desarrollar un Modelo de Regresión con Keras (paquete Deep Learning).**
5. **Desarrollar un modelo no supervisado de clústeres con Mapas de Kohonen.**
6. **Desarrollar un modelo supervisado de clasificación con Mapas de Kohonen.**

**---------------------------------------------------------------------------------------------------------**

**Objetivo 1:**

El procesamiento natural del lenguaje (NLP) permite a empresas tomar datos de reviews (opiniones) de sus clientes para predecir si un determinado producto ha sido de su agrado o no, pero sobretodo, para poder ver cuáles son aquellas características principales que han hecho que su producto guste o no a su clientela para luego hacer políticas de mejora con base a dichos hallazgos. Nuestro ejemplo es de la data de un restaurante, este dataset solo tiene dos columnas, en la primera se muestra una descripción de la experiencia que tuvo el cliente cuando comió allí y en la segunda, la variable respuesta que muestra con 1 si le gustó o con 0 si no le gustó la experiencia de comer allí.

****

**# Procesamiento Natural de Lenguaje**

**# Importando el data set**

dataset\_original = read.delim("Restaurant\_Reviews.tsv", quote = '',

stringsAsFactors = FALSE)

**# Limpieza de textos**

# install.packages("tm")

#install.packages("SnowballC")

library(tm)

library(SnowballC)

**# Creando el cuerpo de texto a manipular (corpus)**

corpus = VCorpus(VectorSource(dataset\_original$Review))

corpus = tm\_map(corpus, content\_transformer(tolower))

**# Consultar el primer elemento del corpus**

as.character(corpus[[1]])

> as.character(corpus[[1]])

[1] "wow... loved this place."

**# Removiendo los números del corpus**

corpus = tm\_map(corpus, removeNumbers)

> as.character(corpus[[841]])

[1] "for 40 bucks a head, i really expect better food." **#** se eliminó 40

**# Removiendo los signos de puntuación del corpus**

corpus = tm\_map(corpus, removePunctuation)

> as.character(corpus[[1]])

[1] "wow loved this place" **#** se eliminaron los puntos suspensivos

# **Removiendo las palabras innecesarias del corpus (preposiciones, artículos, etc.)**

corpus = tm\_map(corpus, removeWords, stopwords(kind = "en"))

> as.character(corpus[[1]])

[1] "wow loved place" **#** se eliminó la palabra this

**# Obtención de palabras raíz del corpus (loved, loving, lovable = love)**

corpus = tm\_map(corpus, stemDocument)

> as.character(corpus[[1]])

[1] "wow love place"

**# Removiendo los espacios sobrantes del corpus**

corpus = tm\_map(corpus, stripWhitespace)

**# Crear el modelo Bag of Words**

dtm = DocumentTermMatrix(corpus)

dtm = removeSparseTerms(dtm, 0.999)

> dtm$i

[1] 1 1 1 2 2 3 3 3 3 4 4 4 4 4 5 5 5 5 6 6 6 6

[23] 6 7 7 7 8 8 8 8 8 8 9 9 10 10 11 11 12 13 13 13 13 13

# Muestra las columnas que son ocupadas por cada review

#La primera review utiliza las tres primeras columnas, la segunda las dos siguientes, etc.

> dtm$j

[1] 361 452 683 143 265 330 400 601 607 340 361 373 492 576 268 381 468 528 148 259 413 446

[23] 655 163 249 599 331 349 365 464 603 617 250 268 268 626 472 532 39 98 99 198 207 427

# Muestra las columnas que son llenadas con unos en orden de aparición

# Las palabras 361, 452 y 683 se corresponden a la primera review.

> dtm$v

[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

[46] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2

# Muestra la cantidad de veces que ha sido escrita una palabra en cada review.

# Casi todas las palabras han aparecido una vez en la muestra salvo la última que apareció 2 veces.

**# Convertir el bag of words a data frame**

dataset = as.data.frame(as.matrix(dtm))

**# Agregando la columna liked (variable respuesta) al final del dataset**

dataset$Liked = dataset\_original$Liked

**# Codificar la variable de clasificación como factor**

dataset$Liked = factor(dataset$Liked, levels = c(0,1))

> dataset$Liked

[1] 1 0 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 0 1 0 0 1 0 1 0 1 1 1 0 1 0 1 0 0 1 0

[46] 1 0 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 1 1 1

# Muestra las respuestas como positiva (1) o negativa (0) de la variable dependiente

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(123)

split = sample.split(dataset$Liked, SplitRatio = 0.80)

training\_set = subset(dataset, split == TRUE)

testing\_set = subset(dataset, split == FALSE)

**# Ajustar el Random Forest con el conjunto de entrenamiento.**

#install.packages("randomForest")

library(randomForest)

classifier = randomForest(x = training\_set[,-692], y = training\_set$Liked, ntree = 10)

**# Predicción de los resultados con el conjunto de testing**

y\_pred = predict(classifier, newdata = testing\_set[,-692])

> y\_pred

4 9 10 16 17 21 24 33 39 40 41 48 56 58 59 61 63 73 76 82 92 93 98 99

1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 1

105 112 113 115 116 122 123 142 150 152 154 157 158 159 161 169 182 183 184 188 190 191 193 199

0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1

**# Crear la matriz de confusión**

cm = table(testing\_set[, 692], y\_pred)

> cm

y\_pred

0 1

0 82 18

1 23 77

**# Hallar ratio de aciertos del modelo Random Forest**

ratio = sum(diag(cm))/sum(cm)

> ratio

[1] 0.795

Concluimos que el modelo logró casi un ratio de aciertos del 80% en la predicción de la variable respuesta (liked), este resultado permitirá tomar gestiones pertinentes de mejora a la empresa con base a los hallazgos encontrados.

**Objetivo 2:**

En el siguiente ejemplo, haremos un análisis de sentimientos, el cual también es un tipo de text mining, con datos de Twitter, en esta ocasión tendremos solo reviews (opiniones) sin variable respuesta, por lo tanto, haremos un Clustering Jerárquico para poder ver cuáles son las principales palabras utilizadas en dichos reviews y con esto determinar esas características (principales valores que toman las variables independientes).

# Análisis de sentimientos en Twiter

library(tm)

tweets <- readLines("Tweets.txt")

**# Crear el corpus (cuerpo del texto)**

corpus <- Corpus(VectorSource(tweets))

**# Crear matriz de documentos**

tdm <- TermDocumentMatrix(corpus, control = list(minWordLength=c(1,Inf)))

t <- removeSparseTerms(tdm, sparse=0.98)

m <- as.matrix(t)

> t

<<TermDocumentMatrix (terms: 83, documents: 2141)>>

Non-/sparse entries: 11741/165962

Sparsity : 93%

Maximal term length: 15

Weighting : term frequency (tf)

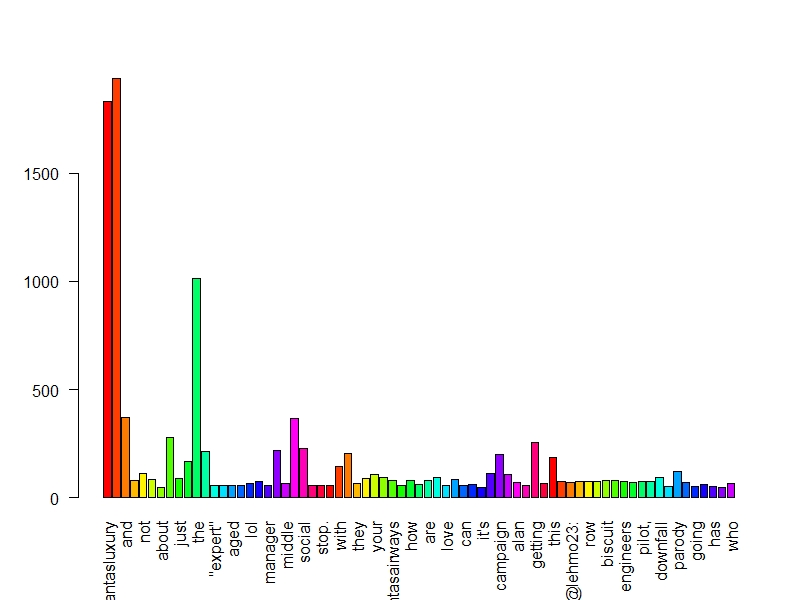
**#L**a matriz creada tiene solo 93% de esparcidad, eso quiere decir que el 93% de las columnas correspondientes a cada palabra ha sido dejada en blanco (no han sido mencionadas por los usauarios

**# Ploteo de términos más frecuentes (más de 50 veces de aparición)**

freq <- rowSums(m)

freq <- subset(freq, freq>=50)

barplot(freq, las=2, col = rainbow(25))



Del gráfico, apreciamos como se genera un histograma del conteo de las palabras que más se repiten en los twitts. Un dendograma debería dejar ver como se forman los clústeres.

**# Generar Clustering Jerárquico usando un Dendograma**

# Si la distancia es larga no deberían estar en el mismo clúster

# Si la distancia es corta deberían estar en el mismo clúster

distance <- dist(scale(m))

print(distance, digits = 2)

hc <- hclust(distance, method = "ward.D")

plot(hc, hang=-1)

rect.hclust(hc, k=12)

> print(distance, digits = 2)

#qantasluxury 22/11/2011 and having not had about for just that the twitter "expert"

22/11/2011 77.0

and 171.2 181.3

having 178.5 185.2 78.1

not 178.0 184.5 81.9 45.1

had 177.3 183.8 84.0 50.4 54.2

about 179.5 186.4 76.2 40.5 45.2 45.8

for 174.4 179.8 89.3 69.9 78.1 78.1 72.2

just 178.2 185.1 80.4 47.5 50.5 51.5 42.1 74.7

that 176.8 182.2 83.7 56.6 59.9 62.6 52.4 81.8 57.2

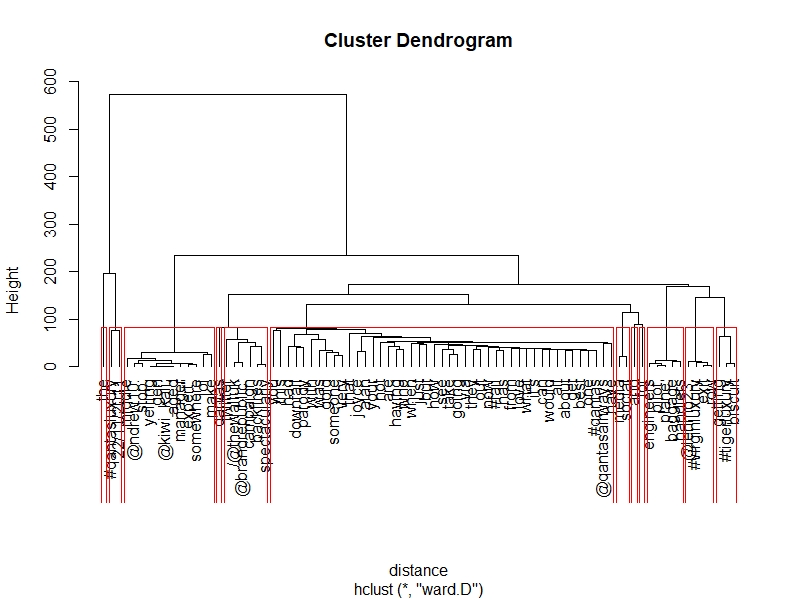
the 165.5 167.5 147.8 137.5 139.6 137.1 137.2 141.4 137.1 137.4

twitter 174.2 181.0 91.0 61.3 63.9 65.4 57.4 84.2 61.6 68.7 142.3

"expert" 180.1 186.8 77.1 37.9 42.8 43.1 32.0 71.7 39.9 53.1 139.0 56.3

@kiwi\_kali: @ndrew10: aged gen lol make manager media middle qantas social somewhere stop.

De la tabla, apreciamos como se forman los clústeres cuando las palabras tienen poca distancia en común, en cambio, lo que tiene valor muy alto deberán ser incluidos en ambos clúster.



Del gráfico, se aprecia cómo se han formado los 12 clústeres con los rectángulos rojos con las palabras cuyas similitudes.

**Objetivo 3:**

**# Red Neuronal de Clasificación con Paquete Keras**

# install\_keras()

library(keras)

**# Importar los datos (la variable dependiente a clasificar es NSP)**

data <- read.csv('Cardiotocographic.csv')

str(data)

> str(data)

'data.frame': 2126 obs. of 22 variables:

$ LB : int 120 132 133 134 132 134 134 122 122 122 ...

$ AC : num 0 0.00638 0.00332 0.00256 0.00651 ...

$ FM : num 0 0 0 0 0 0 0 0 0 0 ...

$ UC : num 0 0.00638 0.00831 0.00768 0.00814 ...

$ DL : num 0 0.00319 0.00332 0.00256 0 ...

$ DS : num 0 0 0 0 0 0 0 0 0 0 ...

$ DP : num 0 0 0 0 0 ...

$ ASTV : int 73 17 16 16 16 26 29 83 84 86 ...

$ MSTV : num 0.5 2.1 2.1 2.4 2.4 5.9 6.3 0.5 0.5 0.3 ...

$ ALTV : int 43 0 0 0 0 0 0 6 5 6 ...

$ MLTV : num 2.4 10.4 13.4 23 19.9 0 0 15.6 13.6 10.6 ...

$ Width : int 64 130 130 117 117 150 150 68 68 68 ...

$ Min : int 62 68 68 53 53 50 50 62 62 62 ...

$ Max : int 126 198 198 170 170 200 200 130 130 130 ...

$ Nmax : int 2 6 5 11 9 5 6 0 0 1 ...

$ Nzeros : int 0 1 1 0 0 3 3 0 0 0 ...

$ Mode : int 120 141 141 137 137 76 71 122 122 122 ...

$ Mean : int 137 136 135 134 136 107 107 122 122 122 ...

$ Median : int 121 140 138 137 138 107 106 123 123 123 ...

$ Variance: int 73 12 13 13 11 170 215 3 3 1 ...

$ Tendency: int 1 0 0 1 1 0 0 1 1 1 ...

$ NSP : int 2 1 1 1 1 3 3 3 3 3 ...

**# Modificar dataset a tipo matriz**

data <- as.matrix(data)

dimnames(data) <- NULL # Remueve el nombre de las variables

**# Normalizar variables independientes (entre la columna 1 a la columna 21)**

data[, 1:21] <- normalize(data[,1:21]) # normalize es una función del paquete Keras

data[,22] <- as.numeric(data[,22]) -1 # Cambia los valores a 0, 1, 2 en la variable respuesta

summary(data)

> summary(data)

V1 V2 V3 V4 V5

Min. :0.2894 Min. :0.000e+00 Min. :0.000e+00 Min. :0.000e+00 Min. :0.000e+00

1st Qu.:0.3747 1st Qu.:0.000e+00 1st Qu.:0.000e+00 1st Qu.:5.398e-06 1st Qu.:0.000e+00

Median :0.3862 Median :4.572e-06 Median :0.000e+00 Median :1.286e-05 Median :0.000e+00

Mean :0.3833 Mean :8.915e-06 Mean :2.721e-05 Mean :1.264e-05 Mean :5.560e-06

3rd Qu.:0.3949 3rd Qu.:1.596e-05 3rd Qu.:7.171e-06 3rd Qu.:1.900e-05 3rd Qu.:9.478e-06

Max. :0.4537 Max. :5.983e-05 Max. :1.446e-03 Max. :4.244e-05 Max. :4.997e-05

**# Generar conjuntos de entrenamiento y testeo**

set.seed(1234)

ind <- sample(2, nrow(data), replace = T, prob = c(0.8, 0.2))

training <- data[ind==1, 1:21] # Particion de entrenamiento para las variables independientes

test <- data[ind==2, 1:21] # Particiòn de testeo para las variables independientes

trainingtarget <- data[ind==1, 22] # Partición de entrenamiento de la variable respuesta

testtarget <- data[ind==2, 22] # Partición de testeo de la variable respuesta

**# Generar variables dummy para la variable respuesta**

trainLabels <- to\_categorical(trainingtarget)

testLabels <- to\_categorical(testtarget)

print(testLabels)

> print(testLabels)

[,1] [,2] [,3]

[1,] 1 0 0

[2,] 1 0 0

[3,] 1 0 0

[4,] 0 0 1

[5,] 0 0 1

[6,] 0 1 0

**# Crear modelo secuencial de red neuronal**

model <- keras\_model\_sequential()

model %>%

layer\_dense(units=12, activation = 'relu', input\_shape = c(21)) %>%

layer\_dense(units=6, activation = 'relu') %>%

layer\_dense(units = 3, activation = 'softmax')

# layer\_dense crea la red neuronal de tipo totalmente conectada

# el primer layer\_dense es para crear la capa interna de nodos

# el segundo layer\_dense es para crear la capa final de nodos hacia la respuesta

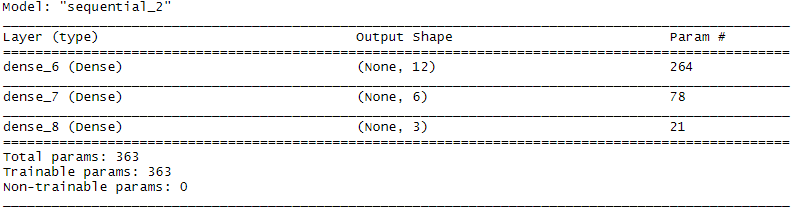
# unit indica el numero de nodos de la red neuronal en su capa oculta

# activation se usa para indicar el tipo de forma de aprender que tendrá la red neuronal

# input\_shape es usado para indicar la cantidad de nodos de entrada (columnas)

summary(model)

> summary(model)



# 264 parámetros = 12 nodos \* 21 entradas + 12 constantes

# 78 parámetros = 6 nodos \* 12 entradas + 6 constantes

# 21 parámetros = 3 nodos \* 6 entradas + 3 constantes

**# Compilar el modelo**

model %>%

compile(loss = 'categorical\_crossentropy', # la pérdida es por entropía cruzada

optimizer = 'adam', # el optimizador elegido es “Adam”

metrics = 'accuracy') # la métrica es el ajuste del modelo

**# Ajustar el modelo**

history <- model %>%

fit(training,

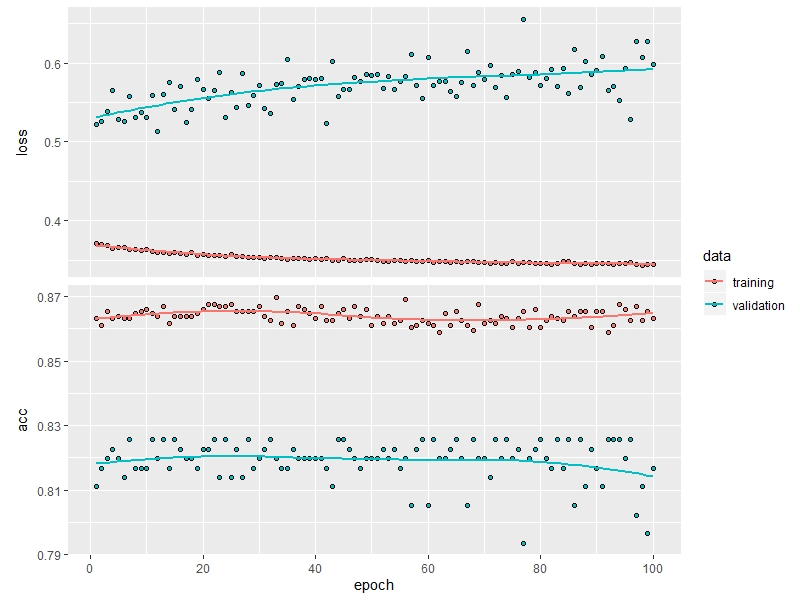
trainLabels,

epoch = 100, # cantidad de iteraciones

batch\_size = 32, # tamaño del lote de iteración

validation\_split = 0.2) # subconjunto de validación

plot(history)



**# Evaluar el modelo con el conjunto de testeo**

model1 <- model %>%

evaluate(test, testLabels)

**# Predicciones y matriz de confusión con el conjunto de testeo**

prob <- model %>%

predict\_proba(test)

pred <- model %>%

predict\_classes(test)

table1 <- table(Predicted = pred, Actual = testtarget)

> table1

Actual

Predicted 0 1 2

0 286 27 5

1 18 40 2

2 3 2 25

cbind(prob, pred, testtarget)

> cbind(prob, pred, testtarget)

pred testtarget

[1,] 9.938437e-01 6.149555e-03 6.721159e-06 0 0

[2,] 9.863304e-01 1.362355e-02 4.614077e-05 0 0

[3,] 9.923028e-01 7.571103e-03 1.260637e-04 0 0

[4,] 3.886975e-03 7.965337e-02 9.164596e-01 2 2

[5,] 3.553885e-03 7.284949e-02 9.235966e-01 2 2

[6,] 9.782860e-01 1.890625e-02 2.807707e-03 0 1

ratio = sum(diag(table1))/sum(table1)

> ratio

[1] 0.8602941

**Objetivo 4:**

**#Red Neuronal de Regresión con paquete Keras**

library(keras)

library(mlbench)

library(dplyr)

library(magrittr)

library(neuralnet)

**# Importar los datos**

data("BostonHousing")

data <- BostonHousing

str(data)

> str(data)

'data.frame': 506 obs. of 14 variables:

$ crim : num 0.00632 0.02731 0.02729 0.03237 0.06905 ...

$ zn : num 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...

$ indus : num 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...

$ chas : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ nox : num 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...

$ rm : num 6.58 6.42 7.18 7 7.15 ...

$ age : num 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...

$ dis : num 4.09 4.97 4.97 6.06 6.06 ...

$ rad : num 1 2 2 3 3 3 5 5 5 5 ...

$ tax : num 296 242 242 222 222 222 311 311 311 311 ...

$ ptratio: num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...

$ b : num 397 397 393 395 397 ...

$ lstat : num 4.98 9.14 4.03 2.94 5.33 ...

$ medv : num 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

**# Convertir las variables de tipo factor a numéricas**

data %<>% mutate\_if(is.factor, as.numeric) #Convierte todas las variables categoricas a numericas

**# Generar la red neuronal (medv es la variable respuesta)**

n <- neuralnet(medv ~ crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+b+lstat,

data = data,

hidden = c(10,5), #dos capas de neuronas internas con 10 nodos y 5 nodos

linear.output = F,

lifesign = 'full',

rep=1)

**# Visualizar una red neuronal**

plot(n,

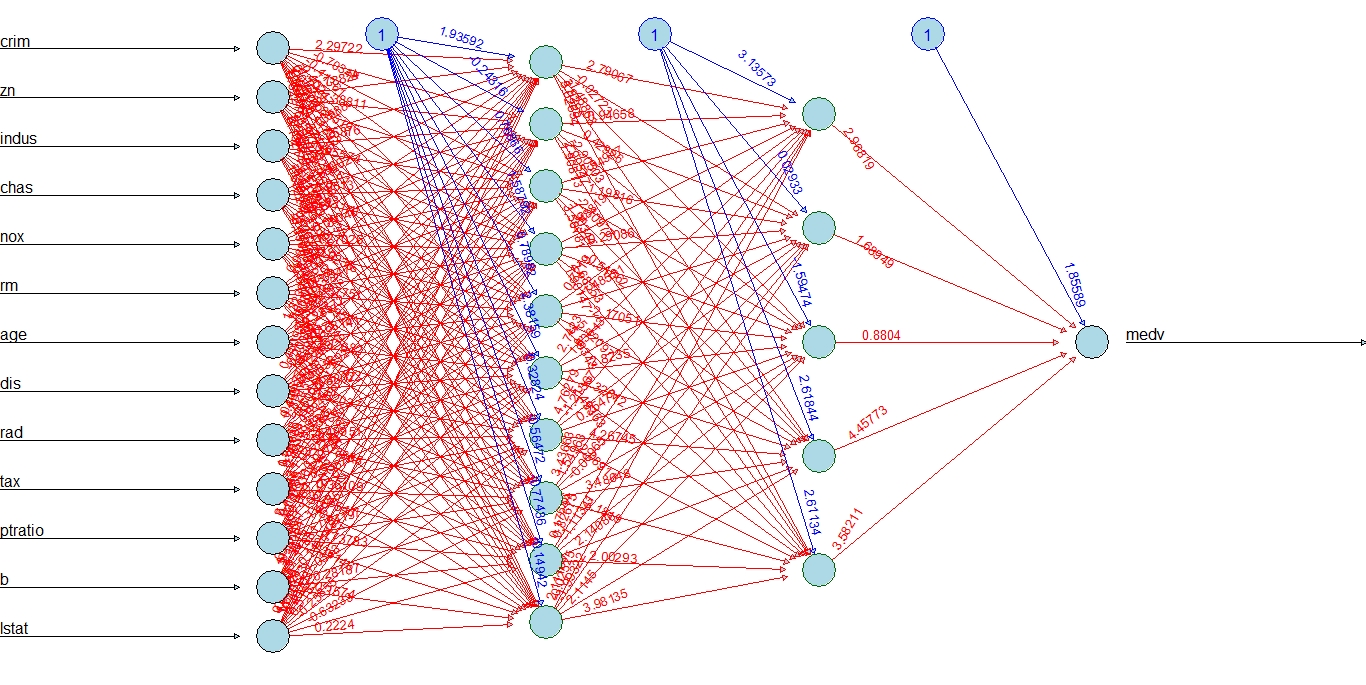
col.hidden = 'darkgreen', #color de los bordes de los nodos

col.hidden.synapse = 'red', #color de las conexiones entre nodos

show.weights = T, #mostrar los pesos de los nodos

information = F, #no mostrar información adicional

fill = 'lightblue') #color de los nodos



**# Convertir el dataset en matriz**

data <- as.matrix(data)

dimnames(data) <- NULL #Eliminar los nombres de las variables

head(data)

> head(data)

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]

[1,] 0.00632 18 2.31 1 0.538 6.575 65.2 4.0900 1 296 15.3 396.90 4.98 24.0

[2,] 0.02731 0 7.07 1 0.469 6.421 78.9 4.9671 2 242 17.8 396.90 9.14 21.6

[3,] 0.02729 0 7.07 1 0.469 7.185 61.1 4.9671 2 242 17.8 392.83 4.03 34.7

[4,] 0.03237 0 2.18 1 0.458 6.998 45.8 6.0622 3 222 18.7 394.63 2.94 33.4

[5,] 0.06905 0 2.18 1 0.458 7.147 54.2 6.0622 3 222 18.7 396.90 5.33 36.2

[6,] 0.02985 0 2.18 1 0.458 6.430 58.7 6.0622 3 222 18.7 394.12 5.21 28.7

**# Generar conjunto de entrenamiento y testeo**

set.seed(1234)

ind <- sample(2, nrow(data), replace = T, prob = c(.7, .3))

training <- data[ind==1,1:13]

test <- data[ind==2, 1:13]

trainingtarget <- data[ind==1, 14]

testtarget <- data[ind==2, 14]

**# Normalizar las variables independientes de los conjuntos de entrenamiento y testeo**

m <- colMeans(training)

s <- apply(training, 2, sd)

training <- scale(training, center = m, scale = s)

test <- scale(test, center = m, scale = s)

**# Crear modelo de red neuronal**

model <- keras\_model\_sequential()

model %>%

layer\_dense(units = 100, activation = 'relu', input\_shape = c(13)) %>%

# 100 nodos para la capa interna y 13 nodos para la capa de entrada

layer\_dropout(rate = 0.4) %>%

# El 40% de los nodos de la capa se ponen en cero para reducir sobreajuste

layer\_dense(units=50, activation = 'relu') %>%

# 50 nodos para la segunda capa interna

layer\_dropout(rate = 0.3) %>%

# El 30% de los nodos de la capa se ponen en cero para reducir sobreajuste

layer\_dense(units=25, activation = 'relu') %>%

# 25 nodos para la tercera capa interna

layer\_dropout(rate = 0.2) %>%

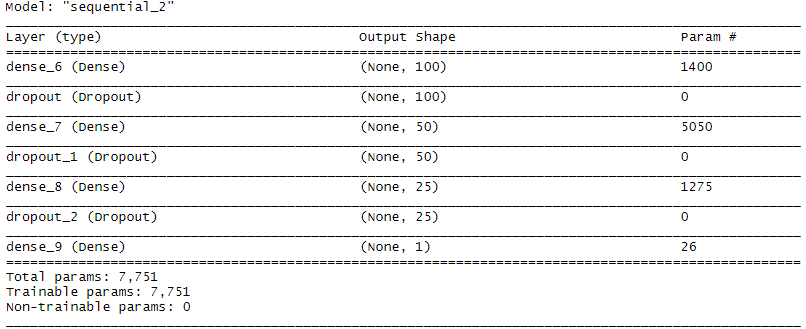
# El 20% de los nodos de la capa se ponen en cero para reducir sobreajuste

layer\_dense(units = 1)

# 1 nodo para la variable respuesta

summary(model)

> summary(model)



**# Compilar el modelo de red neuronal**

model %>% compile(loss = 'mse', # pérdida por defecto (error cuadrático medio)

optimizer = 'rmsprop', # optimizador elegido

metrics = 'mae') # métrica por defecto (error medio absoluto)

**# Ajustar el modelo de red neuronal con el conjunto de entrenamiento**

mymodel <- model %>%

fit(training,

trainingtarget,

epochs = 100,

batch\_size = 32,

validation\_split = 0.2)

**# Evaluar el modelo de red neuronal con el conjunto de testeo**

model %>% evaluate(test, testtarget)

> model %>% evaluate(test, testtarget)

151/151 [==============================] - 0s 31us/sample - loss: 26.6426 - mean\_absolute\_error: 3.2078

$loss

[1] 26.64257

$mean\_absolute\_error

[1] 3.207774

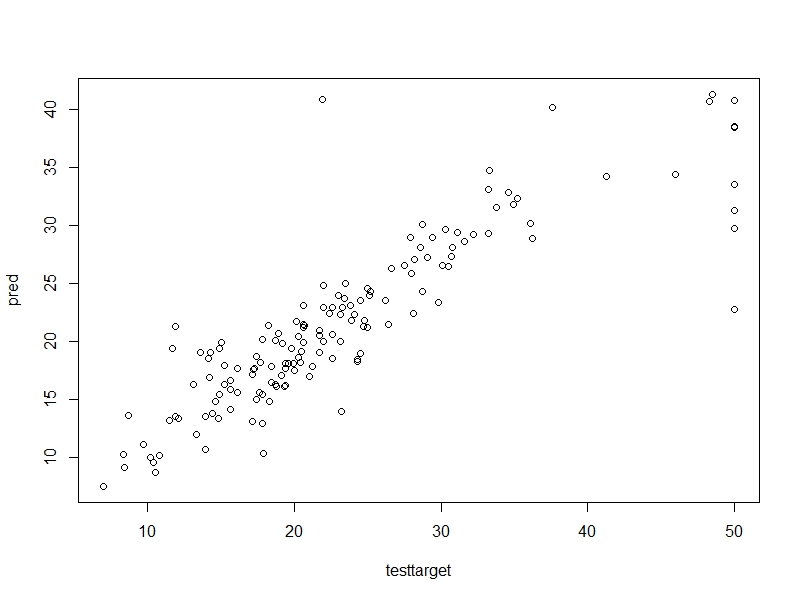
pred <- model %>% predict(test)

mean((testtarget-pred)^2)

> mean((testtarget-pred)^2)

[1] 26.64257

plot(testtarget, pred)



**Objetivo 5:**

Para esta parte del laboratorio utilizaremos el dataset “binary”, este cuenta con información de estudiantes que están postulando a una universidad donde la variable respuesta (admit) será 0 si no logró ingresar y 1 si sí lo logró. Las demás variables son las variables independientes que servirán para generar los clústeres en el objetivo 5 y las predicciones en el objetivo 6.

**# Mapas de Kohen**

**# Modelo no Supervisado**

**# Importar dataset**

library(kohonen)

data <- read.csv('binary.csv')

str(data)

X <- scale(data[,-1])

summary(X)

> summary(X)

gre gpa rank

Min. :-3.18309 Min. :-2.9690 Min. :-1.5723

1st Qu.:-0.58606 1st Qu.:-0.6829 1st Qu.:-0.5135

Median :-0.06666 Median : 0.0134 Median :-0.5135

Mean : 0.00000 Mean : 0.0000 Mean : 0.0000

3rd Qu.: 0.62588 3rd Qu.: 0.7360 3rd Qu.: 0.5453

Max. : 1.83783 Max. : 1.6031 Max. : 1.6041

Apréciese como todas las variables independientes ahora tienen media 0 por estar normalizadas.

**# Crear la cuadrícula de los mapas**

set.seed(1234)

g <- somgrid(xdim = 4, ydim = 4, topo = "rectangular" ) # Existirán 4\*4 = 16 nodos en el mapa

# Crear Mapas de Kohonen

map <- som(X, # Datos a utilizar en formato numérico

grid = g, # Cuadrícula predfinida

alpha = c(0.05, 0.01), # Tasa de aprendizaje

radius = 1) # Radio del vecindario

map$unit.classif

> map$unit.classif

[1] 8 10 12 4 1 7 13 3 5 11 14 13 12 7 12 5 14 15 12 2 5 11 1 4 6 12 2 9 6 13 9 4 5 14

[35] 13 3 13 15 16 15 15 16 16 5 7 5 16 1 1 3 10 1 4 6 5 14 5 3 8 1 16 9 10 10 10 11 4 6

[69] 2 12 10 1 9 11 4 14 5 14 13 2 4 7 15 1 8 15 16 16 6 11 11 12 12 7

De la instrucción anterior, vemos a que nodo pertenece cada aplicación de un estudiante, pudiendo estar entre los nodos 1 al 16. Concentrémonos en el cuarto valor, que es de 4, este está en el nodo 4.

head(data)

> head(data)

admit gre gpa rank

1 0 380 3.61 3

2 1 660 3.67 3

3 1 800 4.00 1

4 1 640 3.19 4

5 0 520 2.93 4

6 1 760 3.00 2

De la instrucción anterior, vemos los valores de las variables independientes de los 6 primeros estudiantes, concentrémonos en el cuarto estudiante (ver su nodo), este tiene valores altos de Rank (4), valores en el medio de gre (380) y valores bajos de gpa (3.61).

map$codes

> map$codes

[[1]]

gre gpa rank

V1 -0.9202516 -1.216066396 1.60409094

V2 -0.1663800 0.889279587 -1.57232676

V3 -1.8370357 -0.658111629 0.09880132

V4 0.9594603 -0.191861538 1.44394118

V5 -0.2269390 -0.300214117 0.54528504

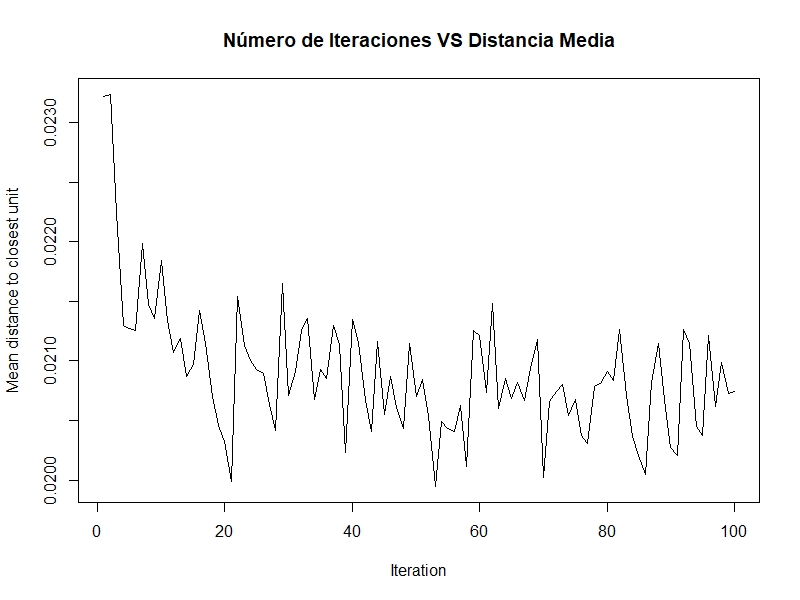
V6 0.8578956 -0.003674739 -0.81863377

De la instrucción anterior, se ven los valores que asumen cada uno de los 16 nodos (los primeros 6 se ven acá) en sus variables. El valor más alto para la variable significa que tendrá un pedazo de torta más grande en el nodo.

plot(map, type = 'changes', main= "Número de Iteraciones VS Distancia Media")

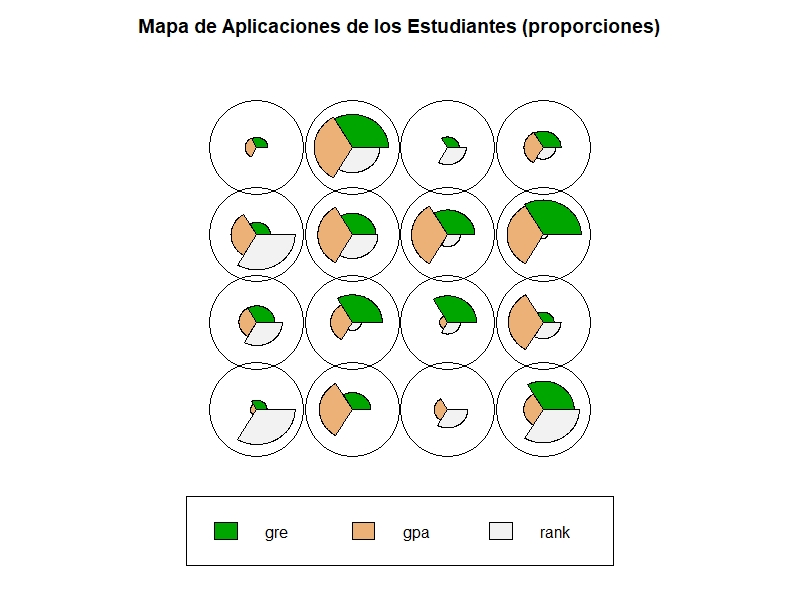
# eje y: distancia media al nodo más cercano

# eje x: número de iteraciones para crear el mapa



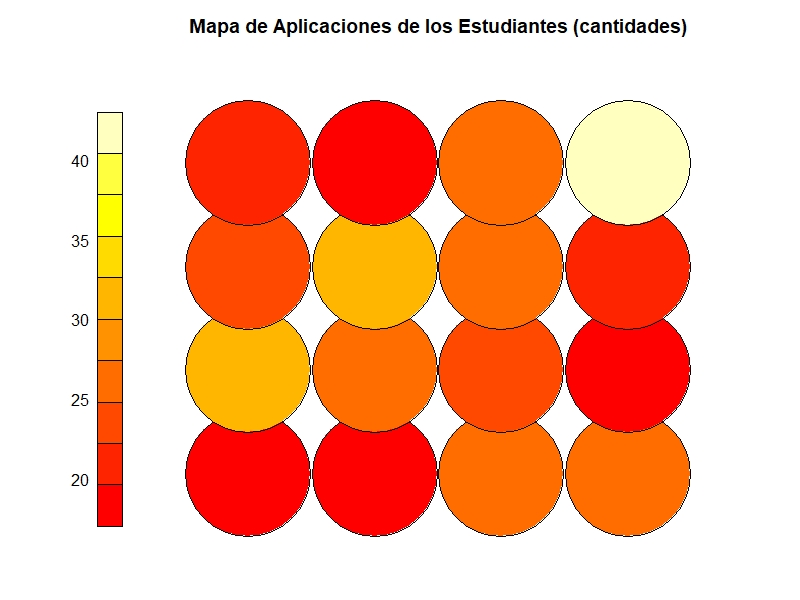
Del gráfico, apreciamos como la distancia promedio entre nodos más cercanos va disminuyendo a razón de que aumenta la cantidad de iteraciones del modelo, se han hecho 100 iteraciones, pero vemos que la media queda aproximadamente estable a partir de la iteración 50.

plot(map, type ='codes', main = 'Mapa de Aplicaciones de los Estudiantes (proporciones)')



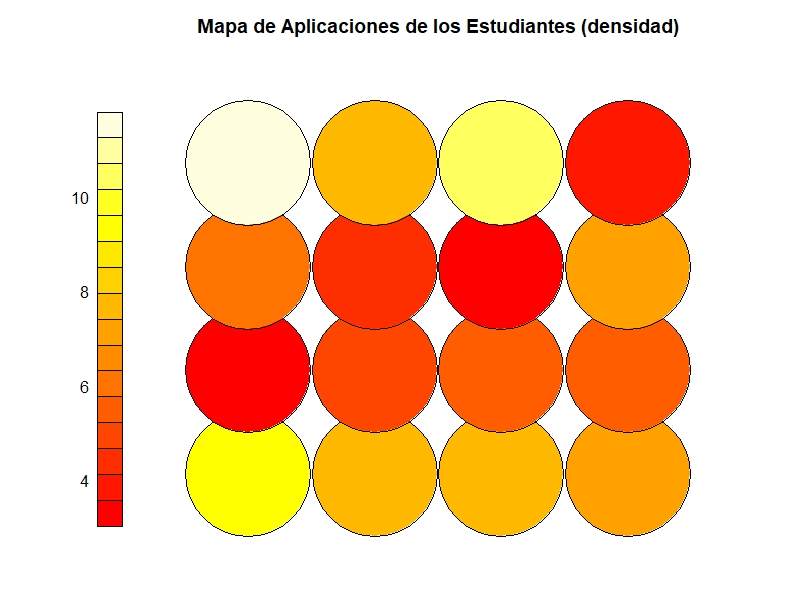
Del gráfico, apreciamos como cada nodo se está definiendo según la cantidad de valores de cada variable, por ejemplo, el nodo que está en la tercera fila (contando desde abajo) y segunda columna (contando desde la izquierda), nodo número 10, tiene mayores valores en “gpa”, seguido por los valores de “gre” y finalmente por los de “rank”.

plot(map, type = 'count', main = 'Mapa de Aplicaciones de los Estudiantes (cantidades)')



Del gráfico, cada nodo muestra la cantidad de estudiantes que tiene por colores, siendo más densos en cantidad los que tienen color más claro y menos densos los de color más oscuro.

plot(map, type = 'dist.neighbours', main = 'Mapa de Aplicaciones de los Estudiantes (densidad)')



Del gráfico, vemos como se distribuyen las aplicaciones de los estudiantes en cada nodo, siendo más cercanos entre ellos los nodos más oscuros, pues hay menor distancia promedio entre los puntos, y los más lejanos entre ellos son los que tienen los nodos más claros, pues existe mayor distancia promedio entre los puntos.

**Objetivo 6:**

Ahora, nuevamente con nuestro dataset “binary”, realizaremos predicciones de clasificación de ingreso tomando en cuenta la variable respuesta (admit) con un Mapa de Kohen, pero esta vez nos interesa contrarrestar contra la variable respuesta, es decir, será un modelo supervisado.

**# Mapas de Kohen**

**# Modelo Supervisado**

**# Importar dataset**

library(kohonen)

data <- read.csv('binary.csv')

str(data)

X <- scale(data[,-1])

summary(X)

**# Generar conjunto de entrenamiento y de testeo**

set.seed(1234)

ind <- sample(2, nrow(data), replace = T, prob = c(0.7, 0.3))

train <- data[ind == 1,]

test <- data[ind == 2,]

**# Normalizar las variables independientes apra ambos conjuntos**

trainX <- scale(train[,-1])

testX <- scale(test[,-1],

center = attr(trainX, "scaled:center"),

scale = attr(trainX, "scaled:scale")) # Escalamos test con la media y desv est de train

trainY <- factor(train[,1])

Y <- factor(test[,1])

test[,1] <- 0

testXY <- list(independent = testX, dependent = test[,1])

**# Generar modelo de clasificación**

set.seed(1234)

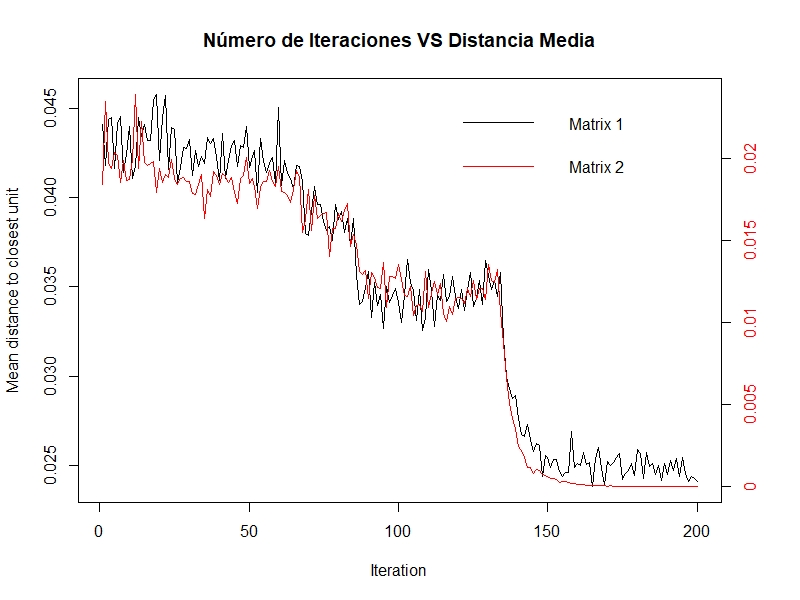
map1 <- xyf(trainX,

classvec2classmat(factor(trainY)),

grid = somgrid(5, 5, "hexagonal"),

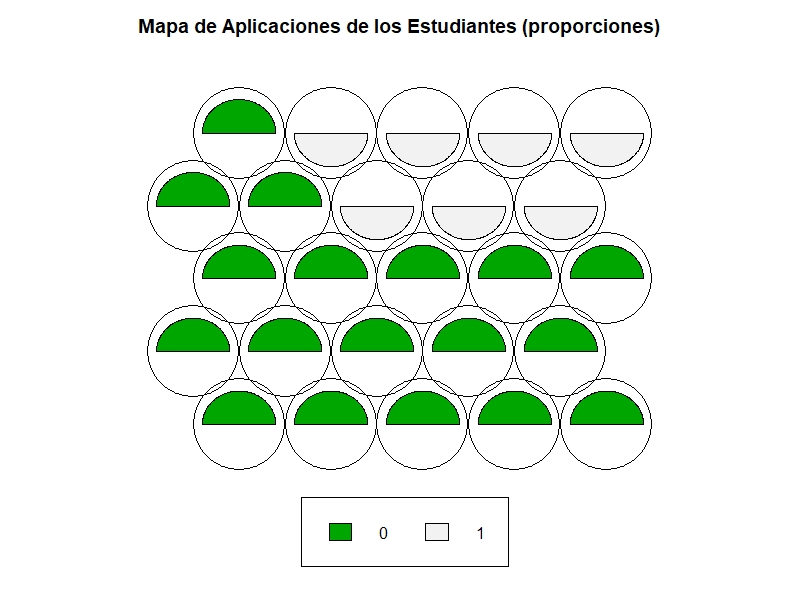
rlen = 200)

plot(map1, type = 'changes', main= "Número de Iteraciones VS Distancia Media")



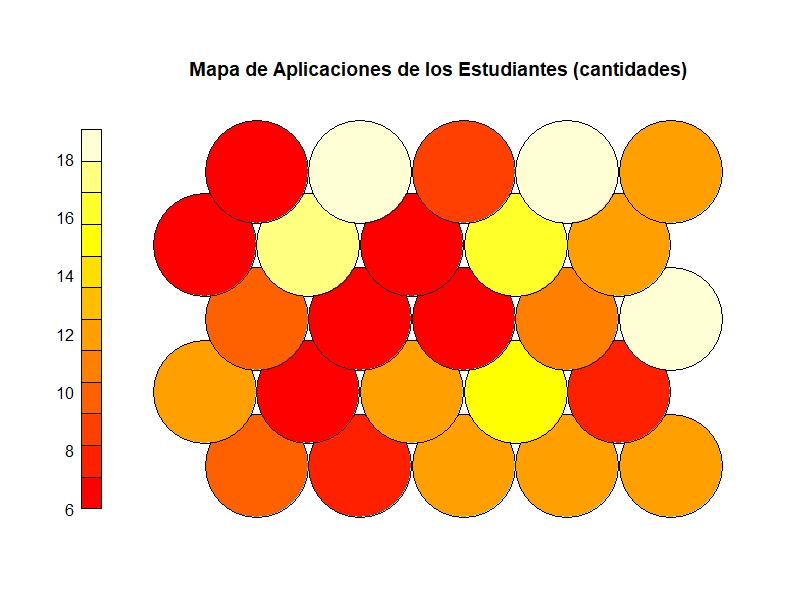
Del gráfico, apreciamos como la distancia media entre nodos va disminuyendo con el paso de las iteraciones, tanto para las variables independientes (color negro), como para la variable dependiente (color rojo).

plot(map1, type ='codes', main = 'Mapa de Aplicaciones de los Estudiantes (proporciones)')



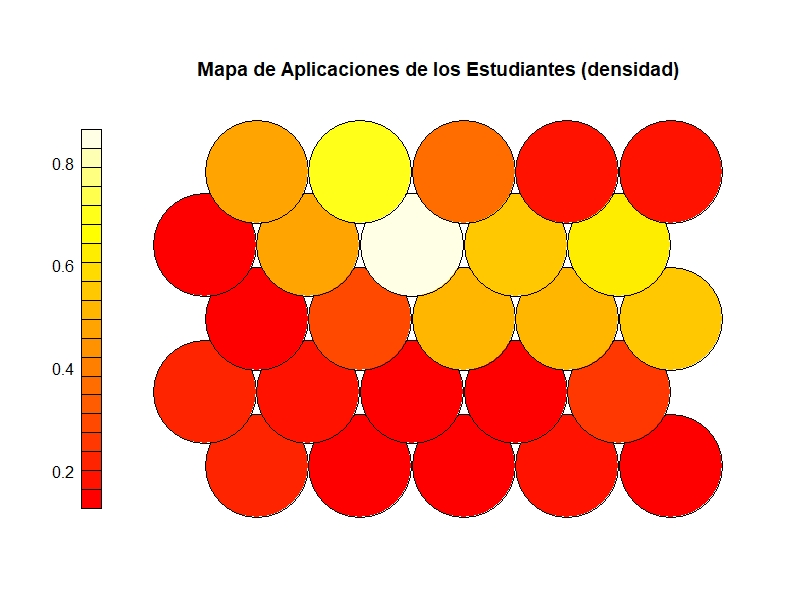
Del gráfico, vemos la distribución de la variable dependiente con los 25 nodos, los nodos de color verde se corresponden con los no admitidos y los nodos de color blanco con los admitidos.

plot(map1, type = 'count', main = 'Mapa de Aplicaciones de los Estudiantes (cantidades)')



Del gráfico, cada nodo muestra la cantidad de estudiantes que tiene por colores, siendo más densos en cantidad los que tienen color más claro y menos densos los de color más oscuro.

plot(map1, type = 'dist.neighbours', main = 'Mapa de Aplicaciones de los Estudiantes (densidad)')



Del gráfico, vemos como se distribuyen las aplicaciones de los estudiantes en cada nodo, siendo más cercanos entre ellos los nodos más oscuros, pues hay menor distancia promedio entre los puntos, y los más lejanos entre ellos son los que tienen los nodos más claros, pues existe mayor distancia promedio entre los puntos.

**# Generar predicción y matriz de confusión**

pred <- predict(map1, newdata = testXY)

> pred

$predictions[[2]]

[1] 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 1 1 1

[52] 0 1 1 0 0 1 0 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0

[103] 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Levels: 0 1

tabla = table(Predicted = pred$predictions[[2]], Actual = Y)

> tabla

Actual

Predicted 0 1

0 56 23

1 26 11

**# Exactitud del modelo (ratio de clasificación)**

ratio= sum(diag(tabla))/sum(tabla)

> ratio

[1] 0.5775862

Podemos concluir, que nuestro modelo de Mapas de Kohen consiguió clasificar correctamente al 57.75%, si se desea mejorar el modelo, queda como tarea aprender a manipular más los parámetros de la función “xyf”.