Modelos NN

Jimena Murillo

2022-05-05

#Paquetes

```
library(keras) # for deep learning
library(tidyverse) # general utility functions
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret) # machine learning utility functions
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tibble)
library(readr)
library(ggplot2)
library(tensorflow)
```

```
##
## Attaching package: 'tensorflow'

## The following object is masked from 'package:caret':
##
##     train
```

#Construir una base con el cantón de Alajuela y partirla en train y test

```r
load("C:/Users/usuario1/Desktop/CIMPA/Github_CIMPA/PRACTICA_CIMPA/base_cantones.RData")
```

```r
Alajuela1 <- basecanton %>% filter(Canton == "Alajuela")

Alajuela1 <- Alajuela1%>%
  dplyr::select(Year,Month,Nino12SSTA,Nino3SSTA, Nino4SSTA, Nino34SSTA,TNA,EVI,NDVI,NDWI,LSD,LSN,Precip_

str(Alajuela1)
```

```
## tibble [235 x 14] (S3: tbl_df/tbl/data.frame)
##  $ Year      : num [1:235] 2001 2001 2001 2001 2002 ...
##  $ Month     : Factor w/ 12 levels "1","2","3","4",..: 9 10 11 12 1 2 3 4 5 6 ...
##  $ Nino12SSTA: num [1:235] -1.13 -1.13 -0.89 -1.02 -0.78 -0.07 0.77 0.82 0.63 0.38 ...
##  $ Nino3SSTA : num [1:235] -0.59 -0.43 -0.68 -0.62 -0.49 -0.24 0.14 0 0.26 0.57 ...
##  $ Nino4SSTA : num [1:235] 0.21 0.19 0.08 0.12 0.4 0.46 0.25 0.41 0.6 0.64 ...
##  $ Nino34SSTA: num [1:235] -0.2 -0.14 -0.37 -0.41 -0.15 -0.04 0.01 0.02 0.31 0.72 ...
##  $ TNA       : num [1:235] 0.51 0.48 0.62 0.66 0.78 0.53 0.37 -0.02 -0.19 -0.13 ...
##  $ EVI       : num [1:235] 0.297 0.297 0.291 0.307 0.274 ...
##  $ NDVI      : num [1:235] 0.532 0.501 0.509 0.523 0.497 ...
##  $ NDWI      : num [1:235] 0.404 0.415 0.372 0.366 0.349 ...
##  $ LSD       : num [1:235] 241 278 202 299 303 ...
##  $ LSN       : num [1:235] 109.3 99.6 147.6 289.6 290.4 ...
##  $ Precip_t  : num [1:235] 4308 5559 2477 1430 944 ...
##  $ RR        : num [1:235] 0.3655 0.3775 0.0996 0.1023 0.7962 ...
```

```r
Alajuela1 = Alajuela1 %>% arrange(Year,Month) %>% ungroup() %>%
  mutate(Month=as.numeric(Month))




if(anyNA(Alajuela1)){
  Alajuela1 <- na.omit(Alajuela1)
}

#Escala


normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

max <- apply(Alajuela1,2,max)
min <- apply(Alajuela1,2,min)

Alajuela1.2 <- apply(Alajuela1, 2, normalize)


#Train y test
```

```r
data_train1 = as.data.frame(Alajuela1.2) %>% filter(Year < 0.85)#PARA ENTRENAR HASTA 2018
data_test1 = as.data.frame(Alajuela1.2) %>% filter(Year >= 0.85)

X_train1 = as.matrix(data_train1[,-ncol(data_train1)])
y_train1 = as.matrix(data_train1[,ncol(data_train1)])

X_test1 = as.matrix(data_test1[,-ncol(data_test1)])
y_test1 = as.matrix(data_test1[,ncol(data_test1)])
```

#Datos con lag

```r
Alajuela <- basecanton %>% filter(Canton == "Alajuela") %>%

  dplyr::select(Year,Month,Nino12SSTA, Nino3SSTA, Nino4SSTA,Nino34SSTA,Nino34SSTA1, Nino34SSTA2, Nino34S

  arrange(Year,Month) %>% ungroup() %>% mutate(Month=as.numeric(Month))


if(anyNA(Alajuela)){
  Alajuela <- na.omit(Alajuela)
}

#Escala


normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

Alajuela2 <- apply(Alajuela, 2, normalize)


#Train y test

data_train = as.data.frame(Alajuela2) %>% filter(Year < 0.85)#PARA ENTRENAR HASTA 2018
data_test = as.data.frame(Alajuela2) %>% filter(Year >= 0.85)

X_train = as.matrix(data_train[,-ncol(data_train)])
y_train = as.matrix(data_train[,ncol(data_train)])

X_test = as.matrix(data_test[,-ncol(data_test)])
y_test = as.matrix(data_test[,ncol(data_test)])
```

#Modelo inicial simple

```r
set.seed(123)
model <- keras_model_sequential()
```

```
## Loaded Tensorflow version 2.8.0
```

```
# our input layer
model %>%
  layer_dense(input_shape = ncol(X_train1), units = 13) %>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  dense_1 (Dense)                    (None, 13)                       182
##
##  dense (Dense)                      (None, 1)                        14
##
## ================================================================================
## Total params: 196
## Trainable params: 196
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model %>% compile(loss = "mse",
                  optimizer = "adam",
                  metric = "mae")

trained_model <- model %>% fit(
  x = X_train1, # sequence we're using for prediction
  y = y_train1, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 80, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model %>% evaluate(X_test1, y_test1)
```

```
##      loss       mae
## 0.0295292 0.1481674
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela1,2,max)
min <- apply(Alajuela1,2,min)

results = model %>% predict(X_test1)
results = denorm(results, max[length(Alajuela1)], min[length(Alajuela1)])

data = cbind(results, Alajuela1[197:235,length(Alajuela1)])
```

```
names(data) = c("Resultados", "RR")


Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue")
p <- p +  geom_line(
    aes(x = Mes, y = Resultados),
    colour = "red")

print(p)
```

#Segundo modelo, se agrega 1 capa

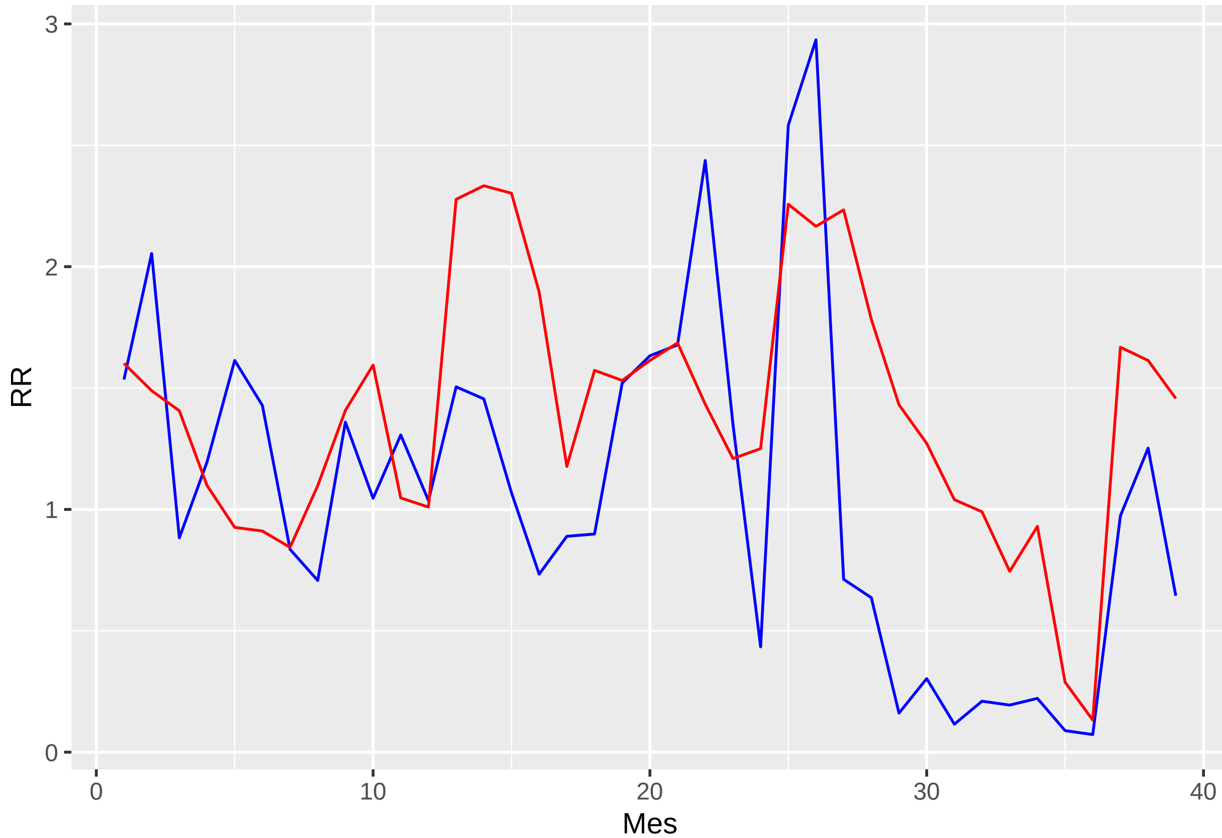
```
set.seed(123)
model2 <- keras_model_sequential()
# our input layer
model2 %>%
  layer_dense(input_shape = ncol(X_train1), units = 13) %>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model2)
```

```
## Model: "sequential_1"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  dense_4 (Dense)                    (None, 13)                       182
##
##  dense_3 (Dense)                    (None, 8)                        112
##
##  dense_2 (Dense)                    (None, 1)                        9
##
## ================================================================================
## Total params: 303
## Trainable params: 303
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model2 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model2 <- model2 %>% fit(
  x = X_train1, # sequence we're using for prediction
  y = y_train1, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 60, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model2 %>% evaluate(X_test1, y_test1)
```

```
##                loss mean_absolute_error
##          0.01759348          0.10696158
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela1,2,max)
min <- apply(Alajuela1,2,min)

results = model2 %>% predict(X_test1)
results = denorm(results, max[length(Alajuela1)], min[length(Alajuela1)])

data = cbind(results, Alajuela1[197:nrow(Alajuela1),length(Alajuela1)])
names(data) = c("Resultados", "RR")


Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



En este modelo se observa una reducción del error cuadrado medio a 0.01.Sin embargo, al graficar se observa una mala predicción

#Modelo con datos con lag

#Preparar datos:

#NN creada con las nuevas variables lag, se ajusta el dropout, y unidades a lo que generó mejores resultados.

```
set.seed(123)
model3 <- keras_model_sequential()
# our input layer
model3 %>%
  layer_dense(input_shape = ncol(X_train), units = 32) %>%
  layer_dropout(rate = 0.2)%>%
  layer_dense(units = 16, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model3)
```

```
## Model: "sequential_2"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
```

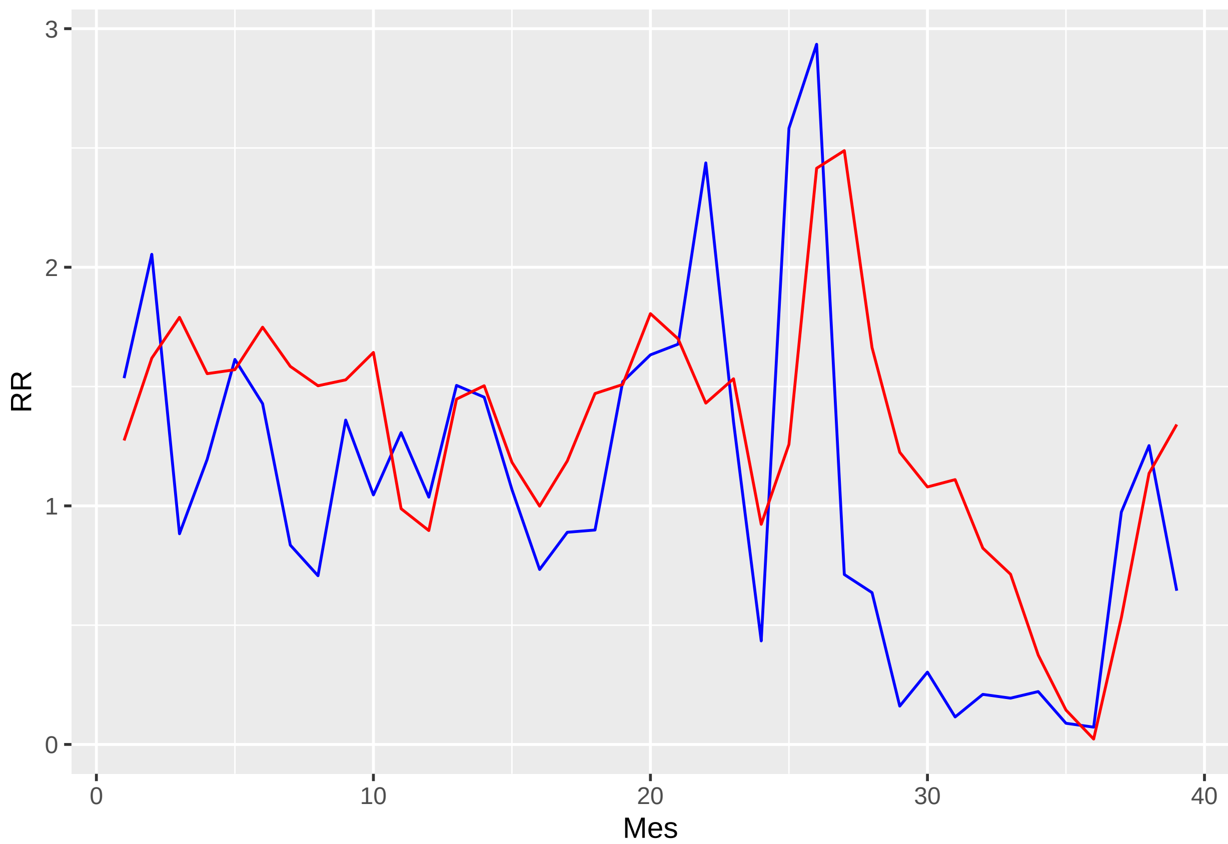
```
## ================================================================================
##  dense_7 (Dense)                    (None, 32)                     1056
##
##  dropout (Dropout)                  (None, 32)                     0
##
##  dense_6 (Dense)                    (None, 16)                     528
##
##  dense_5 (Dense)                    (None, 1)                      17
##
## ================================================================================
## Total params: 1,601
## Trainable params: 1,601
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model3 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model3 <- model3 %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 130, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model3 %>% evaluate(X_test, y_test)
```

```
##                loss mean_absolute_error
##          0.01418507          0.09042546
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model3 %>% predict(X_test)
results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

#Escala

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line(aes(x = Mes, y = Resultados),colour = "red")
```

```
print(p)
```



#Se construye un modelo con rnn

```
model5 <- keras_model_sequential()
# our input layer
model5 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train),1), activation='relu') %>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 12, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model5)
```

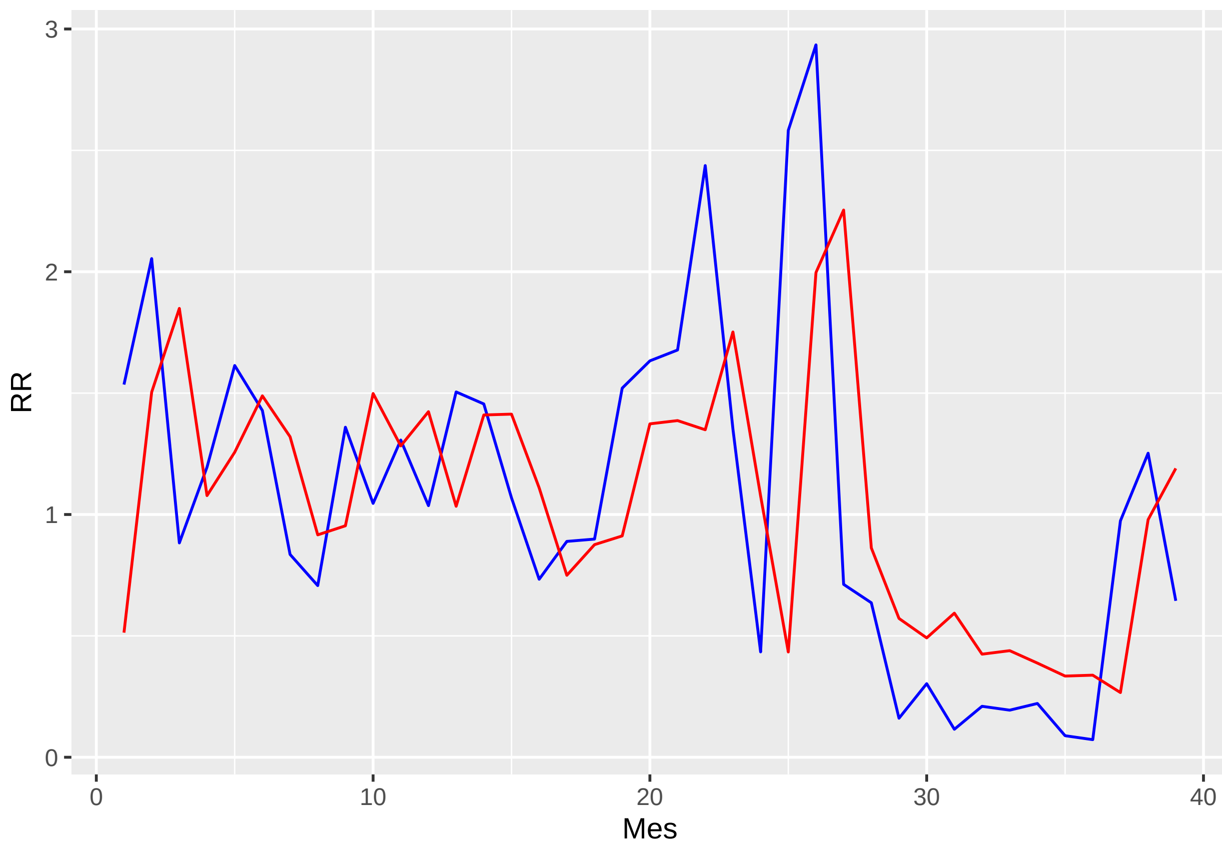
```
## Model: "sequential_3"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  simple_rnn (SimpleRNN)             (None, 24)                        624
##
##  dropout_1 (Dropout)                (None, 24)                        0
##
##  dense_9 (Dense)                    (None, 12)                        300
```

```
##
##  dense_8 (Dense)                    (None, 1)                      13
##
## ================================================================================
## Total params: 937
## Trainable params: 937
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model5 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model5 <- model5 %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.2,
  shuffle = F) # how much data to hold out for testing as we go along

model5 %>% evaluate(X_test, y_test)
```

```
##                  loss mean_absolute_error
##            0.01450328          0.08973631
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model5 %>% predict(X_test)

results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```


RR
3
2
1
0
0
10
20
30
40
Mes