Model 2 Network-based Classification Model

Noel C. Sieras

2022-12-16

# Preliminaries This will prevent some errors in loading some of the chunks and laoding of the dataset.

# Network-based Classification Model
## Load Helper Packages

```
##
## 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

## Warning: package 'ggplot2' was built under R version 4.2.2

## Warning: package 'stringr' was built under R version 4.2.2

##
## Attaching package: 'gridExtra'

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

## Warning: package 'bestNormalize' was built under R version 4.2.2

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8     v purrr   0.3.5
## v tidyr   1.2.1     v forcats 0.5.2
## v readr   2.1.3
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x gridExtra::combine() masks dplyr::combine()
## x dplyr::filter()      masks stats::filter()
## x dplyr::lag()         masks stats::lag()

## Warning: package 'factoextra' was built under R version 4.2.2

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

## Warning: package 'mclust' was built under R version 4.2.2

## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
##
```

```
## Attaching package: 'mclust'
##
## The following object is masked from 'package:purrr':
##
##     map
##
## 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
```

```
## Warning: package 'rsample' was built under R version 4.2.2
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
##
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
## Warning: package 'tfruns' was built under R version 4.2.2
```

```
## Warning: package 'tensorflow' was built under R version 4.2.2
```

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

```
## Warning: package 'tfestimators' was built under R version 4.2.2
```

```
## tfestimators is not recomended for new code. It is only compatible with Tensorflow version 1, and is
##
## Attaching package: 'tfestimators'
##
## The following object is masked from 'package:caret':
##
##     train
```

# Loading of Data Set

Radiomics data contains 197 rows and 431 columns: **Failure.binary**: binary property to predict
## Use `set.seed()` for reproducibility

```
# for reproducibility
set.seed(12345)
```

## Setting up a working directory

## **Importing the normalized dataset normalRAd.csv** The dataset used in this model is a normalized data `normalRad.csv` which was obtain from `radiomics_complete.csv` through pre-processing technuque. It has 197 observations and 431 variables.

```
datard <- read_csv("normalRad.csv", show_col_types = FALSE)
dim(datard)
```

```
## [1] 197 431
```

Splitting

Split the data into training (80%) and testing (30%).

```
datard_n<-datard %>%
  mutate(Failure.binary=ifelse(Failure.binary== "No",0,1))

set.seed(123)
rdsplit = initial_split(datard_n, prop = 0.8, strata = "Failure.binary")
rdtrain <- training(rdsplit)
rdtest  <- testing(rdsplit)

train1 <- rdtrain[,-c(1,2)]%>%as.matrix.data.frame()
train2 <- rdtrain$Failure.binary
test1 <- rdtest[,-c(1,2)]%>%as.matrix.data.frame()
test2 <- rdtest$Failure.binary
```

Reshaping the dataset

```
train1 <- array_reshape(train1, c(nrow(train1), ncol(train1)))
train1 <- train1

test1 <- array_reshape(test1, c(nrow(test1), ncol(test1)))
test1 <- test1

train2 <- to_categorical(train2, num_classes = 2)
```

```
## Loaded Tensorflow version 2.9.2
```

```
test2 <- to_categorical(test2, num_classes = 2)
```

Run the model

```
modeldl <- keras_model_sequential() %>%

   # Network architecture
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(ncol(train1))) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 128, activation = "sigmoid") %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 128, activation = "sigmoid") %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 64, activation = "sigmoid") %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 64, activation = "sigmoid") %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 2, activation = "softmax") %>%

# Backpropagation
 compile(
    loss = "categorical_crossentropy",
    optimizer = optimizer_rmsprop(),
    metrics = c("accuracy")
  )
modeldl
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_5 (Dense)                    (None, 256)                     110080
##  dropout_4 (Dropout)                (None, 256)                     0
##  dense_4 (Dense)                    (None, 128)                     32896
##  dropout_3 (Dropout)                (None, 128)                     0
##  dense_3 (Dense)                    (None, 128)                     16512
##  dropout_2 (Dropout)                (None, 128)                     0
##  dense_2 (Dense)                    (None, 64)                      8256
##  dropout_1 (Dropout)                (None, 64)                      0
##  dense_1 (Dense)                    (None, 64)                      4160
##  dropout (Dropout)                  (None, 64)                      0
##  dense (Dense)                      (None, 2)                       130
## ================================================================================
## Total params: 172,034
## Trainable params: 172,034
## Non-trainable params: 0
## ________________________________________________________________________________
```

Trained the model

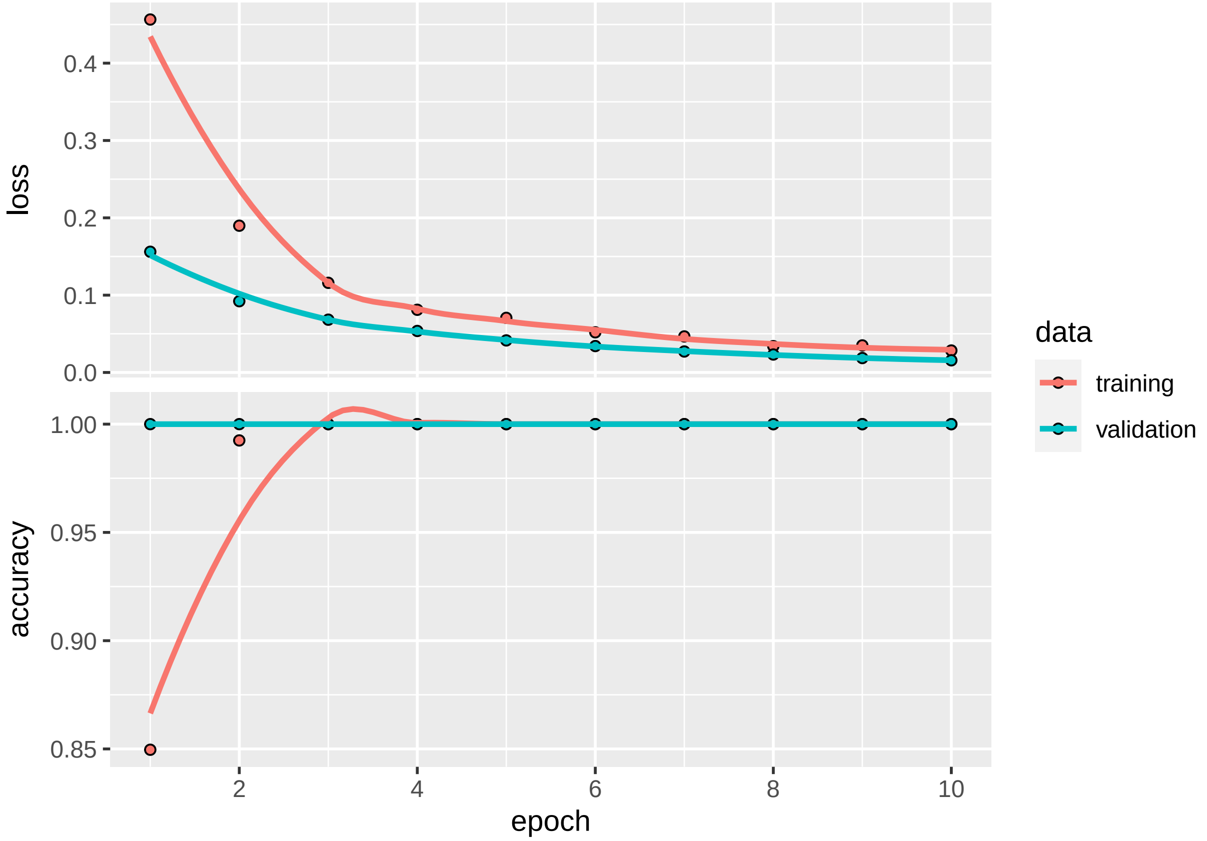
```
#trained model history
fitdl <- modeldl %>%
  fit(train1, train2,
      epochs = 10,
      batch_size = 128,
      validation_split = 0.15)

# Display output
fitdl
```

```
##
## Final epoch (plot to see history):
```

```
##         loss: 0.02841
##     accuracy: 1
##     val_loss: 0.01575
## val_accuracy: 1
```

```
#plot the training and validation performance over 10 epochs
plot(fitdl)
```



Evaluate the trained model using testing dataset

```
modeldl %>%
  evaluate(test1, test2)
```

```
##       loss   accuracy
## 0.01549289 1.00000000
```

```
dim(test1)
```

```
## [1]  40 429
```

```
dim(test2)
```

```
## [1] 40  2
```

Model prediction using testing dataset

```
modeldl %>%
  predict(test1) %>% `>`(0.5) %>% k_cast("int32")
```

```
## tf.Tensor(
## [[0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]
##  [0 1]], shape=(40, 2), dtype=int32)
```