MODEL 2 - NEURAL NETWORK-BASED CLASSIFICATION MODEL

FENDAWN F. RECENTES

2022-12-13

Helper Packages AND Model Packages

```
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(keras)
library(tfruns)
library(rsample)
library(tfestimators)
```

```
## tfestimators is not recomended for new code. It is only compatible with Tensorflow version 1, and is
```

```
library(readr)
library(tensorflow)
library(bestNormalize)
library(tidyverse)
```

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

## v ggplot2 3.4.0     v purrr   0.3.5
## v tibble  3.1.8     v stringr 1.4.1
## v tidyr   1.2.1     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

We use the normalize radiomatics dataset here.

Load and view radiomics dataset

```
radiomics = read_csv("RAD. NORMAL DATA.CSV")
```

```
## Rows: 197 Columns: 431
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (1): Institution
## dbl (430): Failure.binary, Failure, Entropy_cooc.W.ADC, GLNU_align.H.PET, Mi...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(radiomics)
```

```
## # A tibble: 6 x 431
##   Institution Failure.~1 Failure Entro~2 GLNU_~3 Min_h~4 Max_h~5 Mean_~6 Varia~7
##   <chr>            <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 A                    0   1.15     12.9  -0.433  -0.270 -0.257   -0.192  0.0509
## 2 A                    1  -0.533    12.2  -1.02    0.671  0.405    0.490  0.687
## 3 A                    0   2.24     12.8   0.179  -1.41  -1.57    -1.53  -1.57
## 4 A                    1  -0.140    13.5   2.00   -0.218  0.0764  -0.153  0.0127
## 5 A                    0   0.787    12.6   0.153  -1.06  -1.15    -1.45  -1.91
## 6 A                    1  -2.80     13.2   0.391  -1.57  -1.91    -1.72  -1.84
## # ... with 422 more variables: Standard_Deviation_hist.PET <dbl>,
## #   Skewness_hist.PET <dbl>, Kurtosis_hist.PET <dbl>, Energy_hist.PET <dbl>,
## #   Entropy_hist.PET <dbl>, AUC_hist.PET <dbl>, H_suv.PET <dbl>,
## #   Volume.PET <dbl>, X3D_surface.PET <dbl>, ratio_3ds_vol.PET <dbl>,
## #   ratio_3ds_vol_norm.PET <dbl>, irregularity.PET <dbl>,
## #   tumor_length.PET <dbl>, Compactness_v1.PET <dbl>, Compactness_v2.PET <dbl>,
## #   Spherical_disproportion.PET <dbl>, Sphericity.PET <dbl>, ...
```

Split the data into training (80) and testing (20).

```
df <- radiomics %>%
  mutate(Failure.binary=ifelse(Failure.binary== "No",0,1))
df=df[,-1]
set.seed(123)
split = initial_split(df,prop = 0.8 ,strata = "Failure.binary")
radiomics_train <- training(split)
radiomics_test  <- testing(split)

#or

X_train <- radiomics_train[,-c(1,2)]%>%as.matrix.data.frame()
X_test <- radiomics_test[,-c(1,2)]%>%as.matrix.data.frame()
y_train <- radiomics_train$Failure.binary
y_test <- radiomics_test$Failure.binary
```

The model will have five hidden layers with 256, 128, 128, 64 and 64 neurons with activation functions of Sigmoid. The output layer will have 2 neurons for predicting a numeric value and a Softmax activation fuction. Every layer is followed by a dropout to avoid overfitting.

Reshaping the dataset

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

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

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

```
## Loaded Tensorflow version 2.9.3
```

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

model <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(ncol(X_train))) %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 128, activation = "sigmoid") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 128, activation = "sigmoid") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 64, activation = "sigmoid") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 64, activation = "sigmoid") %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 2, activation = "softmax")%>%
 compile(
    loss = "categorical_crossentropy",
    optimizer = optimizer_rmsprop(),
    metrics = c("accuracy")
  )
```

The model will be trained to minimize the categorical_crossentropy loss function using the effective Adam version of stochastic gradient descent.

We will train the model for 10 epochs with a batch size of 128 samples and validation split of 0.15

```
 model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(),
  metrics = c("accuracy")
)

history <- model %>%
  fit(X_train, y_train, epochs = 10, batch_size = 128, validation_split = 0.15)
```

After the model is trained, we will evaluate it on the holdout test dataset

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

```
##        loss    accuracy
## 0.004125958 1.000000000
```

```
dim(X_test)
```

```
## [1]  40 428
```

```
dim(y_test)
```

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

Finally, model prediction using the testing dataset

```
model %>% predict(X_test) %>% `>`(0.8) %>% 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)
```