Model 2

Ying Luo

2022-12-16

Helper packages

```
library(keras)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(rsample)
library(recipes)
```

```
## Loading required package: 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

##
## Attaching package: 'recipes'

## The following object is masked from 'package:stats':
##
##     step
```

Load the dataset

```
library(readr)
df = read.csv("radiomics_completedata.csv")
```

Investigate the statistics of the dataset Use eval=FALSE to save pages when knitting

"'{r} eval=FALSE

summary(df)

```
Remove NA

‘‘‘r
df <- na.omit(df)
```

Investigate the cleaned data Use eval=FALSE to save pages when knitting

“‘{r} eval=FALSE

head(df)

```
Split the training data and testing data by 7 : 3
Extract the features and labels

‘‘‘r
index<-createDataPartition(df$Failure.binary,p=0.7,list=F)

x_train <- data.matrix(df[index,-2])
y_train <- df[index,2]
x_test <- data.matrix(df[-index,-2])
y_test <- df[-index,2]
```

Convert features (x) to matrix and labels (y) to the binary variable

```r
as.matrix(apply(x_train, 2, function(x) (x-min(x))/(max(x) - min(x)))) ->
  x_train

as.matrix(apply(x_test, 2, function(x) (x-min(x))/(max(x) - min(x)))) ->
  x_test

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

```
## Loaded Tensorflow version 2.10.0
```

```r
to_categorical(y_test, num_classes = 2) -> y_test
```

Create five hidden layers with 256, 128, 128, 64 and 64 neurons, respectively with activation functions of Sigmoid Create an output layer with two neurons respectively with activation functions of Softmax Every layer is followed by a dropout to avoid overfitting

```r
model <- keras_model_sequential()

model %>%
  layer_dense(units=256,activation = "sigmoid",input_shape =ncol(x_train))%>%
  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 compiler approach

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

Adam compiler approach

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

Train the model with epoch = 10, batch size = 128 and validation split = 0.15

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

Evaluate the trained model using the testing dataset.

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

```
##      loss  accuracy
## 0.6614779 0.6271186
```

Get the model prediction using the testing dataset

```
model %>%
  predict(x_test)
```

```
##             [,1]      [,2]
##   [1,] 0.6475127 0.3524873
##   [2,] 0.6476544 0.3523456
##   [3,] 0.6474760 0.3525240
##   [4,] 0.6475829 0.3524171
##  [5,] 0.6475290 0.3524710
##  [6,] 0.6475245 0.3524755
##  [7,] 0.6475614 0.3524387
##  [8,] 0.6476569 0.3523432
##  [9,] 0.6474912 0.3525088
## [10,] 0.6475025 0.3524975
## [11,] 0.6476278 0.3523721
## [12,] 0.6475673 0.3524327
## [13,] 0.6478765 0.3521236
## [14,] 0.6476158 0.3523842
## [15,] 0.6476370 0.3523630
## [16,] 0.6476403 0.3523597
## [17,] 0.6475942 0.3524057
## [18,] 0.6476069 0.3523931
## [19,] 0.6476684 0.3523316
## [20,] 0.6478441 0.3521559
## [21,] 0.6476021 0.3523979
## [22,] 0.6476404 0.3523595
## [23,] 0.6474360 0.3525640
## [24,] 0.6475330 0.3524670
## [25,] 0.6475335 0.3524665
## [26,] 0.6475841 0.3524158
## [27,] 0.6474990 0.3525010
## [28,] 0.6472558 0.3527442
## [29,] 0.6472906 0.3527093
## [30,] 0.6472238 0.3527763
## [31,] 0.6473337 0.3526664
## [32,] 0.6473631 0.3526369
## [33,] 0.6473601 0.3526398
## [34,] 0.6476451 0.3523550
## [35,] 0.6477299 0.3522701
## [36,] 0.6473169 0.3526831
## [37,] 0.6472846 0.3527154
## [38,] 0.6467100 0.3532901
## [39,] 0.6466262 0.3533739
## [40,] 0.6467538 0.3532462
## [41,] 0.6468188 0.3531813
## [42,] 0.6488791 0.3511208
## [43,] 0.6489053 0.3510947
## [44,] 0.6488628 0.3511372
## [45,] 0.6488500 0.3511500
## [46,] 0.6489386 0.3510614
## [47,] 0.6488544 0.3511456
## [48,] 0.6490096 0.3509903
## [49,] 0.6488457 0.3511543
## [50,] 0.6489660 0.3510340
## [51,] 0.6488485 0.3511515
## [52,] 0.6490608 0.3509393
## [53,] 0.6490220 0.3509780
## [54,] 0.6488227 0.3511773
## [55,] 0.6487956 0.3512044
## [56,] 0.6487532 0.3512468
## [57,] 0.6485382 0.3514618
## [58,] 0.6485170 0.3514830
## [59,] 0.6486208 0.3513792
```