STATISTICAL COMPUTING FINAL PROJECT

Jobert Jay R. Pandan

2022-12-13

DEEP LEARNING MODEL: Network-based classification model

Needed Libraries

```
library(dplyr)
library(keras)
library(tfruns)
library(rsample)
library(tfestimators)
library(readr)
library(tensorflow)
pacman::p_load(tidyverse)
pacman::p_load(bestNormalize)
```

DATA ACQUISITION and PRE-PROCESSING

The data used in this model undergo normalization, same with previous models in model 1.

```
data=read_csv("radiomics_completedata.csv")
data$Failure.binary = as.factor(data$Failure.binary)
newdf1 = data %>% select_if(is.numeric)
tempDF=apply(newdf1,2,orderNorm)
tempDF=lapply(tempDF, function(x) x$x.t)
tempDF=tempDF%>%as.data.frame()
norm_data = cbind(data[c('Failure.binary')], tempDF)
```

DATA SPLITTING AND PREPARATION

splitting the data into testing and training set. We train the model from training set and a we test the performance in testing set.

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

set.seed(123)
norm_data_split = initial_split(norm_data,prop = 0.8 ,strata = "Failure.binary")
data_train <- training(norm_data_split)
data_test  <- testing(norm_data_split)

#or

trainset1 <- data_train[,-c(1,2)]%>%as.matrix.data.frame()
testset1 <- data_test[,-c(1,2)]%>%as.matrix.data.frame()
trainset2<- data_train$Failure.binary
testset2 <- data_test$Failure.binary
```

#reshaping the dataset

Reshaping the dataset helps to scale the data.

```
reshape_train <- array_reshape(trainset1, c(nrow(trainset1), ncol(trainset1)))


reshape_test <- array_reshape(testset1, c(nrow(testset1), ncol(testset1)))

cat_train <- to_categorical(trainset2, num_classes = 2)
cat_test <- to_categorical(testset2, num_classes = 2)
```

MODELLING

```
model2 <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(ncol(trainset1))) %>%
  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")
model2 %>% # Backpropagation
  compile(
    loss = "categorical_crossentropy",
    optimizer = optimizer_rmsprop(),
    metrics = c("accuracy")
  )
```

Compile the Model

```
 model2 %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(),
  metrics = c("accuracy")
)

history <- model2 %>%
  fit(trainset1, cat_train, epochs = 10, batch_size = 128, validation_split = 0.15)
```

#Evaluate the trained model

Evaluating the trained model to testing data.

```
model2 %>%
  evaluate(testset1, cat_test)
```

```
##       loss   accuracy
## 0.07066088 1.00000000
```

```
dim(testset1)
```

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

```
dim(cat_test)
```

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

#model prediction

```
model2   %>% predict(testset1) %>% `>`(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)
```