MODEL 2

Leonard Dwight Soledad

2022-12-16

Loading the Data

The data contains 197 rows and 431 columns with *Failure.binary* binary output.

```
library(readr)

Model2rawdata <- read_csv("radiomics_completedata.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.
```

```
Model2rawdata
```

```
## # A tibble: 197 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   49.3     12.9    46.3    6.25   17.8     9.78   6.81
##  2 A                   1   12.6     12.2    27.5   11.0    26.5    15.4   12.9
##  3 A                   0   79.8     12.8    90.2    2.78    6.88    4.30   0.923
##  4 A                   1   17.9     13.5   326.     6.30   22.0    10.3    6.65
##  5 A                   0   39.6     12.6    89.6    3.58    7.92    4.45   0.572
##  6 A                   1    4.77    13.2   102.     2.60    6.21    3.77   0.615
##  7 A                   0   25       12.2    36.8    8.65   28.2    14.9   17.7
##  8 A                   0   35.8     12.3    51.0    5.71   13.0     7.73   2.60
##  9 A                   1   35.3     13.4    27.2    5.88   14.8     9.12   4.40
## 10 A                   1   17.8     12.6    20.2    5.70   17.1     8.55   6.12
## # ... with 187 more rows, 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>, ...
```

Data Reprocessing

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v dplyr   1.0.10
## v tibble  3.1.8      v stringr 1.4.1
## v tidyr   1.2.1      v forcats 0.5.2
## v purrr   0.3.5
```

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

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

```
library(bestNormalize)
```

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

Checking for null and missing values

We are using *anyNA()* function to determine if there is any missing value in the data.

```
anyNA(Model2rawdata)
```

```
## [1] FALSE
```

```
#The output will show either *True* or *False*. There are missing values If True, thus you have to omit

#[1] FALSE

# The result is False, hence, the data has no missing values.
```

Checking the Normality of the Data We are using *Shapiro-Wilk's Test* to check the normality of the data.

```
Model2numrawdata <- Model2rawdata%>%select_if(is.numeric)

Model2numrawdata <- Model2numrawdata[ , -1]

Model2SWtest <- apply(Model2numrawdata, 2, function(x){shapiro.test(x)})
```

Next we need to list only the p-value of the respective variables to proceed with the test. We are using the *unlist()* and *lapply()* functions to achieve this goal.

```
Model2DRpvalue <- unlist(lapply(Model2SWtest, function(x) x$p.value))
```

```
sum(Model2DRpvalue<0.05)  # not normally distributed
```

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

```
sum(Model2DRpvalue>0.05)  # normally distributed
```

```
## [1] 1
```

```
Model2SWtest$Entropy_cooc.W.ADC
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.98903, p-value = 0.135
```

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

#  Currently, there are 428 variables that are not normally distributed and only the Entropy_cooc.W.ADC
```

The goal is that all variables should be normally distributed. Next, we are using *orderNorm()* function. And we need to exclude the *Entropy_cooc.W.ADC* since it is already normally distributed.

```
Model2DRtransrawdata <- Model2rawdata[,c(3,5:length(names(Model2rawdata)))]

Model2DRtransrawdata <- apply(Model2DRtransrawdata,2,orderNorm)

Model2DRtransrawdata <- lapply(Model2DRtransrawdata, function(x) x$x.t)

Model2DRtransrawdata <- Model2DRtransrawdata%>%as.data.frame()

Model2SWtest <- apply(Model2DRtransrawdata,2,shapiro.test)

Model2SWtest <- unlist(lapply(Model2SWtest, function(x) x$p.value))
```

Next, we will be testing the data to check the normality or the transformed data.

```
sum(Model2SWtest <0.05)  # for not normally distributed
```

```
## [1] 0
```

```
sum(Model2SWtest >0.05)  # for normally distributed
```

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

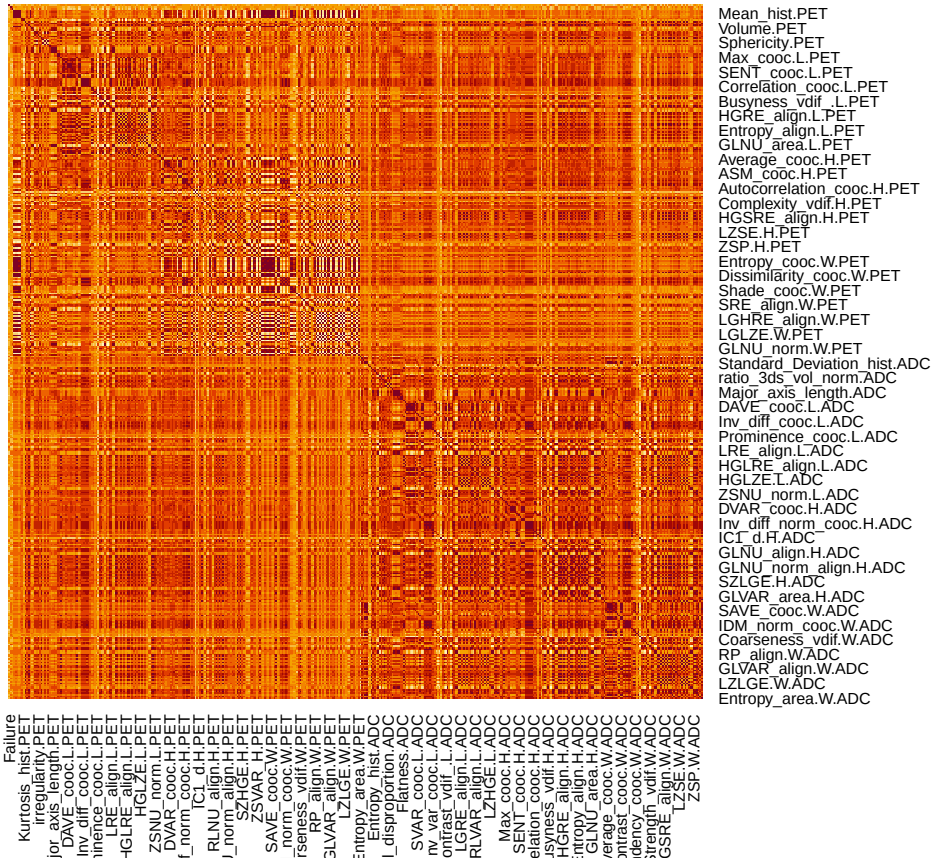
```
#[1] 0
#[1] 428

# Now, the 428 variables that were initially not normally distributed are now normally distributed.
```

```
Model2rawdata[,c(3,5:length(names(Model2rawdata)))] = Model2DRtransrawdata
```

We are getting the correlation of the whole data except the categorical variables

```
Model2CorrMat = cor(Model2rawdata[,-c(1,2)])
heatmap(Model2CorrMat,Rowv=NA,Colv=NA,scale="none",revC = T)
```



Finally, we will convert the data frame output of data reprocessing into "csv" file, which will we use for the entire model.

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

```
fwrite(Model2rawdata, "Model2_Final_Project_Data.csv")
```

Lastly, let's check if the dataframe we have exported to CSV is really the normal data.

```
Model2rawdata1 <- read_csv("Model2_Final_Project_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.
```

```
Model2rawdata1
```

```
## # A tibble: 197 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
##  7 A                   0   0.218    12.2  -0.687   0.284  0.519    0.405  0.915
##  8 A                   0   0.623    12.3  -0.405  -0.461 -0.719   -0.639 -0.623
##  9 A                   1   0.578    13.4  -1.04   -0.391 -0.593   -0.378 -0.244
## 10 A                   1  -0.160    12.6  -1.38   -0.475 -0.310   -0.475 -0.0127
## # ... with 187 more rows, 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>, ...
```

```
Model2numrawdata1 <- Model2rawdata1%>%select_if(is.numeric)

Model2numrawdata1 <- Model2numrawdata1[ , -1]

Model2SWtest1 <- apply(Model2numrawdata1, 2, function(y){shapiro.test(y)})

Model2DRpvalue1 <- unlist(lapply(Model2SWtest1, function(y) y$p.value))

sum(Model2DRpvalue1<0.05)  # not normally distributed
```

```
## [1] 0
```

```
sum(Model2DRpvalue1>0.05)  # normally distributed
```

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

Yes! We were able to produce the correct CSV file and we are now ready to use it for the entire Neural Networking Base Model.

**********NEURAL NETWORKING BASE**********

Helper Packages AND Model Packages

```
library(dplyr)
library(keras)
```

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

```
library(tfruns)
```

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

```
library(rsample)
library(tfestimators)
```

```
## 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
```

Recall that the data *Final_Project_Data.csv* is the output of our data reprocessing, and we noted it to be the data that we will be using for the entire project.

```
Model2Data <- read_csv("Model2_Final_Project_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.
```

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

```
Model2Data <- Model2Data %>%
  mutate(Failure.binary=ifelse(Failure.binary== "No",0,1))


set.seed(123)

snorm <- initial_split(Model2Data,prop = 0.8 ,strata = "Failure.binary")
normrtrain <- training(snorm)
normrtest  <- testing(snorm)

Model2dataXtrain <- normrtrain[,-c(1,2)]%>%as.matrix.data.frame()
Model2dataXtest <- normrtest[,-c(1,2)]%>%as.matrix.data.frame()

Model2dataYtrain <- normrtrain$Failure.binary
Model2dataYtest <- normrtest$Failure.binary
```

*Reshaping** the data set and then run the model. We are going to use *keras_model_sequential()* of the keras package. This will allow us to make the network with a layering technique. As instructed we need to make five hidden layers with 256, 128, 128, 64, and 64 neurons, respectively with activation functions of *Sigmoid* over fitting 2 neurons for the output layer with activation function *Softmax*. Also, to avoid over fitting, each layer is followed by a dropout.

```
Model2dataXtrain <- array_reshape(Model2dataXtrain, c(nrow(Model2dataXtrain), ncol(Model2dataXtrain)))
Model2dataXtrain <- Model2dataXtrain

Model2dataXtest <- array_reshape(Model2dataXtest, c(nrow(Model2dataXtest), ncol(Model2dataXtest)))
Model2dataXtest <- Model2dataXtest

Model2dataYtrain <- to_categorical(Model2dataYtrain, num_classes = 2)
```

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

```
Model2dataYtest <- to_categorical(Model2dataYtest, num_classes = 2)

Model2model <- keras_model_sequential() %>%
```

```
  layer_dense(units = 256, activation = "sigmoid", input_shape = c(ncol(Model2dataXtrain))) %>%
  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")
```

Backpropagation

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

Backpropagation Compiler Approach and train the model

Once we have already built a fundamental model; what remains is to sustain it with some data to train on. For us to accomplish this, we render our training data and model into a *fit()*. function.

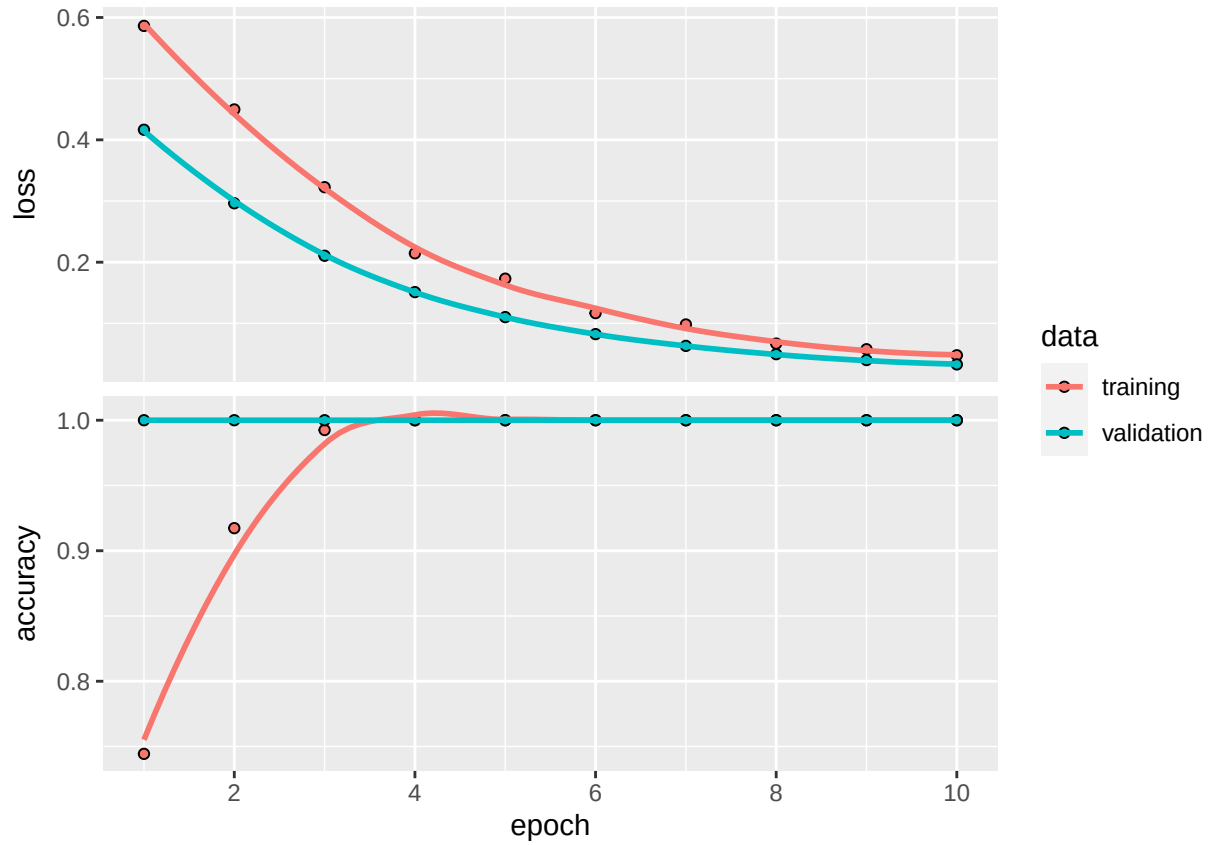
Epoch denotes the number of views of the algorithm to the entire data set. Hence, the epoch will end once all of the samples in the data set had already been inspected by the algorithm. Also, we need to segregate it in smaller portions, since an isolated epoch will be too large to transmit to the computer all at once.

```
 Model2model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_adam(),
  metrics = c("accuracy")
)

Model2history <- Model2model %>%
  fit(Model2dataXtrain, Model2dataYtrain, epochs = 10, batch_size = 128, validation_split = 0.15)
Model2history
```

```
##
## Final epoch (plot to see history):
##         loss: 0.0479
##     accuracy: 1
##     val_loss: 0.03292
## val_accuracy: 1
```

```
plot(Model2history)
```



Evaluate the trained model using testing data set.

```
Model2model %>%
  evaluate(Model2dataXtest, Model2dataYtest)
```

```
##       loss   accuracy
## 0.03246818 1.00000000
```

```
dim(Model2dataXtest)
```

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

```
dim(Model2dataYtest)
```

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

Model prediction

```
Model2model   %>% predict(Model2dataXtest) %>% `>`(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)
```