Model 2

Freddy A. Julkanain II

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)
```

```
## Warning: package 'keras' was built under R version 4.1.3
```

```
library(tfruns)
```

```
## Warning: package 'tfruns' was built under R version 4.1.3
```

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

```
## Warning: package 'tfestimators' was built under R version 4.1.3

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

```
library(readr)
```

Load the dataset

note that the data *normalRad.csv* is the output of our data reprocessing.

```
df=read_csv("CleanedDF.csv")
```

```
## New names:
## Rows: 197 Columns: 432
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Institution dbl (431): ...1, Failure.binary, Failure, Entropy_cooc.W.ADC,
## GLNU_align.H.P...
## 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.
## * `` -> `...1`
```

Split the data into training (70) and testing (30).

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

set.seed(123)
split = initial_split(df,prop = 0.8 ,strata = "Failure.binary")
churn_train = training(split)
churn_test  = testing(split)

#or

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

#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 2.9.2
```

```
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")
```

Backpropagation

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

Compile the Model

```
 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)
```

#Evaluate the trained model

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

```
##      loss  accuracy
## 0.1414207 1.0000000
```

```
dim(X_test)
```

```
## [1]  40 430
```

```
dim(y_test)
```

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

#model prediction

```
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)
```