PredictingCarCrashInjuries.R

Fireraq

Wed Apr 11 22:47:09 2018

```
#Library for high-level neural networks API
library(keras)
#Libraries used for Plotting
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

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

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(caret)
```

```
## Loading required package: lattice
```

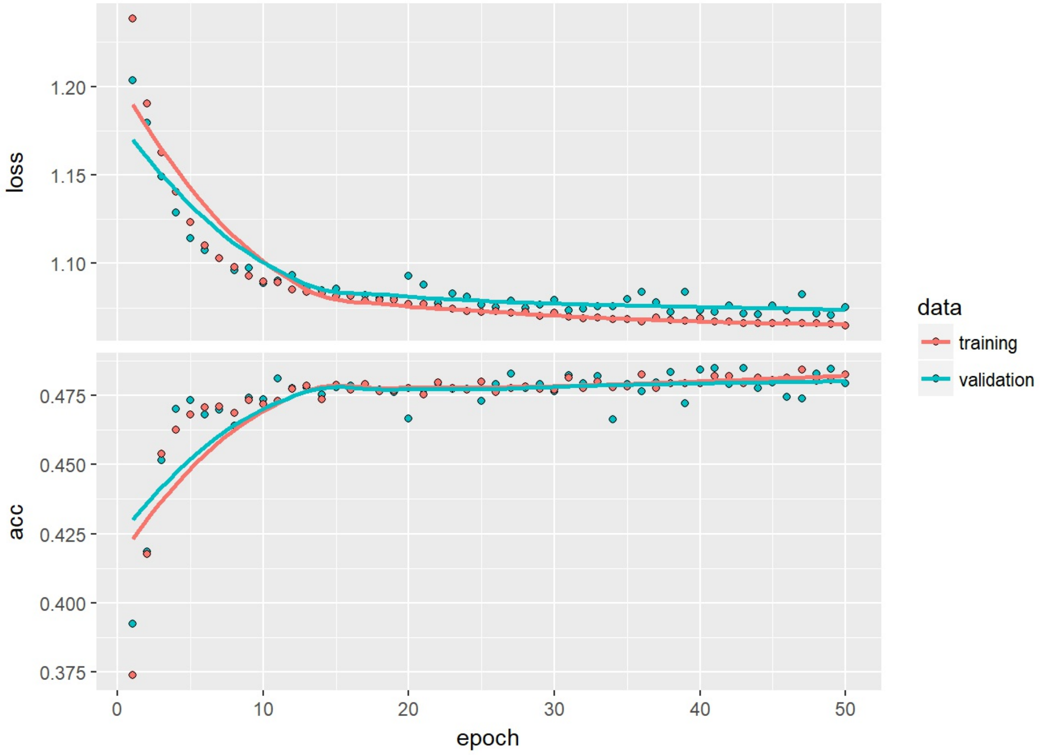
```
library(e1071)

#spliting data into test and train
splitData <- function(df, percent, sample) {
  train_ind <- sample
  train_df <-  data.frame(df[train_ind, ])
  test_df <- data.frame(df[train_ind, ])

  return(list("train" = data.matrix(train_df), "test" = data.matrix(test_df)))
}

# Cleaned carCrashData.
data = read.csv("carCrashDataCleanedNEW.csv")
x_data = subset(data, select=c("dvcat", "airbag","seatbelt","frontal","sex",
                               "ageOFocc","abcat","occRole","deploy", 'ageVeh'))

y_data = subset(data, select=c("injSeverity"))
nrow(y_data) # equals 26,217 colums with row injSeverity
```

```
## [1] 26217
```

```
#Pie chart desplays the injury severity distribution
data <- y_data

colors <- c('rgb(211,94,96)', 'rgb(128,133,133)', 'rgb(144,103,167)',
              'rgb(171,104,87)', 'rgb(114,147,203)')
ytable <- table(y_data)
p <- plot_ly(data, labels = names(ytable), values = ytable, type = 'pie',
             textposition = 'inside',
             textinfo = 'label+percent',
             insidetextfont = list(color = '#FFFFFF', size=18),
             marker = list(colors = colors,
                           line = list(color = '#FFFFFF', width = 1)),
             showlegend = FALSE) %>%
  layout(title = 'Injury Severity Distribution',
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE))
print(p)

#split the data into list(matrix,matrix) with 70%
#training and 30% testing respectively using our splitData function.
set.seed(123)
percent = 0.7
smp_size <- floor(percent * nrow(x_data))
sample = sample(seq_len(nrow(x_data)), size = smp_size)
x_data = splitData(x_data, 0.7, sample)
y_data = splitData(y_data, 0.7, sample)


#convert the y data to a one dimensional array
y_data$train = as.vector(y_data$train)
y_data$test = as.vector(y_data$test)

#convert one dimensional array to a categorical matrix via one-hot encoding
temp = y_data$test
y_data$train <- to_categorical(y_data$train, 4)
y_data$test <- to_categorical(y_data$test, 4)

#model the layers of our network
model <- keras_model_sequential()
model %>%
  layer_dense(units = 12, activation = 'relu', input_shape = c(ncol(x_data$train)),
              kernel_initializer = "random_uniform") %>%
   # increased our units to 5 to reflect all categories of injSeverity
  layer_dense(units = 4, activation = 'softmax')

summary(model)
```

```
## ___________________________________________________________________________
## Layer (type)                     Output Shape                  Param #
## ===========================================================================
## dense_1 (Dense)                  (None, 12)                    132
## ___________________________________________________________________________
## dense_2 (Dense)                  (None, 4)                     52
## ===========================================================================
## Total params: 184
## Trainable params: 184
## Non-trainable params: 0
## ___________________________________________________________________________
```

```
#compile the model
model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = 'adam',
  metrics = c('accuracy')
)

#train and eval
history <- model %>% fit(
  x_data$train, y_data$train,
  epochs = 50, batch_size = 100,
  validation_split = 0.2
)
plot(history)
```



```
model %>% evaluate(x_data$test, y_data$test)
```

```
## $loss
## [1] 1.065909
##
## $acc
## [1] 0.4819356
```

```
#Heat map confustion matrix plot
y_prediction = model %>% predict_classes(x_data$test)
cm = confusionMatrix(as.factor(y_prediction), as.factor(temp),
                     positive = NULL, dnn =c("Prediction","Reference"))

m <- cm$table
#Convert the raw numbers into percentages
for ( c in 1:ncol(m)) {
  c_sum = sum(m[,c])
  for (r in 1:length(m[,c])) {
    m[r,c] = m[r,c] / c_sum
  }
}

p <- plot_ly(
  x = c("0", "1", "2","3"), y = c("0", "1", "2","3"),
  z = m, type = "heatmap" )
print(p)

#Code to test the accuracy of the model
numcorrect = 0
numtests = 1000
for(row in c(0: numtests)) {
  testnum = row
  B = matrix (
    x_data$test[testnum,],
    nrow=1,
    ncol=ncol(x_data$test))
  guess = (model %>% predict_classes(B))
  gold = (which.max(y_data$test[testnum,])-1)

  if (identical(guess[1],gold[1])) {
    numcorrect = numcorrect + 1
  }
}

#print(numcorrect)
number = numcorrect / numtests
sprintf("Accuracy: %f",number)
```

```
## [1] "Accuracy: 0.496000"
```