NN

Yiru Gong, yg2832

2022-05-09

```r
library(tidyverse)
library(summarytools)
library(corrplot)
library(caret)
library(MASS)
library(mlbench)
library(pROC) #ROCR
library(pdp)
library(vip)
library(AppliedPredictiveModeling) #for transparentTheme function
library(ISLR)
library(caret)
library(e1071)
library(kernlab)
library(keras)
library(tfruns)
```

Data Input

```r
data = read.csv('Covid19_vacc_predict_handout.csv')
data = data %>%
  na.omit() %>%
  dplyr::select(-id) %>%
  mutate(
    atlas_type_2015_mining_no = factor(atlas_type_2015_mining_no),
    covid_vaccination = factor(covid_vaccination),
    hum_region = factor(hum_region),
    sex_cd = factor(sex_cd),
    race_cd = factor(race_cd),
    lang_spoken_cd = factor(lang_spoken_cd),
    atlas_low_education_2015_update = factor(atlas_low_education_2015_update)
    )
# summary(data)
# by(data[,c(5,7,8,10,11,17,18)], data$covid_vaccination, summary)
dfSummary(data[,c(5,7,8,10,11,17,18)])
```

Data Frame Summary
Dimensions: 8308 x 7
Duplicates: 7802

No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	atlas_type_2015_mining_no [factor]	1. 0 2. 1	8177 (98.4%) 131 (1.6%)	IIIIIIIIIIIIIIIIIII	8308 (100.0%)	0 (0.0%)
2	covid_vaccination [factor]	1. no_vacc 2. vacc	6682 (80.4%) 1626 (19.6%)	IIIIIIIIIIIIIIII III	8308 (100.0%)	0 (0.0%)
3	hum_region [factor]	1. CALIFORNIA/NEVADA 2. CENTRAL 3. CENTRAL WEST 4. EAST 5. EAST CENTRAL 6. FLORIDA 7. GREAT LAKES/CENTRAL NORTH 8. GULF STATES 9. INTERMOUNTAIN 10. MID-ATLANTIC/NORTH CAROLI [5 others]	299 (3.6%) 551 (6.6%) 238 (2.9%) 491 (5.9%) 1370 (16.5%) 607 (7.3%) 1111 (13.4%) 454 (5.5%) 220 (2.6%) 845 (10.2%) 2122 (25.5%)	I	8308 (100.0%)	0 (0.0%)
4	sex_cd [factor]	1. F 2. M	4527 (54.5%) 3781 (45.5%)	IIIIIIIIII IIIIIIIII	8308 (100.0%)	0 (0.0%)
5	lang_spoken_cd [factor]	1. * 2. CHI 3. CRE 4. ENG 5. KOR 6. OTH 7. SPA 8. VIE	10 (0.1%) 13 (0.2%) 4 (0.0%) 7957 (95.8%) 7 (0.1%) 34 (0.4%) 276 (3.3%) 7 (0.1%)		8308 (100.0%)	0 (0.0%)
6	atlas_low_education_2015_update [factor]	1. 0 2. 1	7769 (93.5%) 539 (6.5%)	IIIIIIIIIIIIIIIIII I	8308 (100.0%)	0 (0.0%)
7	race_cd [factor]	1. 0 2. 1 3. 2 4. 3 5. 4 6. 5 7. 6	160 (1.9%) 7317 (88.1%) 558 (6.7%) 80 (1.0%) 56 (0.7%) 129 (1.6%) 8 (0.1%)	IIIIIIIIIIIIIIIII I	8308 (100.0%)	0 (0.0%)

```
# cat_sum = NULL
# for (n in c(5,8,10,11,17,18)){
#   cat = data[,c(n,7)]
#   name = colnames(cat)[1]
#   cat2 = cat %>%
#     group_by(covid_vaccination,cat[,1]) %>%
#     count() %>%
#     rename(cat=`cat[, 1]`) %>%
#     pivot_wider(
#       names_from = covid_vaccination,
#       values_from = n
#     ) %>%
#     mutate(variable = name) %>%
#     relocate(variable,everything())
#   cat_sum = rbind(cat_sum,cat2)
# }
# knitr::kable(cat_sum)

# cat_sum %>%
#   pivot_longer(
#     c("no_vacc","vacc"),
#     names_to = 'covid_vaccination',
#     values_to = 'count'
#   ) %>%
#   ggplot(aes(variable,count,group=covid_vaccination,fill=cat))+geom_bar(stat = 'identity')

data2 = model.matrix(covid_vaccination ~ ., data)[ ,-1]
```

Data split

```
set.seed(1)
rowTrain <- createDataPartition(y = data$covid_vaccination,
                                p = 0.7,
                                list = FALSE)
x = data2[rowTrain,]
y = data$covid_vaccination[rowTrain]
x2 = data2[-rowTrain,]
y2 = data$covid_vaccination[-rowTrain]

save(x,y,x2,y2,file = "split_data.Rdata")
```

Neural Network

```
## tuning
set.seed(1)
runs <- tuning_run("keras_grid_search.R",
                   flags = list(
                   nodes_layer1 = c(64, 128, 256),
                   nodes_layer2 = c(64, 128, 256),
                   nodes_layer3 = c(64, 128, 256),
                   dropout_layer1 = c(0.2, 0.3, 0.4),
                   dropout_layer2 = c(0.2, 0.3, 0.4),
                   dropout_layer3 = c(0.2, 0.3, 0.4)),
                   confirm = FALSE,
                   echo = FALSE,
                   sample = 0.01) # try more after class
```

```
## 729 total combinations of flags
```

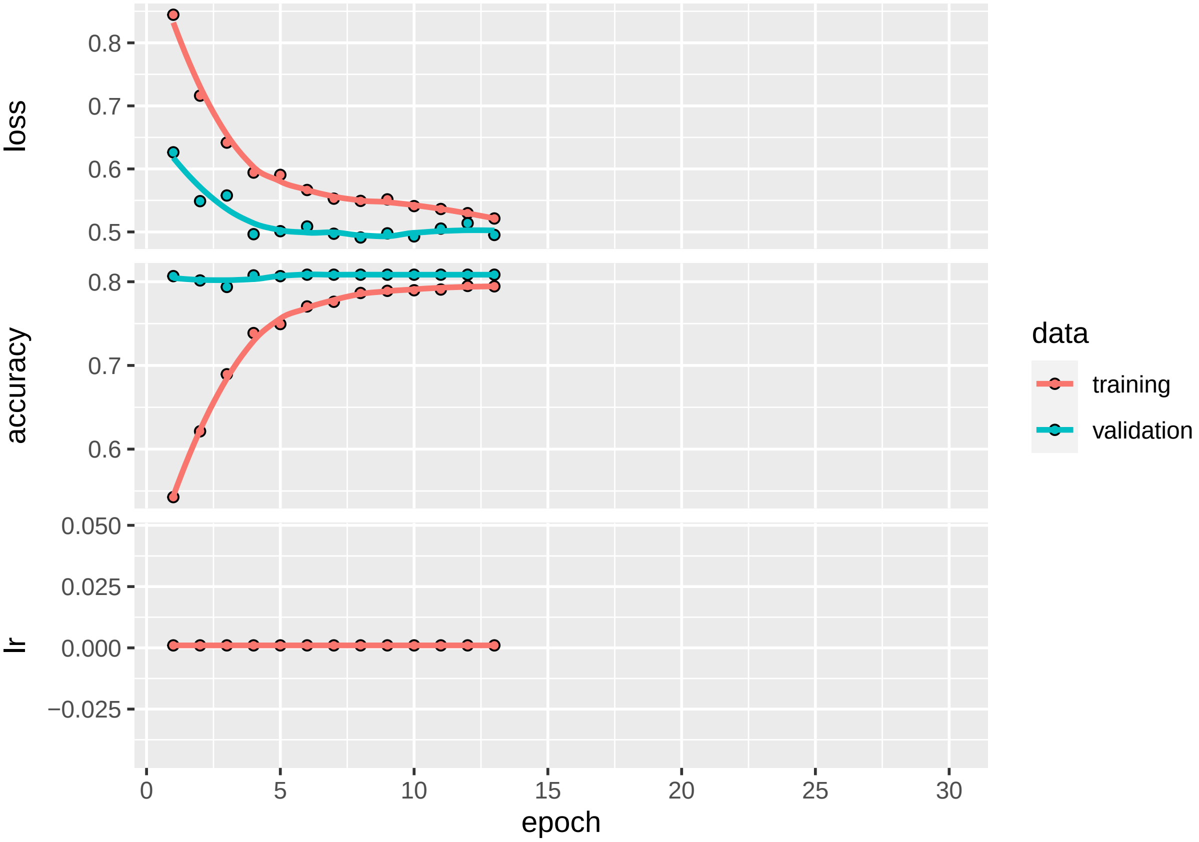
```
## (sampled to 8 combinations)
## Training run 1/8 (flags = list(64, 128, 64, 0.3, 0.4, 0.4))

## Using run directory runs/2022-05-10T00-55-54Z

## Loaded Tensorflow version 2.8.0

##
## Run completed: runs/2022-05-10T00-55-54Z

## Training run 2/8 (flags = list(256, 64, 256, 0.3, 0.3, 0.2))

## Using run directory runs/2022-05-10T00-56-08Z

##
## Run completed: runs/2022-05-10T00-56-08Z

## Training run 3/8 (flags = list(128, 128, 256, 0.2, 0.2, 0.4))

## Using run directory runs/2022-05-10T00-56-14Z

##
## Run completed: runs/2022-05-10T00-56-14Z

## Training run 4/8 (flags = list(256, 64, 128, 0.4, 0.4, 0.3))

## Using run directory runs/2022-05-10T00-56-18Z

##
## Run completed: runs/2022-05-10T00-56-18Z

## Training run 5/8 (flags = list(128, 64, 64, 0.4, 0.2, 0.3))

## Using run directory runs/2022-05-10T00-56-26Z

##
## Run completed: runs/2022-05-10T00-56-26Z

## Training run 6/8 (flags = list(256, 256, 256, 0.2, 0.2, 0.3))

## Using run directory runs/2022-05-10T00-56-32Z

##
## Run completed: runs/2022-05-10T00-56-32Z

## Training run 7/8 (flags = list(64, 256, 256, 0.2, 0.4, 0.2))

## Using run directory runs/2022-05-10T00-56-42Z
```

```
##
## Run completed: runs/2022-05-10T00-56-42Z

## Training run 8/8 (flags = list(64, 64, 128, 0.4, 0.2, 0.3))

## Using run directory runs/2022-05-10T00-56-48Z

##
## Run completed: runs/2022-05-10T00-56-48Z
```

```
best = runs[which.max(runs$metric_val_accuracy),]
```

```
y_c = ifelse(y=="vacc",1,0)
y_c <- to_categorical(y_c, 2)
y2_c = ifelse(y2=="vacc",1,0)
y2_c <- to_categorical(y2_c, 2)

model.nn <- keras_model_sequential() %>%
  layer_dense(units = best$flag_nodes_layer1, activation = "relu", input_shape = ncol(x)) %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = best$flag_dropout_layer1) %>%
  layer_dense(units = best$flag_nodes_layer2, activation = "relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = best$flag_dropout_layer2) %>%
  layer_dense(units = best$flag_nodes_layer3, activation = "relu") %>%
  layer_batch_normalization() %>%
  layer_dropout(rate = best$flag_dropout_layer3) %>%
  layer_dense(units = 2, activation = "sigmoid") %>%
  compile(loss = "categorical_crossentropy",
          optimizer = optimizer_rmsprop(),
          metrics = "accuracy")
fit.nn = model.nn %>%
  fit(x = x,
      y = y_c,
      epochs = 30,
      batch_size = 256,
      validation_split = 0.2,
      callbacks = list(callback_early_stopping(patience = 5),
                       callback_reduce_lr_on_plateau()),
      verbose = 2)
plot(fit.nn)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
## testing and evaluation
score <- model.nn %>% evaluate(x2, y2_c)
score
```

```
##      loss  accuracy
## 0.5063494 0.8044962
```