Analyzing FIFA 19 Ratings with Keras

William Morris

11/19/2020

```r
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
```

```r
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(stringr)


data_raw <- read.csv('data.csv', na.strings = c('',' ', 'NA'))
outputs <- subset(data_raw, select = c(Value, Wage))
names <- subset(data_raw, select = c(Name))
df <- subset(data_raw, select = -c(Flag,Club.Logo, Photo, Value, Wage, Name, Loaned.From))
```

```r
#I'm keeping this subset command separate because I may include these later.
#They don't seem super useful
df <- subset(df, select = -c(LS:RB))

#The following rows contain only NA values for most columns, so I'm removing them
rows_to_drop <- 13237:13284
df <- df[-rows_to_drop,]

#The following is commented out because these rows are still useful.
#They are missing info on Club, Position, Jersey.Number, Contract.Valid.Until, and Joined
#All instances are from India or Bolivia
#rows_to_drop <- c(which(is.na(df$Joined)))
#df <- df[-rows_to_drop]

#Convert Contract.Valid.Until to integer. There are various strings denoting dates,
#so I converted  to datetime first.
df$Contract.Valid.Until <- parse_date_time(df$Contract.Valid.Until,
                                           orders = c('Y', 'bdY', 'dbY'))
df$Contract.Valid.Until <- as.integer(year(df$Contract.Valid.Until))
df$Contract.Valid.Until[is.na(df$Contract.Valid.Until)] <- 0

#Do the same with Joined
df$Joined <- parse_date_time(df$Joined,
                             orders = c('bdY', 'dbY'))
df$Joined <- as.integer(year(df$Joined))
df$Joined[is.na(df$Joined)] <- 0

#Assign integer to Nationality and Club
df$Nationality <- mapvalues(df$Nationality,
                            from = unique(df$Nationality),
                            to = 1:length(unique(df$Nationality)))
df$Nationality <- as.integer(df$Nationality)
df$Club <- mapvalues(df$Club,
                     from = unique(df$Club),
                     to = 1:length(unique(df$Club)))
df$Club <- as.integer(df$Club)

#Assign 1 or -1 to Preferred Foot. If NA, assign 0
df$Preferred.Foot[is.na(df$Preferred.Foot)] <- 0
df$Preferred.Foot <- ifelse(df$Preferred.Foot == 'Left', 1, -1)

#Same for Real.Face
df$Real.Face[is.na(df$Real.Face)] <- 0
df$Real.Face <- ifelse(df$Real.Face == 'Yes', 1, -1)

#Assign integer to Body.Type and Work.Rate. If NA, choose most average string value
df$Body.Type[is.na(df$Body.Type)] <- 'Normal'
df$Body.Type <- mapvalues(df$Body.Type,
                          from = unique(df$Body.Type),
                          to = 1:length(unique(df$Body.Type)))
df$Body.Type <- as.integer(df$Body.Type)
df$Work.Rate[is.na(df$Work.Rate)] <- 'Medium/Medium'
df$Work.Rate <- mapvalues(df$Work.Rate,
                            from = unique(df$Work.Rate),
                            to = 1:length(unique(df$Work.Rate)))
df$Work.Rate <- as.integer(df$Work.Rate)

#Assign integer to Position. I don't have a good way to handle NA values yet,
#so they're their own class for now
df$Position <- mapvalues(df$Position,
                           from = unique(df$Position),
                           to = 1:length(unique(df$Position)))
df$Position <- as.integer(df$Position)

#Assign NAs in Jersey.Number to -1
df$Jersey.Number[is.na(df$Jersey.Number)] <- -1

#Convert character string of Height in feet'inches to integer with inches
convert_to_height <- function(character) {
  feet_inches <- str_split(character, pattern = "'")
  feet_inches <- as.integer(unlist(feet_inches))
  inches <- 12*feet_inches[1] + feet_inches[2]
  return(inches)
}
df$Height <- sapply(df$Height, FUN = convert_to_height)

#Convert Weight to numeric without the 'lbs'
convert_to_weight <- function(character) {
  weight_string <- str_split(character, pattern = 'lbs')
  weight <- as.integer(unlist(weight_string))
  return(weight[1])
}
df$Weight <- sapply(df$Weight, FUN = convert_to_weight)

#Remove € and 'M' from Release.Clause and convert to integer (in millions)
convert_from_euro <- function(character) {
  euro_string <- str_extract(character, pattern = '\\€[0-9,.]+')
  euro_string <- str_remove(euro_string, '\\€')
  euro <- as.numeric(euro_string) * 10^6
  return(euro)
}
df$Release.Clause <- sapply(df$Release.Clause, FUN = convert_from_euro)
df$Release.Clause[is.na(df$Release.Clause)] <- 0

#Export data to file
#write.csv(df, 'CleanData.csv')

#Clean output data, we'll decide later which one to use
outputs <- outputs[-rows_to_drop,]
outputs$Value <- sapply(outputs$Value, FUN = convert_from_euro)
outputs$Wage <- sapply(outputs$Wage, FUN = convert_from_euro)
#write.csv(outputs, 'Outputs.csv')
```

```
library(keras)
library(caret)
```

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

```
#Import clean data
fifa_data <- subset(read.csv('CleanData.csv'), select = -c(X.1))
outputs <- subset(read.csv('Outputs.csv'), select = -c(X))

#Normalize Data
preprocess_fifa <- preProcess(fifa_data, method=c("range"))
preprocess_output <- preProcess(outputs, method = c('range'))
fifa_data <- predict(preprocess_fifa, fifa_data)
outputs <- predict(preprocess_output, outputs)


#Separate into training and testing data
index <- 1:nrow(fifa_data)
ratio <- 0.2
set.seed(12)
index_test <- sample(index, floor(ratio*nrow(fifa_data)), replace = FALSE)
input_train <- as.matrix(fifa_data[-index_test,])
input_test <- as.matrix(fifa_data[index_test,])
output_train <- outputs[-index_test,1]
output_test <- outputs[index_test,1]

# Build Keras model
tensorflow::tf$random$set_seed(12)
fifamodel <- keras_model_sequential()

# Architecture
fifamodel %>%
  layer_dense(units = 37, activation = 'relu', input_shape = 56) %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 24, activation = 'relu') %>%
  layer_dropout(rate = 0.2) %>%
  #layer_dense(units = 16, activation = 'relu') %>%
  #layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = 'relu')

# Compilation Info
fifamodel %>% compile(
  loss = 'mse',
  optimizer = 'adam',
  metrics = c('mean_absolute_error')
)

# Training the model
fifa_history <- fifamodel %>% fit(
  input_train,
  output_train,
  epochs = 100,
  batch_size = 1800,
  validation_split = 0.2
)

training_scores <- fifamodel %>% evaluate(input_train, output_train, verbose = 0)
```

```r
cat('\nTraining and Validation MAE: ', training_scores[2], '\n')
```

```
##
## Training and Validation MAE:  0.08785104
```

```r
test_scores <- fifamodel %>% evaluate(input_test, output_test, verbose = 0)
cat('Testing MAE: ', test_scores[2], '\n')
```

```
## Testing MAE:  0.0881471
```