Artificial Neural Networks - Reuters 1986 & Major League Baseball 1986-1987 dataset

Problem 1— Using ANN for Classifying News Articles

Loading the library and reading in the dataset

```
library(keras)
```

```
## Warning: package 'keras' was built under R version 3.6.3
```

```
#install_keras()

reuters=dataset_reuters(num_words=10000)
```

Overall structure and summary statistics of variables.

```
str(reuters$train$x, list.len = 10)
```

```
## List of 8982
##  $ : int [1:87] 1 2 2 8 43 10 447 5 25 207 ...
##  $ : int [1:56] 1 3267 699 3434 2295 56 2 7511 9 56 ...
##  $ : int [1:139] 1 53 12 284 15 14 272 26 53 959 ...
##  $ : int [1:224] 1 4 686 867 558 4 37 38 309 2276 ...
##  $ : int [1:101] 1 8295 111 8 25 166 40 638 10 436 ...
##  $ : int [1:116] 1 4 37 38 309 213 349 1632 48 193 ...
##  $ : int [1:100] 1 56 5539 925 149 8 16 23 931 3875 ...
##  $ : int [1:100] 1 53 648 26 14 749 26 39 6207 5466 ...
##  $ : int [1:82] 1 178 53 321 26 14 948 26 178 39 ...
##  $ : int [1:106] 1 56 7224 81 40 1175 174 2 6 1793 ...
##   [list output truncated]
```

One-hot encoding of the dataset

```
one_hot_encoding=function(x, dimension=10000) {
  encoded=matrix(0,length(x),dimension)
  for (i in 1:length(x))
  encoded[i, x[[i]]]=1
  return (encoded)
}
```

```
data_train <- one_hot_encoding(reuters$train$x)
data_test <- one_hot_encoding(reuters$test$x)
```

```r
dim(data_train)
```

```
## [1]  8982 10000
```

```r
dim(data_test)
```

```
## [1]  2246 10000
```

1. Fitting ANN on the data

```r
model <- keras_model_sequential()

model %>%
  layer_dense(units = 500, activation = 'relu', input_shape = dim(data_train)[2]) %>%
  layer_dense(units = 250, activation = 'relu') %>%
  layer_dense(units = 100, activation = 'relu') %>%
  layer_dense(units = 46, activation = 'softmax')

model %>%
  compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics = c('accuracy'))

set.seed(123)

history <- model %>%
  fit(data_train, reuters$train$y, batch_size = 50, epochs = 100,
      validation_data = list(data_test, reuters$test$y))
```

```r
print('Metrics:' )
```

```
## [1] "Metrics:"
```

```r
model %>% evaluate(data_test, reuters$test$y)
```

```
##      loss  accuracy
## 3.3891776 0.7822796
```

2. Tuning the ANN

```r
library(tfruns)
```

```
## Warning: package 'tfruns' was built under R version 3.6.3
```

```r
# Train\Test split

testX <- data_train[1:1000,]
testY <- reuters$train$y[1:1000]
trainX <- data_train[1001:8982,]
trainY <-reuters$train$y[1001:8982]
```

```
# Parameter testing

runs <- tuning_run("reuters_train.R",
                   flags = list(
                   nodes_hlayer1 = c(700, 500, 250),
                   nodes_hlayer2 = c(200, 150, 100),
                   learning_rate = c(0.01, 0.05, 0.001, 0.0001),
                   batch_size=c(100,200,350,500),
                   epochs=c(35,60,100),
                   activation=c("relu","sigmoid","tanh")),
                   sample = 0.02
)
```

```
#runs
#view_run(runs$run_dir[9])

runsReuters <- runs[order(runs$metric_val_accuracy, decreasing = TRUE),][1,]
```

Best performing with params: nodes_hlayer1 = 500, nodes_hlayer2 = 100, batch_size = 500, activation = tanh, learning_rate = 0.001, epochs = 60.

The model fit using these parameters is not overfitting since it performing okay on both seen and unseen data (0.9632 training accuracy, 0.9704 testing accuracy). At around 10 epochs, the validation loss stopped decreasing significantly in this model. Even though it did decrease afterwards but it was not as much.

Re-fit with tuned parameters

```
model <- keras_model_sequential()

model %>%
  layer_dense(units = 500, activation = runsReuters$flag_activation, input_shape = dim(data_train)[2]) 
  layer_dense(units = runsReuters$flag_nodes_hlayer1, activation = runsReuters$flag_activation) %>%
  layer_dense(units = runsReuters$flag_nodes_hlayer2, activation = runsReuters$flag_activation) %>%
  layer_dense(units = 46, activation = 'softmax')

model %>%
  compile(optimizer = optimizer_adam(lr = runsReuters$flag_learning_rate), loss = 'sparse_categorical_c

set.seed(123)

model %>%
  fit(data_train, reuters$train$y, batch_size = runsReuters$flag_batch_size, epochs = runsReuters$flag_
      validation_data = list(data_test, reuters$test$y))
```

```
print('Metrics: ')
```

```
## [1] "Metrics: "
```

```
model %>% evaluate(data_test, reuters$test$y)
```

```
##      loss  accuracy
## 1.3544534 0.7853963
```

Problem 2 — Predicting Baseball players' salaries

1.Data load and exploration

```
hitters <- read.csv("hitters.csv", stringsAsFactors = FALSE)
```

```
cat("Number of Observations:", dim(hitters)[1])
```

```
## Number of Observations: 322
```

```
str(hitters)
```

```
## 'data.frame':    322 obs. of  20 variables:
##  $ AtBat    : int  293 315 479 496 321 594 185 298 323 401 ...
##  $ Hits     : int  66 81 130 141 87 169 37 73 81 92 ...
##  $ HmRun    : int  1 7 18 20 10 4 1 0 6 17 ...
##  $ Runs     : int  30 24 66 65 39 74 23 24 26 49 ...
##  $ RBI      : int  29 38 72 78 42 51 8 24 32 66 ...
##  $ Walks    : int  14 39 76 37 30 35 21 7 8 65 ...
##  $ Years    : int  1 14 3 11 2 11 2 3 2 13 ...
##  $ CAtBat   : int  293 3449 1624 5628 396 4408 214 509 341 5206 ...
##  $ CHits    : int  66 835 457 1575 101 1133 42 108 86 1332 ...
##  $ CHmRun   : int  1 69 63 225 12 19 1 0 6 253 ...
##  $ CRuns    : int  30 321 224 828 48 501 30 41 32 784 ...
##  $ CRBI     : int  29 414 266 838 46 336 9 37 34 890 ...
##  $ CWalks   : int  14 375 263 354 33 194 24 12 8 866 ...
##  $ League   : chr  "A" "N" "A" "N" ...
##  $ Division : chr  "E" "W" "W" "E" ...
##  $ PutOuts  : int  446 632 880 200 805 282 76 121 143 0 ...
##  $ Assists  : int  33 43 82 11 40 421 127 283 290 0 ...
##  $ Errors   : int  20 10 14 3 4 25 7 9 19 0 ...
##  $ Salary   : num  NA 475 480 500 91.5 750 70 100 75 1100 ...
##  $ NewLeague: chr  "A" "N" "A" "N" ...
```

```
summary(hitters)
```

```
##      AtBat            Hits         HmRun            Runs       
##  Min.   : 16.0   Min.   :  1   Min.   : 0.00   Min.   :  0.00  
##  1st Qu.:255.2   1st Qu.: 64   1st Qu.: 4.00   1st Qu.: 30.25  
##  Median :379.5   Median : 96   Median : 8.00   Median : 48.00  
##  Mean   :380.9   Mean   :101   Mean   :10.77   Mean   : 50.91  
##  3rd Qu.:512.0   3rd Qu.:137   3rd Qu.:16.00   3rd Qu.: 69.00  
##  Max.   :687.0   Max.   :238   Max.   :40.00   Max.   :130.00  
##                                                                 
##       RBI             Walks            Years            CAtBat       
##  Min.   :  0.00   Min.   :  0.00   Min.   : 1.000   Min.   :   19.0  
##  1st Qu.: 28.00   1st Qu.: 22.00   1st Qu.: 4.000   1st Qu.:  816.8  
##  Median : 44.00   Median : 35.00   Median : 6.000   Median : 1928.0  
##  Mean   : 48.03   Mean   : 38.74   Mean   : 7.444   Mean   : 2648.7  
##  3rd Qu.: 64.75   3rd Qu.: 53.00   3rd Qu.:11.000   3rd Qu.: 3924.2  
##  Max.   :121.00   Max.   :105.00   Max.   :24.000   Max.   :14053.0
```

```
##
##      CHits            CHmRun           CRuns             CRBI        
##  Min.   :   4.0   Min.   :  0.00   Min.   :   1.0   Min.   :   0.00  
##  1st Qu.: 209.0   1st Qu.: 14.00   1st Qu.: 100.2   1st Qu.:  88.75  
##  Median : 508.0   Median : 37.50   Median : 247.0   Median : 220.50  
##  Mean   : 717.6   Mean   : 69.49   Mean   : 358.8   Mean   : 330.12  
##  3rd Qu.:1059.2   3rd Qu.: 90.00   3rd Qu.: 526.2   3rd Qu.: 426.25  
##  Max.   :4256.0   Max.   :548.00   Max.   :2165.0   Max.   :1659.00  
##
##      CWalks           League            Division            PutOuts      
##  Min.   :   0.00   Length:322         Length:322         Min.   :   0.0  
##  1st Qu.:  67.25   Class :character   Class :character   1st Qu.: 109.2  
##  Median : 170.50   Mode  :character   Mode  :character   Median : 212.0  
##  Mean   : 260.24                                         Mean   : 288.9  
##  3rd Qu.: 339.25                                         3rd Qu.: 325.0  
##  Max.   :1566.00                                         Max.   :1378.0  
##
##     Assists          Errors          Salary         NewLeague        
##  Min.   :  0.0   Min.   : 0.00   Min.   :  67.5   Length:322        
##  1st Qu.:  7.0   1st Qu.: 3.00   1st Qu.: 190.0   Class :character  
##  Median : 39.5   Median : 6.00   Median : 425.0   Mode  :character  
##  Mean   :106.9   Mean   : 8.04   Mean   : 535.9                     
##  3rd Qu.:166.0   3rd Qu.:11.00   3rd Qu.: 750.0                     
##  Max.   :492.0   Max.   :32.00   Max.   :2460.0                     
##                                  NA's   :59
```

16 Numeric features and 3 categorical features with 1 numeric target variables. Salary have 59 null values.

2. Removing null values rows

```
hitters <- na.omit(hitters, cols = 'Salary')
```

3. Correlations

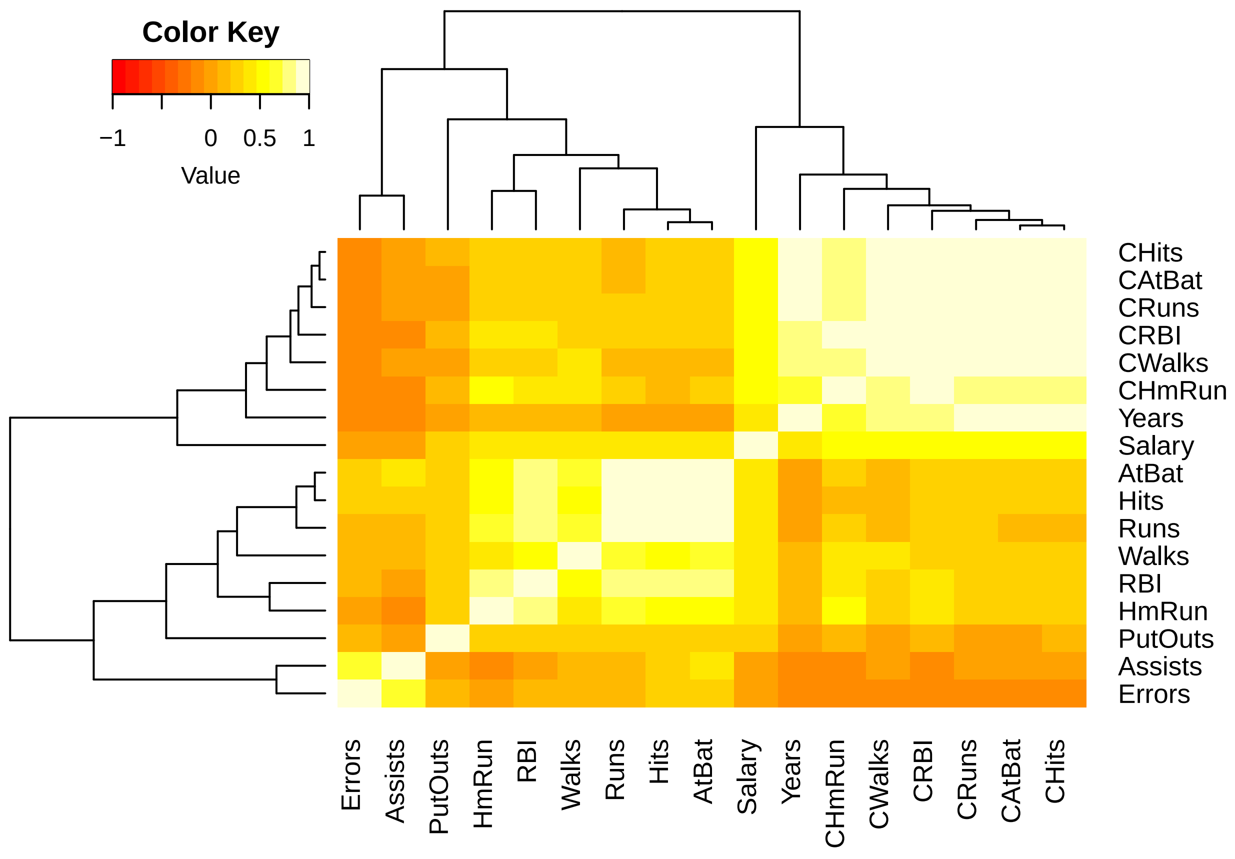
```
# Numerical Variables
library("gplots")
```

```
## Warning: package 'gplots' was built under R version 3.6.3
```

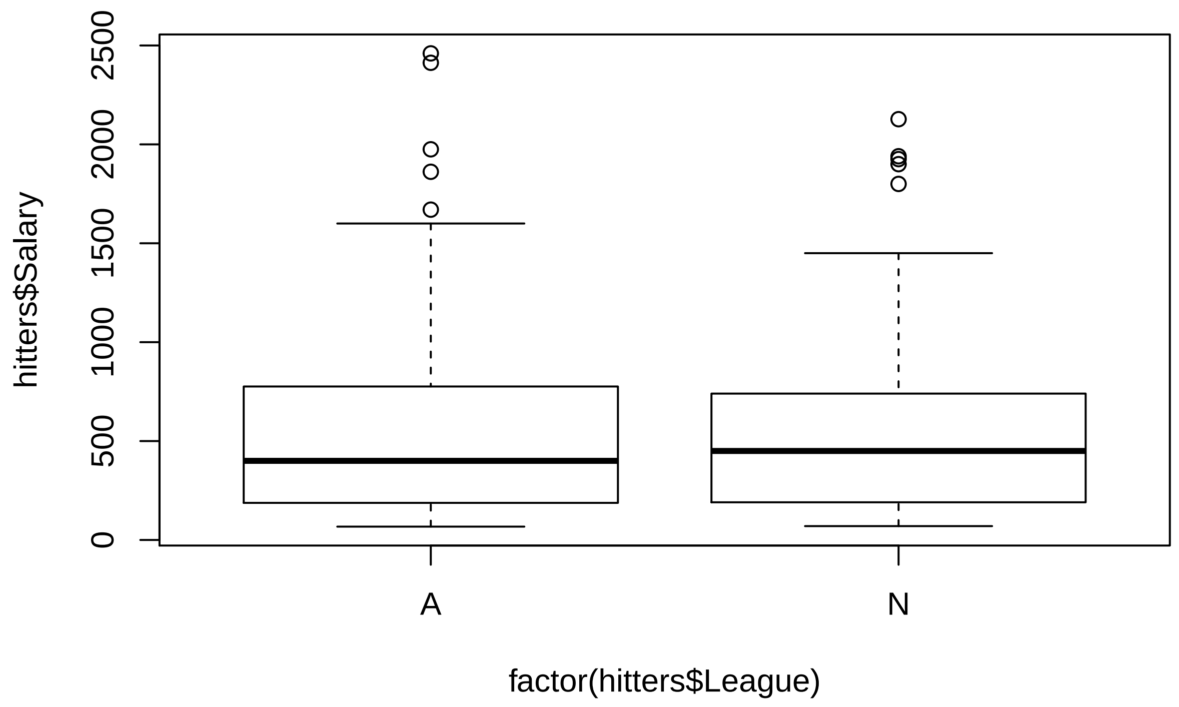
```
##
## Attaching package: 'gplots'
```

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

```
# pairs(hitters[c(1:13,16:19)]) # practically useless because a lot of variables
# cor(hitters[c(1:13,16:19)])

heatmap.2(cor(hitters[c(1:13,16:19)]), density.info = "none", trace = "none")
```



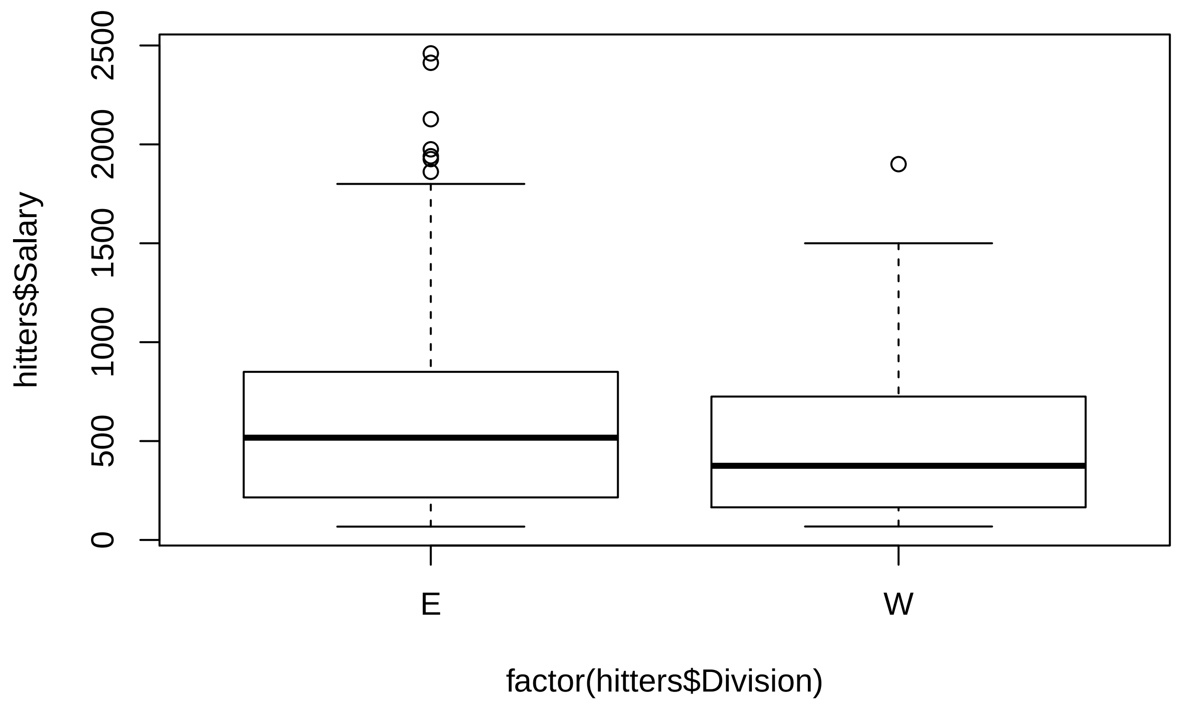
```r
# Categorical Variables

plot(hitters$Salary~factor(hitters$League))
```



```
t.test(hitters$Salary~factor(hitters$League), alternative = 'two.sided')
```

```
##
##  Welch Two Sample t-test
##
## data:  hitters$Salary by factor(hitters$League)
## t = 0.23157, df = 260.26, p-value = 0.8171
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -96.66039 122.42448
## sample estimates:
## mean in group A mean in group N
##        541.9995        529.1175
```

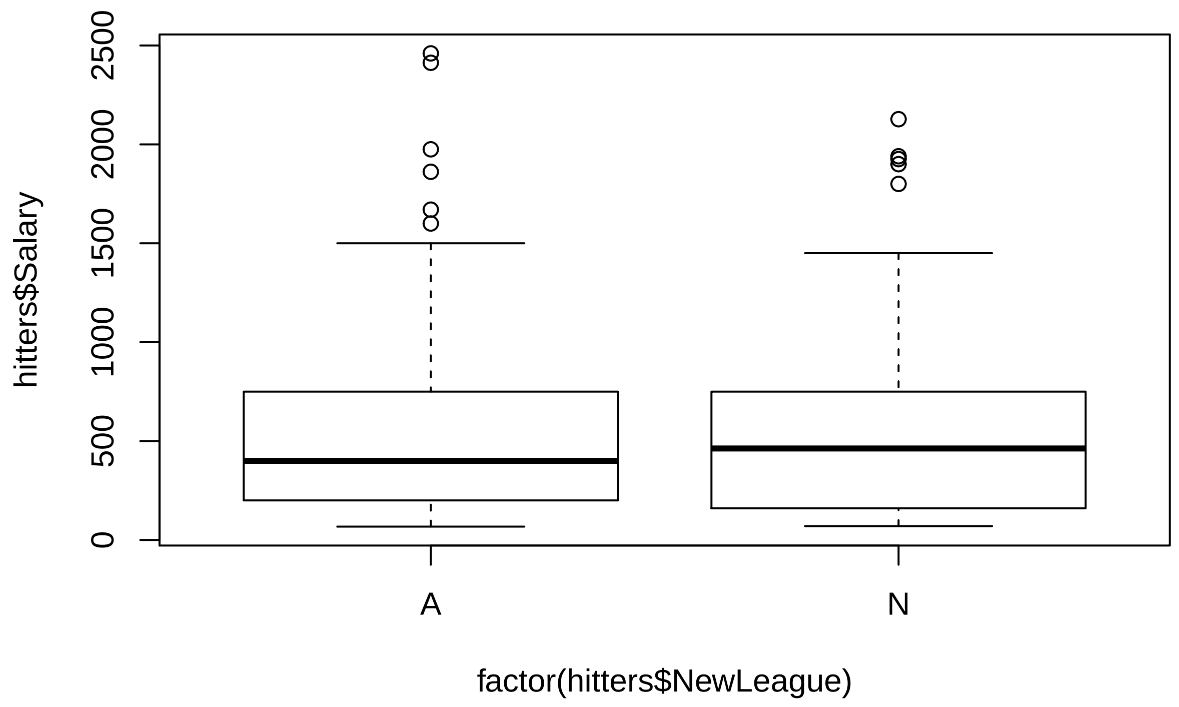
```
plot(hitters$Salary~factor(hitters$Division))
```



```r
t.test(hitters$Salary~factor(hitters$Division), alternative = 'two.sided')
```

```
##
##  Welch Two Sample t-test
##
## data:  hitters$Salary by factor(hitters$Division)
## t = 3.145, df = 218.46, p-value = 0.001892
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   64.73206 282.05692
## sample estimates:
## mean in group E mean in group W
##        624.2714        450.8769
```

```r
plot(hitters$Salary~factor(hitters$NewLeague))
```



```r
t.test(hitters$Salary~factor(hitters$NewLeague), alternative = 'two.sided')
```

```
##
##  Welch Two Sample t-test
##
## data:  hitters$Salary by factor(hitters$NewLeague)
## t = 0.045921, df = 258.07, p-value = 0.9634
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -107.1832  112.3016
## sample estimates:
## mean in group A mean in group N
##        537.1130        534.5539
```

The categorical variable 'Division' is associated with Salary (assuming alpha = 0.01).

4. Setting seed

```r
set.seed(123)
```

6,7,9. Data transformations

```r
# Categorical Encoding

hitters$League <- ifelse(hitters$League == "A", 1, 0)
```

```r
hitters$Division <- ifelse(hitters$Division == "E", 1, 0)
hitters$NewLeague <- ifelse(hitters$NewLeague == "A", 1, 0)
```

```r
# Log transformation of Salary

hitters$Salary <- log(hitters$Salary)
```

```r
# Scaling Features

hitters[,c(1:13,16:19)] <- scale(hitters[,c(1:13,16:19)])
```

8. Data partition

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```r
partitionIndex <- createDataPartition(hitters$Salary, p = 0.9, list = FALSE, times = 1)

hitters_train<- as.matrix(hitters[partitionIndex,])
hitters_test <- as.matrix(hitters[-partitionIndex,])

hitters_train_y <- hitters_train[,19]
hitters_test_y <- hitters_test[,19]

hitters_train <- hitters_train[,-19]
hitters_test <- hitters_test[,-19]
```

9. Modelling

```r
runs <- tuning_run("hitters_train.R",
                  flags = list(
                  nodes_hlayer1 = c(30, 20, 15),
                  nodes_hlayer2 = c(20, 15, 10),
                  learning_rate = c(0.01, 0.05, 0.001, 0.0001),
                  batch_size=c(10,20,50,75),
                  epochs=c(30,50,100),
                  activation=c("relu","sigmoid","tanh")),
                  sample = 0.02
)
```

```r
#runs
#view_run(runs$run_dir[9])

runsHitter <- runs[order(runs$metric_val_loss, decreasing = FALSE),][1,]
```

Best performing with params: nodes_hlayer1 = 15, nodes_hlayer2 = 20, batch_size = 75, activation = sigmoid, learning_rate = 0.01, epochs = 100.

The model fit using these parameters is not overfitting since it performing okay on both seen and unseen data (0.1621 training loss, 0.1887 testing loss). At around 30 epochs, the validation loss stopped decreasing significantly in this model. Even though it did decrease afterwards but it was not as much.

10. Evaluation of model

```r
set.seed(123)
model <- keras_model_sequential()

model %>%
  layer_dense(units = 20, activation = runsHitter$flag_activation, input_shape = dim(hitters_test)[2]) 
  layer_dense(units = runsHitter$flag_nodes_hlayer1, activation = runsHitter$flag_activation) %>%
  layer_dense(units = runsHitter$flag_nodes_hlayer2, activation = runsHitter$flag_activation) %>%
  layer_dense(units = 1)

model %>%
  compile(optimizer = optimizer_adam(lr = runsHitter$flag_learning_rate), loss = 'mse')

set.seed(123)

model %>%
  fit(hitters_train, hitters_train_y, batch_size = runsHitter$flag_batch_size, epochs = runsHitter$flag
      validation_data = list(hitters_test, hitters_test_y))
```

```r
predictions <- model %>% predict(hitters_test)

cat('RMSE:', RMSE(predictions, hitters_test_y))
```

```
## RMSE: 0.4921082
```