Predicting NHL players' salaries

UH

3/19/2021

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

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(ggplot2)
library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

library(corrplot)

## corrplot 0.84 loaded

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

# library(extraTrees)
# library(rJava)
library(gbm)

## Loaded gbm 2.1.8

library(xgboost)

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice

library(h2o)

##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit https://docs.h2o.ai
##
## ----------------------------------------------------------------------

##
## Attaching package: 'h2o'

## The following objects are masked from 'package:stats':
##
##     cor, sd, var

## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```r
setwd("~/Datascience/Data Science Courses/HarvardX Course/Individual project"
)

# Data for the 2020/21 season from HockeyReference (https://www.hockey-refere
nce.com/friv/current_nhl_salaries.cgi) and NHL (http://www.nhl.com/stats/skat
ers).

dat = read.csv("NHL_players_stats_merged.csv", header = T)

head(dat)

##          ï..Player Team   Salary S.C Pos GP  G  A  P plus_minus PIM PperGP
EVG
## 1 Auston Matthews  TOR 15900000   L   C 27 21 15 36          8   6   1.33
13
## 2 Mitchell Marner  TOR 15000000   R   R 30 11 28 39         15  14   1.30
11
## 3  Connor McDavid  EDM 14000000   L   C 30 17 35 52         10  14   1.73
11
## 4  Artemi Panarin  NYR 13000000   R   L 15  5 14 19          0   0   1.27
3
## 5  Mikko Rantanen  COL 12000000   L   R 26 13 15 28          7   6   1.08
8
## 6   Erik Karlsson  SJS 12000000   R   D 21  2  7  9          1   4   0.43
0
##   EVP PPG PPP SHG SHP OTG GWG   S  TOI
## 1  26   8  10   0   0   1   7 112 1323
## 2  25   0  13   0   1   0   2  82 1358
## 3  30   6  22   0   0   1   4 119 1349
## 4  12   2   7   0   0   0   0  45 1229
## 5  18   5  10   0   0   0   3  87 1210
## 6   5   2   4   0   0   0   0  41 1452

# Legend: S.C=Skater shoots, Pos=Player position, GP=Games played, G=Goals, A
=Assists, P=Points, plus_minus, PIM=Penalty minutes, PperGP=Points per game p
layed, EVG=Even strength goals, EVP=Even strength points,
# PPG=Powerplay goals, PPP=Powerplay points, SHG=Shorthanded goals, SHP=Short
handed points, OTG=Overtime goals, GWG=Game-winning goals, S=Shots, TOI=Time
on ice
```

Introduction

The goal of this project is to predict the salaries of NHL players. I am using data from the current 2020/21 season, which is publicly available. I am new to machine learning and am exploring various methods and approaches. I am a hockey fan, hence the topic. The csv files are posted on github.

Legend: S.C=Skater shoots, Pos=Player position, GP=Games played, G=Goals, A=Assists, P=Points, plus_minus, PIM=Penalty minutes, PperGP=Points per game played, EVG=Even strength goals, EVP=Even strength points, PPG=Powerplay goals, PPP=Powerplay points, SHG=Shorthanded goals, SHP=Shorthanded points, OTG=Overtime goals, GWG=Game-winning goals, S=Shots, TOI=Time on ice

```
# Wrangling and getting descriptive numbers
is.na(dat$salary)

## logical(0)

cat("\n\n Sort data frame by salary in descending order\n")

##
##
##  Sort data frame by salary in descending order

# sort data frame by salary in descending order
dat_sorted <- dat[with(dat, order(-dat$Salary)), ]

# print(dat_sorted)

# Make sure numbers are in numeric format
dat_sorted$Salary <- as.numeric(dat_sorted$Salary)
dat_sorted$GP <- as.numeric(dat_sorted$GP)
dat_sorted$G <- as.numeric(dat_sorted$G)
dat_sorted$A <- as.numeric(dat_sorted$A)
dat_sorted$plus_minus <- as.numeric(dat_sorted$plus_minus)
dat_sorted$PIM <- as.numeric(dat_sorted$PIM)
dat_sorted$PperGP<- as.numeric(dat_sorted$PperGP)
dat_sorted$EVG <- as.numeric(dat_sorted$EVG)
dat_sorted$EVP <- as.numeric(dat_sorted$EVP)
dat_sorted$PPG <- as.numeric(dat_sorted$PPG)
dat_sorted$PPP <- as.numeric(dat_sorted$PPP)
dat_sorted$SHG<- as.numeric(dat_sorted$SHG)
dat_sorted$SHP <- as.numeric(dat_sorted$SHP)
dat_sorted$OTG <- as.numeric(dat_sorted$OTG)
dat_sorted$GWG <- as.numeric(dat_sorted$GWG)
dat_sorted$S<- as.numeric(dat_sorted$S)
dat_sorted$TOI<- as.numeric(dat_sorted$TOI)

# Examine the structure of the dat dataset
str(dat)
```

```
## 'data.frame':    593 obs. of  22 variables:
##  $ ï..Player : chr  "Auston Matthews" "Mitchell Marner" "Connor McDavid" "
Artemi Panarin" ...
##  $ Team      : chr  "TOR" "TOR" "EDM" "NYR" ...
##  $ Salary    : int  15900000 15000000 14000000 13000000 12000000 12000000
12000000 12000000 10570000 10570000 ...
##  $ S.C       : chr  "L" "R" "L" "R" ...
##  $ Pos       : chr  "C" "R" "C" "L" ...
##  $ GP        : int  27 30 30 15 26 21 30 27 27 1 ...
##  $ G         : int  21 11 17 5 13 2 9 10 10 0 ...
##  $ A         : int  15 28 35 14 15 7 17 20 15 1 ...
##  $ P         : int  36 39 52 19 28 9 26 30 25 1 ...
##  $ plus_minus: int  8 15 10 0 7 1 11 -1 7 0 ...
##  $ PIM       : int  6 14 14 0 6 4 6 4 12 0 ...
##  $ PperGP    : num  1.33 1.3 1.73 1.27 1.08 0.43 0.87 1.11 0.93 1 ...
##  $ EVG       : int  13 11 11 3 8 0 4 7 6 0 ...
##  $ EVP       : int  26 25 30 12 18 5 17 19 16 1 ...
##  $ PPG       : int  8 0 6 2 5 2 5 3 3 0 ...
##  $ PPP       : int  10 13 22 7 10 4 9 11 8 0 ...
##  $ SHG       : int  0 0 0 0 0 0 0 0 1 0 ...
##  $ SHP       : int  0 1 0 0 0 0 0 0 1 0 ...
##  $ OTG       : int  1 0 1 0 0 0 0 0 0 0 ...
##  $ GWG       : int  7 2 4 0 3 0 1 0 2 0 ...
##  $ S         : int  112 82 119 45 87 41 83 49 66 1 ...
##  $ TOI       : int  1323 1358 1349 1229 1210 1452 1090 1132 1160 638 ...
```

```
# Create a summary for the dat dataset
summary(dat)
```

```
##   ï..Player             Team               Salary              S.C
##  Length:593         Length:593         Min.   :  700000   Length:593
##  Class :character   Class :character   1st Qu.:  925000   Class :character
##  Mode  :character   Mode  :character   Median : 2050000   Mode  :character
##                                        Mean   : 2928660
##                                        3rd Qu.: 4100000
##                                        Max.   :15900000
##      Pos                  GP              G                A
##  Length:593         Min.   : 1.00   Min.   : 0.000   Min.   : 0.00
##  Class :character   1st Qu.:19.00   1st Qu.: 1.000   1st Qu.: 2.00
##  Mode  :character   Median :24.00   Median : 3.000   Median : 5.00
##                     Mean   :22.28   Mean   : 3.826   Mean   : 6.41
##                     3rd Qu.:27.00   3rd Qu.: 6.000   3rd Qu.:10.00
##                     Max.   :31.00   Max.   :21.000   Max.   :35.00
##        P           plus_minus            PIM              PperGP
##  Min.   : 0.00   Min.   :-24.0000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.: 4.00   1st Qu.: -4.0000   1st Qu.: 4.000   1st Qu.:0.1900
##  Median : 8.00   Median :  0.0000   Median : 8.000   Median :0.3600
##  Mean   :10.24   Mean   :  0.1046   Mean   : 9.997   Mean   :0.4245
##  3rd Qu.:15.00   3rd Qu.:  3.0000   3rd Qu.:14.000   3rd Qu.:0.6100
##  Max.   :52.00   Max.   : 25.0000   Max.   :53.000   Max.   :1.7300
```

```
##       EVG              EVP              PPG               PPP
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 1.000   1st Qu.: 3.000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 2.000   Median : 7.000   Median : 0.0000   Median : 1.000
##  Mean   : 2.877   Mean   : 7.567   Mean   : 0.8583   Mean   : 2.497
##  3rd Qu.: 4.000   3rd Qu.:11.000   3rd Qu.: 1.0000   3rd Qu.: 4.000
##  Max.   :13.000   Max.   :30.000   Max.   :10.0000   Max.   :22.000
##       SHG              SHP              OTG               GWG
##  Min.   :0.00000   Min.   :0.000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.00000   Median :0.000   Median :0.00000   Median :0.0000
##  Mean   :0.09106   Mean   :0.172   Mean   :0.09275   Mean   :0.6054
##  3rd Qu.:0.00000   3rd Qu.:0.000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :4.00000   Max.   :4.000   Max.   :3.00000   Max.   :7.0000
##        S                TOI
##  Min.   :  0.00   Min.   : 248.0
##  1st Qu.: 21.00   1st Qu.: 796.0
##  Median : 35.00   Median : 973.0
##  Mean   : 38.29   Mean   : 973.8
##  3rd Qu.: 53.00   3rd Qu.:1132.0
##  Max.   :132.00   Max.   :1614.0
```

Now we have some basic numbers about the dataset. There are 593 skaters for which I could match stats from the NHL.com website and salaries. The highest salaries are for Matthews, Marner, and McDavid; these are all forwards. The highest salaries for defensemen are for Karlsson, Aho, and Trouba.

I thought that forwards and defensemen will be evaluated based on different parameters. For instance, goals per season will likely not be the main criterium for a defenseman. Hence I am splitting the dataset into forwards and defensemen. How do they differ in terms of salary? I thought initially that forwards would get paid better on average.

```
# Separate forwards and defenders; first forwards
Forwards<-subset(dat, dat$Pos !='D')
head(Forwards)
```

```
##          ï..Player Team   Salary S.C Pos GP  G  A  P plus_minus PIM PperGP
EVG
## 1 Auston Matthews  TOR 15900000   L   C 27 21 15 36          8   6   1.33
13
## 2 Mitchell Marner  TOR 15000000   R   R 30 11 28 39         15  14   1.30
11
## 3  Connor McDavid  EDM 14000000   L   C 30 17 35 52         10  14   1.73
11
## 4  Artemi Panarin  NYR 13000000   R   L 15  5 14 19          0   0   1.27
3
## 5  Mikko Rantanen  COL 12000000   L   R 26 13 15 28          7   6   1.08
8
## 7    John Tavares  TOR 12000000   L   C 30  9 17 26         11   6   0.87
4
```

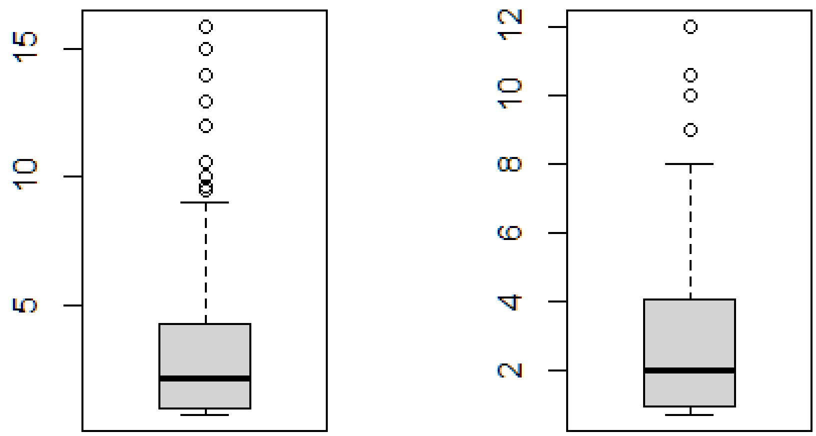
```
##    EVP PPG PPP SHG SHP OTG GWG   S  TOI
## 1  26   8  10   0   0   1   7 112 1323
## 2  25   0  13   0   1   0   2  82 1358
## 3  30   6  22   0   0   1   4 119 1349
## 4  12   2   7   0   0   0   0  45 1229
## 5  18   5  10   0   0   0   3  87 1210
## 7  17   5   9   0   0   0   1  83 1090
```

```
# Now create a dataframe with the defensemen
Def<-subset(dat, dat$Pos == 'D')
head(Def)
```

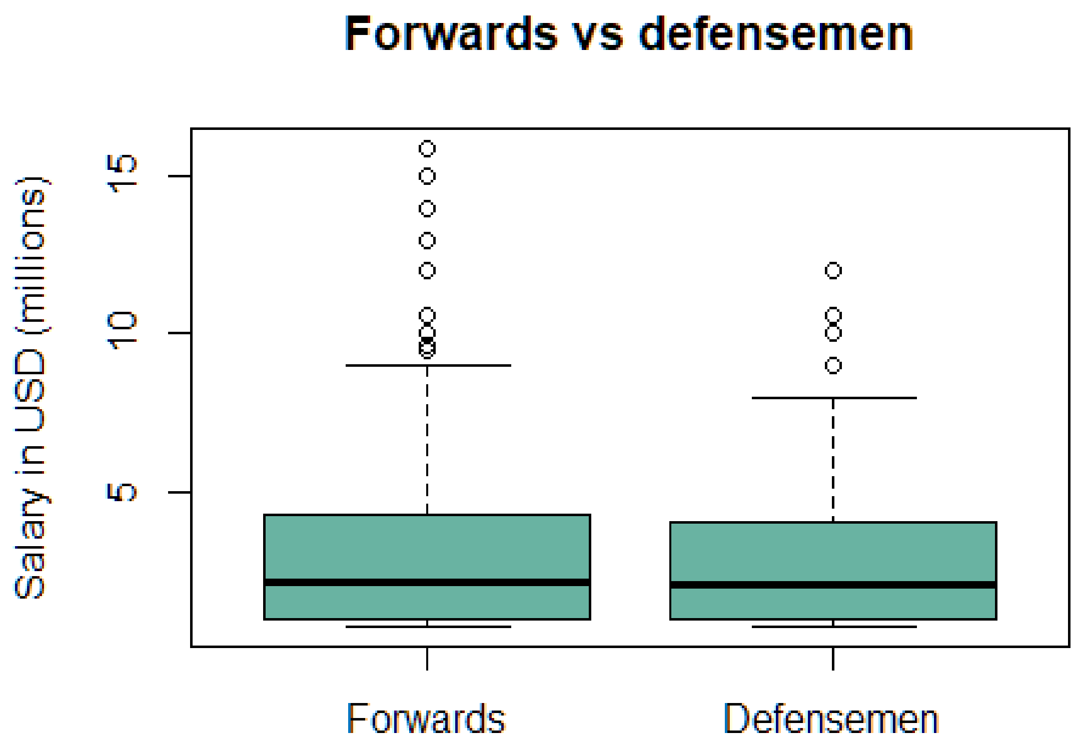
```
##          ï..Player Team   Salary S.C Pos GP G  A  P plus_minus PIM PperGP E
VG
## 6   Erik Karlsson  SJS 12000000   R   D 21 2  7  9          1   4   0.43
0
## 10  Sebastian Aho  NYI 10570000   L   D  1 0  1  1          0   0   1.00
0
## 11   Jacob Trouba  NYR 10000000   R   D 18 0  5  5         -1  14   0.28
0
## 14   Drew Doughty  LAK 10000000   R   D 27 6 16 22          0  12   0.81
2
## 15    Brent Burns  SJS 10000000   R   D 25 5  9 14        -13  16   0.56
3
## 24 Jared Spurgeon  MIN  9000000   R   D 25 0  5  5          1   0   0.20
0
##     EVP PPG PPP SHG SHP OTG GWG  S  TOI
## 6     5   2   4   0   0   0   0 41 1452
## 10    1   0   0   0   0   0   0  1  638
## 11    5   0   0   0   0   0   0 30 1307
## 14    7   4  14   0   1   0   0 46 1589
## 15    9   2   5   0   0   0   1 67 1614
## 24    5   0   0   0   0   0   0 50 1309
```

First I am getting some basic numbers from the two datasets. I will do some data wrangling as described below. I will then make some graphs and tables to get a better understanding of datasets.

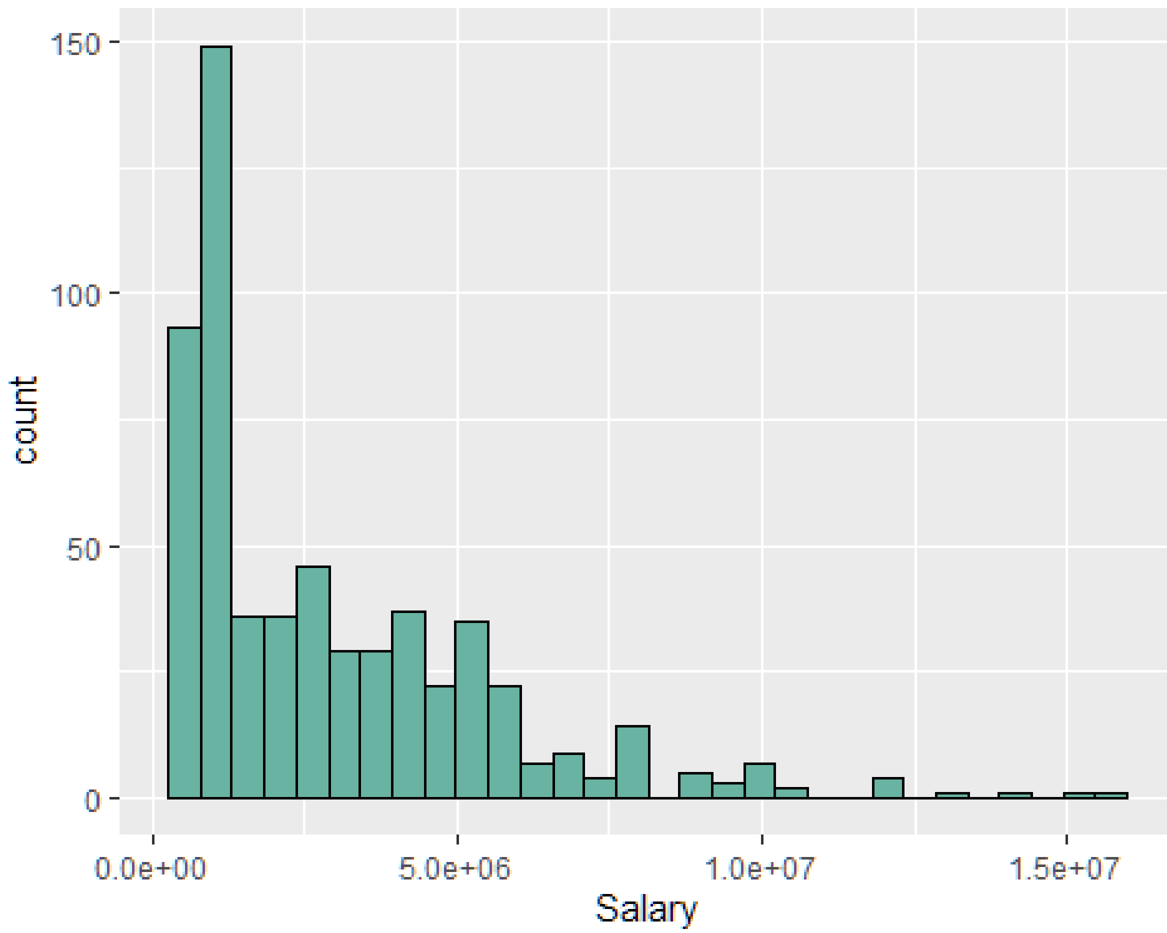
```
# Compare the salary in the Forwards and Defensemen datasets with a boxplot
p1 = dat$Salary[which(dat$Pos !='D')]/1000000
p2 = dat$Salary[which(dat$Pos =='D')]/1000000
par(mfrow=c(1,2))
boxplot(p1)
boxplot(p2)
```




```
par(mfrow=c(1,1))
boxplot(p1,p2, main = "Forwards vs defensemen", ylab = "Salary in USD (millio
ns)", names = c("Forwards", "Defensemen"), col = "#69b3a2")
```




```
# Make salary histogram
Histo <- ggplot(dat, aes(x=Salary)) +
  geom_histogram(fill= "#69b3a2", col = "black")
Histo

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Compare the salary in the Forwards and Defensemen datasets with numbers
summary(Forwards)
```

```
##    ï..Player              Team               Salary               S.C
##  Length:389         Length:389         Min.   :  700000   Length:389
##  Class :character   Class :character   1st Qu.:  900000   Class :character
##  Mode  :character   Mode  :character   Median : 2150000   Mode  :character
##                                        Mean   : 2975710
##                                        3rd Qu.: 4250000
##                                        Max.   :15900000
##      Pos                  GP              G                A
##  Length:389         Min.   : 1.00   Min.   : 0.000   Min.   : 0.000
##  Class :character   1st Qu.:20.00   1st Qu.: 2.000   1st Qu.: 3.000
##  Mode  :character   Median :25.00   Median : 4.000   Median : 5.000
##                     Mean   :22.82   Mean   : 5.064   Mean   : 6.817
##                     3rd Qu.:27.00   3rd Qu.: 7.000   3rd Qu.:10.000
##                     Max.   :31.00   Max.   :21.000   Max.   :35.000
##        P           plus_minus            PIM             PperGP
##  Min.   : 0.00   Min.   :-17.000000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.: 5.00   1st Qu.: -4.000000   1st Qu.: 4.000   1st Qu.:0.2500
##  Median :10.00   Median :  0.000000   Median : 8.000   Median :0.4300
##  Mean   :11.88   Mean   :  0.005141   Mean   : 9.715   Mean   :0.4882
```

```
##  3rd Qu.:18.00   3rd Qu.:  3.000000   3rd Qu.:13.000   3rd Qu.:0.7000
##  Max.   :52.00   Max.   : 20.000000   Max.   :53.000   Max.   :1.7300
##       EVG              EVP              PPG              PPP
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 2.000   1st Qu.: 4.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 3.000   Median : 8.000   Median : 0.000   Median : 1.000
##  Mean   : 3.784   Mean   : 8.787   Mean   : 1.152   Mean   : 2.889
##  3rd Qu.: 5.000   3rd Qu.:12.000   3rd Qu.: 2.000   3rd Qu.: 4.000
##  Max.   :13.000   Max.   :30.000   Max.   :10.000   Max.   :22.000
##       SHG              SHP              OTG              GWG
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.1285   Mean   :0.2057   Mean   :0.1337   Mean   :0.8175
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :4.0000   Max.   :4.0000   Max.   :3.0000   Max.   :7.0000
##        S               TOI
##  Min.   :  0.00   Min.   : 248.0
##  1st Qu.: 24.00   1st Qu.: 751.0
##  Median : 38.00   Median : 891.0
##  Mean   : 41.74   Mean   : 892.8
##  3rd Qu.: 58.00   3rd Qu.:1044.0
##  Max.   :132.00   Max.   :1358.0

mean(Forwards$Salary)

## [1] 2975710

median(Forwards$Salary)

## [1] 2150000

max(Forwards$Salary)

## [1] 15900000

min(Forwards$Salary)

## [1] 700000

summary(Def)

##   ï..Player            Team               Salary             S.C
##  Length:204         Length:204         Min.   :  700000   Length:204
##  Class :character   Class :character   1st Qu.:  925000   Class :character
##  Mode  :character   Mode  :character   Median : 2000000   Mode  :character
##                                        Mean   : 2838944
##                                        3rd Qu.: 4025044
##                                        Max.   :12000000
##      Pos                  GP              G                A
##  Length:204         Min.   : 1.00   Min.   : 0.000   Min.   : 0.000
##  Class :character   1st Qu.:17.00   1st Qu.: 0.000   1st Qu.: 2.000
```

```
##  Mode  :character   Median :24.00   Median : 1.000   Median : 4.000  
##                     Mean   :21.25   Mean   : 1.466   Mean   : 5.632  
##                     3rd Qu.:27.00   3rd Qu.: 2.000   3rd Qu.: 9.000  
##                     Max.   :31.00   Max.   :11.000   Max.   :22.000  
##        P            plus_minus            PIM             PperGP      
##  Min.   : 0.000   Min.   :-24.0000   Min.   : 0.00   Min.   :0.0000  
##  1st Qu.: 2.000   1st Qu.: -3.0000   1st Qu.: 4.00   1st Qu.:0.1375  
##  Median : 5.500   Median :  0.0000   Median : 8.00   Median :0.2500  
##  Mean   : 7.098   Mean   :  0.2941   Mean   :10.53   Mean   :0.3032  
##  3rd Qu.:10.250   3rd Qu.:  4.0000   3rd Qu.:14.25   3rd Qu.:0.4300  
##  Max.   :26.000   Max.   : 25.0000   Max.   :45.00   Max.   :1.0000  
##       EVG              EVP             PPG             PPP       
##  Min.   :0.000   Min.   : 0.00   Min.   :0.000   Min.   : 0.00  
##  1st Qu.:0.000   1st Qu.: 2.00   1st Qu.:0.000   1st Qu.: 0.00  
##  Median :1.000   Median : 5.00   Median :0.000   Median : 0.00  
##  Mean   :1.147   Mean   : 5.24   Mean   :0.299   Mean   : 1.75  
##  3rd Qu.:2.000   3rd Qu.: 8.00   3rd Qu.:0.000   3rd Qu.: 3.00  
##  Max.   :8.000   Max.   :16.00   Max.   :5.000   Max.   :16.00  
##       SHG               SHP              OTG               GWG       
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.000  
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000  
##  Median :0.00000   Median :0.0000   Median :0.00000   Median :0.000  
##  Mean   :0.01961   Mean   :0.1078   Mean   :0.01471   Mean   :0.201  
##  3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.000  
##  Max.   :1.00000   Max.   :2.0000   Max.   :1.00000   Max.   :2.000  
##        S              TOI        
##  Min.   : 0.00   Min.   : 568.0  
##  1st Qu.:15.75   1st Qu.: 961.5  
##  Median :30.00   Median :1132.5  
##  Mean   :31.73   Mean   :1128.2  
##  3rd Qu.:46.25   3rd Qu.:1321.2  
##  Max.   :88.00   Max.   :1614.0
```

```
mean(Def$Salary)
```

```
## [1] 2838944
```

```
median(Def$Salary)
```

```
## [1] 2e+06
```

```
max(Def$Salary)
```

```
## [1] 12000000
```

```
min(Def$Salary)
```

```
## [1] 700000
```

While the top salaries are reserved for forwards, defensemen have similar salaries in terms of mean and median.

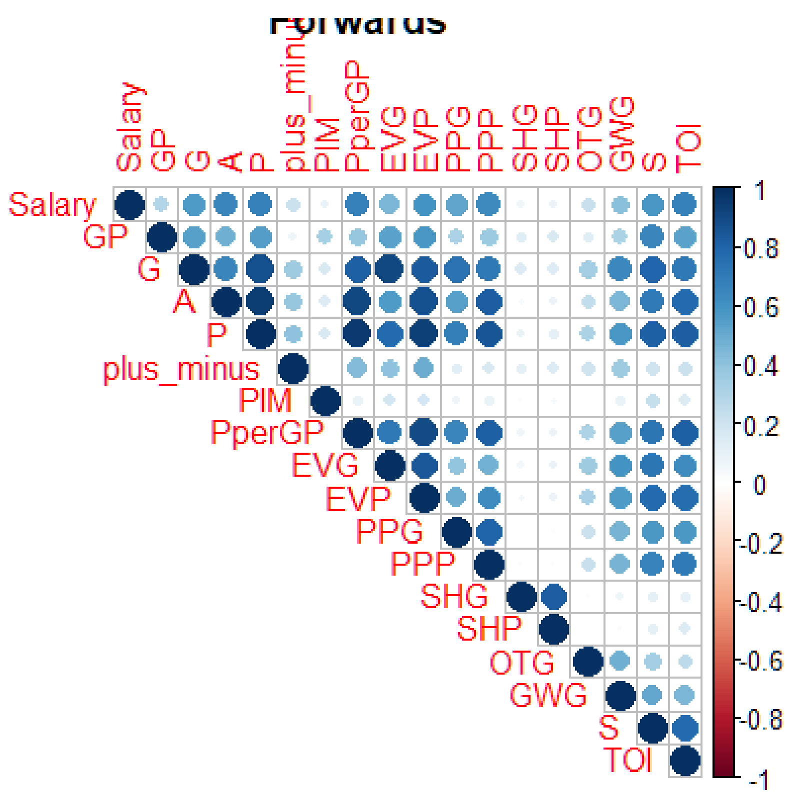
The boxplots and histogram tell us a few things: (1) There are many outliers for extremely well paid skaters, and (2) the salaries are not normally distributed. Still, let's try multiple linear regression as a first approach.

```
# Build correlation matrix of all parameters. First exclude non-numeric data.
# For forwards
sapply(Forwards, is.numeric)

##  ï..Player       Team     Salary        S.C        Pos         GP
G
##      FALSE      FALSE       TRUE      FALSE      FALSE       TRUE        TR
UE
##          A          P plus_minus        PIM     PperGP        EVG         E
VP
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE        TR
UE
##        PPG        PPP        SHG        SHP        OTG        GWG
S
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE        TR
UE
##        TOI
##       TRUE

Forwards_num_data <- Forwards[, sapply(Forwards, is.numeric)]
cor(Forwards_num_data, use = "complete.obs", method = "pearson")

##                Salary         GP         G          A         P  plus_minu
s
## Salary     1.00000000 0.29855469 0.5688157 0.65067772 0.6755561 0.21067993
8
## GP         0.29855469 1.00000000 0.5412164 0.48593962 0.5565427 0.07406654
1
## G          0.56881565 0.54121637 1.0000000 0.65818317 0.8731069 0.35628175
1
## A          0.65067772 0.48593962 0.6581832 1.00000000 0.9417041 0.38327364
3
## P          0.67555606 0.55654272 0.8731069 0.94170415 1.0000000 0.40741440
7
## plus_minus 0.21067994 0.07406654 0.3562818 0.38327364 0.4074144 1.00000000
0
## PIM        0.09336739 0.33337439 0.1644578 0.14895819 0.1699550 0.00927598
3
## PperGP     0.67497667 0.38663889 0.8178614 0.90653432 0.9525366 0.43419902
6
## EVG        0.45292699 0.53644744 0.9031630 0.56735475 0.7710139 0.40270019
4
## EVP        0.59459280 0.58534638 0.8360137 0.87041134 0.9372564 0.49063225
4
## PPG        0.52182429 0.31318935 0.7397810 0.54454107 0.6832272 0.12262101
2
## PPP        0.63118408 0.36409513 0.7103241 0.82328113 0.8505672 0.16895420
```

```
3
## SHG        0.06356257 0.13312371 0.1513386 0.03688819 0.0915189 0.11105985
8
## SHP        0.08757546 0.16823725 0.1468328 0.08361223 0.1197625 0.15834274
4
## OTG        0.22738305 0.13913596 0.3497504 0.24571196 0.3154147 0.20458545
8
## GWG        0.41131968 0.31020592 0.6458213 0.45052559 0.5803562 0.35110797
3
## S          0.58010268 0.65496266 0.7934833 0.70747705 0.8127386 0.20219035
8
## TOI        0.67030509 0.53554053 0.7179934 0.77467838 0.8225208 0.21497405
4
##                     PIM     PperGP        EVG        EVP          PPG
## Salary      0.0933673853 0.67497667 0.45292699 0.59459280  0.521824293
## GP          0.3333743868 0.38663889 0.53644744 0.58534638  0.313189348
## G           0.1644578306 0.81786144 0.90316301 0.83601370  0.739781046
## A           0.1489581870 0.90653432 0.56735475 0.87041134  0.544541071
## P           0.1699549931 0.95253658 0.77101391 0.93725640  0.683227214
## plus_minus  0.0092759827 0.43419903 0.40270019 0.49063225  0.122621012
## PIM         1.0000000000 0.09915262 0.18623755 0.18749468  0.074706109
## PperGP      0.0991526176 1.00000000 0.71810767 0.89306021  0.651392559
## EVG         0.1862375530 0.71810767 1.00000000 0.84291157  0.397107445
## EVP         0.1874946773 0.89306021 0.84291157 1.00000000  0.497566982
## PPG         0.0747061093 0.65139256 0.39710744 0.49756698  1.000000000
## PPP         0.0988448071 0.81325894 0.47645129 0.62037529  0.808313802
## SHG        -0.0266492540 0.06520372 0.05813610 0.05440679  0.001476032
## SHP         0.0464283757 0.08921445 0.09252494 0.08092964 -0.019857744
## OTG         0.0006747926 0.30664446 0.35425113 0.32611068  0.213541309
## GWG         0.0978716537 0.54815737 0.59357441 0.56603193  0.467347969
## S           0.2396343195 0.72348917 0.72226140 0.77673376  0.579858069
## TOI         0.1520555116 0.81595070 0.62153922 0.76014044  0.573594934
##                   PPP          SHG         SHP           OTG        GWG
## Salary     0.63118408  0.063562572  0.08757546  0.2273830499 0.41131968
## GP         0.36409513  0.133123714  0.16823725  0.1391359582 0.31020592
## G          0.71032409  0.151338573  0.14683281  0.3497504482 0.64582130
## A          0.82328113  0.036888187  0.08361223  0.2457119608 0.45052559
## P          0.85056724  0.091518905  0.11976247  0.3154147174 0.58035617
## plus_minus 0.16895420  0.111059858  0.15834274  0.2045854584 0.35110797
## PIM        0.09884481 -0.026649254  0.04642838  0.0006747926 0.09787165
## PperGP     0.81325894  0.065203718  0.08921445  0.3066444649 0.54815737
## EVG        0.47645129  0.058136100  0.09252494  0.3542511254 0.59357441
## EVP        0.62037529  0.054406786  0.08092964  0.3261106828 0.56603193
## PPG        0.80831380  0.001476032 -0.01985774  0.2135413094 0.46734797
## PPP        1.00000000  0.010659768  0.01006036  0.2276268759 0.46761493
## SHG        0.01065977  1.000000000  0.82193297 -0.0103139446 0.07365052
## SHP        0.01006036  0.821932969  1.00000000  0.0036239001 0.02027842
## OTG        0.22762688 -0.010313945  0.00362390  1.0000000000 0.48891424
## GWG        0.46761493  0.073650522  0.02027842  0.4889142439 1.00000000
## S          0.66602667  0.113038315  0.11807334  0.3406281889 0.51768473
```

```
## TOI        0.70903205  0.108284007  0.14575022  0.2644176808 0.45390051
##                    S       TOI
## Salary     0.5801027 0.6703051
## GP         0.6549627 0.5355405
## G          0.7934833 0.7179934
## A          0.7074770 0.7746784
## P          0.8127386 0.8225208
## plus_minus 0.2021904 0.2149741
## PIM        0.2396343 0.1520555
## PperGP     0.7234892 0.8159507
## EVG        0.7222614 0.6215392
## EVP        0.7767338 0.7601404
## PPG        0.5798581 0.5735949
## PPP        0.6660267 0.7090321
## SHG        0.1130383 0.1082840
## SHP        0.1180733 0.1457502
## OTG        0.3406282 0.2644177
## GWG        0.5176847 0.4539005
## S          1.0000000 0.7770869
## TOI        0.7770869 1.0000000
```

```
corrplot(cor(Forwards_num_data), method = "circle", type = "upper", title = "
Forwards")
```




```
# For defensemen
sapply(Def, is.numeric)
```

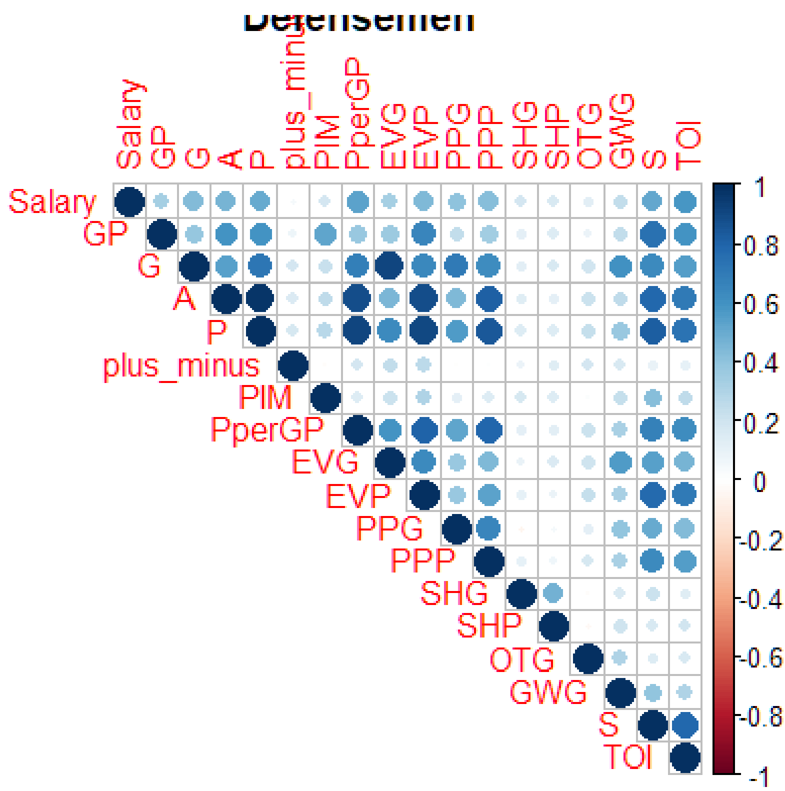
```
##  ï..Player       Team     Salary        S.C        Pos         GP
G
##      FALSE      FALSE       TRUE      FALSE      FALSE       TRUE         TR
UE
##          A          P plus_minus        PIM     PperGP        EVG          E
VP
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE         TR
UE
##        PPG        PPP        SHG        SHP        OTG        GWG
S
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE         TR
UE
##        TOI
##       TRUE
```

```
Def_num_data <- Def[, sapply(Def, is.numeric)]
cor(Def_num_data, use = "complete.obs", method = "pearson")
```

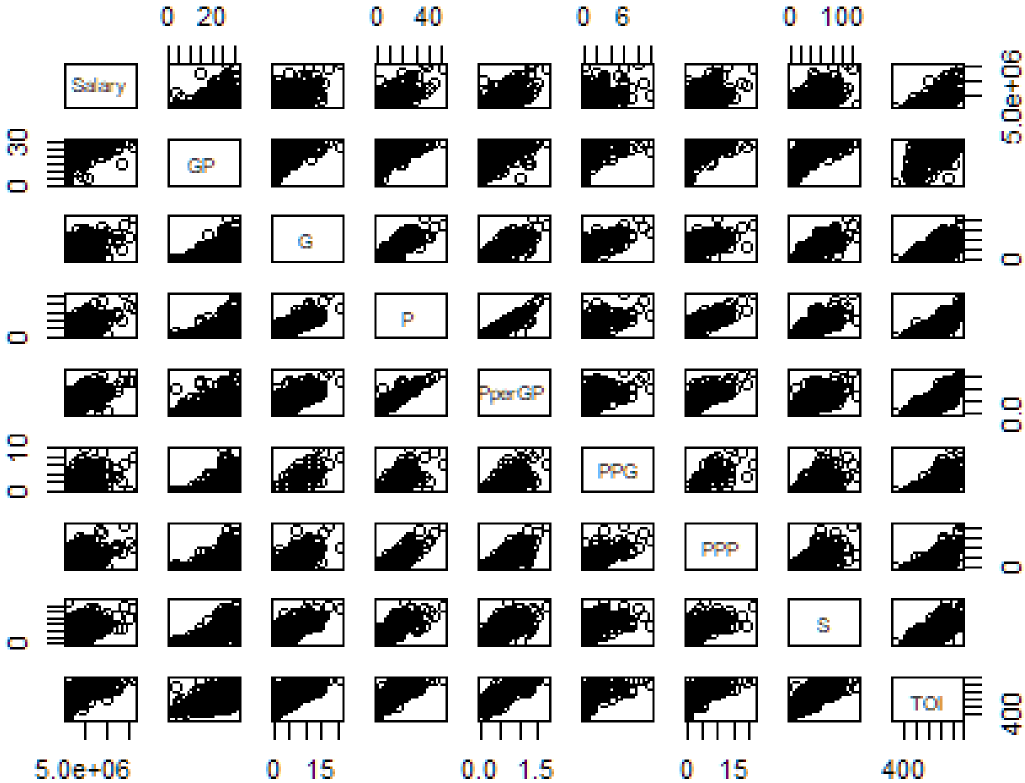
```
##                Salary         GP         G         A         P   plus_minu
s
## Salary     1.00000000 0.33035269 0.4387764 0.4623040 0.5020896  0.04765473
1
## GP         0.33035269 1.00000000 0.3843629 0.5926371 0.5933326  0.07805371
7
## G          0.43877638 0.38436294 1.0000000 0.5434578 0.7269852  0.18829865
3
## A          0.46230400 0.59263707 0.5434578 1.0000000 0.9714875  0.15090415
8
## P          0.50208965 0.59333262 0.7269852 0.9714875 1.0000000  0.17662171
0
## plus_minus 0.04765473 0.07805372 0.1882987 0.1509042 0.1766217  1.00000000
0
## PIM        0.18768978 0.52503350 0.2243964 0.2531786 0.2704769 -0.01156341
1
## PperGP     0.53892738 0.37613683 0.6806634 0.8837416 0.9151414  0.18670947
9
## EVG        0.33423738 0.36164743 0.9171186 0.4542442 0.6306000  0.24667509
7
## EVP        0.44387831 0.65591805 0.6421331 0.8830397 0.9036846  0.26489883
0
## PPG        0.40443912 0.24210271 0.7048778 0.4476925 0.5652954 -0.00692995
4
## PPP        0.42472169 0.34186677 0.6235477 0.8138997 0.8418794  0.00362728
3
## SHG        0.18400309 0.11760204 0.1232040 0.1349607 0.1451947  0.08075430
0
## SHP        0.16712284 0.14669964 0.1697434 0.1160291 0.1428534  0.13877775
4
## OTG        0.12318907 0.08805761 0.1989225 0.2163868 0.2331867  0.18898242
3
```

```
## GWG        0.24189822 0.24618945 0.6073766 0.2541496 0.3794403  0.16818053
7
## S          0.51731854 0.74335396 0.6388341 0.7823879 0.8204205  0.09919940
2
## TOI        0.58633036 0.59409893 0.5513459 0.7074666 0.7344251  0.10259192
8
##                    PIM    PperGP        EVG        EVP           PPG
PPP
## Salary      0.18768978 0.5389274 0.33423738 0.44387831  0.404439124 0.4247
21686
## GP          0.52503350 0.3761368 0.36164743 0.65591805  0.242102713 0.3418
66771
## G           0.22439642 0.6806634 0.91711857 0.64213313  0.704877794 0.6235
47654
## A           0.25317856 0.8837416 0.45424418 0.88303966  0.447692531 0.8138
99747
## P           0.27047693 0.9151414 0.63060000 0.90368464  0.565295396 0.8418
79393
## plus_minus -0.01156341 0.1867095 0.24667510 0.26489883 -0.006929954 0.0036
27283
## PIM         1.00000000 0.1488959 0.21420917 0.30246589  0.117079106 0.1425
38863
## PperGP      0.14889586 1.0000000 0.59359570 0.80961509  0.528272134 0.7927
49261
## EVG         0.21420917 0.5935957 1.00000000 0.63268491  0.373130938 0.4451
68385
## EVP         0.30246589 0.8096151 0.63268491 1.00000000  0.370918958 0.5343
23069
## PPG         0.11707911 0.5282721 0.37313094 0.37091896  1.000000000 0.6560
39576
## PPP         0.14253886 0.7927493 0.44516838 0.53432307  0.656039576 1.0000
00000
## SHG         0.17177863 0.1111783 0.09136408 0.10812851 -0.055896301 0.1018
72927
## SHP         0.14897732 0.1270469 0.15017056 0.08677591  0.046104325 0.0668
66063
## OTG         0.01106507 0.2113783 0.20230596 0.23047938  0.113197510 0.1792
69047
## GWG         0.23192432 0.3245068 0.56744527 0.32408227  0.392595209 0.3265
07616
## S           0.42019891 0.6706758 0.54338204 0.77906076  0.500394376 0.6378
57373
## TOI         0.25957973 0.6280183 0.46909289 0.70720705  0.439931956 0.5538
86137
##                    SHG          SHP          OTG       GWG          S        T
OI
## Salary      0.18400309  0.16712284  0.12318907 0.2418982 0.5173185 0.58633
04
## GP          0.11760204  0.14669964  0.08805761 0.2461894 0.7433540 0.59409
89
```

```
## G           0.12320400  0.16974337  0.19892251 0.6073766 0.6388341 0.55134
59
## A           0.13496065  0.11602914  0.21638685 0.2541496 0.7823879 0.70746
66
## P           0.14519467  0.14285342  0.23318670 0.3794403 0.8204205 0.73442
51
## plus_minus  0.08075430  0.13877775  0.18898242 0.1681805 0.0991994 0.10259
19
## PIM         0.17177863  0.14897732  0.01106507 0.2319243 0.4201989 0.25957
97
## PperGP      0.11117827  0.12704689  0.21137834 0.3245068 0.6706758 0.62801
83
## EVG         0.09136408  0.15017056  0.20230596 0.5674453 0.5433820 0.46909
29
## EVP         0.10812851  0.08677591  0.23047938 0.3240823 0.7790608 0.70720
71
## PPG        -0.05589630  0.04610432  0.11319751 0.3925952 0.5003944 0.43993
20
## PPP         0.10187293  0.06686606  0.17926905 0.3265076 0.6378574 0.55388
61
## SHG         1.00000000  0.47462149 -0.01727737 0.1695899 0.2101078 0.13611
10
## SHP         0.47462149  1.00000000 -0.03871344 0.2069646 0.1665362 0.19270
24
## OTG        -0.01727737 -0.03871344  1.00000000 0.3021632 0.1408493 0.16488
72
## GWG         0.16958994  0.20696460  0.30216325 1.0000000 0.3917126 0.30206
38
## S           0.21010783  0.16653623  0.14084929 0.3917126 1.0000000 0.78172
29
## TOI         0.13611104  0.19270238  0.16488716 0.3020638 0.7817229 1.00000
00

corrplot(cor(Def_num_data), method = "circle", type = "upper", title = "Defen
semen")
```




```
# Focus on Forwards and remove predictors that show no strong correlation and
name this dataframe FWD
FWD <- Forwards_num_data %>% select(Salary, GP, G, P, PperGP, PPG, PPP, S, TO
I)
head(FWD)

##     Salary GP  G  P PperGP PPG PPP   S  TOI
## 1 15900000 27 21 36   1.33   8  10 112 1323
## 2 15000000 30 11 39   1.30   0  13  82 1358
## 3 14000000 30 17 52   1.73   6  22 119 1349
## 4 13000000 15  5 19   1.27   2   7  45 1229
## 5 12000000 26 13 28   1.08   5  10  87 1210
## 7 12000000 30  9 26   0.87   5   9  83 1090

pairs(FWD)
```



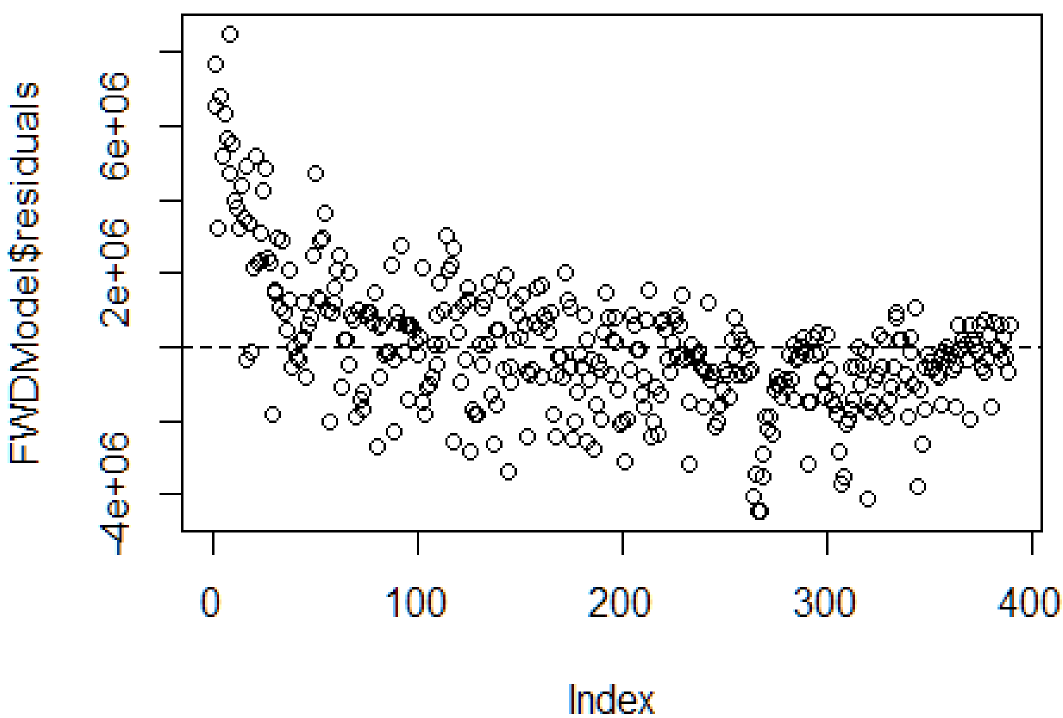
This gives us an idea which predictors are influential, and which ones are not helpful in predicting the salary. Note that points is autocorrelated with goals and assists.

I will only consider forwards from now on, because the predictors seem better suited to evaluate their value.

```
# Create a linear model with promising predictors
FWDModel = lm(Salary~ GP+P+PPP+S+TOI, data=FWD)
summary(FWDModel)

##
## Call:
## lm(formula = Salary ~ GP + P + PPP + S + TOI, data = FWD)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4440804 -1056066  -198149   790465  8511183
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1457163.3   652315.0  -2.234  0.02607 *
## GP            -67450.2    21208.2  -3.180  0.00159 **
```

```
## P                88636.9    29537.9   3.001  0.00287 **
## PPP              98434.8    49515.6   1.988  0.04753 *
## S                 8796.5     7873.5   1.117  0.26460
## TOI               4779.8      874.7   5.465 8.37e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1872000 on 383 degrees of freedom
## Multiple R-squared:   0.52,	Adjusted R-squared:  0.5137
## F-statistic: 82.97 on 5 and 383 DF,  p-value: < 2.2e-16
```

```
# Plot residuals - Salaries are not normally distributed
plot(FWDModel$residuals)
abline(h=0, lty=2)
```



Looking at the residuals, this does not look like a normal distribution, as we suspected. I will not pursue linear regression. Log-transforming the salary did not help either, data not shown. Let's try other approaches.

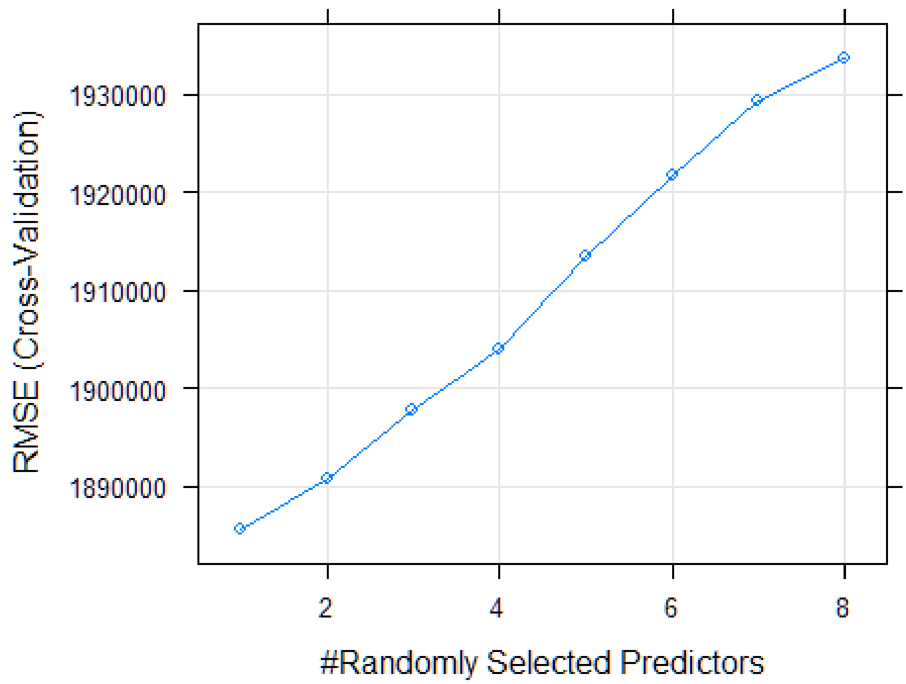
First, I am creating a training and a test set.

```
# I tried log-transforming salaries and it doesn't work either. Hence, I will
abandon linear regression and will instead focus on other techniques.

options(java.parameters = "-Xmx4g")
# I will use RMSE as my metric, and will define it as follows:
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}

# Then we split the dataset, 80:20 in this case, i.e., 80% of the data will g
o to the training set and 20% will go to the test set.
FWD1 = sort(sample(nrow(FWD), nrow(FWD)*.8))

#creating training data set by selecting the output row values
train<-FWD[FWD1,]

#creating test data set by not selecting the output row values
test<-FWD[-FWD1,]

head(train)

##     Salary GP  G  P PperGP PPG PPP   S  TOI
## 1 15900000 27 21 36   1.33   8  10 112 1323
## 3 14000000 30 17 52   1.73   6  22 119 1349
## 4 13000000 15  5 19   1.27   2   7  45 1229
## 5 12000000 26 13 28   1.08   5  10  87 1210
## 7 12000000 30  9 26   0.87   5   9  83 1090
## 8 12000000 27 10 30   1.11   3  11  49 1132

nrow(train)

## [1] 311

head(test)

##      Salary GP  G  P PperGP PPG PPP  S  TOI
## 2  15000000 30 11 39   1.30   0  13 82 1358
## 13 10000000 26 14 19   0.73   6   6 55 1034
## 20  9500000 24 12 25   1.04   6  10 63 1109
## 32  8000000 28 12 24   0.86   3   6 69 1179
## 37  8000000 25  3 12   0.48   0   2 54  957
## 45  7000000 25 12 28   1.12   7   8 60 1004

nrow(test)

## [1] 78
```

Random Forest.

```
# Random Forest
# We set up cross-validation 5 fold and create a grid of mtry values (Here, t
```

```r
rying all possible values.)
cv_5 = trainControl(method = "cv", number = 5)
rf_grid =  expand.grid(mtry = 1:8)

rf_fit = train(Salary ~ ., data = train,
               method = "rf",
               trControl = cv_5,
               tuneGrid = rf_grid)
rf_fit$bestTune

##   mtry
## 1    1

plot(rf_fit)
```

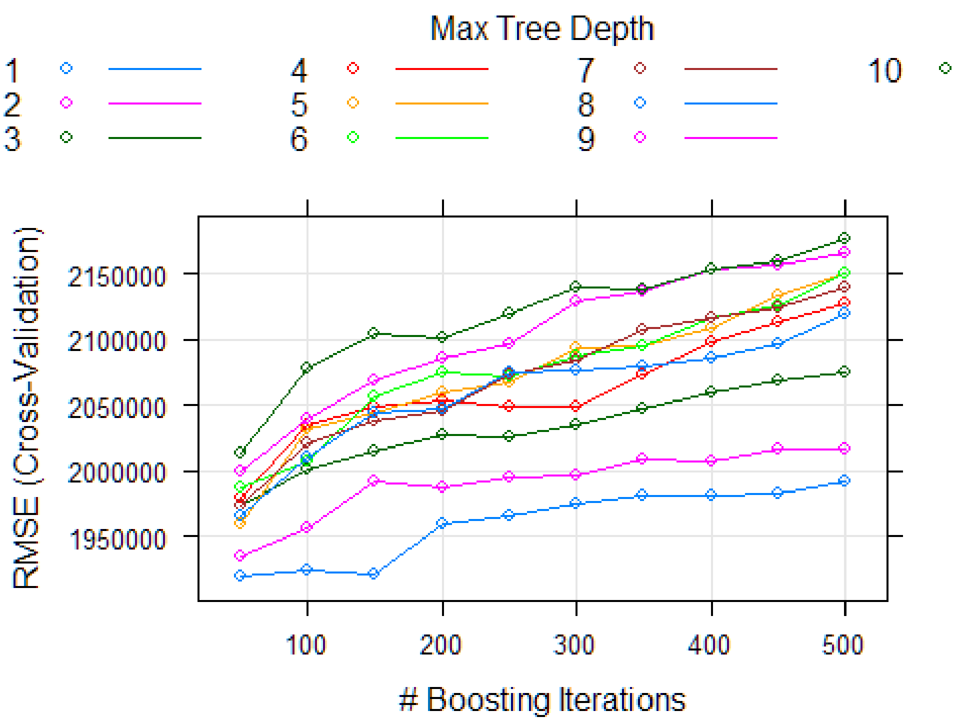



```r
rmse(predict(rf_fit, test), test$Salary)

## [1] 1804505

# The resulting test RMSE with mtry = 2 is 1791422
```

Extremely Randomized Trees. Sorry won't run with this PC because my Java version is incompatible. Not the best approach anyways. Code works if the software environment is correct. RMSE: 2014886.

Next, Generalized Boosted Regression Modeling, gbm.

```
# Generalized Boosted Regression Modeling, gbm
gbm_fit = train(Salary ~ ., data = train,
                method = "gbm",
                trControl = cv_5,
                verbose = FALSE,
                tuneLength = 10)

gbm_fit$bestTune

##   n.trees interaction.depth shrinkage n.minobsinnode
## 1      50                 1       0.1             10

plot(gbm_fit)
```




```
rmse(predict(gbm_fit, test), test$Salary)

## [1] 1792469
```

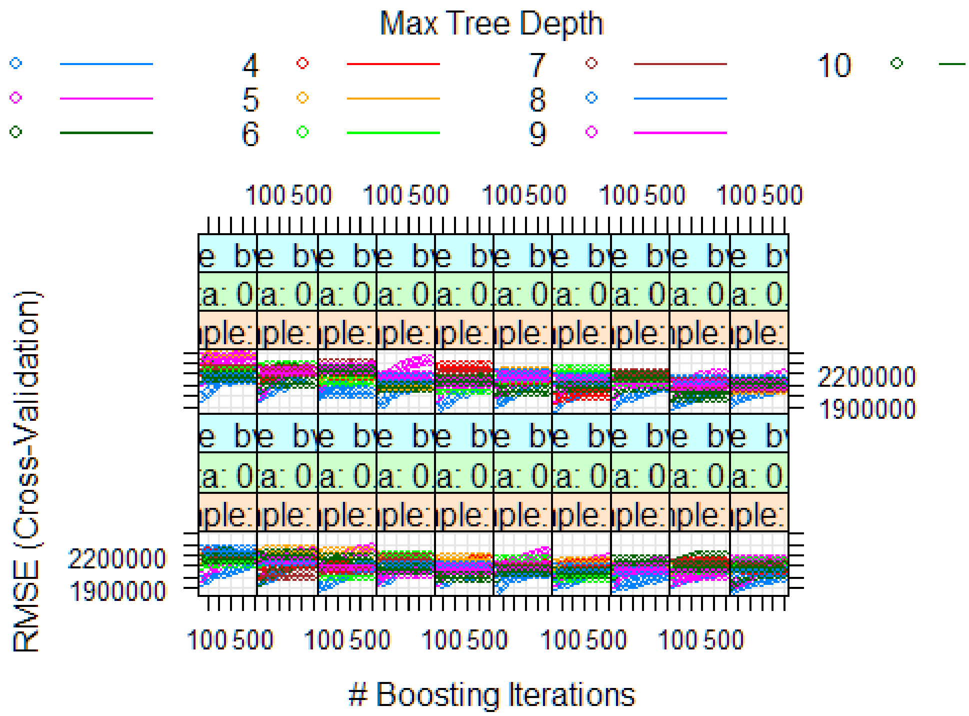
Extreme Gradient Boosting, xgboost. Tons of warning messages while processing.

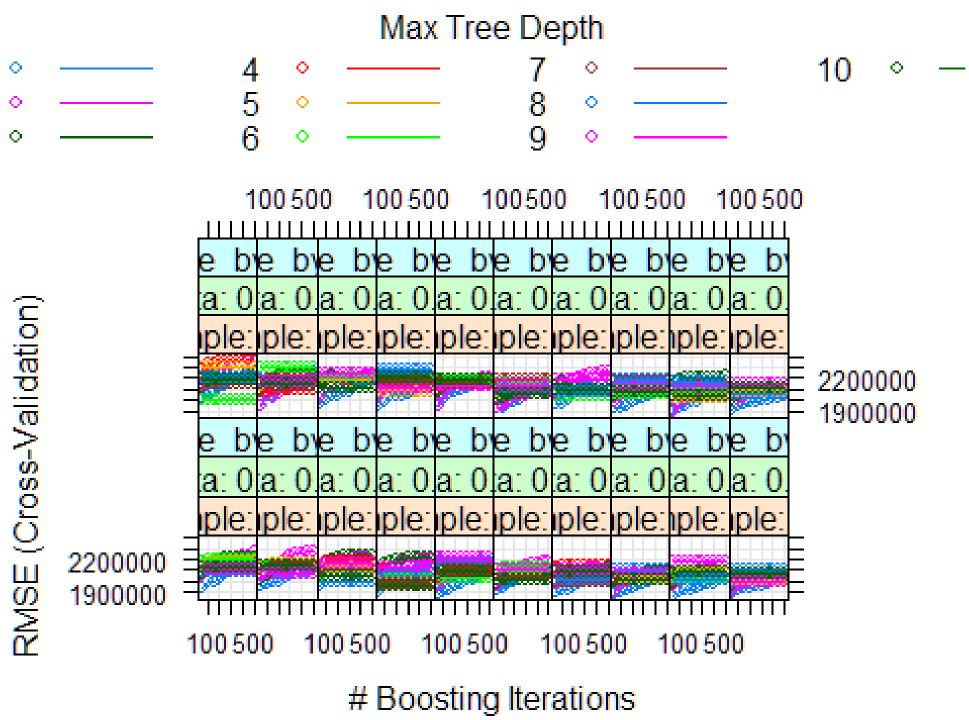
```
# Extreme Gradient Boosting, xgboost. I get tons of warning messages.
xgb_fit = train(Salary ~ ., data = train,
                method = "xgbTree",
                trControl = cv_5,
```

```
                 verbose = FALSE,
                 tuneLength = 10,
                 numThreads = 8)
```

-warning messages deleted-

```
plot(xgb_fit)
```






```
rmse(predict(xgb_fit, test), test$Salary)

## [1] 1796426
```

Finally, I wanted to have a comparison with h2o models. Turns out these aren't much better than the models described above.

```
# Finally, let's see how h2o models compare
h2o.init(nthreads = -1)

##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 days 2 hours
##     H2O cluster timezone:       Europe/Berlin
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.5
##     H2O cluster version age:    2 days
##     H2O cluster name:           H2O_started_from_R_ughac_dkv520
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   3.33 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetE
ncoder, Core V4
##     R Version:                  R version 4.0.4 (2021-02-15)

Fwd_concise = read.csv("Fwd_concise.csv", header = T)
head(Fwd_concise)

##              Player Team   Salary S.C Pos GP  G  A  P plus_minus PIM PperG
P EVG
## 1   Auston Matthews  TOR 15900000   L   C 27 21 15 36          8   6   1.3
3  13
## 2    Connor McDavid  EDM 14000000   L   C 30 17 35 52         10  14   1.7
3  11
## 3      John Tavares  TOR 12000000   L   C 30  9 17 26         11   6   0.8
7   4
## 4 Nicklas Backstrom  WSH 12000000   L   C 27 10 20 30         -1   4   1.1
1   7
## 5     Sebastian Aho  CAR 10570000   L   C 27 10 15 25          7  12   0.9
3   6
## 6       Jack Eichel  BUF 10000000   R   C 21  2 16 18         -9   6   0.8
6   1
##   EVP PPG PPP SHG SHP OTG GWG   S  TOI
## 1  26   8  10   0   0   1   7 112 1323
## 2  30   6  22   0   0   1   4 119 1349
## 3  17   5   9   0   0   0   1  83 1090
## 4  19   3  11   0   0   0   0  49 1132
```

```
## 5  16   3   8   1   1   0   2  66 1160
## 6   9   1   9   0   0   0   1  61 1230

names(Fwd_concise) <- c("Player", "Salary", "GP", "G", "P", "PperGP", "PPG",
"PPP", "S", "TOI")
head(names)

##
## 1 .Primitive("names")

FWD_h2o <- h2o.importFile("FWD_concise.csv")

##   |
|                                                                      |   0%
|
|======================================================================| 100%

# Create the training dataset and test dataset (80% and 20%)
partitions <- h2o.splitFrame(data = as.h2o(FWD_h2o),
                             ratios = c(0.8),
                             seed = 1)

data_train_h2o   <- h2o.assign(data = partitions[[1]], key = "data_train_H2O"
)
data_test_h2o    <- h2o.assign(data = partitions[[2]], key = "data_test_H2O")
y1  <-  "Salary"
x1  <-  setdiff(names(data_train_h2o), y1)

# Applies the H2O AutoML Machine Learning Platform
aml <- h2o.automl(x = x1, y = y1,
                  training_frame = data_train_h2o,
                  validation_frame = data_test_h2o,
                  stopping_metric = "RMSE",
                  seed = 1,
                  sort_metric = "RMSE")

##   |
|                                                                      |   0%
## 22:49:47.685: User specified a validation frame with cross-validation stil
l enabled. Please note that the models will still be validated using cross-va
lidation only, the validation frame will be used to provide purely informativ
e validation metrics on the trained models.
## 22:49:47.685: AutoML: XGBoost is not available; skipping it.  |
|=========                                                             |  12%
|
|===============                                                       |  21%
|
|==================                                                    |  25%
|
|=======================                                               |  33%
|
```

```
|===========================================                          |  62%
|
|===========================================                          |  63%
|
|============================================                         |  64%
|
|============================================                         |  65%
|
|=============================================                        |  65%
|
|=============================================                        |  66%
|
|=================================================                    |  71%
|
|=================================================                    |  72%
|
|==================================================                   |  72%
|
|==================================================                   |  73%
|
|===================================================                  |  74%
|
|===================================================                  |  75%
|
|======================================================               |  79%
|
|=======================================================              |  80%
|
|========================================================             |  81%
|
|========================================================             |  82%
|
|=========================================================            |  82%
|
|=========================================================            |  83%
|
|==========================================================           |  84%
|
|==========================================================           |  85%
|
|===========================================================          |  85%
|
|===========================================================          |  86%
|
|============================================================         |  87%
|
|============================================================         |  88%
|
|===============================================================      |  92%
```

```
|
|======================================================================| 100%

lb <- aml@leaderboard
print(lb, n = nrow(lb))

##                                                 model_id    rmse
## 1                        XRT_1_AutoML_20210318_224947 1923438
## 2   StackedEnsemble_BestOfFamily_AutoML_20210318_224947 1925153
## 3                        DRF_1_AutoML_20210318_224947 1937265
## 4                        GBM_1_AutoML_20210318_224947 1976159
## 5           GBM_grid__1_AutoML_20210318_224947_model_5 1979149
## 6                        GBM_4_AutoML_20210318_224947 1989975
## 7           GBM_grid__1_AutoML_20210318_224947_model_7 1992915
## 8                        GBM_2_AutoML_20210318_224947 1995185
## 9                        GBM_3_AutoML_20210318_224947 1996622
## 10          GBM_grid__1_AutoML_20210318_224947_model_6 2009496
## 11          GBM_grid__1_AutoML_20210318_224947_model_2 2015722
## 12          GBM_grid__1_AutoML_20210318_224947_model_4 2022996
## 13          GBM_grid__1_AutoML_20210318_224947_model_1 2033074
## 14  DeepLearning_grid__1_AutoML_20210318_224947_model_4 2040069
## 15  DeepLearning_grid__2_AutoML_20210318_224947_model_6 2086859
## 16  DeepLearning_grid__1_AutoML_20210318_224947_model_5 2094879
## 17  DeepLearning_grid__2_AutoML_20210318_224947_model_4 2104743
## 18  DeepLearning_grid__3_AutoML_20210318_224947_model_6 2107548
## 19  DeepLearning_grid__3_AutoML_20210318_224947_model_4 2115614
## 20  DeepLearning_grid__1_AutoML_20210318_224947_model_3 2119638
## 21  DeepLearning_grid__3_AutoML_20210318_224947_model_2 2123925
## 22  DeepLearning_grid__1_AutoML_20210318_224947_model_7 2130639
## 23  DeepLearning_grid__2_AutoML_20210318_224947_model_5 2131762
## 24                       GBM_5_AutoML_20210318_224947 2142509
## 25  DeepLearning_grid__1_AutoML_20210318_224947_model_1 2144648
## 26  DeepLearning_grid__1_AutoML_20210318_224947_model_6 2146826
## 27  DeepLearning_grid__1_AutoML_20210318_224947_model_2 2148779
## 28  DeepLearning_grid__2_AutoML_20210318_224947_model_2 2148787
## 29          GBM_grid__1_AutoML_20210318_224947_model_3 2151576
## 30  DeepLearning_grid__2_AutoML_20210318_224947_model_7 2216577
## 31  DeepLearning_grid__3_AutoML_20210318_224947_model_1 2219481
## 32  DeepLearning_grid__2_AutoML_20210318_224947_model_1 2221836
## 33  DeepLearning_grid__3_AutoML_20210318_224947_model_9 2222617
## 34 DeepLearning_grid__3_AutoML_20210318_224947_model_14 2236299
## 35 DeepLearning_grid__3_AutoML_20210318_224947_model_11 2240925
## 36 DeepLearning_grid__3_AutoML_20210318_224947_model_12 2254824
## 37  DeepLearning_grid__3_AutoML_20210318_224947_model_3 2270675
## 38  DeepLearning_grid__3_AutoML_20210318_224947_model_7 2279337
## 39  DeepLearning_grid__3_AutoML_20210318_224947_model_5 2280686
## 40  DeepLearning_grid__3_AutoML_20210318_224947_model_8 2330969
## 41                DeepLearning_1_AutoML_20210318_224947 2331625
## 42  DeepLearning_grid__2_AutoML_20210318_224947_model_3 2363551
## 43 DeepLearning_grid__3_AutoML_20210318_224947_model_10 2470764
```

```
## 44     StackedEnsemble_AllModels_AutoML_20210318_224947 2768827
## 45                         GLM_1_AutoML_20210318_224947 2770200
## 46 DeepLearning_grid__3_AutoML_20210318_224947_model_13 3183626
##    mean_residual_deviance          mse     mae     rmsle
## 1            3.699614e+12 3.699614e+12 1379985 0.6361620
## 2            3.706213e+12 3.706213e+12 1364086 0.6238121
## 3            3.752996e+12 3.752996e+12 1331328 0.6168050
## 4            3.905205e+12 3.905205e+12 1379592 0.6477449
## 5            3.917031e+12 3.917031e+12 1391064 0.6316241
## 6            3.960001e+12 3.960001e+12 1395110 0.6287476
## 7            3.971711e+12 3.971711e+12 1394647 0.6307653
## 8            3.980764e+12 3.980764e+12 1408327 0.6232471
## 9            3.986501e+12 3.986501e+12 1406207 0.6324638
## 10           4.038073e+12 4.038073e+12 1408592 0.6287941
## 11           4.063136e+12 4.063136e+12 1403251 0.6359773
## 12           4.092515e+12 4.092515e+12 1445739 0.6642061
## 13           4.133389e+12 4.133389e+12 1435456 0.6381711
## 14           4.161883e+12 4.161883e+12 1492363 0.7007795
## 15           4.354979e+12 4.354979e+12 1471578 0.6999430
## 16           4.388519e+12 4.388519e+12 1577195       NaN
## 17           4.429944e+12 4.429944e+12 1483405       NaN
## 18           4.441759e+12 4.441759e+12 1528982 0.7242408
## 19           4.475823e+12 4.475823e+12 1561941 0.7391848
## 20           4.492863e+12 4.492863e+12 1537420       NaN
## 21           4.511056e+12 4.511056e+12 1533009 0.6969662
## 22           4.539621e+12 4.539621e+12 1556328       NaN
## 23           4.544411e+12 4.544411e+12 1581337 0.7381308
## 24           4.590344e+12 4.590344e+12 1485106 0.6669821
## 25           4.599517e+12 4.599517e+12 1548487       NaN
## 26           4.608863e+12 4.608863e+12 1546240       NaN
## 27           4.617250e+12 4.617250e+12 1606584       NaN
## 28           4.617284e+12 4.617284e+12 1617629 0.7409023
## 29           4.629278e+12 4.629278e+12 1495947 0.6704016
## 30           4.913213e+12 4.913213e+12 1670437 0.7569916
## 31           4.926096e+12 4.926096e+12 1684527 0.7682585
## 32           4.936554e+12 4.936554e+12 1650548 0.7689965
## 33           4.940024e+12 4.940024e+12 1572785       NaN
## 34           5.001034e+12 5.001034e+12 1648111       NaN
## 35           5.021743e+12 5.021743e+12 1702681 0.7646135
## 36           5.084232e+12 5.084232e+12 1661689 0.7544574
## 37           5.155967e+12 5.155967e+12 1652783 0.7612042
## 38           5.195377e+12 5.195377e+12 1741074 0.7842688
## 39           5.201531e+12 5.201531e+12 1768061 0.7913690
## 40           5.433418e+12 5.433418e+12 1630328 0.7594464
## 41           5.436475e+12 5.436475e+12 1767910       NaN
## 42           5.586372e+12 5.586372e+12 1811536 0.8285262
## 43           6.104674e+12 6.104674e+12 2004179 0.9080212
## 44           7.666403e+12 7.666403e+12 2143457 0.9391688
## 45           7.674006e+12 7.674006e+12 2143914 0.9394052
## 46           1.013548e+13 1.013548e+13 2656635       NaN
```

```
##
## [46 rows x 6 columns]
```

```
aml@leader
```

```
## Model Details:
## ==============
##
## H2ORegressionModel: drf
## Model ID:  XRT_1_AutoML_20210318_224947
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1              35                       35               94972        16
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1        20   19.17143        146        213   195.00000
##
##
## H2ORegressionMetrics: drf
## ** Reported on training data. **
## ** Metrics reported on Out-Of-Bag training samples **
##
## MSE:  3.978484e+12
## RMSE:  1994614
## MAE:  1441659
## RMSLE:  0.6565352
## Mean Residual Deviance :  3.978484e+12
##
##
## H2ORegressionMetrics: drf
## ** Reported on validation data. **
##
## MSE:  3.144704e+12
## RMSE:  1773331
## MAE:  1239874
## RMSLE:  0.6654103
## Mean Residual Deviance :  3.144704e+12
##
##
## H2ORegressionMetrics: drf
## ** Reported on cross-validation data. **
## ** 5-fold cross-validation on training data (Metrics computed for combined
holdout predictions) **
##
## MSE:  3.699614e+12
## RMSE:  1923438
## MAE:  1379985
## RMSLE:  0.636162
## Mean Residual Deviance :  3.699614e+12
##
##
```

```
## Cross-Validation Metrics Summary:
##                                mean            sd    cv_1_valid    cv_2_v
alid
## mae                       1379842.6     143136.69     1210061.6     15941
62.2
## mean_residual_deviance 3.69990618E12 5.47875029E11 3.02002104E12 4.2887500
3E12
## mse                    3.69990618E12 5.47875029E11 3.02002104E12 4.2887500
3E12
## r2                        0.5041692   0.103992954     0.6245669     0.535
1587
## residual_deviance      3.69990618E12 5.47875029E11 3.02002104E12 4.2887500
3E12
## rmse                      1919185.6     144191.27     1737820.8     20709
29.8
## rmsle                    0.63391995   0.059637815     0.5621784    0.7029
0357
##                          cv_3_valid    cv_4_valid    cv_5_valid
## mae                       1428679.9     1348144.8     1318164.9
## mean_residual_deviance 3.90807054E12 4.05456185E12 3.22812799E12
## mse                    3.90807054E12 4.05456185E12 3.22812799E12
## r2                        0.5564315    0.35579178     0.4488973
## residual_deviance      3.90807054E12 4.05456185E12 3.22812799E12
## rmse                      1976884.0     2013594.2     1796699.2
## rmsle                     0.5824985      0.650782    0.67123735
```

```
# test prediction of the leader model
pred <- h2o.predict(aml, data_test_h2o)
```

```
##   |
|                                                                      |   0%
|
|======================================================================| 100%
```

```
# retrieve the leaderboard
lb <- h2o.get_leaderboard(object = aml, extra_columns = 'ALL')
lb
```

```
##                                              model_id    rmse
## 1                        XRT_1_AutoML_20210318_224947 1923438
## 2 StackedEnsemble_BestOfFamily_AutoML_20210318_224947 1925153
## 3                        DRF_1_AutoML_20210318_224947 1937265
## 4                        GBM_1_AutoML_20210318_224947 1976159
## 5          GBM_grid__1_AutoML_20210318_224947_model_5 1979149
## 6                        GBM_4_AutoML_20210318_224947 1989975
##   mean_residual_deviance          mse     mae     rmsle training_time_ms
## 1           3.699614e+12 3.699614e+12 1379985 0.6361620              143
## 2           3.706213e+12 3.706213e+12 1364086 0.6238121              127
## 3           3.752996e+12 3.752996e+12 1331328 0.6168050               89
## 4           3.905205e+12 3.905205e+12 1379592 0.6477449               33
## 5           3.917031e+12 3.917031e+12 1391064 0.6316241               14
```

```
## 6           3.960001e+12 3.960001e+12 1395110 0.6287476               35
##   predict_time_per_row_ms           algo
## 1                0.041775            DRF
## 2                0.042058 StackedEnsemble
## 3                0.007266            DRF
## 4                0.004837            GBM
## 5                0.003211            GBM
## 6                0.004408            GBM
##
## [46 rows x 9 columns]
```

Conclusions:

There are several methods that produces similar RMSE values, including Random Forest, gbm, xgboost, and h2o models. I could not create a linear regression model. It is inherently difficult to predict salaries in professional sports. There are many reasons why a closer fit may not be achievable, for instance the time when a contract was signed.