PH125.9x - Capstone Project - Your Own Project - Predicting NBA players salary

Matthew Manasterski, MBA

02/01/2020

Contents

1 Introduction

The following report is for the second Capstone choose-your-own project for the course: HarvardX - PH125.9x, which is based on choosing your own dataset from available sources like Kaggle or UCI Machine Learning Repository, choosing your own Methodology and Data Anaslysis, showcasing what you have learned in the course thru data exploration, visualization and data manipulation. Running the dataset thru Machine Learning Models and discussing the results.

2 Overview

In NBA, like many professional sports, salaries are not equally distributed among the league. There are few players at the top making very large sums of money, multiple times of the salaries that majority of the players make, skewing the league salary average far away from the median. The aim of this choose-your-own capstone project is to predict if an NBA players salary is above or below/equal league average using 3 seasons of data and basing the decision on criteria like player position, Team, offensive stats, defensive stats, and if the player played in the all-star game. The approach we will take on this project is to first get familiar with the data by exploring and visualizing the data, manipulating the data and then running models on it to make our predictions.

3 Data

The Data for this Capstone project has been sourced from publicly available dataset available on Kaggle website https://www.kaggle.com/davra98/nba-players-20162019

This Dataset was created for an University project in Milan for a different project to predict the All Star Game score for each player. This Data contains various stats and salaries for last three NBA seasons from 2016-2017 season to 2018-2019 season. This dataset contains 1408 observations with 45 variables for each observation. Although the data has 45 observations we will only choose few variables that will help us predict how good the player is and determine if he is payed above or below league average salary.

Some variables of interest in the dataset:
POS1 = Main position (some players have a second position called POS2)
G = Games played
GS = Games started
MP = Minutes played
FG = Field Goals Per Game
FGA = Field Goal Attempts Per Game
FG.= Field Goal Percentage
X3P = 3-Point Field Goals Per Game
X3PA = 3-Point Field Goal Attempts Per Game
X3P. = FG% on 3-Pt FGAs.
X2P = 2-Point Field Goals Per Game
X2PA = 2-Point Field Goal Attempts Per Game
X2P. = FG% on 2-Pt FGAs.
eFG. = Effective Field Goal Percentage
FT = Free Throws Per Game
FTA = Free Throw Attempts Per Game
FT.= Free Throw Percentage
ORB = Offensive Rebounds Per Game
DRB = Defensive Rebounds Per Game
TRB = Total Rebounds Per Game
AST = Assists Per Game
STL= Steals Per Game
BLK = Blocks Per Game

TOV = Turnovers Per Game
PF = Personal Fouls Per Game
PTS = Points Per Game
MEAN_VIEWS = Daily views on wikipedia
PLAY = If the player played in the all star game

4 Methods and Analysis

We will first get the data by reading the csv data file "nba_final.csv" and inspect the dataset with head, str, and summary functions. Based on what we see in the dataset from these functions and our basic knowledge of the NBA game, we will then pick a subset of variables that we think are useful to us. We will then use data visualization techniques to get familiar with the data and confirm our variables. We will then transform Salary column into 2 factor AboveAvg column which will state if the player has above league average salary or that of below/equal league average. Next we will split the dataset into train and test sets and run our Machine Learning models.

4.1 Data - Getting and Exploring the Data

We start by loading the necessary libraries we know we will need.

```
# Load necessary libraries
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(caTools)) install.packages("caTools", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
```

We then load our downloaded dataset from the working directory.

```
# Read in the dataset
nba_players <- read.csv('nba_final.csv')
```

Let's explore the loaded dataset with looking at the first few columns.

```
# Explore the structure of the dataset by looking at the head columnns
head(nba_players)
```

```
##    Rk         Player.x Player_ID Pos1 Pos2 Age  Tm  G GS   MP  FG  FGA   FG. X3P
## 1 170     A.J. Hammons hammoaj01    C <NA>  24 DAL 22  0  7.4 0.8  1.9 0.405 0.2
## 2  58     Aaron Brooks brookaa01   PG <NA>  32 IND 65  0 13.8 1.9  4.6 0.403 0.7
## 3 157     Aaron Gordon gordoaa01   SF <NA>  21 ORL 80 72 28.7 4.9 10.8 0.454 1.0
## 4 352    Adreian Payne paynead01   PF <NA>  25 MIN 18  0  7.5 1.3  3.0 0.426 0.2
## 5  10 Al-Farouq Aminu aminual01   PF <NA>  26 POR 61 25 29.1 3.0  7.6 0.393 1.1
## 6 203      Al Horford horfoal01    C <NA>  30 BOS 68 68 32.3 5.6 11.8 0.473 1.3
##   X3PA  X3P. X2P X2PA  X2P.  eFG.  FT FTA   FT. ORB DRB TRB AST STL BLK TOV  PF
## 1  0.5 0.500 0.5  1.5 0.375 0.464 0.4 0.9 0.450 0.4 1.3 1.6 0.2 0.0 0.6 0.5 1.0
## 2  2.0 0.375 1.1  2.6 0.424 0.483 0.5 0.6 0.800 0.3 0.8 1.1 1.9 0.4 0.1 1.0 1.4
## 3  3.3 0.288 4.0  7.5 0.528 0.499 2.0 2.7 0.719 1.5 3.6 5.1 1.9 0.8 0.5 1.1 2.2
## 4  0.8 0.200 1.1  2.2 0.513 0.454 0.8 1.1 0.737 0.5 1.3 1.8 0.4 0.4 0.4 0.4 1.8
## 5  3.5 0.330 1.9  4.2 0.445 0.468 1.6 2.2 0.706 1.3 6.1 7.4 1.6 1.0 0.7 1.5 1.7
## 6  3.6 0.355 4.3  8.2 0.524 0.527 1.6 2.0 0.800 1.4 5.4 6.8 5.0 0.8 1.3 1.7 2.0
##    PTS   Salary mean_views  Season Conference  Role  Fvot FRank Pvot PRank Mvot
## 1  2.2       NA    3.32000 2016-17       West Front   786   123   NA    NA   NA
## 2  5.0  2700000   11.15574 2016-17        Est  Back  2474    64   NA    NA   NA
## 3 12.7  4351320 1713.98634 2016-17        Est Front 22774    29   NA    NA   NA
```

```
## 4  3.5  2022240  205.85519 2016-17       West Front   861   120    1    52   NA
## 5  8.7  7680965  604.34153 2016-17       West Front  4971    69    7    23   NA
## 6 14.0 26540100 1556.38251 2016-17        Est Front 12219    14    8    16    2
##   MRank Score Play
## 1    NA  83.5   No
## 2    NA  48.2   No
## 3    NA  40.0   No
## 4    NA  75.5   No
## 5    NA  42.8   No
## 6     6  12.5   No
```

We see right away that this dataset has many variables that are mostly numeric as they represent players statistics, there are also some other values like players names, players id and positon.

We will take closer look at the structure of the dataset.

```r
str(nba_players)
```

```
## 'data.frame':    1408 obs. of  45 variables:
##  $ Rk        : int  170 58 157 352 10 203 221 12 464 65 ...
##  $ Player.x  : Factor w/ 660 levels "A.J. Hammons",..: 1 2 3 6 9 7 8 10 11 12 ...
##  $ Player_ID : Factor w/ 660 levels "abrinal01","acyqu01",..: 240 87 224 482 16 277 305 17 629 101 .
##  $ Pos1      : Factor w/ 5 levels "C","PF","PG",..: 1 3 4 2 2 1 1 4 1 5 ...
##  $ Pos2      : Factor w/ 4 levels "C","PF","SF",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ Age       : int  24 32 21 25 26 30 32 34 24 25 ...
##  $ Tm        : Factor w/ 30 levels "ATL","BOS","BRK",..: 7 12 22 18 25 2 12 13 24 29 ...
##  $ G         : int  22 65 80 18 61 68 66 30 47 42 ...
##  $ GS        : int  0 0 72 0 25 68 1 0 0 0 ...
##  $ MP        : num  7.4 13.8 28.7 7.5 29.1 32.3 14.1 10.3 15.1 15.5 ...
##  $ FG        : num  0.8 1.9 4.9 1.3 3 5.6 3.6 1 2.9 2.4 ...
##  $ FGA       : num  1.9 4.6 10.8 3 7.6 11.8 7.1 2.7 5.7 5.9 ...
##  $ FG.       : num  0.405 0.403 0.454 0.426 0.393 0.473 0.499 0.375 0.517 0.399 ...
##  $ X3P       : num  0.2 0.7 1 0.2 1.1 1.3 0 0.5 0 0.6 ...
##  $ X3PA      : num  0.5 2 3.3 0.8 3.5 3.6 0 1.5 0 1.8 ...
##  $ X3P.      : num  0.5 0.375 0.288 0.2 0.33 0.355 0 0.318 0 0.329 ...
##  $ X2P       : num  0.5 1.1 4 1.1 1.9 4.3 3.6 0.5 2.9 1.8 ...
##  $ X2PA      : num  1.5 2.6 7.5 2.2 4.2 8.2 7.1 1.2 5.7 4.1 ...
##  $ X2P.      : num  0.375 0.424 0.528 0.513 0.445 0.524 0.5 0.444 0.519 0.43 ...
##  $ eFG.      : num  0.464 0.483 0.499 0.454 0.468 0.527 0.499 0.463 0.517 0.45 ...
##  $ FT        : num  0.4 0.5 2 0.8 1.6 1.6 1 0.4 1.5 1.4 ...
##  $ FTA       : num  0.9 0.6 2.7 1.1 2.2 2 1.3 0.5 2.4 1.9 ...
##  $ FT.       : num  0.45 0.8 0.719 0.737 0.706 0.8 0.765 0.75 0.625 0.769 ...
##  $ ORB       : num  0.4 0.3 1.5 0.5 1.3 1.4 1.1 0.1 2 0.4 ...
##  $ DRB       : num  1.3 0.8 3.6 1.3 6.1 5.4 3.1 0.7 4.2 2.5 ...
##  $ TRB       : num  1.6 1.1 5.1 1.8 7.4 6.8 4.2 0.8 6.2 2.9 ...
##  $ AST       : num  0.2 1.9 1.9 0.4 1.6 5 0.9 0.4 0.5 0.7 ...
##  $ STL       : num  0 0.4 0.8 0.4 1 0.8 0.3 0.1 0.6 0.4 ...
##  $ BLK       : num  0.6 0.1 0.5 0.4 0.7 1.3 0.2 0 0.7 0.1 ...
##  $ TOV       : num  0.5 1 1.1 0.4 1.5 1.7 0.5 0.2 0.8 0.8 ...
##  $ PF        : num  1 1.4 2.2 1.8 1.7 2 1.9 1.2 2.7 1.2 ...
##  $ PTS       : num  2.2 5 12.7 3.5 8.7 14 8.1 2.9 7.4 6.7 ...
##  $ Salary    : int  NA 2700000 4351320 2022240 7680965 26540100 10230179 1315448 874636 9904495 ...
##  $ mean_views: num  3.32 11.16 1713.99 205.86 604.34 ...
##  $ Season    : Factor w/ 3 levels "2016-17","2017-18",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Conference: Factor w/ 2 levels "Est","West": 2 1 1 2 2 1 1 2 2 2 ...
##  $ Role      : Factor w/ 2 levels "Back","Front": 2 1 2 2 2 2 2 2 2 1 ...
```

```
##  $ Fvot      : int  786 2474 22774 861 4971 12219 2936 3096 607 1981 ...
##  $ FRank     : int  123 64 29 120 69 14 84 88 127 72 ...
##  $ Pvot      : int  NA NA NA 1 7 8 1 1 NA NA ...
##  $ PRank     : int  NA NA NA 52 23 16 60 52 NA NA ...
##  $ Mvot      : int  NA NA NA NA NA 2 NA NA NA NA ...
##  $ MRank     : int  NA NA NA NA NA 6 NA NA NA NA ...
##  $ Score     : num  83.5 48.2 40 75.5 42.8 12.5 60 59.5 85.5 53 ...
##  $ Play      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

From the structure above we can see that the dataset has 1408 observations across 45 variables. For the purpose of this project 45 variables is way too many, and we will have to get rid of those not useful to us. Right away we can see variables that will not be useful to us like Player.x and Player_ID, both of which have Factor of 660, these are player names and player ids, the purpose of this project is to determine if the player is above average salary or below, we do not need to know the player's name. Before we get rid of any variables lets lastly look at the dataset summary.

```
#Display Summary of the set
summary(nba_players)
```

```
##        Rk                    Player.x        Player_ID     Pos1       Pos2
##  Min.   :  1.0   Aaron Gordon    :   3   abrinal01:   3   C :277   C    :   2
##  1st Qu.:127.0   Al Horford      :   3   adamsst01:   3   PF:295   PF   :   2
##  Median :257.0   Al-Farouq Aminu:   3   aldrila01:   3   PG:282   SF   :   3
##  Mean   :257.7   Alec Burks      :   3   aminual01:   3   SF:230   SG   :   5
##  3rd Qu.:385.2   Álex Abrines    :   3   anderju01:   3   SG:324   NA's:1396
##  Max.   :540.0   Alex Len        :   3   anderky01:   3
##                  (Other)         :1390   (Other)  :1390
##       Age             Tm              G               GS               MP
##  Min.   :19.00   MIL    :  51   Min.   : 1.0   Min.   : 0.00   Min.   : 0.70
##  1st Qu.:23.00   PHI    :  51   1st Qu.:36.0   1st Qu.: 1.00   1st Qu.:13.30
##  Median :25.00   DET    :  50   Median :62.0   Median :13.00   Median :20.00
##  Mean   :26.14   IND    :  50   Mean   :54.1   Mean   :25.91   Mean   :20.16
##  3rd Qu.:29.00   LAC    :  50   3rd Qu.:74.0   3rd Qu.:52.00   3rd Qu.:27.50
##  Max.   :42.00   DEN    :  49   Max.   :82.0   Max.   :82.00   Max.   :37.80
##                  (Other):1107
##        FG               FGA               FG.               X3P
##  Min.   : 0.000   Min.   : 0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 1.600   1st Qu.: 3.700   1st Qu.:0.4040   1st Qu.:0.2000
##  Median : 2.800   Median : 6.150   Median :0.4440   Median :0.7000
##  Mean   : 3.262   Mean   : 7.173   Mean   :0.4476   Mean   :0.8562
##  3rd Qu.: 4.500   3rd Qu.: 9.900   3rd Qu.:0.4900   3rd Qu.:1.3000
##  Max.   :10.800   Max.   :24.500   Max.   :1.0000   Max.   :5.1000
##                                    NA's   :4
##       X3PA             X3P.              X2P              X2PA
##  Min.   : 0.000   Min.   :0.0000   Min.   :0.000   Min.   : 0.000
##  1st Qu.: 0.700   1st Qu.:0.2860   1st Qu.:1.000   1st Qu.: 2.100
##  Median : 2.050   Median :0.3400   Median :1.900   Median : 3.800
##  Mean   : 2.421   Mean   :0.3125   Mean   :2.406   Mean   : 4.753
##  3rd Qu.: 3.700   3rd Qu.:0.3750   3rd Qu.:3.300   3rd Qu.: 6.700
##  Max.   :13.200   Max.   :1.0000   Max.   :9.700   Max.   :19.200
##                   NA's   :99
##       X2P.             eFG.               FT              FTA
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000   Min.   : 0.000
##  1st Qu.:0.4540   1st Qu.:0.4730   1st Qu.:0.500   1st Qu.: 0.700
##  Median :0.4980   Median :0.5100   Median :1.000   Median : 1.300
```

```
##  Mean   :0.4967   Mean   :0.5054   Mean   :1.417   Mean   : 1.855
##  3rd Qu.:0.5450   3rd Qu.:0.5490   3rd Qu.:1.900   3rd Qu.: 2.500
##  Max.   :1.0000   Max.   :1.5000   Max.   :9.700   Max.   :11.000
##  NA's   :15       NA's   :4
##       FT.              ORB              DRB              TRB
##  Min.   :0.0000   Min.   :0.0000   Min.   : 0.000   Min.   : 0.00
##  1st Qu.:0.6780   1st Qu.:0.3000   1st Qu.: 1.500   1st Qu.: 1.90
##  Median :0.7650   Median :0.6000   Median : 2.400   Median : 3.10
##  Mean   :0.7408   Mean   :0.8475   Mean   : 2.825   Mean   : 3.67
##  3rd Qu.:0.8270   3rd Qu.:1.1000   3rd Qu.: 3.700   3rd Qu.: 4.70
##  Max.   :1.0000   Max.   :5.4000   Max.   :11.100   Max.   :16.00
##  NA's   :47
##       AST              STL              BLK              TOV
##  Min.   : 0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.: 0.700   1st Qu.:0.3000   1st Qu.:0.1000   1st Qu.:0.600
##  Median : 1.300   Median :0.6000   Median :0.3000   Median :0.900
##  Mean   : 1.935   Mean   :0.6408   Mean   :0.4027   Mean   :1.131
##  3rd Qu.: 2.500   3rd Qu.:0.9000   3rd Qu.:0.5000   3rd Qu.:1.500
##  Max.   :11.200   Max.   :2.4000   Max.   :2.7000   Max.   :5.700
##
##        PF             PTS             Salary          mean_views
##  Min.   :0.000   Min.   : 0.000   Min.   :   56845   Min.   :    1.14
##  1st Qu.:1.200   1st Qu.: 4.300   1st Qu.: 1471382   1st Qu.:  203.57
##  Median :1.700   Median : 7.300   Median : 3384298   Median :  410.92
##  Mean   :1.726   Mean   : 8.794   Mean   : 6790048   Mean   :  988.80
##  3rd Qu.:2.200   3rd Qu.:12.000   3rd Qu.:10432339   3rd Qu.:  930.16
##  Max.   :4.000   Max.   :36.100   Max.   :37457154   Max.   :34147.96
##                                   NA's   :62         NA's   :138
##      Season    Conference    Role          Fvot             FRank
##  2016-17:436   Est :704   Back :629   Min.   :      3   Min.   :  1.00
##  2017-18:498   West:704   Front:779   1st Qu.:   2136   1st Qu.: 30.00
##  2018-19:474                          Median :   6843   Median : 60.00
##                                       Mean   : 117696   Mean   : 61.60
##                                       3rd Qu.:  29252   3rd Qu.: 90.25
##                                       Max.   :4620809   Max.   :145.00
##
##       Pvot             PRank            Mvot              MRank
##  Min.   :  0.000   Min.   : 1.00   Min.   :  0.000   Min.   :1.000
##  1st Qu.:  0.000   1st Qu.:24.00   1st Qu.:  0.000   1st Qu.:6.000
##  Median :  1.000   Median :43.00   Median :  0.000   Median :7.000
##  Mean   :  8.135   Mean   :43.27   Mean   :  2.938   Mean   :7.149
##  3rd Qu.:  4.000   3rd Qu.:60.00   3rd Qu.:  0.000   3rd Qu.:8.000
##  Max.   :269.000   Max.   :88.00   Max.   :100.000   Max.   :9.000
##  NA's   :159       NA's   :159     NA's   :404       NA's   :404
##      Score          Play
##  Min.   :  1.00   No :1335
##  1st Qu.: 45.00   Yes:  73
##  Median : 69.90
##  Mean   : 75.58
##  3rd Qu.:109.20
##  Max.   :166.80
##
```

Between the structure and the summary of the dataset we can see right away there are some columns that will not be useful to us at all. Rk is not useful to us as it represents alphabetical order of the players, we already mentioned Player.x Player_ID, is not useful as it has too many factors, Pos2 has minimal values and 1396 NA's so it also is not useful.

With 45 variables, using our basic knowledge of the NBA game lets discuss those variable that will be useful in our analysis and those we will keep rather than those that we will drop.

Pos1 (position) is worth taking a look at, Age will defnietly be factor as older players that have proven themselves in their prime earn more than younger players. G - Games Played, GS - Games Started and MP - Minutes played is definitely worth exploring as starters and those players that play more minutes earn more. This dataset provides us with extensive statistics, for our purpose we will explore few offensive and defensive statistics (FG, X3P, X2P, FT, TRB, AST, STL, BLK, PTS). We will also explore if Season plays a role as seasons progress salaries rise, and finally if the player played in the all-star game (Play). We of course need to keep the Salary variable we are trying to predict at least for now for the purpose of data exploration, later we will factor salary into factor of 2, above league average or below.

Select the variables we will keep.

```
nba_players <- select(nba_players, Pos1, Age, Tm, G, GS, MP, FG, X3P, X2P, FT, TRB, AST, STL, BLK, PTS,
```

Before exploring the data check for missing values in nba_players, strip if any found.

```
nba_players <- na.omit(nba_players)
```

Review the new structure.

```
str(nba_players)
```

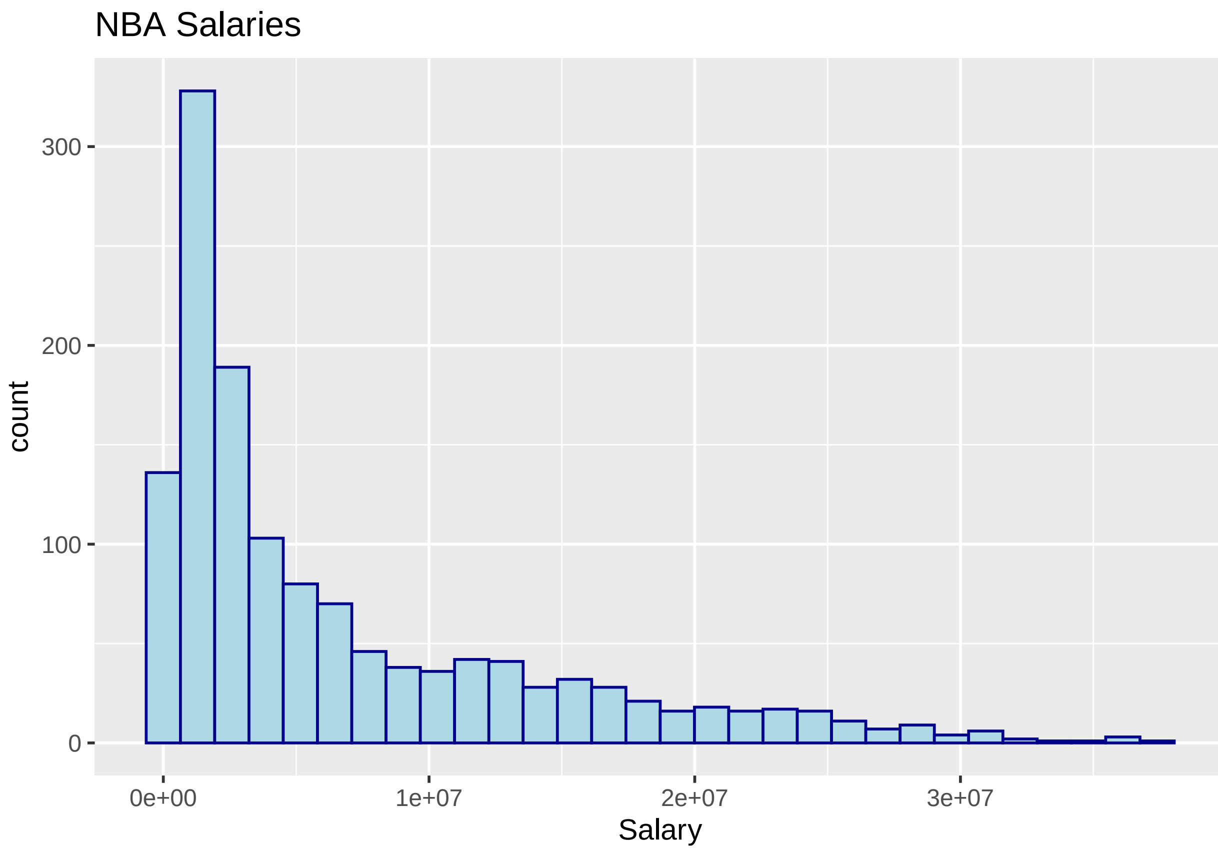
```
## 'data.frame':    1346 obs. of  18 variables:
##  $ Pos1  : Factor w/ 5 levels "C","PF","PG",..: 3 4 2 2 1 1 4 1 5 5 ...
##  $ Age   : int  32 21 25 26 30 32 34 24 25 23 ...
##  $ Tm    : Factor w/ 30 levels "ATL","BOS","BRK",..: 12 22 18 25 2 12 13 24 29 21 ...
##  $ G     : int  65 80 18 61 68 66 30 47 42 68 ...
##  $ GS    : int  0 72 0 25 68 1 0 0 0 6 ...
##  $ MP    : num  13.8 28.7 7.5 29.1 32.3 14.1 10.3 15.1 15.5 15.5 ...
##  $ FG    : num  1.9 4.9 1.3 3 5.6 3.6 1 2.9 2.4 2 ...
##  $ X3P   : num  0.7 1 0.2 1.1 1.3 0 0.5 0 0.6 1.4 ...
##  $ X2P   : num  1.1 4 1.1 1.9 4.3 3.6 0.5 2.9 1.8 0.6 ...
##  $ FT    : num  0.5 2 0.8 1.6 1.6 1 0.4 1.5 1.4 0.6 ...
##  $ TRB   : num  1.1 5.1 1.8 7.4 6.8 4.2 0.8 6.2 2.9 1.3 ...
##  $ AST   : num  1.9 1.9 0.4 1.6 5 0.9 0.4 0.5 0.7 0.6 ...
##  $ STL   : num  0.4 0.8 0.4 1 0.8 0.3 0.1 0.6 0.4 0.5 ...
##  $ BLK   : num  0.1 0.5 0.4 0.7 1.3 0.2 0 0.7 0.1 0.1 ...
##  $ PTS   : num  5 12.7 3.5 8.7 14 8.1 2.9 7.4 6.7 6 ...
##  $ Salary: int  2700000 4351320 2022240 7680965 26540100 10230179 1315448 874636 9904495 5994764 ...
##  $ Season: Factor w/ 3 levels "2016-17","2017-18",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Play  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int  1 55 56 57 78 95 161 163 164 165 ...
##   ..- attr(*, "names")= chr  "1" "55" "56" "57" ...
```

4.2 Data Visualization

4.2.1 NBA Salaries

Let's start by visualizing players' salaries.

```r
# Plot Players Salary
ggplot(nba_players,aes(Salary)) +
  geom_histogram(color='darkblue',fill='lightblue') +
  ggtitle('NBA Salaries')
```

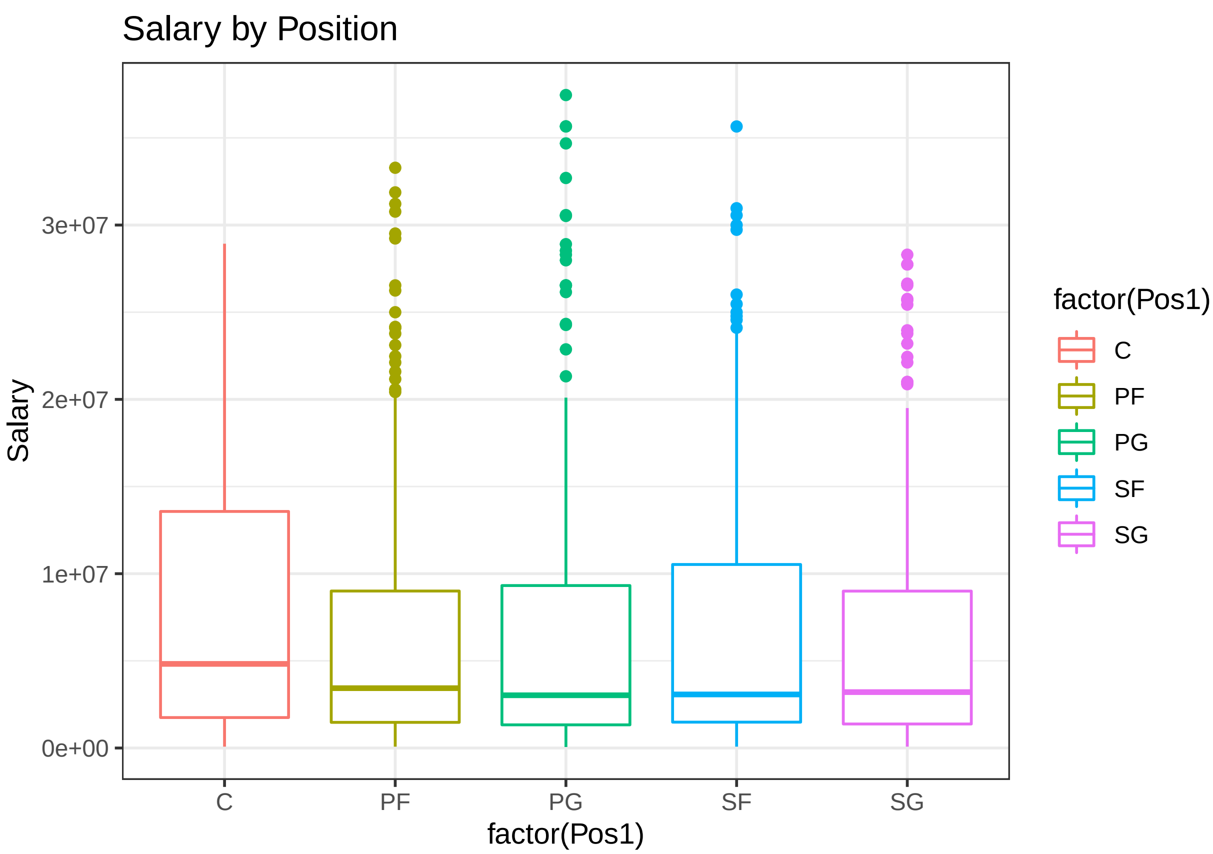


We see that majority of the players make under 5 Million dollars, in fact we know from our earlier summary that median salary is $3,384,298 well below average salary of $6,790,048 with max salary $37,457,154. What this tells us is that there are number of players that make a lot more money than majority of the players causing the league mean Salary value to be around double of the median value.

Let's a explore some predictors that might good indicators of Salary and decide what we want to keep. Let's start with Position.

4.2.2 Position (Pos1)

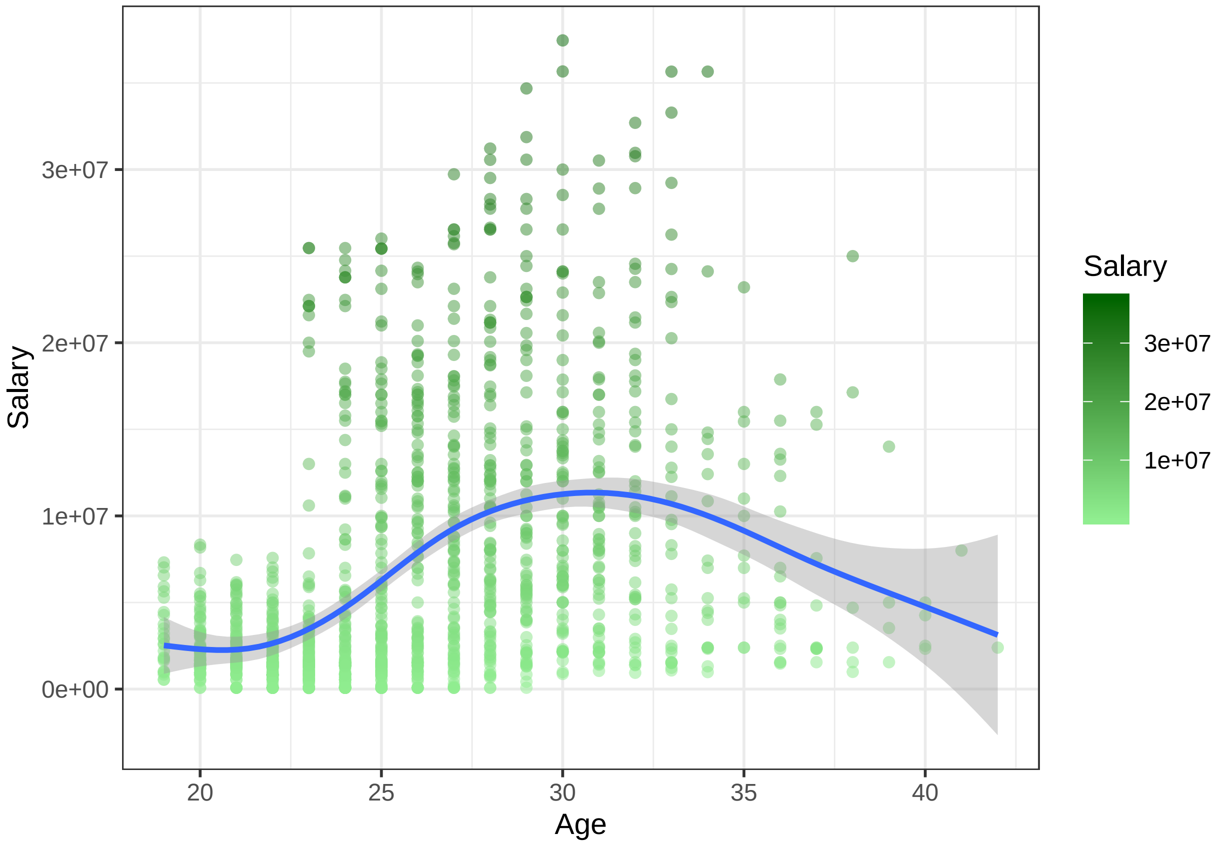
```
# Plot Salary by Position
ggplot(nba_players,aes(factor(Pos1),Salary)) +
  geom_boxplot(aes(color=factor(Pos1))) +
  theme_bw() +
  ggtitle('Salary by Position')
```



From the above boxplot we see that Center (C) position has the highest average of salaries but does not have any outliers, Other position averages are fairly close, but what is important to notice that Point Guard (PG) has the most and highest paid outliers in salary, this could skew average salary. We should keep this indicator.

4.2.3 Age

```
#Plot Salary by Age
ggplot(nba_players,aes(Age,Salary)) +
  geom_point(aes(color=Salary),alpha=0.5)  +
  scale_color_continuous(low='lightgreen',high='darkgreen') +
  geom_smooth() + theme_bw()
```

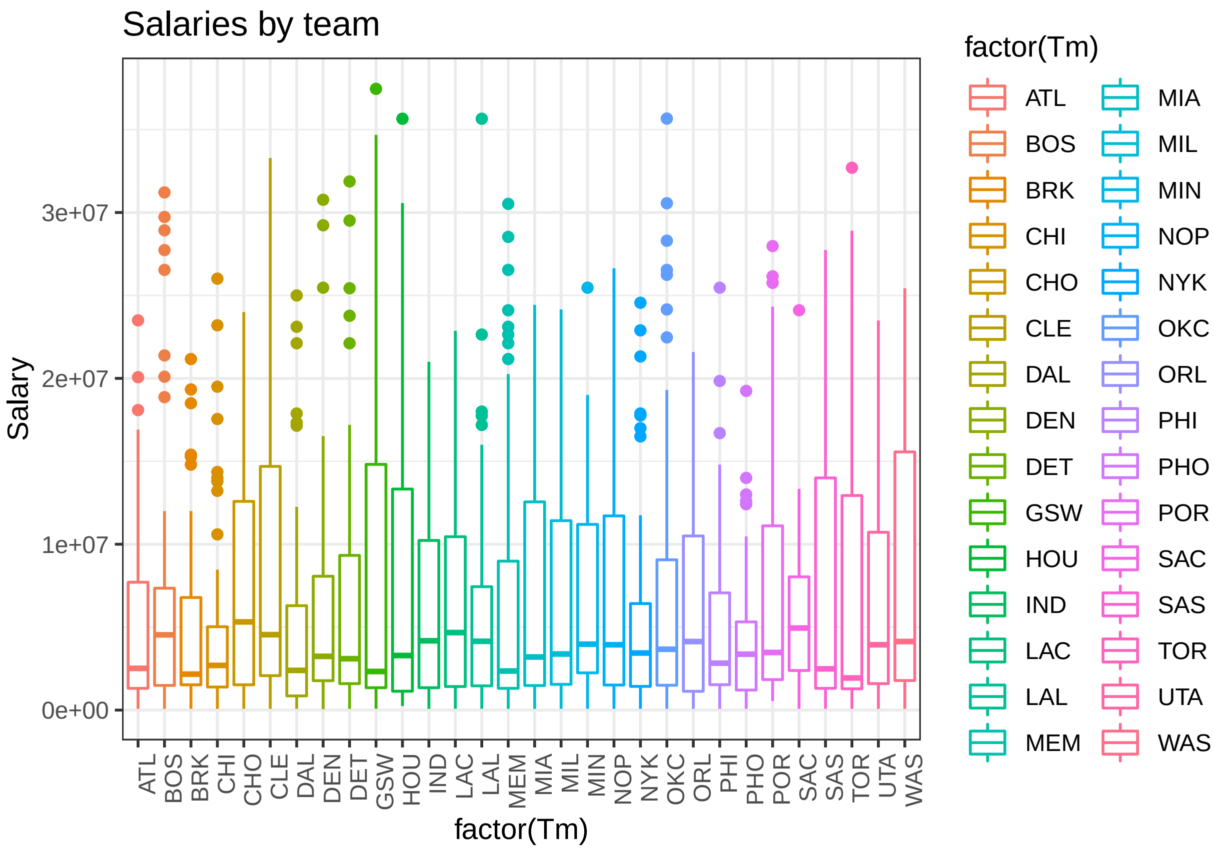


From this plot we see that on average players earn most at their peak between ages of 28 to 34, then start trending back down the older they get. This is a very good indicator of Salary to keep.

4.2.4 Team

Does it matter what team does the player play on?

```
#Plot Salary by team
ggplot(nba_players,aes(factor(Tm),Salary)) +
  geom_boxplot(aes(color=factor(Tm))) +theme_bw() +
  ggtitle('Salaries by team') +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

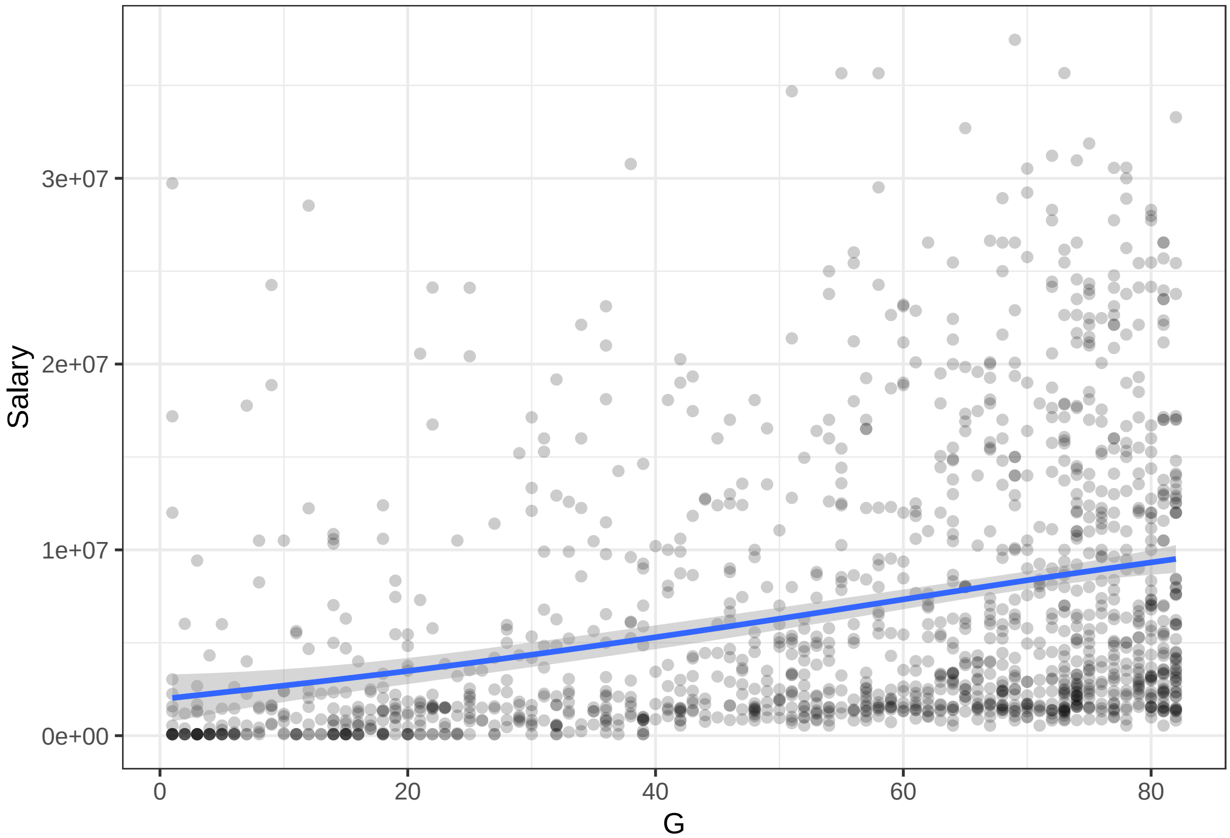


It does seem that some teams have bigger budgets than others, so it is a factor we should keep.

4.2.5 Game Playes stats - G, GS, Min

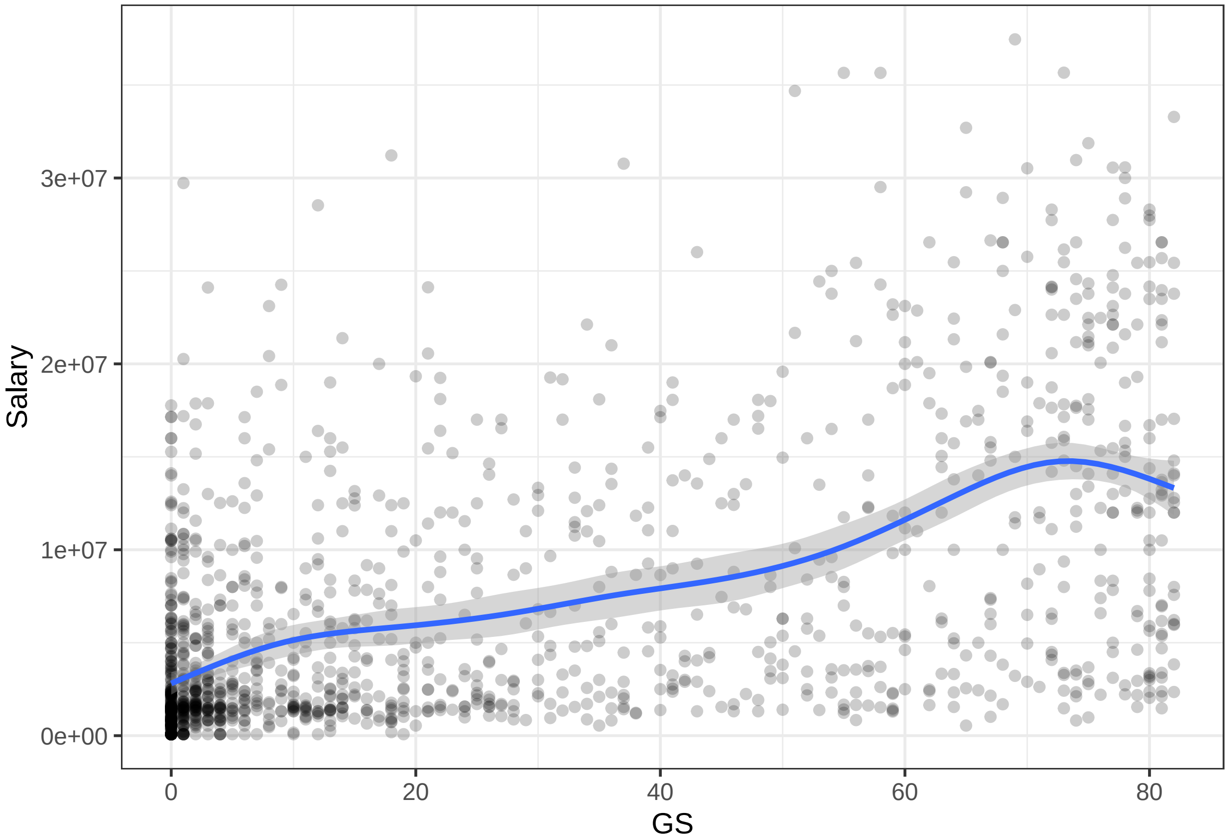
Plot Games Played, Games Started and Minutes played

```
# Plot G-Games Played
ggplot(nba_players,aes(G,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



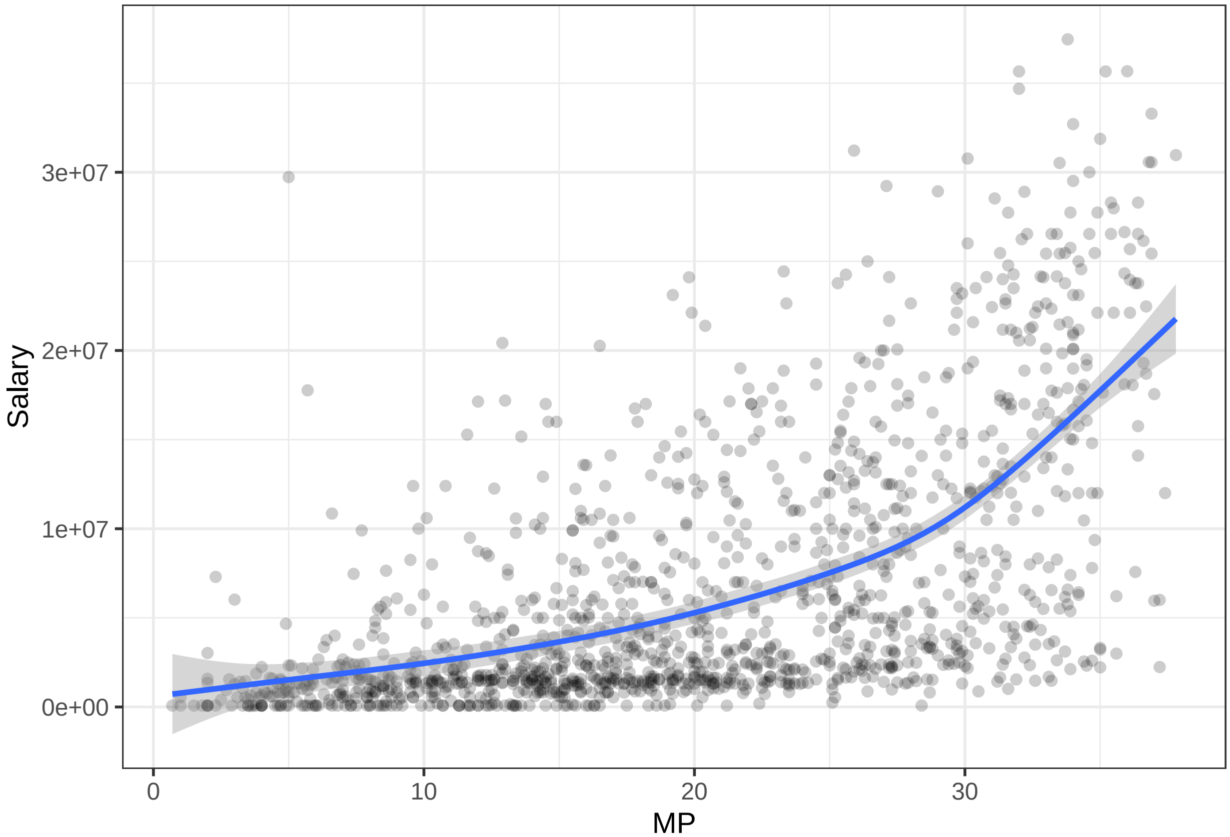
Games played seems like a good steady indicator.

```r
# Plot Games Started
ggplot(nba_players,aes(GS,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



We see that Games started is a good indicator, although it peaks at around 73 Games, this probably to account for injuries and maintenance days for the stars which would throw off our prediction, we might want to get rid of this indicator.

```r
# Plot Minutes played
ggplot(nba_players,aes(MP,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```

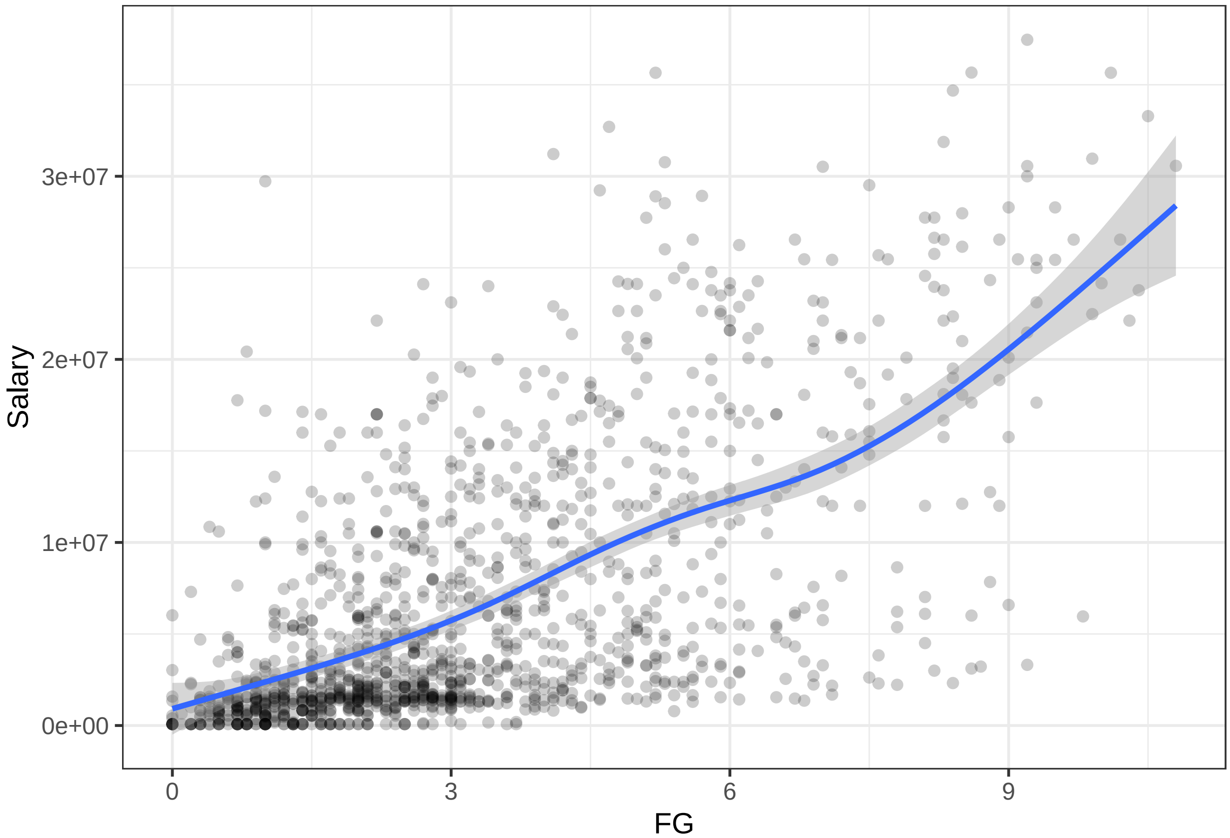


We see that players that earn more definitely play more minutes, players earning over $10 Million play over 30 minutes. MP is probably the most important out of the three variables.

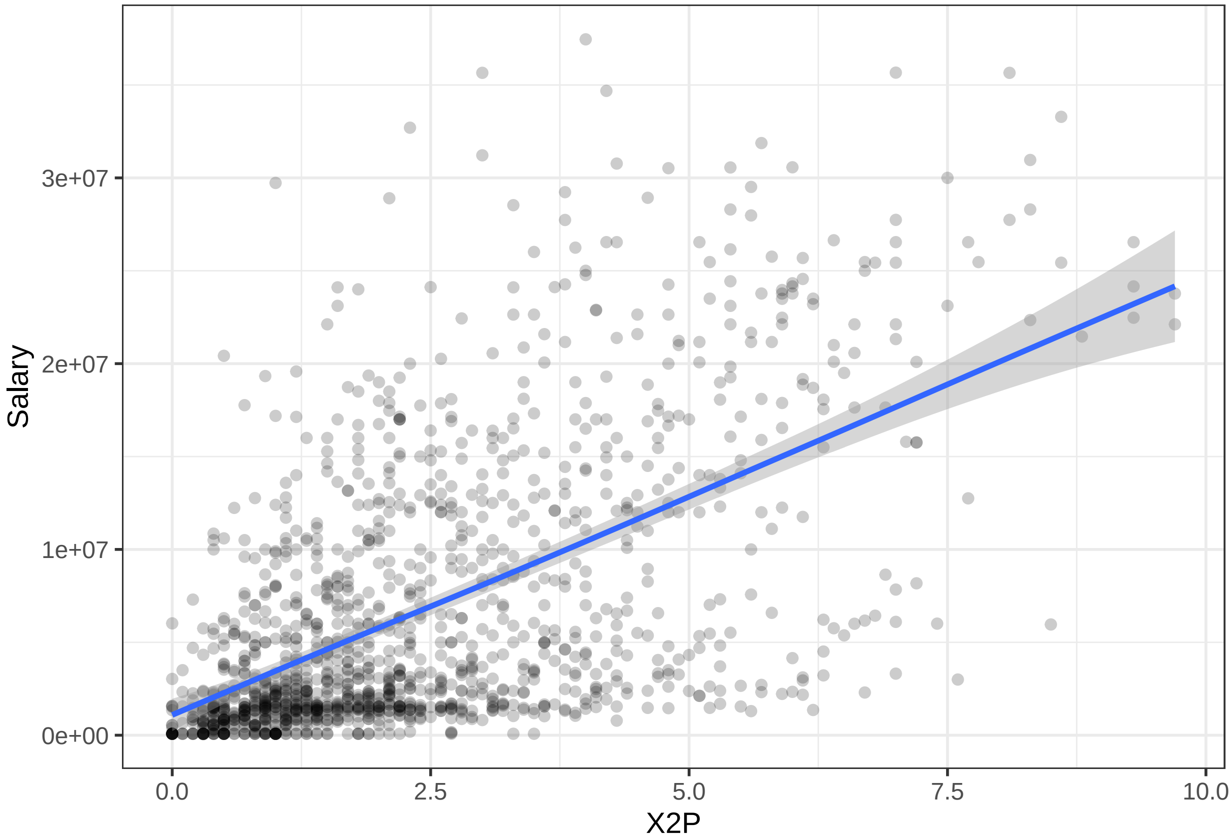
4.2.6 Offensive Stats

We will look at offensive stats next.

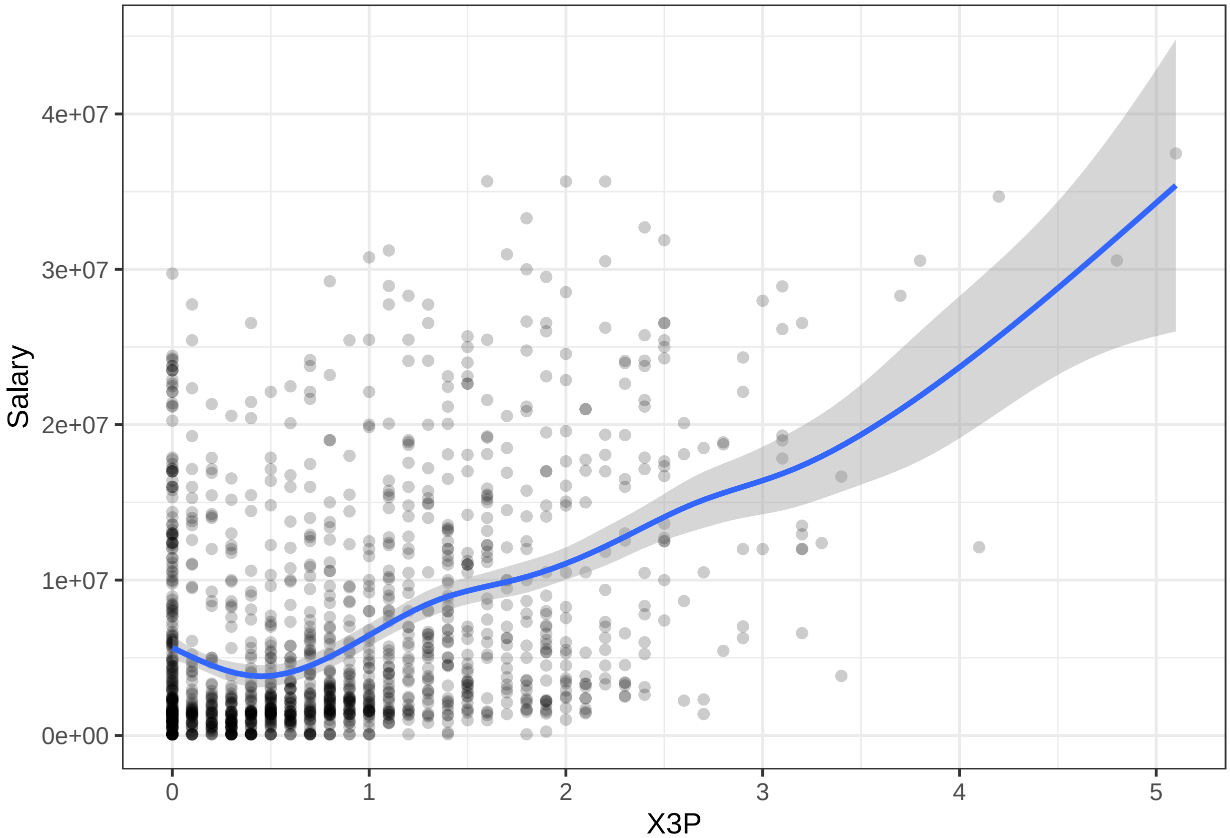
```
# Plot FG = Field Goals Per Game
ggplot(nba_players,aes(FG,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



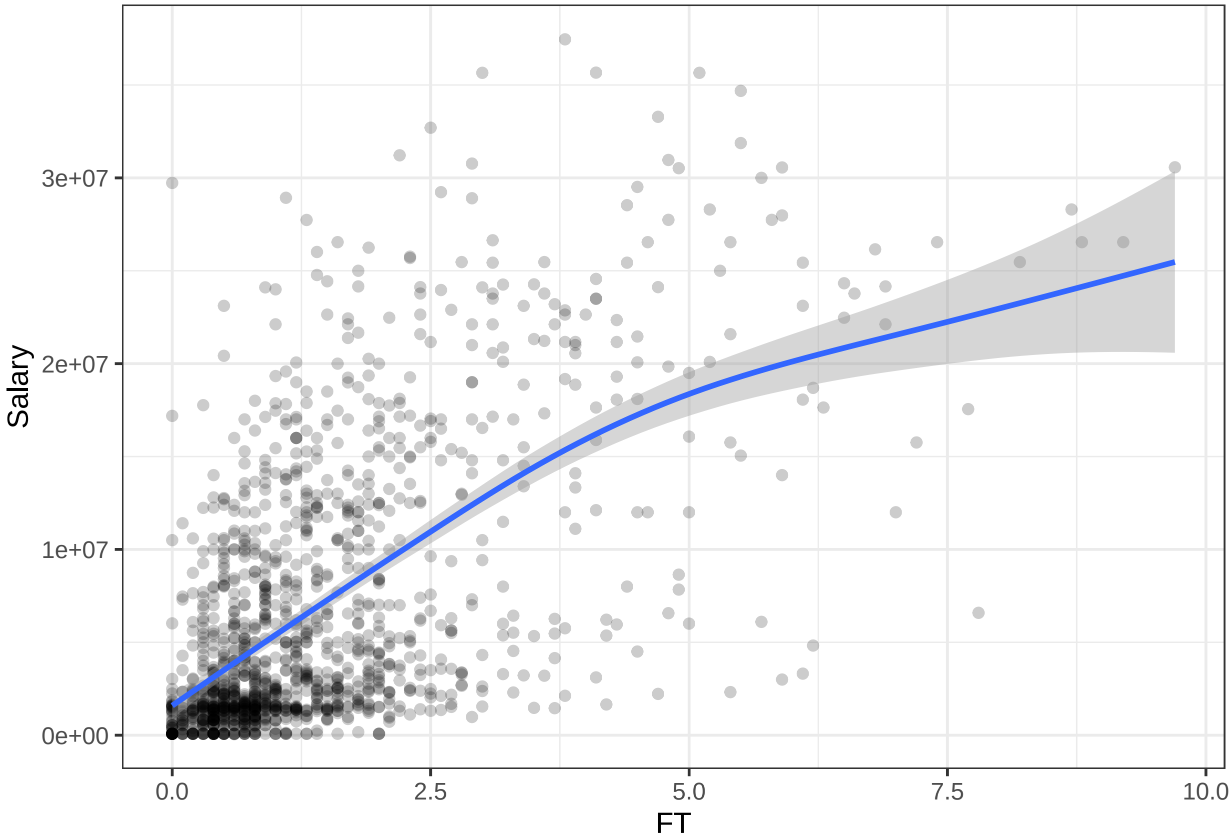
```r
# Plot X2P = 2-Point Field Goals Per Game
ggplot(nba_players,aes(X2P,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



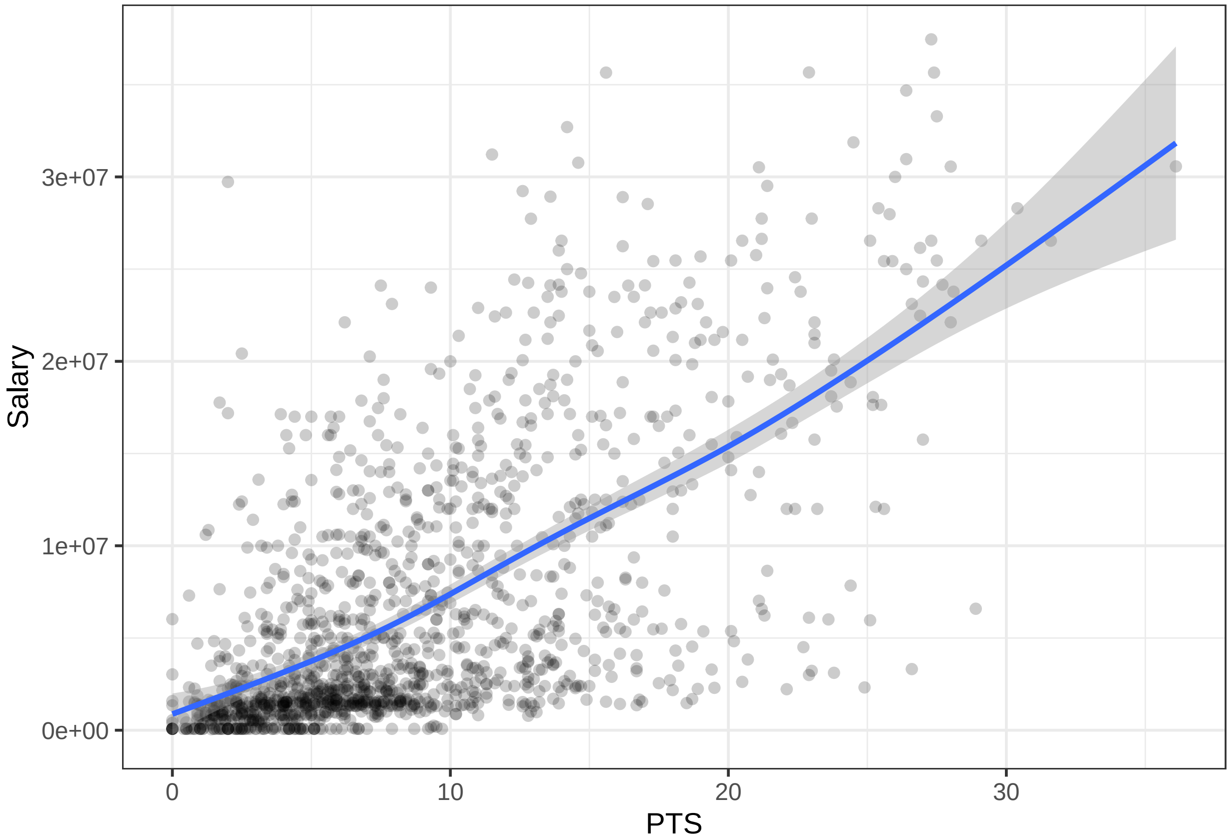
```r
# Plot X3P = 3-Point Field Goals Per Game
ggplot(nba_players,aes(X3P,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



```r
# Plot FT - Free throws
ggplot(nba_players,aes(FT,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



```r
# Plot PTS - Points Per Game
ggplot(nba_players,aes(PTS,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



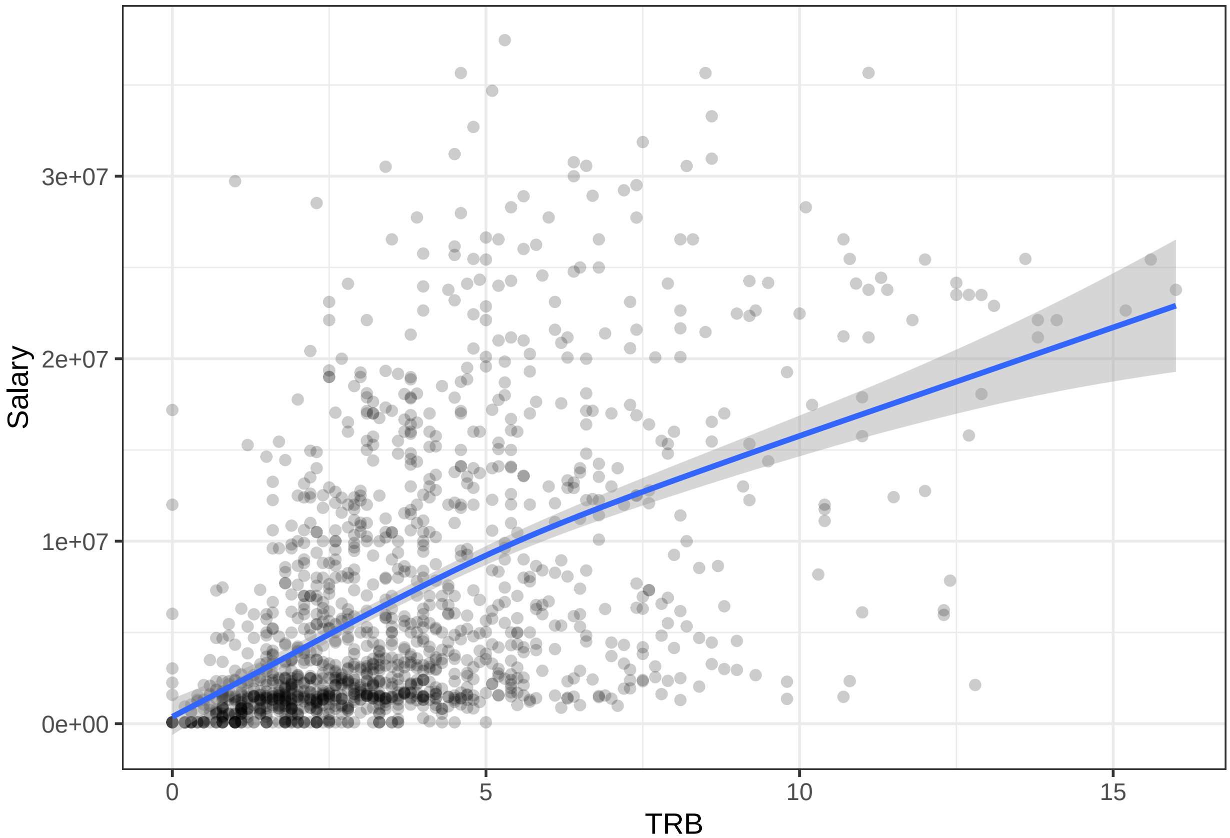
Field Goals (FG) is a combination of X2P and X3P, we want to separate the field goals as X3P is closer related to position of Point Guard. To avoid over-fitting we will get rid of FG.

All other offensive stats variables look like good indicators in the above plots, we should keep them all.

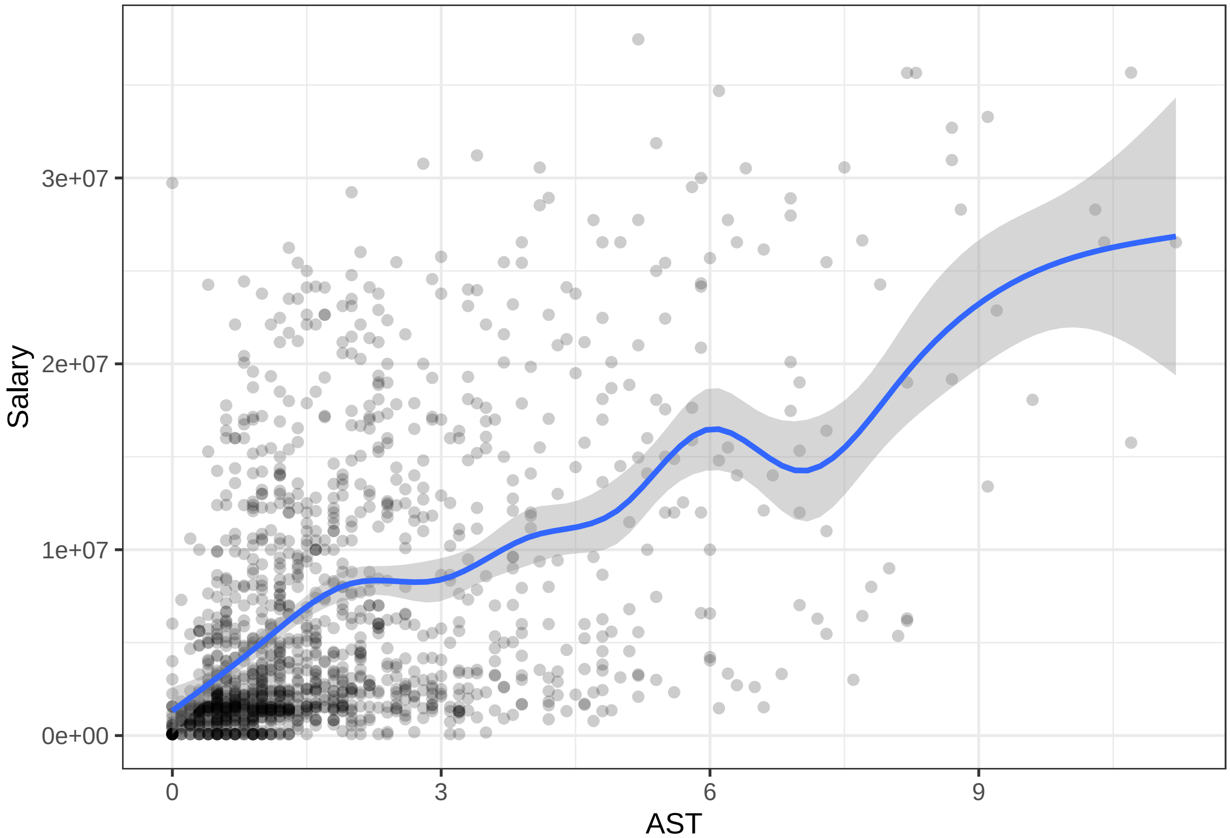
4.2.7 Defensive Stats

Some players are better defensive players than offensive and get payed well for being defensive players, we will next look at some of these stats - TRB, AST, STL, BLK

```
# Plot TRB = Total Rebounds Per Game
ggplot(nba_players,aes(TRB,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```

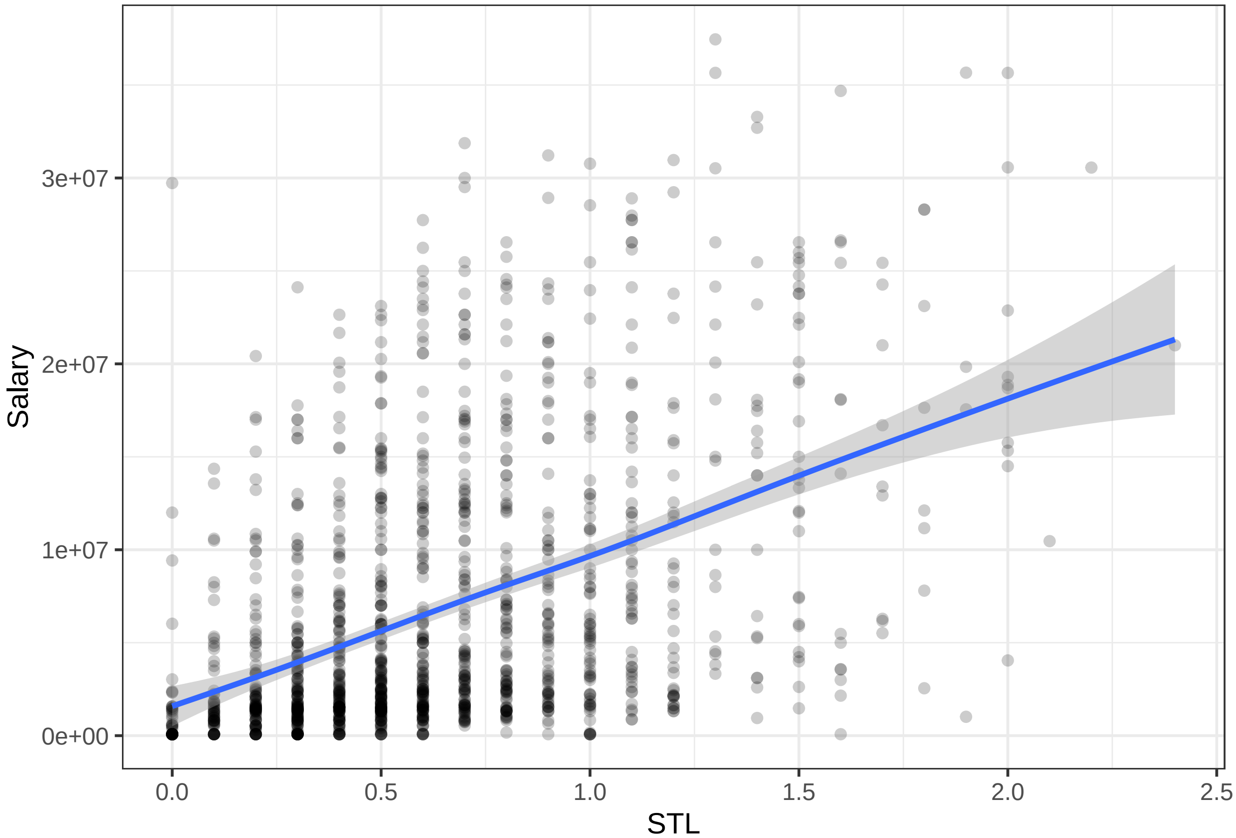


```r
# AST = Assits Per Game
ggplot(nba_players,aes(AST,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```



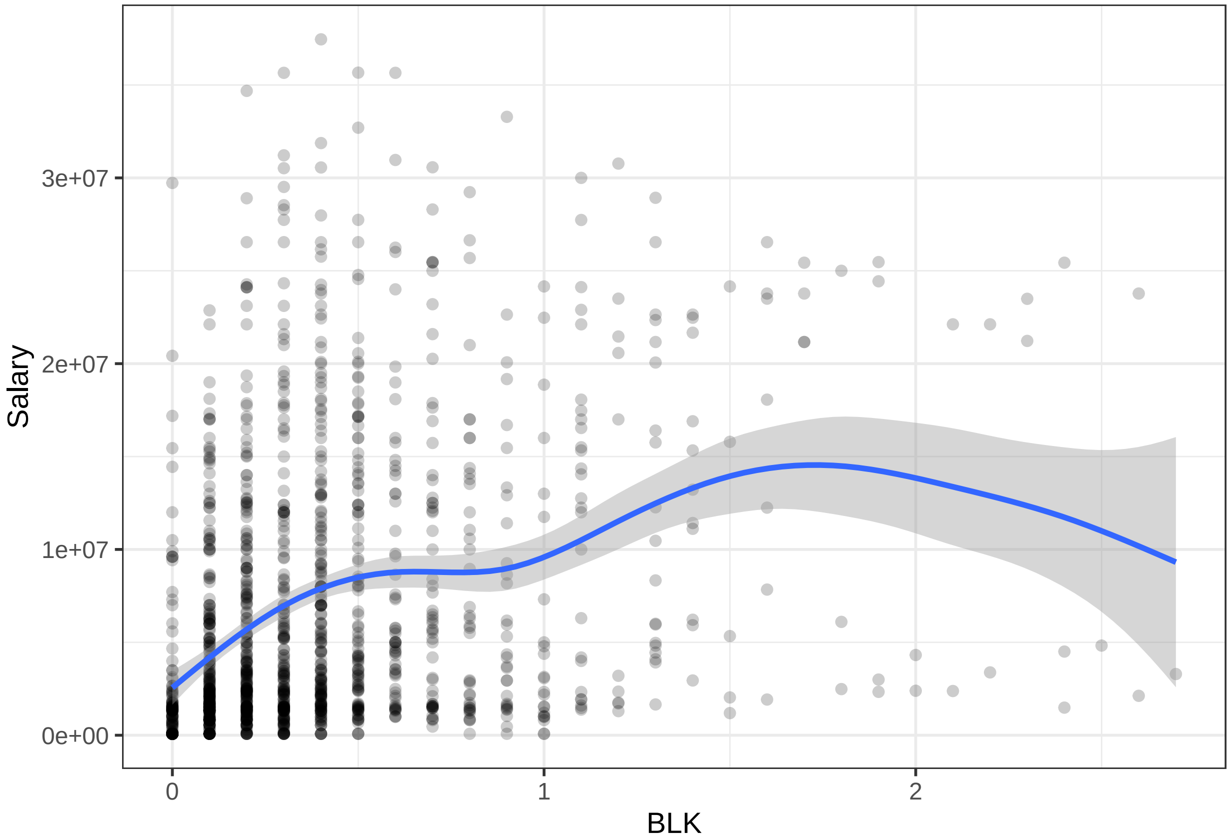

```r
# Assists also look like a good indicator.
```

```r
# STL= Steals Per Game
ggplot(nba_players,aes(STL,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```




```r
# Steals also look like a good indicator.
```

```r
# BLK = Blocks Per Game
ggplot(nba_players,aes(BLK,Salary)) +
  geom_point(alpha=0.2) +
  geom_smooth() +
  theme_bw()
```

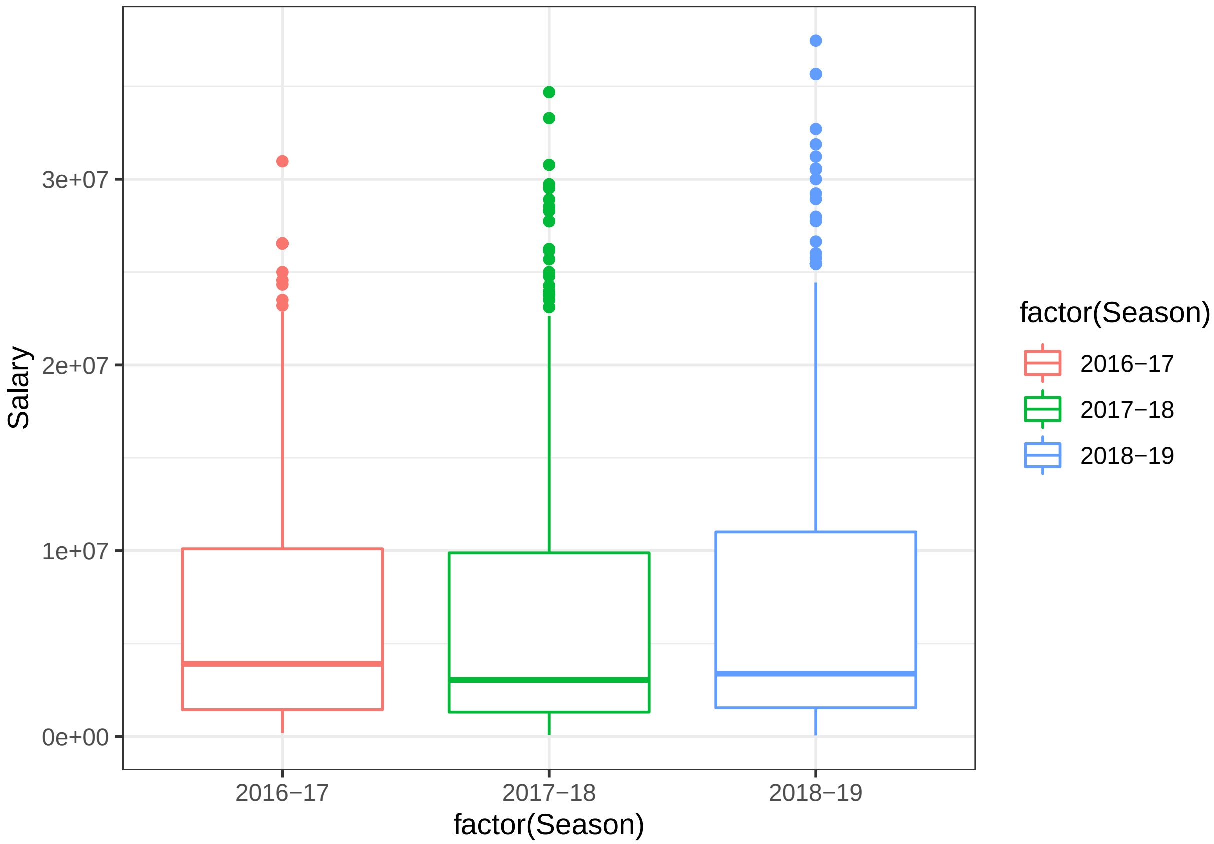


All the defensive stats seem relevant to us from looking at the plots.

4.2.8 Season

Lets look if the seasons plays a role.

```
# plot 3 Seasons
ggplot(nba_players,aes(factor(Season),Salary)) +
  geom_boxplot(aes(color=factor(Season))) +
  theme_bw()
```

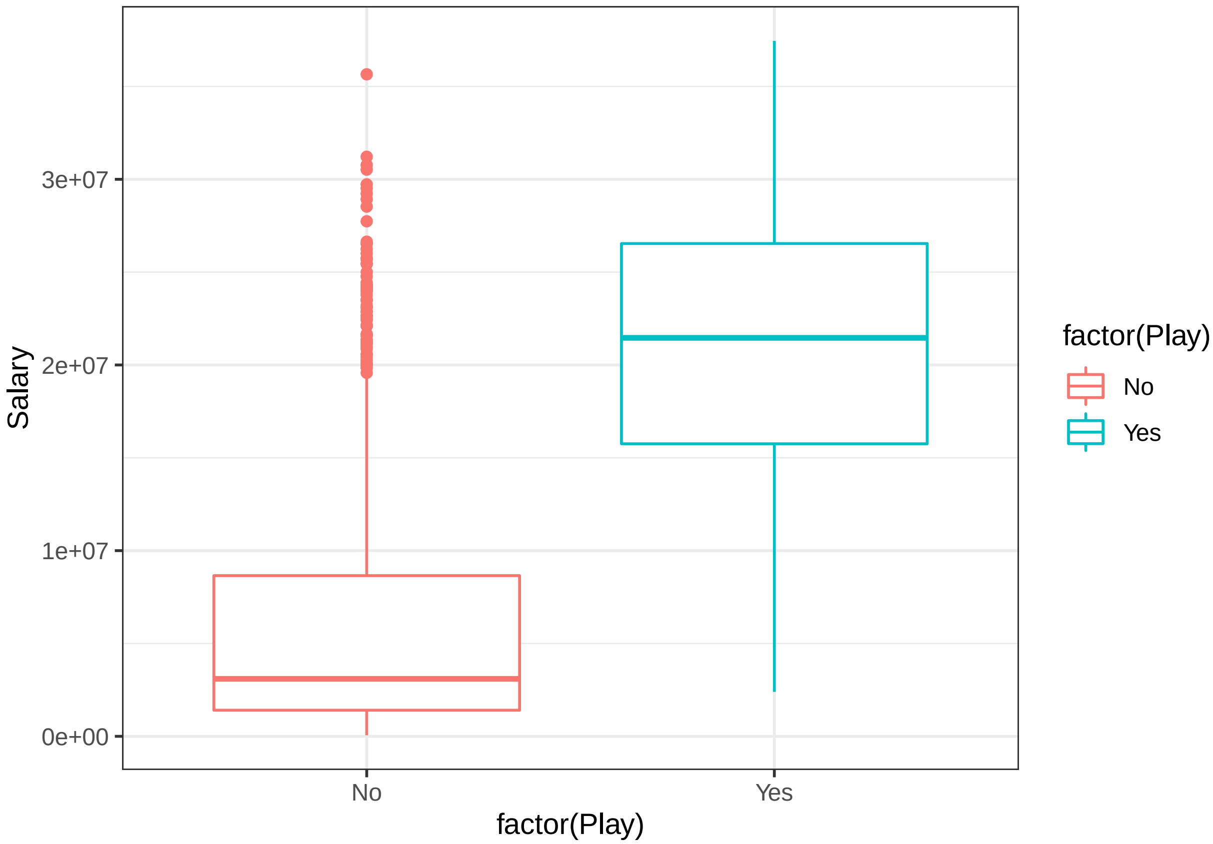


As seasons progress the salaries are going up slightly especially for the outlier players making more money, we that 2018-2019 has the highest salaries.

4.2.9 All Star

Last factor to look at is if the player played in the All-Star Game.

```
#Played in allstar game
ggplot(nba_players,aes(factor(Play),Salary)) +
  geom_boxplot(aes(color=factor(Play))) +
  theme_bw()
```



Although there are few outliers for those that did not play in all-star game, we can see that on average those that played in the all-star game earn more than double of those that did not.

4.3 Data Manipulaton

We will first get rid of the indicators we identified in the previous section: GS and FG.

```
nba_players <- select(nba_players, -GS, -FG)
```

Next we will calculate the league average salary and create a new column AboveAvg indicating if the player is above or below/equal league average salary. To do that will use a function aboveAvg.

```
# What is the Average salary in the NBA
mu_salary <- mean(nba_players$Salary)
mu_salary
```

```
## [1] 6790048
```

```
# Function that determines if the player is above average salary
aboveAvg <- function(x){
  if (x>mu_salary){
    return("Yes")
  }else{
    return("No")
  }
}

# Add a new column AboveAvg to the dataset
nba_players$AboveAvg <-sapply(nba_players$Salary,aboveAvg)
```

Next we will need to re-factor the new column.

```
# Refactor the new column
nba_players$AboveAvg <- factor(nba_players$AboveAvg)
```

Now we can drop our Salary column and check out the new stucture of our dataset.

```
# Drop the Salary column
nba_players <- select(nba_players, -Salary)

# Check the structure of the Dataset before proceeding with the model
str(nba_players)
```

```
## 'data.frame':    1346 obs. of  16 variables:
##  $ Pos1    : Factor w/ 5 levels "C","PF","PG",..: 3 4 2 2 1 1 4 1 5 5 ...
##  $ Age     : int  32 21 25 26 30 32 34 24 25 23 ...
##  $ Tm      : Factor w/ 30 levels "ATL","BOS","BRK",..: 12 22 18 25 2 12 13 24 29 21 ...
##  $ G       : int  65 80 18 61 68 66 30 47 42 68 ...
##  $ MP      : num  13.8 28.7 7.5 29.1 32.3 14.1 10.3 15.1 15.5 15.5 ...
##  $ X3P     : num  0.7 1 0.2 1.1 1.3 0 0.5 0 0.6 1.4 ...
##  $ X2P     : num  1.1 4 1.1 1.9 4.3 3.6 0.5 2.9 1.8 0.6 ...
##  $ FT      : num  0.5 2 0.8 1.6 1.6 1 0.4 1.5 1.4 0.6 ...
##  $ TRB     : num  1.1 5.1 1.8 7.4 6.8 4.2 0.8 6.2 2.9 1.3 ...
##  $ AST     : num  1.9 1.9 0.4 1.6 5 0.9 0.4 0.5 0.7 0.6 ...
##  $ STL     : num  0.4 0.8 0.4 1 0.8 0.3 0.1 0.6 0.4 0.5 ...
##  $ BLK     : num  0.1 0.5 0.4 0.7 1.3 0.2 0 0.7 0.1 0.1 ...
##  $ PTS     : num  5 12.7 3.5 8.7 14 8.1 2.9 7.4 6.7 6 ...
##  $ Season  : Factor w/ 3 levels "2016-17","2017-18",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Play    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ AboveAvg: Factor w/ 2 levels "No","Yes": 1 1 1 2 2 2 1 1 2 1 ...
##  - attr(*, "na.action")= 'omit' Named int  1 55 56 57 78 95 161 163 164 165 ...
```

```
##   ..- attr(*, "names")= chr  "1" "55" "56" "57" ...
```

5 Modeling/Results

5.1 Create Train and test sets

Before we start modeling our data we will need to split of dataset into train and test split. We will use 70% split for train set and 30% for test set.

```
# Set a random seed
set.seed(101)

# Split data nba_players assigning split ratio to TRUE
sample_nba <- sample.split(nba_players$AboveAvg, SplitRatio = 0.70)

# Training Data
train = subset(nba_players, sample_nba == TRUE)

# Testing Data
test = subset(nba_players, sample_nba == FALSE)
```

5.2 Model 1: Logistic Regression Model

We first run logistic regression model on our train data.

```
# Run Logistic Regression model
model_lgr = glm(AboveAvg ~ ., family = binomial(logit), data = train)
```

Lets see the results of the model.

```
# Print Summary of the model
summary(model_lgr)
```

```
##
## Call:
## glm(formula = AboveAvg ~ ., family = binomial(logit), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5437  -0.5487  -0.2118   0.4915   2.4620
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -11.632845   1.130137 -10.293  < 2e-16 ***
## Pos1PF        -0.794464   0.335214  -2.370 0.017787 *
## Pos1PG        -2.132942   0.507583  -4.202 2.64e-05 ***
## Pos1SF        -1.627411   0.440867  -3.691 0.000223 ***
## Pos1SG        -1.364850   0.446476  -3.057 0.002236 **
## Age            0.282156   0.026982  10.457  < 2e-16 ***
## TmBOS         -0.213782   0.813532  -0.263 0.792719
## TmBRK          0.312070   0.751962   0.415 0.678137
## TmCHI          0.580933   0.820240   0.708 0.478792
## TmCHO          1.106630   0.818238   1.352 0.176230
## TmCLE          0.546269   0.764830   0.714 0.475081
## TmDAL         -0.544331   0.894229  -0.609 0.542713
## TmDEN          0.078666   0.764115   0.103 0.918003
```

```
## TmDET          0.522172   0.760470   0.687 0.492307
## TmGSW          0.448524   0.797718   0.562 0.573940
## TmHOU         -0.096934   0.848110  -0.114 0.909004
## TmIND          1.403669   0.780516   1.798 0.072116 .
## TmLAC         -0.288811   0.785785  -0.368 0.713213
## TmLAL         -0.118774   0.819970  -0.145 0.884828
## TmMEM          0.047100   0.817259   0.058 0.954042
## TmMIA         -0.177703   0.747958  -0.238 0.812203
## TmMIL          0.692209   0.782104   0.885 0.376125
## TmMIN          0.444812   0.798504   0.557 0.577489
## TmNOP          0.849496   0.760983   1.116 0.264288
## TmNYK         -0.516915   0.892120  -0.579 0.562304
## TmOKC          0.295604   0.797998   0.370 0.711061
## TmORL          1.120790   0.756155   1.482 0.138281
## TmPHI          0.748447   0.752537   0.995 0.319948
## TmPHO         -0.269163   0.866195  -0.311 0.755997
## TmPOR          1.451197   0.828223   1.752 0.079743 .
## TmSAC          0.782270   0.780403   1.002 0.316154
## TmSAS          0.588554   0.819050   0.719 0.472399
## TmTOR          1.504848   0.795958   1.891 0.058676 .
## TmUTA          0.942125   0.735637   1.281 0.200301
## TmWAS          0.263476   0.802790   0.328 0.742760
## G             -0.016534   0.005929  -2.789 0.005291 **
## MP             0.221353   0.038036   5.820 5.90e-09 ***
## X3P           -1.001997   2.722387  -0.368 0.712830
## X2P           -0.440064   1.814704  -0.242 0.808394
## FT            -0.240399   0.928469  -0.259 0.795697
## TRB            0.009569   0.084568   0.113 0.909912
## AST            0.036219   0.109228   0.332 0.740200
## STL            0.233338   0.383951   0.608 0.543367
## BLK           -1.124671   0.346308  -3.248 0.001164 **
## PTS            0.268985   0.909642   0.296 0.767456
## Season2017-18  0.216643   0.240009   0.903 0.366714
## Season2018-19  0.500193   0.244768   2.044 0.040999 *
## PlayYes        0.588546   0.558031   1.055 0.291571
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1207.34  on 941  degrees of freedom
## Residual deviance:  698.22  on 894  degrees of freedom
## AIC: 794.22
##
## Number of Fisher Scoring iterations: 6
```

From the summary we see that the most important factor is Age with the lowest Pr value, followed by Minutes Played (MP) and then position of Point Guard (Pos1PG). This makes sense as we recall from our plots, players that played over 30 minutes had significantly higher salaries and those of Pos1 of PG had also the highest salaries.

Let's now see how well our Model performs predicting if salary is above or below/equal to league average.

```
# Predict the values using the model and the test data
test$predicted.AboveAvg = predict(model_lgr, newdata=test, type="response")
```

```r
#Lets see how well we predicted the results with confusion matrix
table(test$AboveAvg, test$predicted.AboveAvg > 0.5)
```

```
##
##       FALSE TRUE
##   No    233   34
##   Yes    37  100
```

Our model predicted correctly that 100 players out 137 players have above league average salary, getting 37 wrong. It also predicted 233 out of 267 players have below or equal salary to that of league average getting 34 players wrong.

Let's Calculate the accuracy of our model with confusion matrix.

```r
(233+100)/(233+100+34+37)
```

```
## [1] 0.8242574
```

We were able to achieve the accuracy of 82.43%.

Let's look at the stats with confusion matrix function.

To use the confusion matrix function we will first have to refactor our test$predicted.AboveAvg result so it's the same factor as test.AboveAvg. We will use the following code.

```r
# Refactor function
aboveAvgTest <- function(x){
  if (x > 0.5){
    return("Yes")
  }else{
    return("No")
  }
}

test$predicted.AboveAvg <-sapply(test$predicted.AboveAvg,aboveAvgTest)
# Factor test$predicted.AboveAvg now with 2 values.
test$predicted.AboveAvg <- factor(test$predicted.AboveAvg)
str(test$predicted.AboveAvg)
```

```
##  Factor w/ 2 levels "No","Yes": 1 2 2 1 2 1 1 2 1 2 ...
```

Now that test$predicted.AboveAvg and test.AboveAvg are of the same factor call confusion Matrix.

```r
# Print Confusion Matrix
confusionMatrix(data = test$predicted.AboveAvg, reference = test$AboveAvg)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  233  37
##        Yes  34 100
##
##                Accuracy : 0.8243
##                  95% CI : (0.7836, 0.8601)
##     No Information Rate : 0.6609
##     P-Value [Acc > NIR] : 1.852e-13
##
##                   Kappa : 0.6058
```

```
##
##  Mcnemar's Test P-Value : 0.8124
##
##             Sensitivity : 0.8727
##             Specificity : 0.7299
##          Pos Pred Value : 0.8630
##          Neg Pred Value : 0.7463
##              Prevalence : 0.6609
##          Detection Rate : 0.5767
##    Detection Prevalence : 0.6683
##       Balanced Accuracy : 0.8013
##
##        'Positive' Class : No
##
```

We see that the function gave us the same accuracy and some additonal stats like Sensitivity of 0.8727 and Specificity of 0.7299, along with p-value of 1.852e-13.

5.3 Model 2: Logistic Regression Model with Step function

Let's see if we can improve this Model with build in step function that will help us get rid of factors that are not as important. We start by calling step function on our model_lgr.

```r
model_lgr_step <- step(model_lgr)
```

Let's look at the summary of our new model.

```r
# Print Summary
summary(model_lgr_step)
```

```
##
## Call:
## glm(formula = AboveAvg ~ Pos1 + Age + G + MP + X2P + BLK, family = binomial(logit),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7668  -0.6152  -0.2342   0.5922   2.5894
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.72589    0.83997 -12.769  < 2e-16 ***
## Pos1PF       -0.78158    0.30624  -2.552 0.010704 *
## Pos1PG       -1.74294    0.37053  -4.704 2.55e-06 ***
## Pos1SF       -1.41392    0.36906  -3.831 0.000128 ***
## Pos1SG       -1.21666    0.36241  -3.357 0.000788 ***
## Age           0.27233    0.02450  11.117  < 2e-16 ***
## G            -0.01277    0.00544  -2.348 0.018900 *
## MP            0.19879    0.02364   8.407  < 2e-16 ***
## X2P           0.18485    0.08743   2.114 0.034488 *
## BLK          -0.85849    0.28581  -3.004 0.002667 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 1207.34  on 941  degrees of freedom
## Residual deviance:  737.64  on 932  degrees of freedom
## AIC: 757.64
##
## Number of Fisher Scoring iterations: 5
```

We see that our new step model only kept 9 of the most important factors, those with at least one star. Let's see how this new model preforms in making the predictions.

First we need to drop test$predicted.AboveAvg column from our previous iteration before running new prediction, then run the predict function.

```r
test <- select(test, -predicted.AboveAvg)

# Get new predicted AboveAvg values
test$predicted.AboveAvg = predict(model_lgr_step, newdata=test, type="response")

table(test$AboveAvg, test$predicted.AboveAvg > 0.5)
```

```
##
##       FALSE TRUE
##   No    224   43
##   Yes    41   96
```

It does not look like the new step model preformed as well as our original model. Let's calculate our new accuracy.

```r
(224+96)/(224+96+43+41)
```

```
## [1] 0.7920792
```

Here we see that our accuracy actually got worst with the new step model, it when down from 82.43% to 79.21% , a loss or 3.22%.

5.4 Model 3: SVM Model

Let's see if we can improve our prediction with SVM model.

```r
# call the svm model
model_svm <- svm(AboveAvg ~ ., data = train)

# print summary of the model
summary(model_svm)
```

```
##
## Call:
## svm(formula = AboveAvg ~ ., data = train)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  460
##
##  ( 232 228 )
##
```

```
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

Again, we need to drop test$predicted.AboveAvg column from our previous iteration before running new prediction, then run the predict function.

```
test <- select(test, -predicted.AboveAvg)
# Call predict
test$predicted.AboveAvg <- predict(model_svm,test[1:15])

table(test$predicted.AboveAvg,test$AboveAvg)
```

```
##
##        No Yes
##   No  239  47
##   Yes  28  90
```

We see that SVM model was a little better than our original logistic regression model at predicting those players with below/equal average salary, getting 239 out of 267 players right, but it did worst for those players above league average, getting only 90 out 137 right. Let's Calculate the accuracy of our model to see how close they are in accuracy.

```
(239+90)/(239+90+47+28)
```

```
## [1] 0.8143564
```

We see that the accuracy of our SVM model is actually around 1% lower at 81.44% than our Logistic model which had accuracy of 82.43%.

6 Conclusion

Our first and best model, Logistic Regression model predicted correctly that 100 players out 137 players have above league average salary, getting 37 wrong. It also predicted 233 out of 267 players have below or equal salary to that of league average getting 34 players wrong, giving us overall accuracy of 82.43%. With all the outside factors like missing games to injury, having better agents negotiating contracts, playoff success not accounted in the dataset and other such factors, we think 82.43% accuracy is a very good in those circumstances, so our model is a good model.

We tried to improve our Logistic Regression model by using the step function, however that actually resulted in loss of accuracy from 82.43% to 79.21%, a loss or 3.22%. SVM model faired a little better at 81.44% but at end our first model, Logistic Regression model ended up being the best model at accuracy of 82.43%.