Predicting NBA Players' Salaries - MGT 6203 Final Report

Wissam Afyouni, Nameer Rehman, Fuat Yurekli, Ahmad El-hendawy, and Maliha Rishat

2023-04-16

Overview

Project Context

The National Basketball Association (NBA) is a multi-billion-dollar industry, with player salaries being a significant component of team expenditures. NBA players are one of the highest paid athletes—a 2015 Insider article names the NBA "the highest-paying sports league in the world" (Gaines 2015). Team owners are given a soft salary cap by the league that they cannot surpass when determining player salaries without potentially sacrificing free agency perks, and the team's general manager operates within this cap when negotiating contracts with players. Teams and agents must make informed decisions about which players to sign and at what price, considering a variety of factors such as age, performance metrics, position, team, and contract length. However, accurately predicting NBA player salaries is a complex task due to the large number of interrelated factors involved.

Problem Overview

Literature Review

Salary prediction for NBA players is a popular topic often explored by many sports enthusiasts and stakeholders in the sports industry. One interesting research paper on the topic is Ioanna Papadaki and Michail Tsagris's article on "Are NBA Players' Salaries in Accordance with Their Performance on Court?". Papadaki and Tsagris walk through past approaches to this question, which largely use linear regression to estimate team revenues and player salaries. They analyzed 2016-2019 seasons data using LASSO for variable selection, and using k-fold CV for validation and testing. Using the Random Forest algorithm on experience, total minutes participated in games, and the number of times a player was in the starting five position, they were able to predict salaries with an AUC of ~0.8 across the three seasons. While this is a good AUC, the Random Forest approach lacks interpretability in a business scenario and may be hard to sell to stakeholders if the predictive accuracy is not sufficiently high.

The Kaggle notebook NBA Salary Prediction uses extreme gradient boosting to regress on age, points per game, player position, steals, goals, rebounds, assists, fouls, and minutes played. This method can lead to overfitting more than others, and outliers can skew the model to a larger extent than other models because of its iterative learning nature.

Problem Statement

This project seeks to explore the relationship between player performance data and salary data to develop predictive models that can help a team's general manager make informed decisions about player contracts. We will develop a model for GMs to use when negotiating player contracts, allowing them to make data-driven decisions that maximize their team's potential for success.

Primary and Supporting Research Questions

- What factors have the strongest impact on NBA player salaries and how accurately can we predict player salaries based on these factors?
  - Which factors/characteristics should be explored in predicting how much of the salary cap certain players should receive?
  - Are teams signing players to purposely under-perform and build a team for the future?
  - What types of relationships can we observe when looking at NBA salaries and successful teams?

Methodology

The current methodology of our project revolves around two primary tasks: (1) ETL and (2) Modeling. These tasks are subdivided and further elaborated on in the following sections. The former involves data sourcing via web harvesting and data cleaning; the latter includes model exploration, model selection and final model testing.

ETL

Extract

The ETL Process involves several stages. Our data was exclusively extracted from the website basketball reference, which is considered one of the most reliable open source data hubs for NBA statistics. In order to extract the data without overwhelming the website's servers, we decided to extract and save the raw html files for each player's profile to our local environment, then extract the information we wanted from each html file. This way, we wouldn't have to re-pull information from basketball reference's website each time we decided we wanted some other information that would be contained on the player's basketball reference profile page.

First, we manually downloaded player lists for each season of the NBA from basketball reference, then extracted the URL for each of those players and concatenated them into a list.

```
library(httr)
library(xml2)
library(rvest)
library(tools)
library(pbapply)
library(progressr)
library(rstudioapi)

import_player_lists <- function(extract_path) {
  # Get list of HTML files in extract_path
  player_lists <- list.files(path = extract_path, pattern = "\\.html$",
                             full.names = TRUE)

  # Initialize empty list to store player links
  player_links <- list()

  # Loop through each HTML file
  for (player_list in player_lists) {
    print(player_list)

    # Read and Parse HTML using rvest package
    text <- rvest::read_html(player_list)

    # Find all td elements with class 'rank-name player'
    td_elements <- rvest::html_nodes(text, 'td[data-stat="player"]')

    # Loop thru td elements & extract href attr of elements with class 'team-name'
    for (td in td_elements) {
      a_elements <- rvest::html_nodes(td, "a")
      for (a in a_elements) {
        player_links[[length(player_links) + 1]] <- a %>% rvest::html_attr("href")
      }
    }
  }
  # Remove duplicates from player_links list and return
  return(unique(player_links))
}
```

Next, before we started downloading the NBA player's profiles, we checked to make sure that we had not already downloaded them (as to avoid downloading the player's profile twice and needlessly pinging Basketball Reference's servers a second time). Then, we'd have a full list of URLs for player's profiles we'd like to extract from Basketball Reference.

```
get_links_to_pull <- function(all_player_links, extract_path) {
  # Links that have been extracted
  links_extracted <- list.files(path = extract_path, full.names = TRUE)
  links_extracted = paste0("https://www.basketball-reference.com/players/",
                          gsub("-", "/",
                               tools::file_path_sans_ext(basename(links_extracted)
                                                          )
                               ),
                          '.html')

  links_extracted <- as.list(links_extracted)
  links_to_pull <- setdiff(all_player_links, links_extracted)
  return(links_to_pull)
}
```

And lastly, we'd extract and save each NBA player's profile page, making sure to wait a few seconds between requests in order to ensure that we weren't overwhelming Basketball Reference's servers (as they point out in their Terms of Service). This would complete the extraction process.

```
extract_html <- function(links_to_pull, extract_path) {
  msgp1 <- "\nExtracting missing Basketball Reference data"
  msgp2 <- " and saving it into local working directory...\n"
  cat(msgp1, msgp2)
  links_to_pull <- sample(links_to_pull)
  extract_with_progress <- function(x) {
    p <- progressr::progressor(along=length(links_to_pull))
    sum <- 0
    for (i in seq(links_to_pull)) {
      x = as.character(links_to_pull[i])
      response <- GET(x)
      site_parsed <- rvest::read_html(response$content)
      file_name <- tail(strsplit(x, "/")[[1]], 3)[3]
      file_path <- paste0(extract_path, "/", file_name, ".html")
      write_html(site_parsed, file_path)
      sum <- sum + i
      p(message = sprintf("Adding %g", i))
      Sys.sleep(sample(6:10, 1))
    }
  }
  progressr::with_progress(y <- extract_with_progress(links_to_pull))
  cat("\nExtracting Data Complete.\n")
}
```

Transform

To carry out the transformation process, we would go through every HTML file and retrieve the necessary data for each player. Our primary objective centered around obtaining player performance and salary data, with the intention of gathering as many features as possible to test in our models. Our aim was to gain insight into which variables were most significant in influencing a player's salary. To accomplish this, we extracted both their regular season and playoff data, along with the stats categorized as 'regular' performance analytics data, and also the 'advanced' performance analytics data for every player. Lastly, we pulled the salary data for each player as well. The aim was to create five separate tables for each of these data sources, and keep the data organized in one-to-one relationships so they could easily be joined for analysis as necessary.

We modularized the code, and created functions for each step of the cleaning process that could be applied to more than one data source.

```
library(dplyr)
library(stringr)

clean_salary_html_table <- function(page, id_name, player_code) {
  div <- page %>% html_node(id_name)
  if (is.na(div)) {
    message(paste0(id_name, ' not found for ', player_code))
    return(page)
  }
  div <- gsub("<!--", "", as.character(div))
  div <- gsub("-->", "", as.character(div))
  div <- read_html(div)
  return(div)
}
clean_salary_table <- function(df) {
  if (nrow(df) == 0) {
    return(data.frame())
  }
  # Extract Digits from Column
  df$Salary <- gsub("[^[:digit:]]", "", df$Salary)
  df$Salary <- as.numeric(df$Salary)
  return(df)
}

clean_performance_table <- function(df, season_type) {
  if (nrow(df) == 0) {
    return(data.frame())
  }
  df[df == ''] <- NA
  # Convert Stats Columns to Floats
  columns_to_not_convert <- c('Player_Code', 'Player_Name', 'Season', 'Tm', 'Pos')
  columns_to_convert <- setdiff(names(df), columns_to_not_convert)
  df[columns_to_convert] <- apply(df[columns_to_convert], 2, as.numeric)
  # Add Season_Type to Column
  column_names <- ifelse(names(df) %in% columns_to_convert,
                         paste0(names(df), '_', season_type),
                         names(df))
  names(df) <- column_names
  return(df)
}
```

We initiated the task of loading the HTML profile page of each player and extracting the required tables by utilizing the get_table extraction function that we had specifically developed for this purpose. The function was designed to search for the specified div table as per the function parameters within each page.

```
get_table <- function(page, id_name, player_name, player_code) {
  div <- page %>% html_node(id_name)
  if (is.na(div)) {
    message(paste0(id_name, ' not found for ', player_code))
    return(data.frame())
  }
  table <- div %>% html_node('table')
  df <- html_table(table)
  df$Player_Code <- player_code
  df$Player_Name <- player_name
  df <- df[str_detect(df$Season, '\\d{4}-\\d{2}'), ]
  df$Season <- as.integer(substr(df$Season, 1, 4))
  df <- df[, !(names(df) %in% c('Lg', ''))]
  df <- as.data.frame(df)
  return(df)
}
```

Finally, we would take the data from the extracted tables, and utilize the modularized functions that we had developed for the purpose of transforming and cleaning each data set. Once this was done, we appended the tables to a dataframe and saved each as a distinct csv file. This process was executed for every HTML page once it was transformed.

```
transform <- function(files, transform_path) {
  df_salary <- data.frame()
  df_performance_rs_totals <- data.frame()
  df_performance_po_totals <- data.frame()
  df_performance_rs_advanced <- data.frame()
  df_performance_po_advanced <- data.frame()

  for (file in files) {
    page <- rvest::read_html(paste0(as.character(extract_path), '/', file))
    # Get Player_Name
    player_name <- page %>%
      rvest::html_nodes("h1") %>%
      html_text(trim = TRUE)
    # Get Player_Code:
    player_code <- strsplit(strsplit(file, "-")[[1]][2], "\\.html")[[1]][1]
    print(player_code)
    # Get BBall_Ref Player Code
    salary_html <- clean_salary_html_table(page, '#all_all_salaries',
                                           player_code)
    salary <- get_table(salary_html, '#div_all_salaries', player_name,
                        player_code)
    performance_rs_totals <- get_table(page, '#div_totals', player_name,
                                       player_code)
    performance_po_totals <- get_table(page, '#div_playoffs_totals',
                                       player_name, player_code)
    performance_rs_advanced <- get_table(page, '#div_advanced',
                                         player_name, player_code)
    performance_po_advanced <- get_table(page, '#div_playoffs_advanced',
```

```
                                         player_name, player_code)
    # Clean BBall_Ref Tables
    salary <- clean_salary_table(salary)
    performance_rs_totals <- clean_performance_table(performance_rs_totals,
                                                     'rs')
    performance_po_totals <- clean_performance_table(performance_po_totals,
                                                     'po')
    performance_rs_advanced <- clean_performance_table(performance_rs_advanced,
                                                       'rs')
    performance_po_advanced <- clean_performance_table(performance_po_advanced,
                                                       'po')
    # Append to DataFrame
    df_salary <- bind_rows(df_salary, salary)
    df_performance_rs_totals <- bind_rows(df_performance_rs_totals,
                                          performance_rs_totals)
    df_performance_po_totals <- bind_rows(df_performance_po_totals,
                                          performance_po_totals)
    df_performance_rs_advanced <- bind_rows(df_performance_rs_advanced,
                                            performance_rs_advanced)
    df_performance_po_advanced <- bind_rows(df_performance_po_advanced,
                                            performance_po_advanced)
  }
  # Write Tables
  write.csv(df_salary, file=paste0(transform_path, 'df_salary.csv'),
            row.names = FALSE)
  write.csv(df_performance_rs_totals, file=paste0(transform_path,
                                                  'df_performance_rs_totals.csv'),
            row.names = FALSE)
  write.csv(df_performance_po_totals, file=paste0(transform_path,
                                                  'df_performance_po_totals.csv'),
            row.names = FALSE)
  write.csv(df_performance_rs_advanced, file=paste0(transform_path,
                                                    'df_performance_rs_advanced.csv'),
            row.names = FALSE)
  write.csv(df_performance_po_advanced, file=paste0(transform_path,
                                                    'df_performance_po_advanced.csv'),
            row.names = FALSE)
}
```

Load

With all the data ingested, the individual tables need to be joined to create an encompassing data set with all the required factors we intended to explore. The individual tables to be joined are summarized below.

1. Player regular season counting stats by team and season
2. Player regular season advanced stats by team and season
3. Player salary by team and season
4. NBA overall salary cap by season

Fortunately, since each data set was obtained from the same source, the structure of each of the tables were very similar. However, since the data spanned multiple seasons, a unique ID needed to be created to join tables on, by combining unique player id and season. Additionally, there were some particulars with players found during data exploration that required additional transformations.

1. Players traded mid-season played on multiple teams in a season & had their performance metrics split up into multiple records. Only the summed stats were kept for these players.
2. Players on occasion could have multiple salaries in a single season as they were released from a team and signed to another. These player's salaries were averaged out for the year.

Once data was transformed and joined, a player's salary as a percent of the cap was calculated to be used as a potential response variable as opposed to raw salary.

```r
library(tidyverse)

df_rs_totals <- read.csv('bball_ref_performance_rs_totals.csv',encoding="cp1252")
df_advanced <- read.csv('bball_ref_performance_rs_advanced.csv',encoding="cp1252")

df_rs_totals$ID <- paste(df_rs_totals$Player_Name,
                         as.character(df_rs_totals$Season),
                         sep="")
df_advanced$ID <- paste(df_advanced$Player_Name,
                        as.character(df_advanced$Season),
                        sep="")

#join counting stats and advanced stats
df_rs <- merge(df_rs_totals, df_advanced, by="ID") %>%
  select(-Season.y, -Player_Name.y, -Player_Code.y) %>%
  rename(Season=Season.x, Player_name=Player_Name.x, Player_Code=Player_Code.x)

#get only the "TOT" row for Players that played for multiple teams in a season
df_rs$Team.x <- ifelse(df_rs$Team.x=="TOT", "9", df_rs$Team.x)
df_rs <- df_rs %>% group_by(ID) %>% slice(1) %>%
  mutate(Team.x=ifelse(Team.x=="9","TOT", Team.x)) %>% ungroup()

df_salary <- read.csv('bball_ref_salary.csv',encoding="cp1252")
df_salary$ID <- paste(df_salary$Player_Name,
                      as.character(df_salary$Season),
                      sep="")
```

```r
df_salary <- df_salary %>% group_by(Player_Name, Season, Team, ID) %>%
  summarize(Salary=mean(Salary, na.rm=TRUE)) %>% ungroup()

df_rs_salary <- merge(df_rs, df_salary, by="ID", all.x=TRUE) %>%
  drop_na(Salary) %>%
  select(-Season.y) %>%
  rename(Season=Season.x)
df_cap <- read.csv('salary_cap.csv')
df_rs_salary <- merge(df_rs_salary, df_cap, by="Season", all.x=TRUE)

#get salary as % of cap
df_rs_salary$cap_perc <- df_rs_salary$Salary/df_rs_salary$salary_cap

#save as csv
write.csv(df_rs_salary, 'rs performance and salary.csv', row.names=FALSE)
```

EDA

Below is a glimpse of the final data set split into multiple tables given the large number of feature columns present.

Table 1: Quick Glimpse of the Loaded Dataset

ID	Player_Code	Player_name	Last_Team	Season	Age
Aaron Brooks2007	brookaa01	Aaron Brooks	MIN	2007	23
Aaron Brooks2008	brookaa01	Aaron Brooks	MIN	2008	24
Aaron Brooks2009	brookaa01	Aaron Brooks	MIN	2009	25
Aaron Brooks2010	brookaa01	Aaron Brooks	MIN	2010	26

Team_x	Pos	G	GS	MP	FG	FGA	FG%
HOU	PG	51	0	608	93	225	0.413
HOU	PG	80	35	1998	316	783	0.404
HOU	PG	82	82	2919	575	1331	0.432
TOT	PG	59	12	1284	220	587	0.375

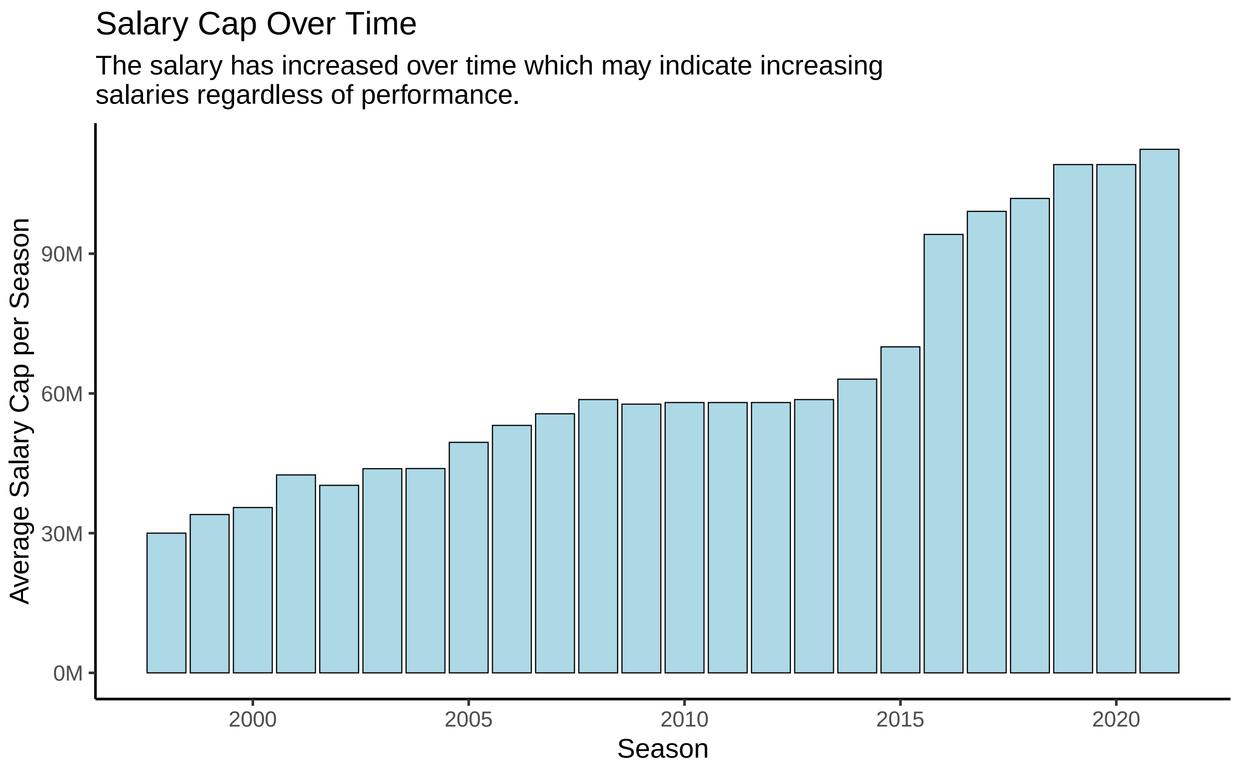
3P	3PA	3P%	2P	2PA	2P%	eFG%	FT	FTA	FT%	ORB	DRB	TRB
36	109	0.330	57	116	0.491	0.493	42	49	0.857	13	43	56
113	309	0.366	203	474	0.428	0.476	149	172	0.866	33	124	157
209	525	0.398	366	806	0.454	0.511	245	298	0.822	54	161	215
70	236	0.297	150	351	0.427	0.434	124	140	0.886	20	58	78

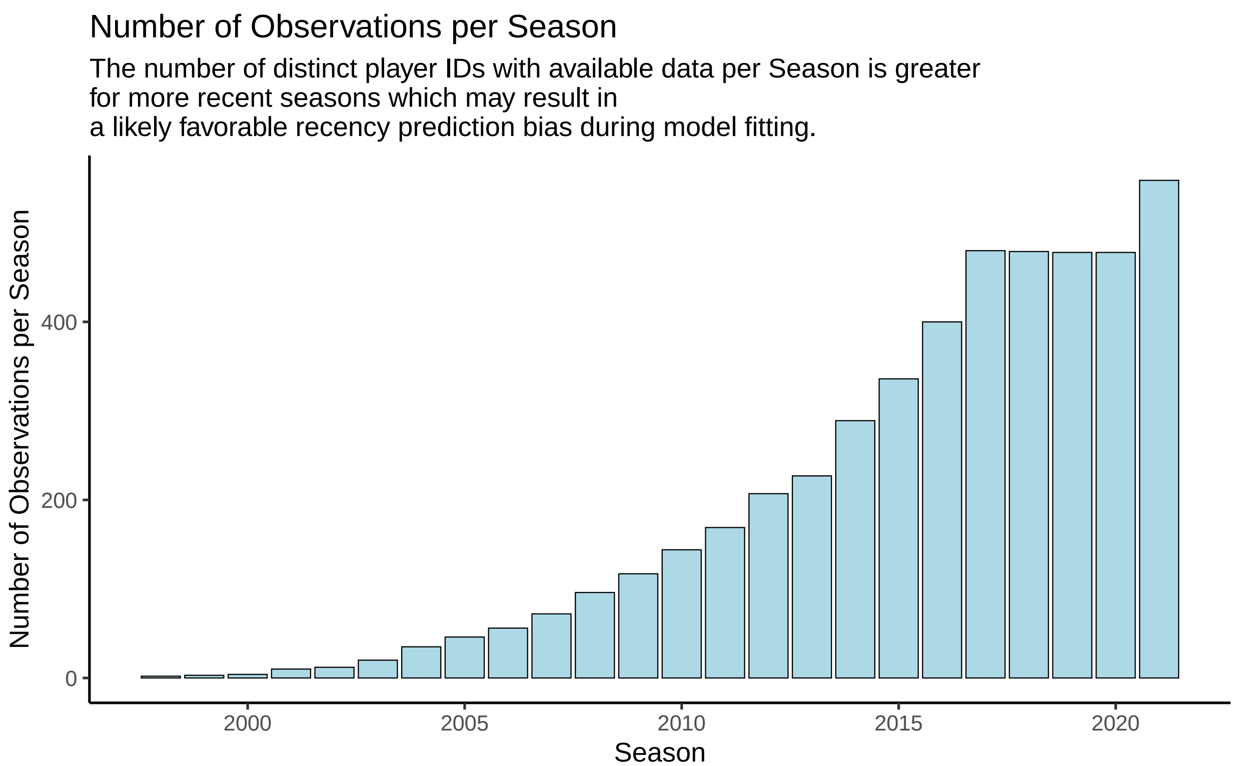
AST	STL	BLK	TOV	PF	PTS	Trp-Dbl	TS%	PER	USG%
87	13	5	44	69	264	NA	0.535	13.1	21.8
238	46	8	125	152	894	NA	0.521	12.9	22.9
434	69	14	232	199	1604	NA	0.549	16.0	25.7
233	34	3	99	115	634	NA	0.489	13.1	25.9

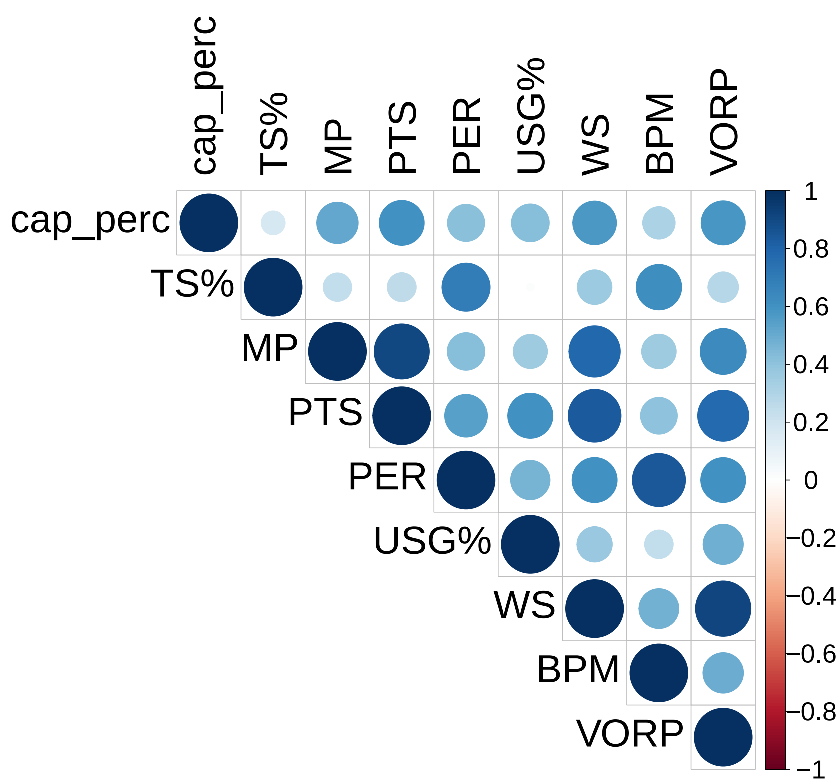
WS	WS/48	BPM	VORP	Tm	Player_Name
1.4	0.112	-0.2	0.3	HOU	Aaron Brooks
3.6	0.086	-0.9	0.6	HOU	Aaron Brooks
5.5	0.091	0.7	1.9	HOU	Aaron Brooks
1.1	0.040	-3.0	-0.3	TOT	Aaron Brooks

Player_Name	Team_y	Salary	salary_cap	2022_dollar_value	cap_perc
Aaron Brooks	Houston Rockets	972720	55630000	75594893	0.0174855
Aaron Brooks	Houston Rockets	1045560	58680000	80023973	0.0178180
Aaron Brooks	Houston Rockets	1118520	57700000	77414680	0.0193851
Aaron Brooks	Phoenix Suns	2016692	58044000	75491545	0.0347442

With the final data set obtained, a correlation matrix was plotted for initial analysis on factors that may influence higher salaries. There are also histograms of both salary and the salary cap which provide insight into the possible distributions of both types of responses during later modeling. It is worth noting that the distribution of the salary cap is quite comparable to that of salary and will likely provide no tangible advantage as a response variable for the modeling phase. Given such, we omitted it from the modeling sections. Moreover, the use of a log transformation on the Salary response variable indicates a possible way for dealing with the highly skewed distribution.

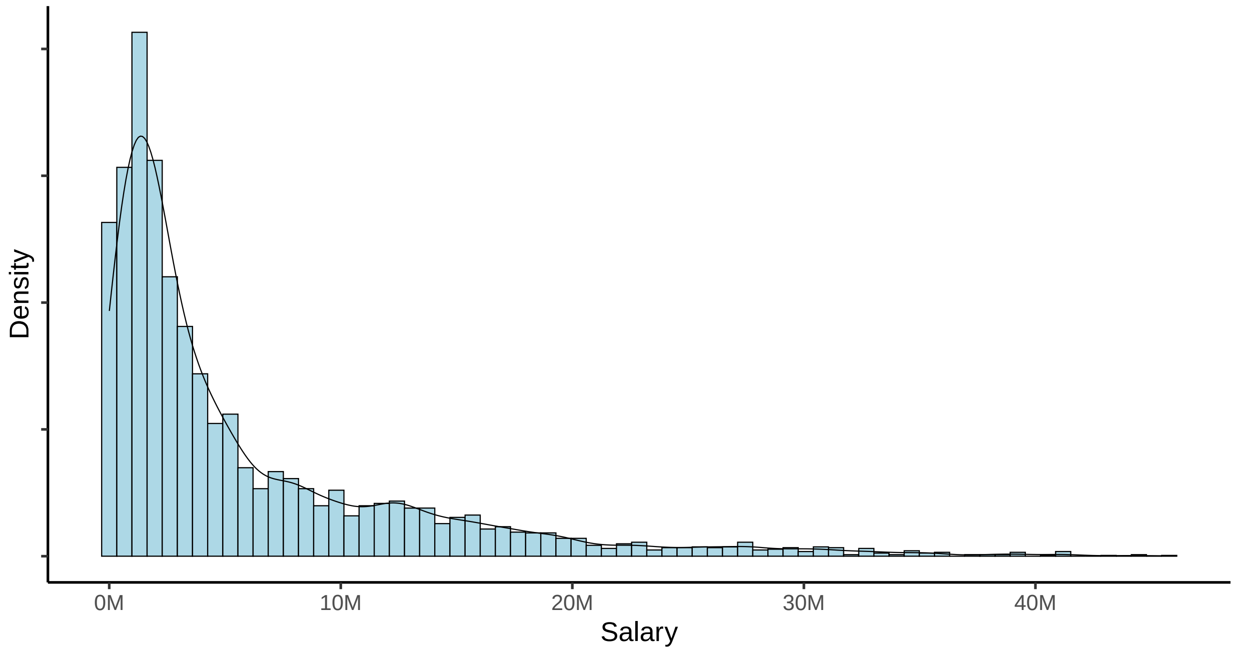



Number of Observations per Season
The number of distinct player IDs with available data per Season is greater
for more recent seasons which may result in
a likely favorable recency prediction bias during model fitting.
Number of Observations per Season
400
200
0
2000
2005
2010
2015
2020
Season


cap_perc
TS%
MP
PTS
PER
USG%
WS
BPM
VORP
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

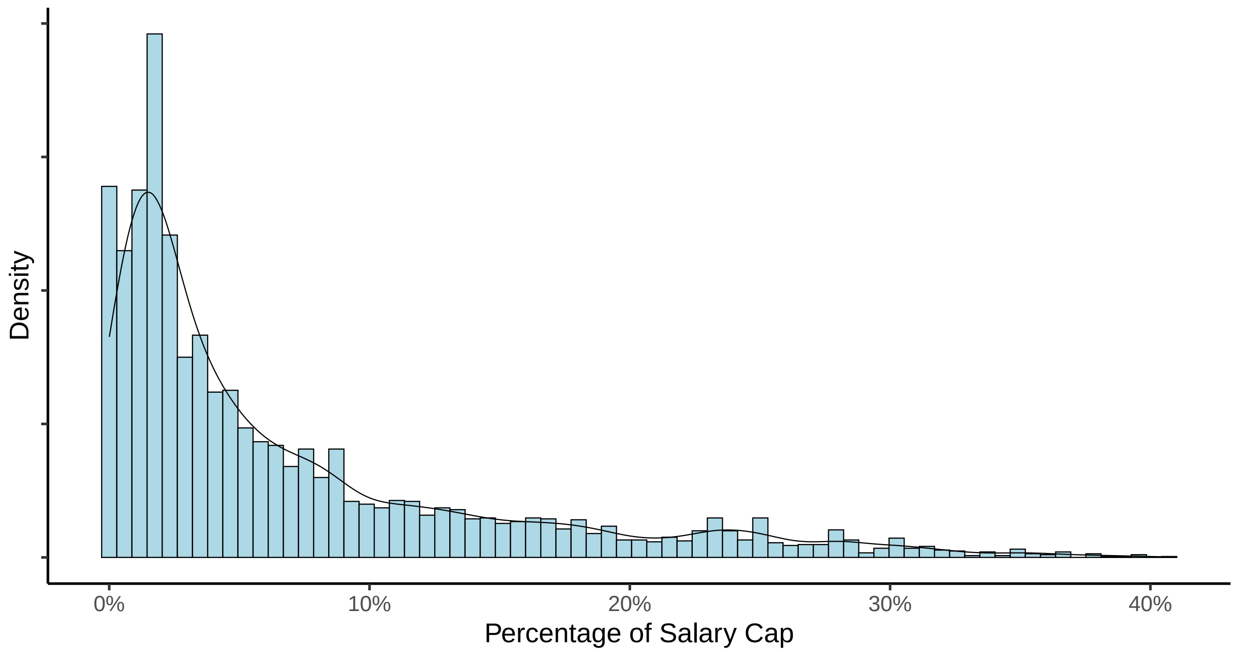
KDE Plot for Salary

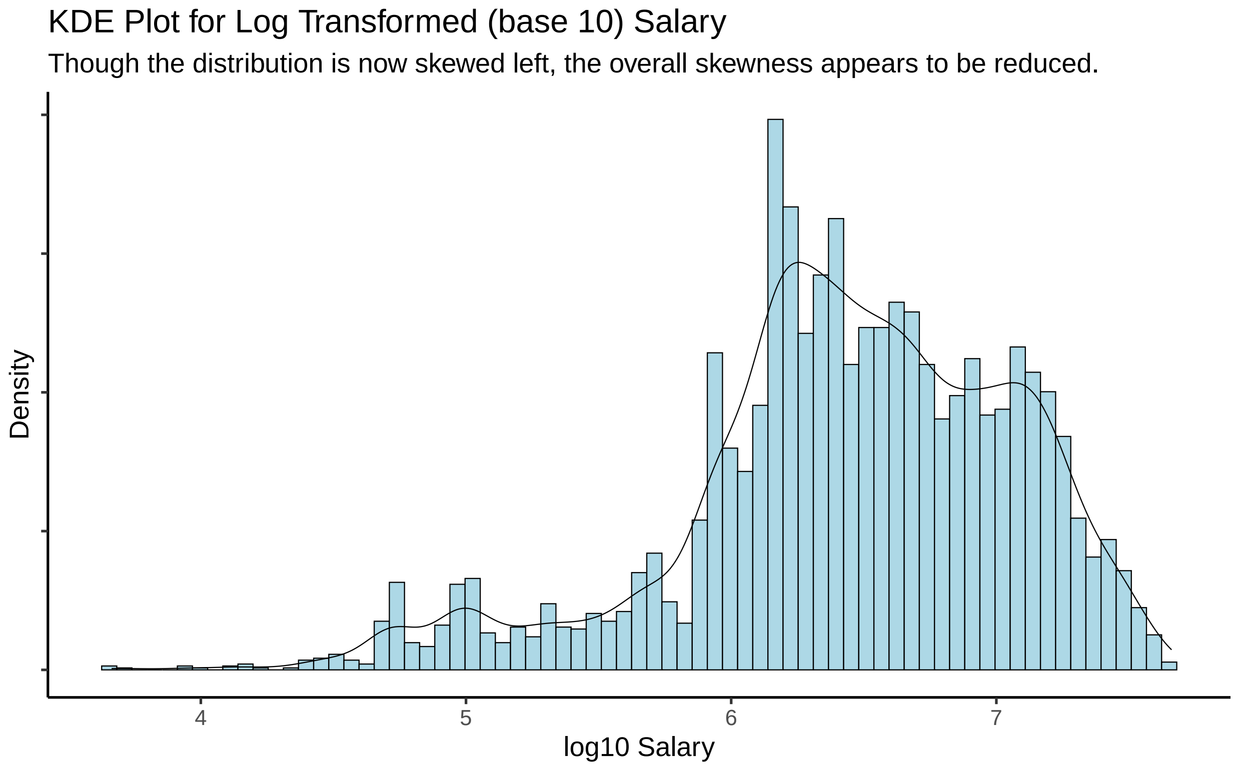
The distribution is highly skewed and suggests there may be problems with using a linear model.



KDE Plot for Percentage of Salary Cap

The use of Salary Cap Percentage appears to offer no significant advantage in addressing the skewed distribution.




KDE Plot for Log Transformed (base 10) Salary
Though the distribution is now skewed left, the overall skewness appears to be reduced.
Density
4
5
6
7
log10 Salary

Modeling

Our approach for model exploration, selection and testing is based on the use of separate training, validation and testing data sets. The final testing data set is the most recent Season's data, and the remaining observations are used for model exploration, training and selection via 10-fold cross-validation. Using the most-recent Season's data for final testing of the selected model places an emphasis on our interest in the estimated predictive capability of our chosen model as the quality of any predictions in a future season will ultimately be of greatest importance.

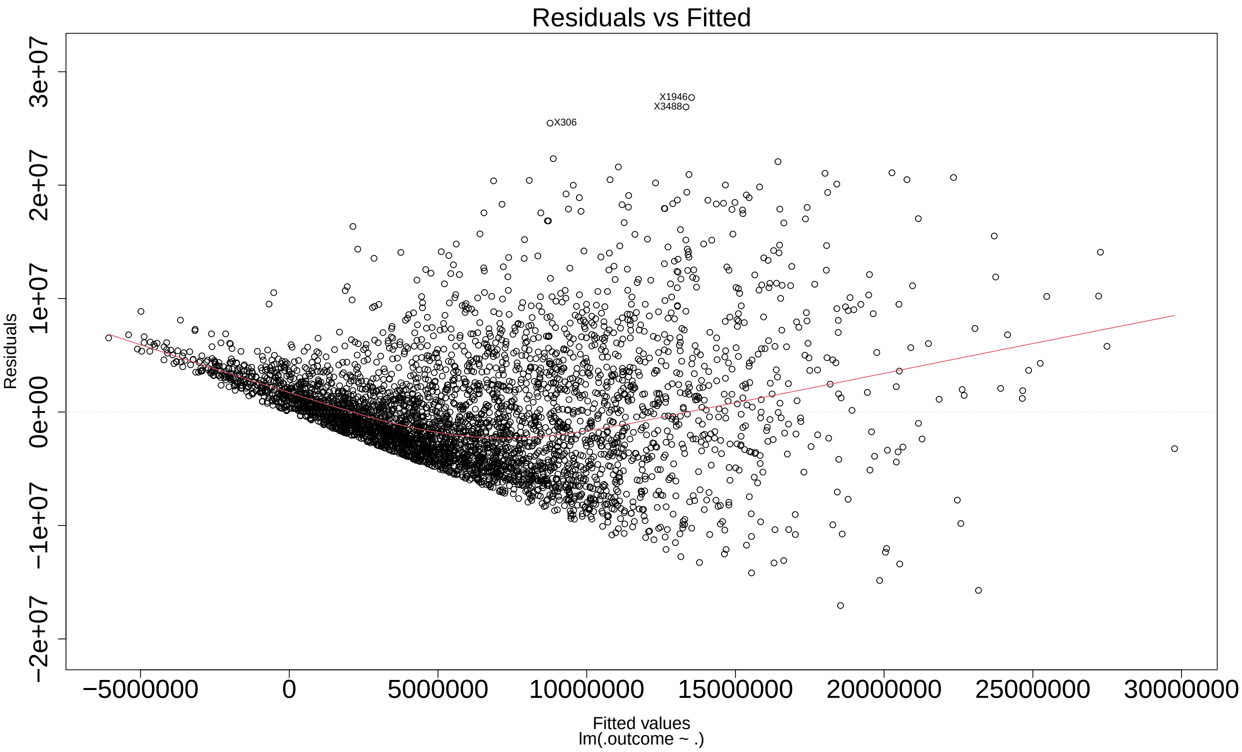
Model Exploration

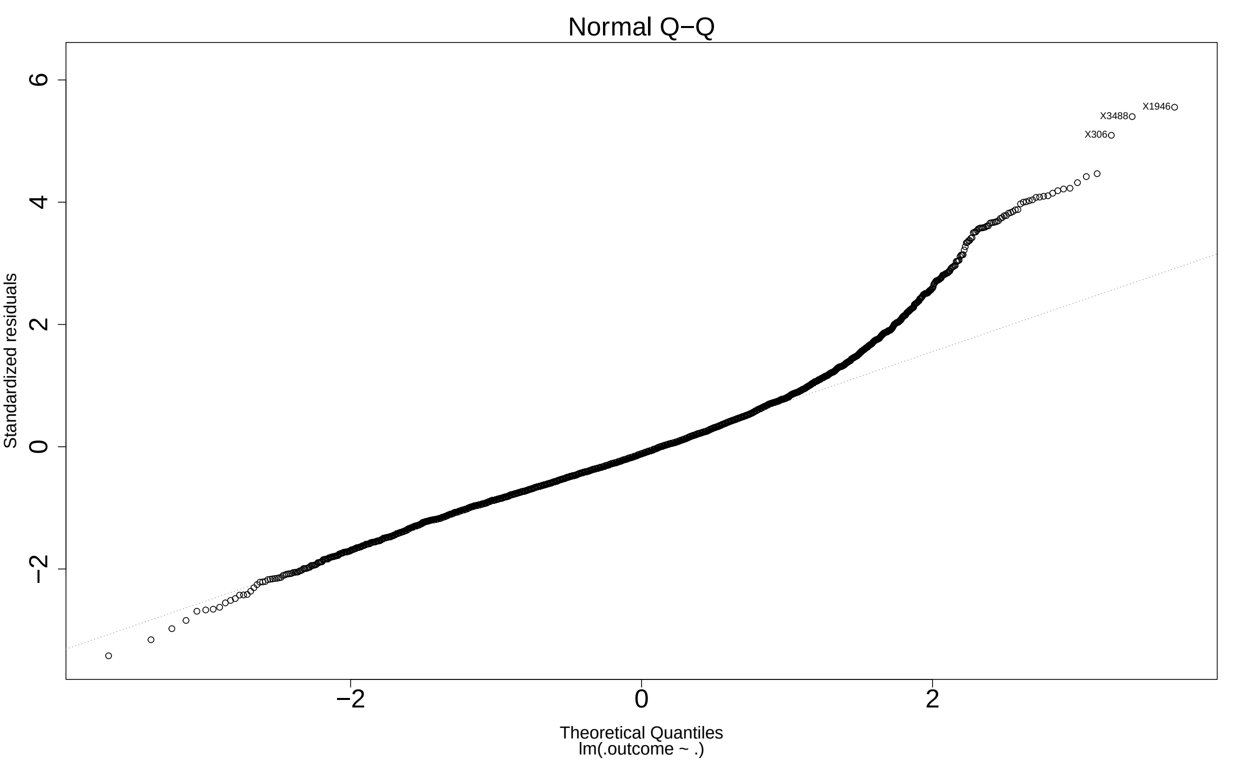
Linear Regression

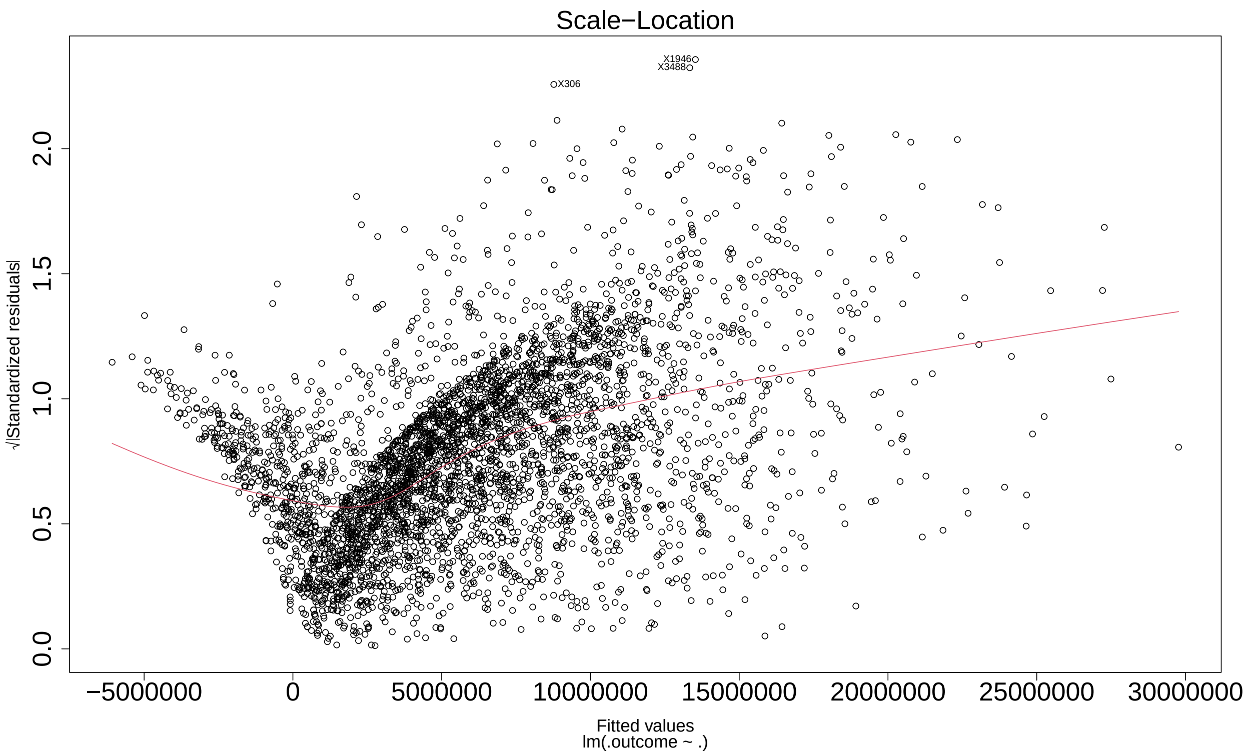
The first type of modeling we explored is linear regression. The reasoning for this is due to higher interpretability offered as well as to form a baseline comparison to existing approaches from the literature review which are fundamentally based on linear models; gradient boosting uses linear models on the residuals and the residuals of the residuals, etc. and random forest models are based on decision trees which apply linear models to the corresponding nodes. Below we can see the model summary along with some informative plots of the final regression model after performing cross-validation with the predictor variables selected using a backward step-wise selection approach (by AIC criterion). There is also a plot of the variance inflation factor for each of the resulting predictor variables. Note that the predictors used for the initial model in the backward step-wise variable selection process are from a predetermined subset that was agreed upon during prior model exploration, and the remaining set used for cross-validation was further refined post-automated selection for reasons that include but are not limited to coefficient significance, multicollinearity, existing literature and domain knowledge. For more details of the column subset used for this and all the following models regardless of model type please refer to the appendix.

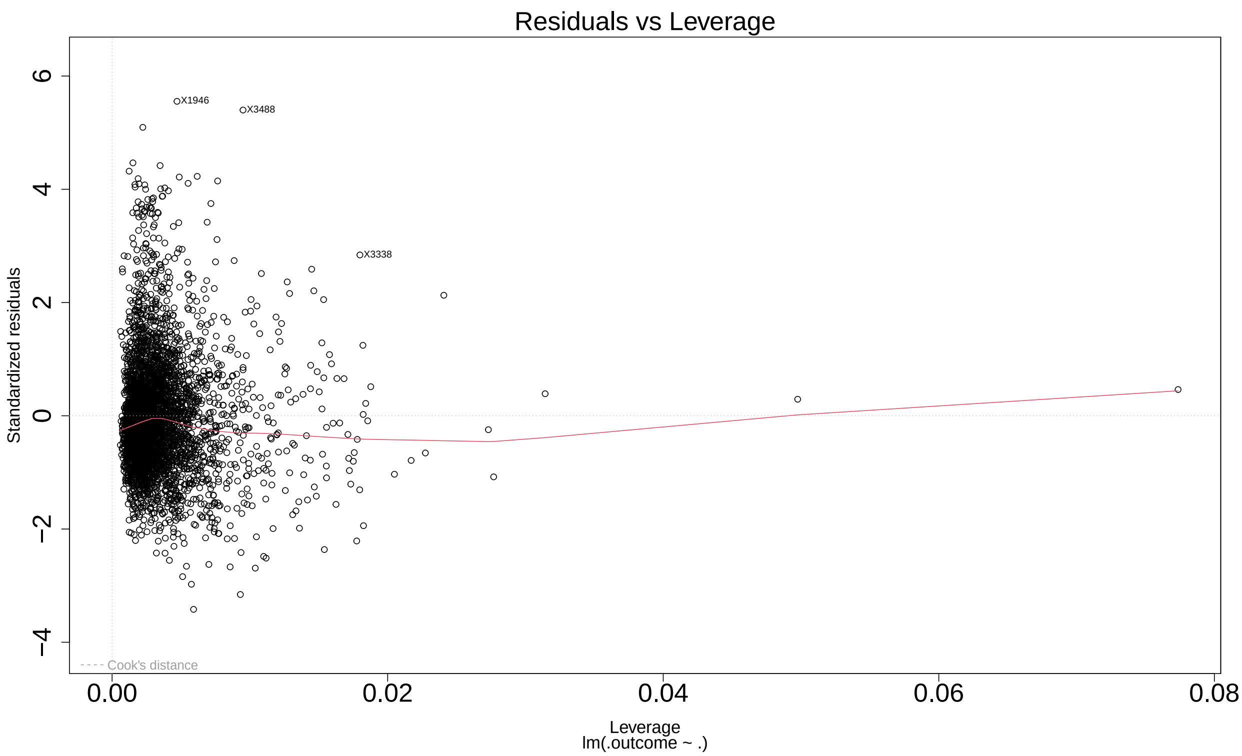
```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
      Min        1Q    Median        3Q       Max
-17061775  -3142251   -581152   2363651  27727392

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -699289141   44089654 -15.861  < 2e-16 ***
BPM               900177      77534  11.610  < 2e-16 ***
AST                 8164       1046   7.801 7.80e-15 ***
‘\\‘WS/48\\‘‘  -26797987    2943660  -9.104  < 2e-16 ***
VORP              377205     119651   3.153  0.00163 **
DRB                13924       1184  11.760  < 2e-16 ***
G                 -46208       6466  -7.146 1.05e-12 ***
STL               -32325       4643  -6.962 3.89e-12 ***
GS                 54630       4640  11.775  < 2e-16 ***
BLK                -8277       4067  -2.035  0.04192 *
‘\\‘FT%\\‘‘      2061250     754706   2.731  0.00634 **
Season            344410      21936  15.701  < 2e-16 ***
‘\\‘3P%\\‘‘     -1979780     694883  -2.849  0.00441 **
Age               465207      20929  22.228  < 2e-16 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

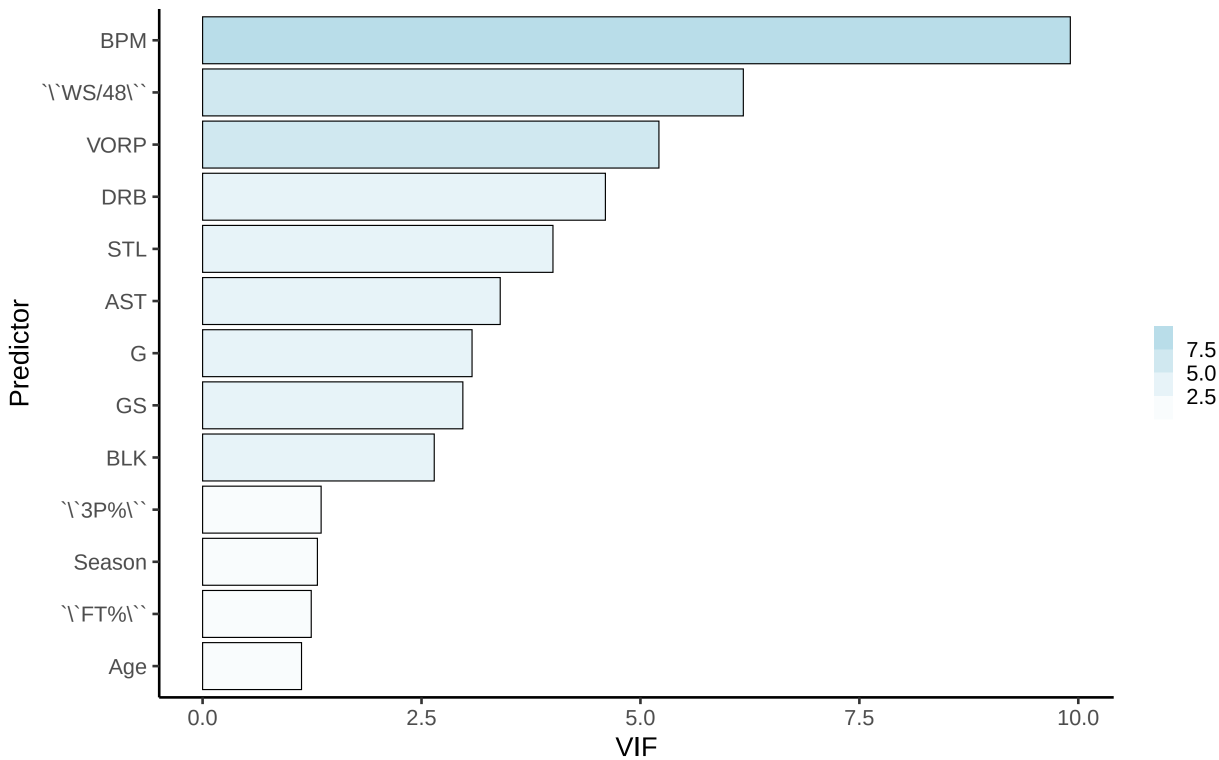
```
Residual standard error: 5005000 on 4009 degrees of freedom
Multiple R-squared:  0.4824,    Adjusted R-squared:  0.4807
F-statistic: 287.4 on 13 and 4009 DF,  p-value: < 2.2e-16
```






Scale−Location
√|Standardized residuals|
Fitted values
lm(.outcome ~ .)
X1946
X3488
X306

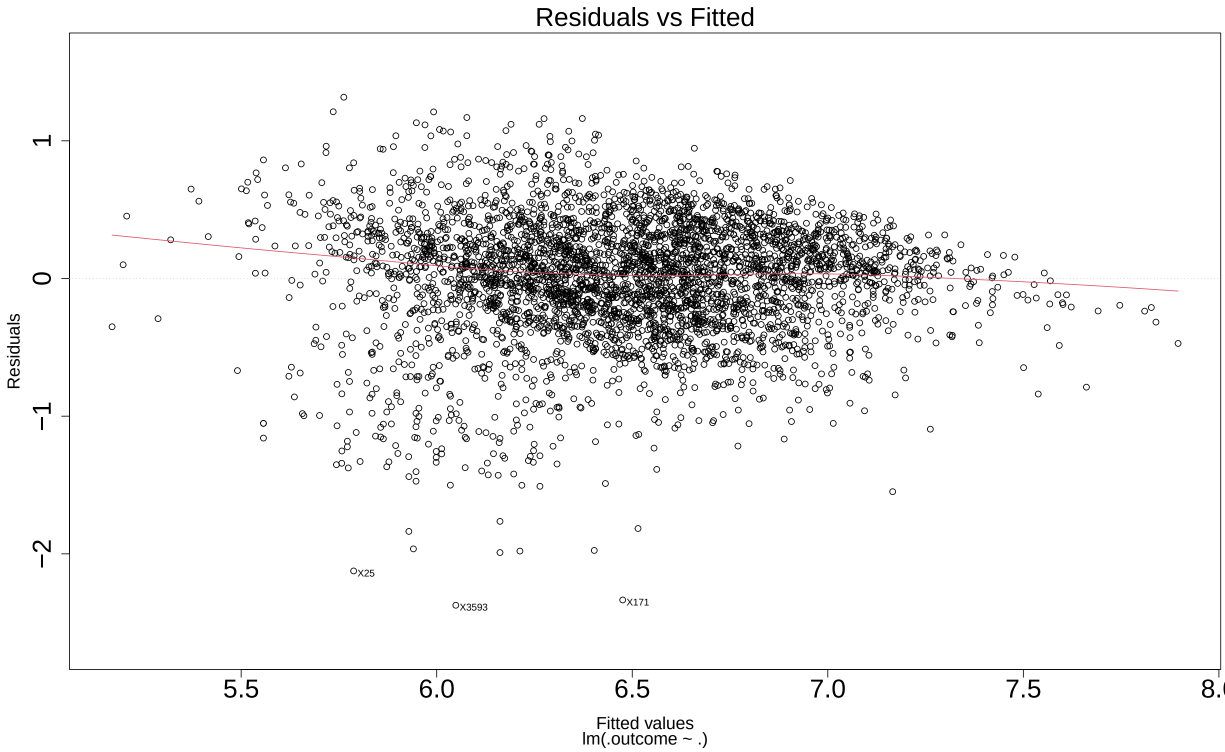

Residuals vs Leverage
Standardized residuals
Leverage
lm(.outcome ~ .)
X1946
X3488
X3338
Cook's distance

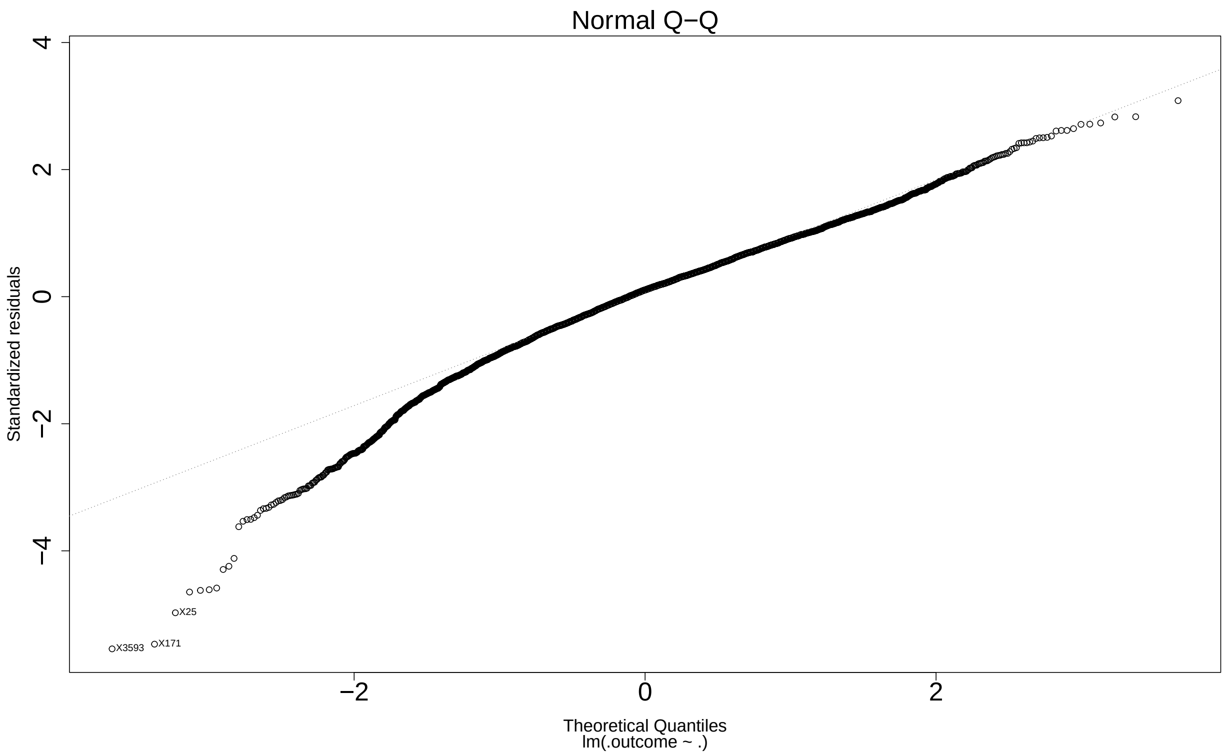


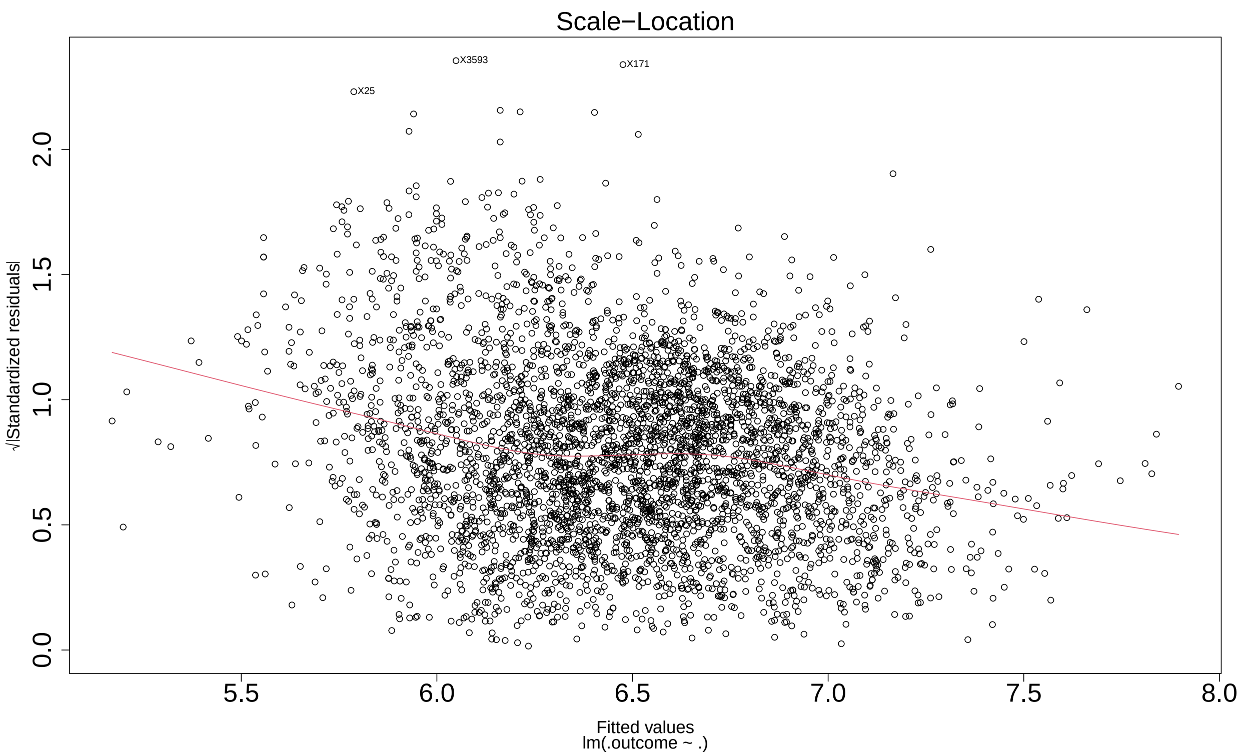
Looking at the plots of the Residuals vs Fitted and Scale-Location, there is clear evidence of heteroscedasticity. To address this, we repeated the previous modeling approach using a log (base-10) transformation on Salary. The resulting plots appears to be promising in their resemblance to Gaussian noise about zero over the range of Salary values but there is still some heteroscedasticity present as the variance in the residuals appears to decrease with increasing Salary. Note that in the log-linear model we removed the BLK and FT% predictor variables as their associated p-values were not below our pre-chosen type-I error rate of 5%.

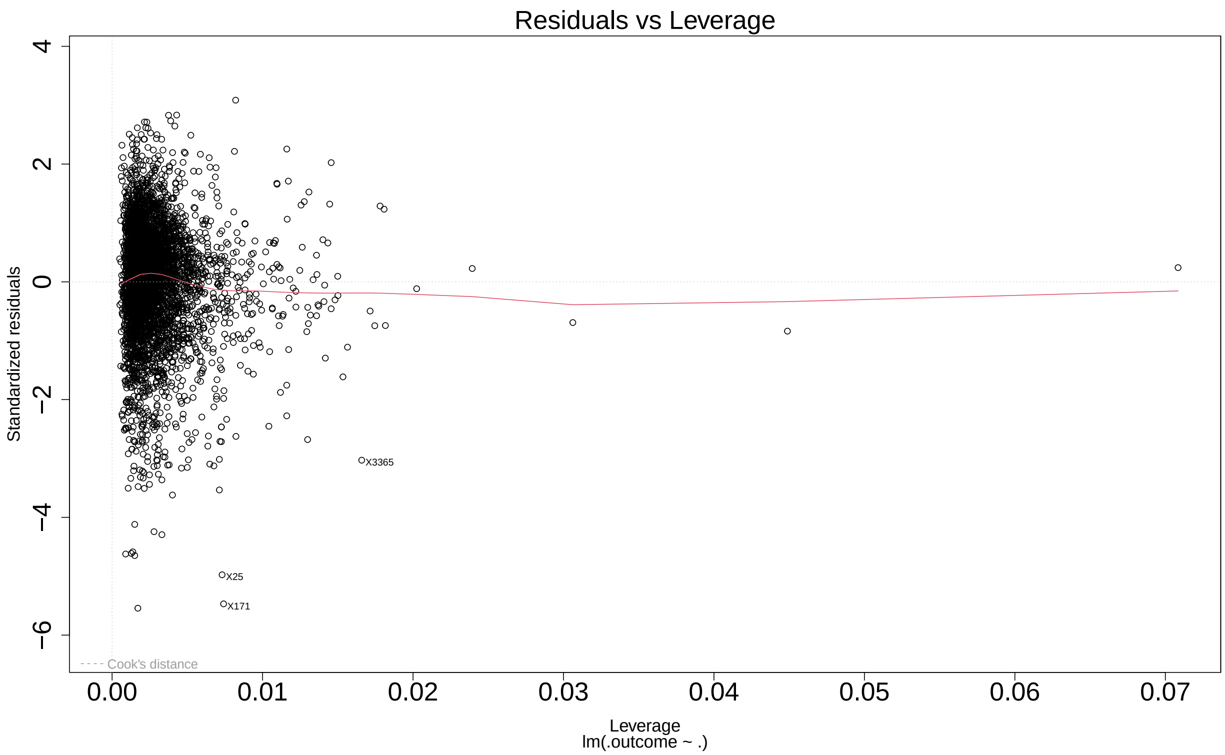
```
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-2.37335 -0.22954  0.04505  0.28315  1.31672

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -4.183e+01  3.775e+00 -11.081  < 2e-16 ***
BPM            7.779e-02  6.599e-03  11.788  < 2e-16 ***
AST            4.481e-04  8.504e-05   5.269 1.44e-07 ***
‘\\‘WS/48\\‘‘ -1.873e+00  2.519e-01  -7.435 1.27e-13 ***
VORP          -2.872e-02  1.015e-02  -2.831 0.004664 **
DRB            7.332e-04  9.100e-05   8.057 1.02e-15 ***
G              4.646e-03  5.427e-04   8.560  < 2e-16 ***
STL           -1.496e-03  3.948e-04  -3.791 0.000152 ***
GS             4.056e-03  3.928e-04  10.325  < 2e-16 ***
Season         2.344e-02  1.878e-03  12.480  < 2e-16 ***
‘\\‘3P%\\‘‘   -1.515e-01  5.707e-02  -2.655 0.007967 **
Age            3.329e-02  1.782e-03  18.679  < 2e-16 ***
```

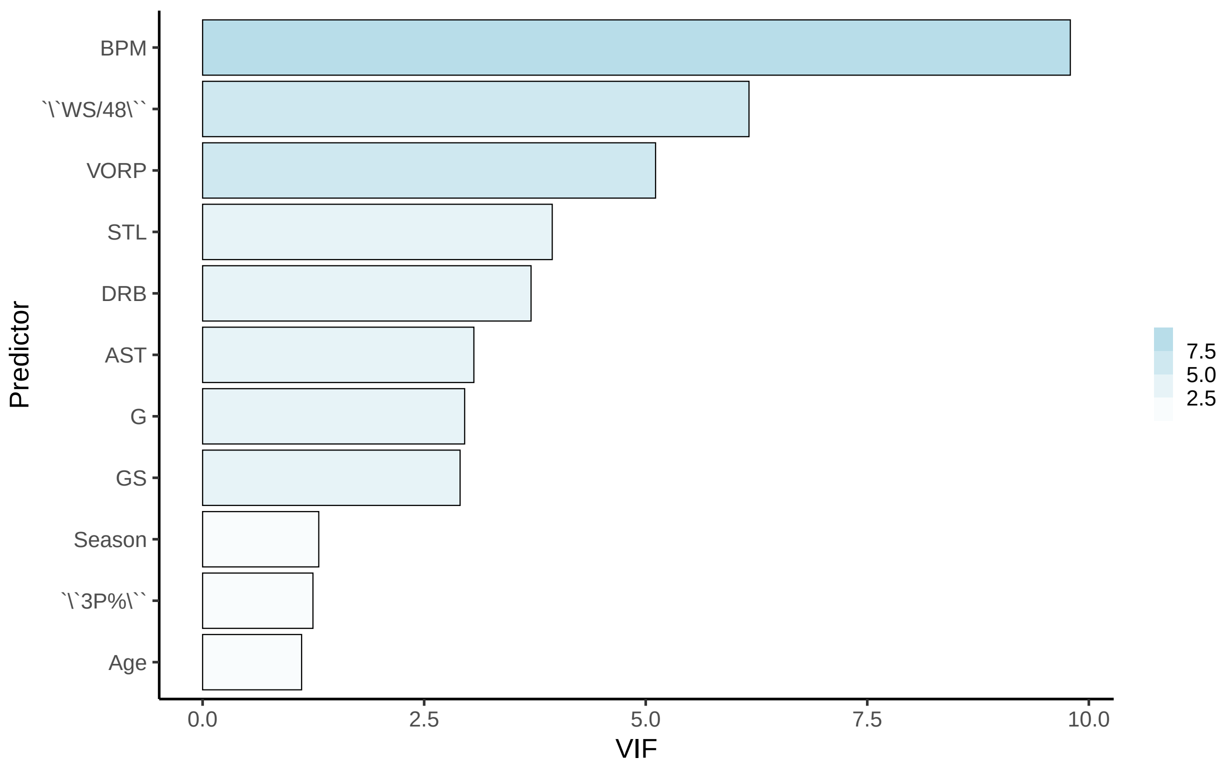
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4285 on 4011 degrees of freedom
Multiple R-squared:  0.4432,	Adjusted R-squared:  0.4417
F-statistic: 290.2 on 11 and 4011 DF,  p-value: < 2.2e-16
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(.outcome ~ .)
X3593
X171
X25

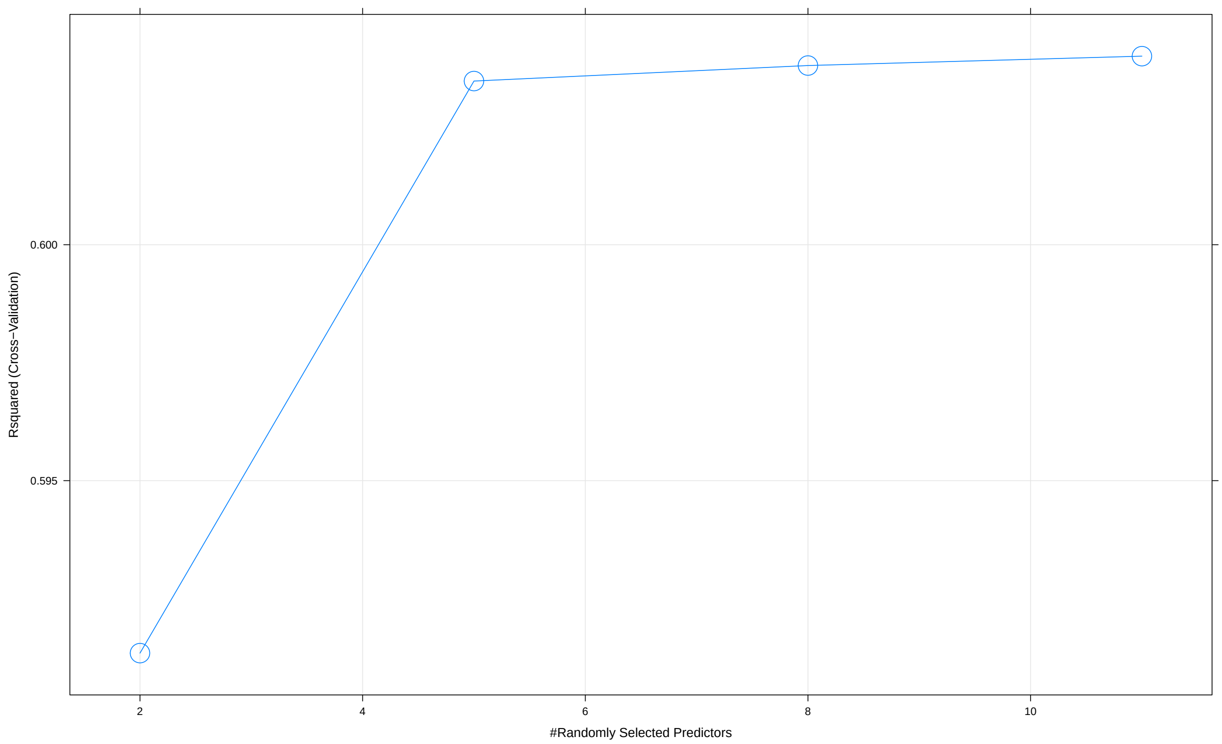

Scale−Location
√|Standardized residuals|
Fitted values
lm(.outcome ~ .)
X3593
X171
X25


Residuals vs Leverage
Standardized residuals
4
2
0
−2
−4
−6
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Leverage
lm(.outcome ~ .)
Cook's distance
X3365
X25
X171

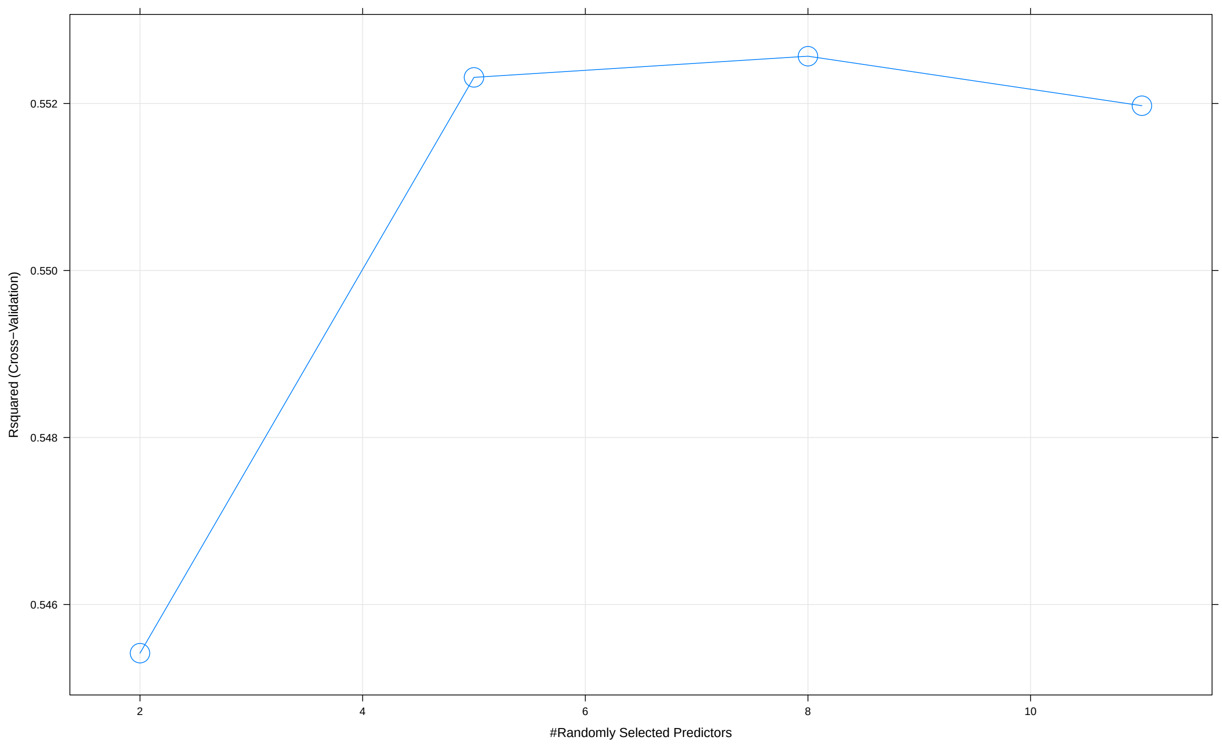

BPM
`\`WS/48\``
VORP
STL
DRB
AST
G
GS
Season
`\`3P%\``
Age
Predictor
0.0
2.5
5.0
7.5
10.0
VIF
7.5
5.0
2.5

Random Forest

Another model we explored is the random forest for not only feasibility in predictive capability but also comparing with the existing literature. In this case, the hyper-parameter of interest is the number of randomly selected predictor variables from the subset. Tuning this parameter led to the lowest associated RMSE using 11 randomly selected predictor variables.

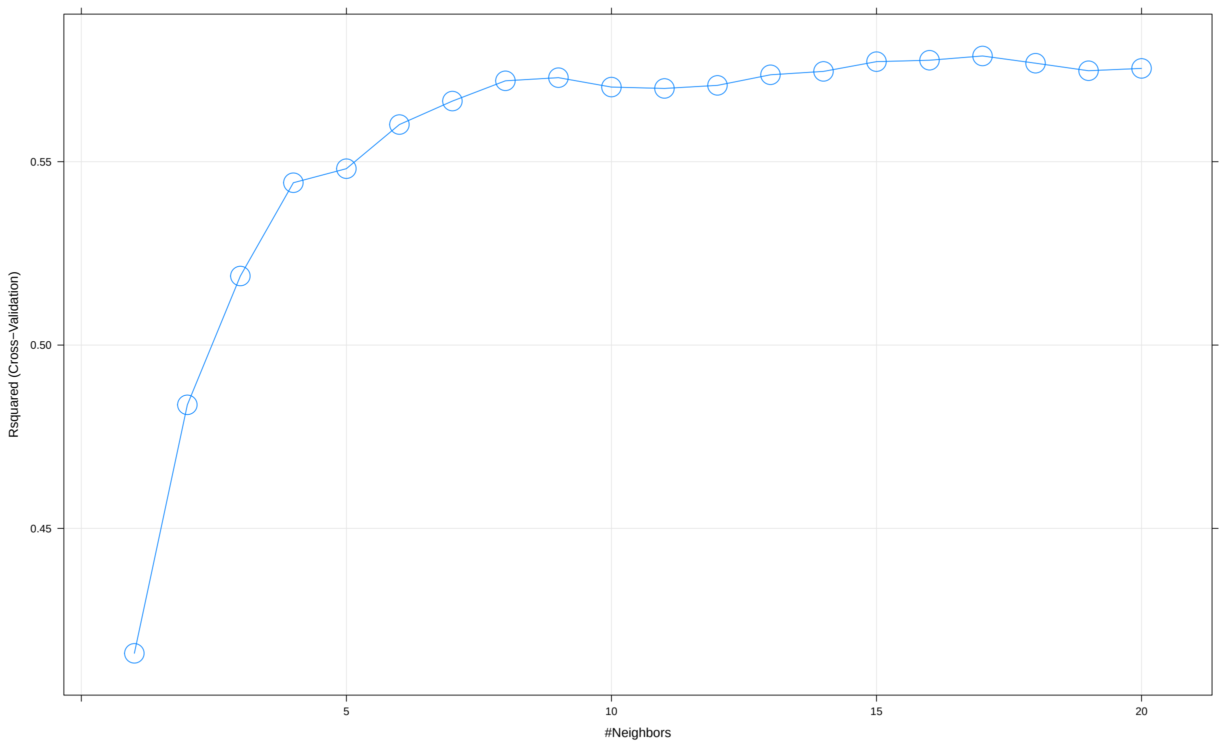


We also ran the model with the log transformed salary variable as an alternative. In that case, 8 was the optimal number of randomly chosen variables which is loosely consistent with the reduced number of features we observed in the log-linear model above.


0.552
0.550
0.548
0.546
Rsquared (Cross−Validation)
2
4
6
8
10
#Randomly Selected Predictors

KNN

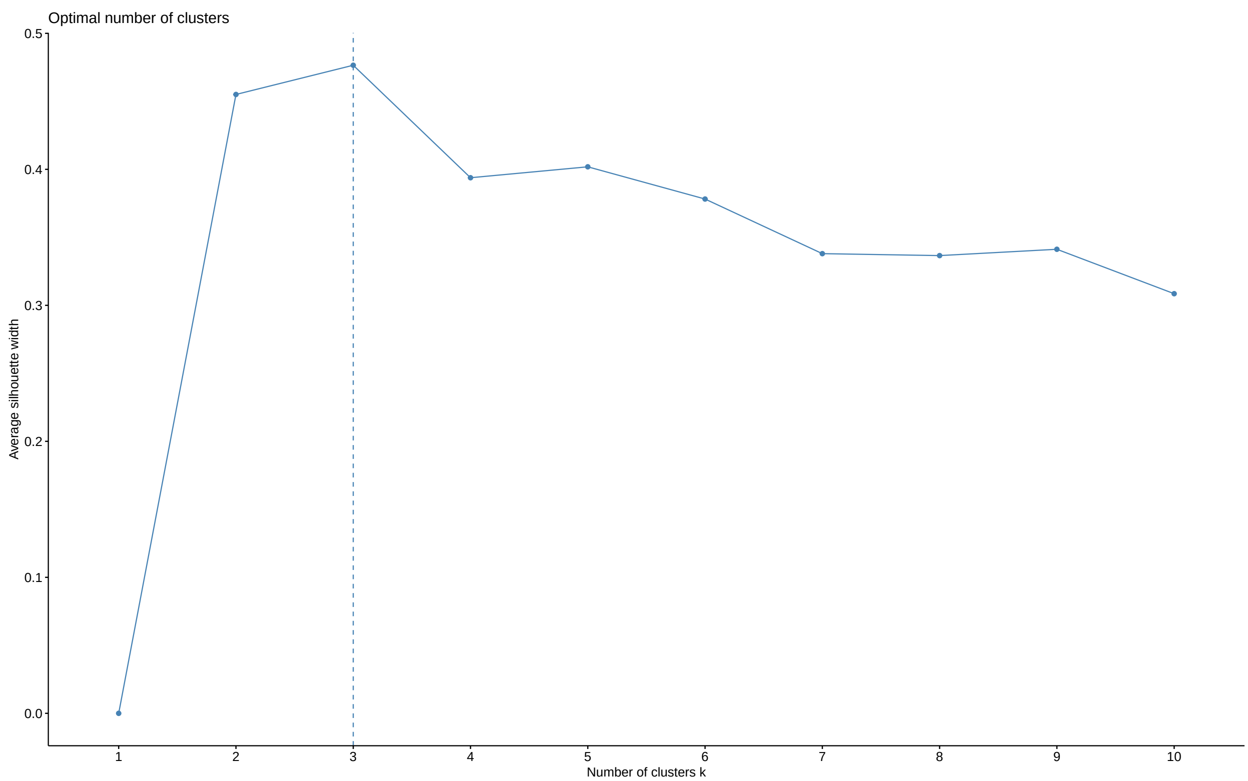
The next model in our exploration and analysis is the K-nearest-neighbor (KNN) model. We chose this model as it has greater flexibility (depending on the choice for the number of neighbors) which can account for non-linear and complex patterns in the data set to make predictions. Such flexibility is favorable if one recalls the skewed distribution from the EDA as well as the likely high-levels of multicollinearity. A 10-fold cross-validation method was used on the training set to tune the hyper parameters to those that will yield the best results with the given seed. The range of values of K used was K=1 through K=20. Additionally, the training data was pre-processed using Yeo-Johnson transformations with all predictors of zero-variance dropped; the predictor variables were also standardized to confirm that we would not be potentially affecting the fitted estimates based on the magnitude and scale of any variable's values. We found that the KNN model produced the best results when the number of neighbors is 17.

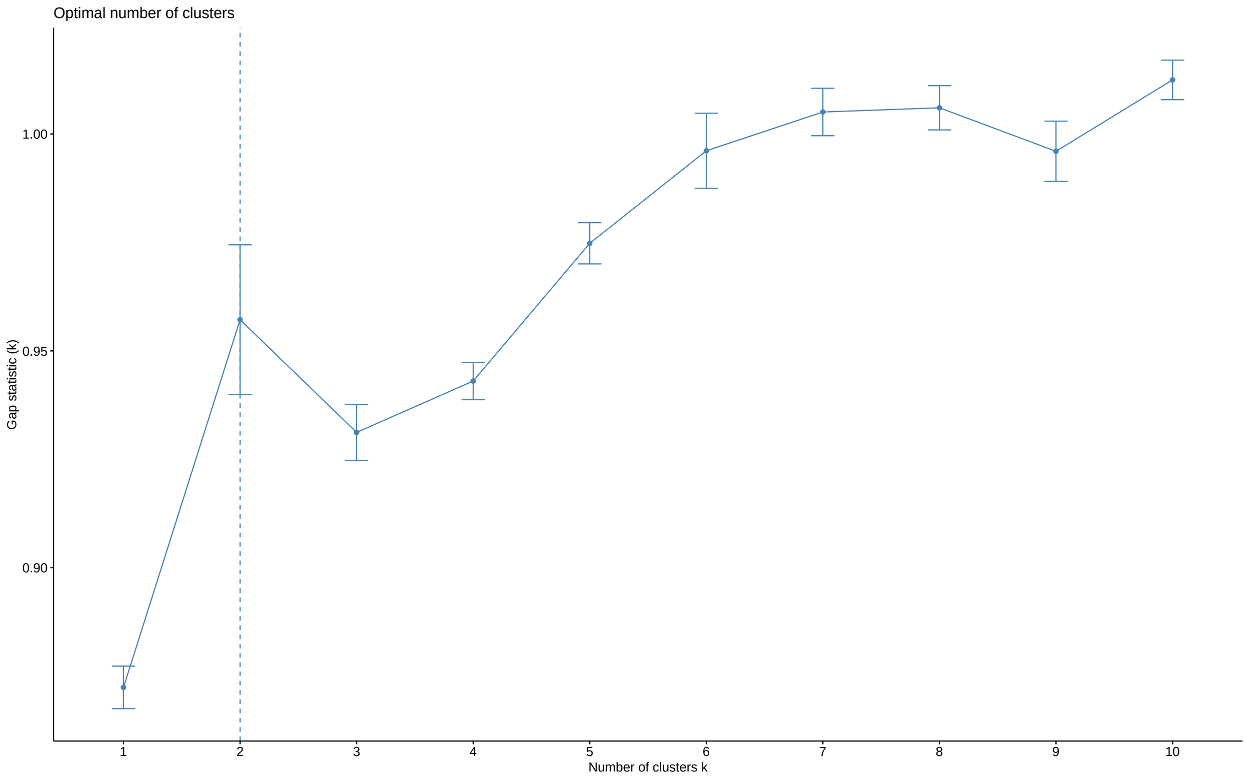


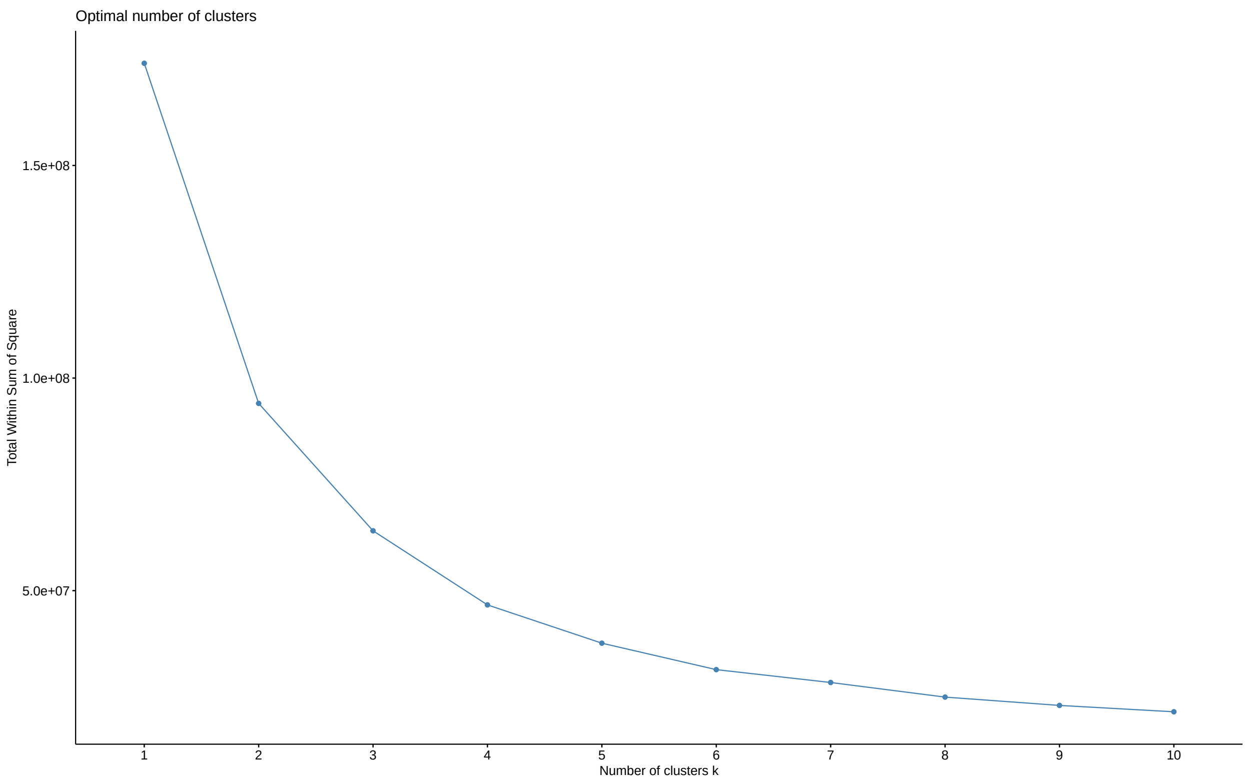
Hybrid Model:

The last model we explored is one that combines clustering with linear regression. The essential idea is we can use clustering to determine whether there are are particular player groups that can be better modeled separately using linear regression. This method can be thought of as a decision tree with multiple branches resulting in clusters as the final nodes, however the simpler nature of the 'tree' may be favorable to reduce the possibility of over-fitting while maintaining increased flexibility relative to linear regression. We used the training data set to cluster the data and get the approximate cluster centers with kmeans. This allows us to find the cluster of a future test observation and predict its value using the model trained on the other members of the same cluster from the training set. In our case the relevant hyper-parameter to start the ensemble modeling process off is the number of clusters. To determine which value would be most appropriate we looked at three different types of plots to make our final assessment.

The first of the three plots we used is the silhouette plot which indicates that the likely optimal value using that metric is 3. The second is the gap statistic calculated using a similar form of bootstrapping with 10 iterations for each number of clusters. The outcome highlights a different optimal value of 2 clusters. The third plot is the scree plot showing the decrease in the within sum of squares with the increasing number of clusters. In the end, we decided to choose three clusters as the KDE plot for Salary (refer to EDA section) has two primary 'central' modes with multiple smaller modes in the right tail that may be better captured with a third cluster.







Now that the number of clusters is chosen, we partition our data appropriately by cluster and perform cross validation as it was done in the first linear regression model for each individual cluster; however, we also removed non-significant features since we inted for each cluster's linear model to differ. Below we can see that some clusters have a greater adjusted R-squared value than others though it appears to be comparable

Table 2: Final Cluster Centers for Hybrid Model using k = 3

cluster	BPM	AST	WS/48	VORP	DRB	G	STL	GS	BLK	FT%	Season	3P%	Age
1	1.11	150.26	0.13	1.75	386.56	72.40	64.26	58.26	63.65	0.75	2014.14	0.28	25.56
2	-1.84	74.38	0.07	0.19	111.05	48.81	30.08	14.60	16.19	0.74	2015.92	0.30	25.46
3	2.03	398.22	0.12	2.42	241.43	70.74	87.50	56.78	23.75	0.80	2013.78	0.34	25.89

to that of the linear regression model as a whole. In fact, it may even be slightly worse since the cluster with the lowest adjusted R-squared also had the most observations. Lastly and as expected, the majority of the feature set were shared across all three clusters, but there were different significant predictor variables for each cluster. Lastly, we computed the RMSE, R-squared and MAE along with each of their associated standard deviations by pooling the corresponding values for each cluster to calculate the overall metric value using the number of observations for the associated weights.

```
#### Cluster  1 :
--------------------------------------------------------------
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
      Min        1Q    Median        3Q       Max
-15623638  -3938463   -411122   3162136  21003423

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -608460479   98099473  -6.202 8.65e-10 ***
BPM             486916     105913   4.597 4.93e-06 ***
AST              14161       3261   4.343 1.57e-05 ***
DRB              10032       1923   5.217 2.28e-07 ***
G              -223775      24079  -9.293  < 2e-16 ***
GS               46905       9491   4.942 9.31e-07 ***
Season          301283      48593   6.200 8.77e-10 ***
Age             692366      53255  13.001  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5594000 on 855 degrees of freedom
Multiple R-squared:  0.4528,    Adjusted R-squared:  0.4483
F-statistic: 101.1 on 7 and 855 DF,  p-value: < 2.2e-16


--------------------------------------------------------------


#### Cluster  2 :
--------------------------------------------------------------
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
      Min        1Q    Median        3Q       Max
-10021125  -2479625   -631178   1546506  31031267

Coefficients:
```

```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -303667541   48033684  -6.322 3.05e-10 ***
BPM               536517      69741   7.693 2.05e-14 ***
AST                13477       1988   6.779 1.50e-11 ***
‘\\‘WS/48\\‘‘  -18333558    2879437  -6.367 2.28e-10 ***
DRB                23389       2047  11.424  < 2e-16 ***
G                 -64154       6836  -9.384  < 2e-16 ***
STL               -22729       6408  -3.547 0.000397 ***
GS                 48856       5560   8.788  < 2e-16 ***
‘\\‘FT%\\‘‘      1369234     668276   2.049 0.040575 *
Season            149029      23886   6.239 5.15e-10 ***
Age               310837      20664  15.042  < 2e-16 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 4055000 on 2511 degrees of freedom
Multiple R-squared:  0.308, Adjusted R-squared:  0.3052
F-statistic: 111.8 on 10 and 2511 DF,  p-value: < 2.2e-16


--------------------------------------------------------------


#### Cluster  3 :
--------------------------------------------------------------
Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
      Min        1Q    Median        3Q       Max
-18457267  -3224418   -368539   3086881  20269515

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)   -1.041e+09  1.105e+08  -9.425  < 2e-16 ***
BPM            1.035e+06  2.397e+05   4.316 1.85e-05 ***
‘\\‘WS/48\\‘‘ -2.858e+07  1.098e+07  -2.603  0.00946 **
DRB            2.020e+04  3.583e+03   5.639 2.59e-08 ***
G             -2.127e+05  2.690e+04  -7.908 1.18e-14 ***
GS             7.864e+04  1.110e+04   7.087 3.70e-12 ***
‘\\‘FT%\\‘‘    1.680e+07  3.326e+06   5.050 5.81e-07 ***
Season         5.072e+05  5.506e+04   9.213  < 2e-16 ***
‘\\‘3P%\\‘‘   -1.186e+07  4.211e+06  -2.817  0.00500 **
Age            1.075e+06  6.588e+04  16.318  < 2e-16 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 5518000 on 628 degrees of freedom
Multiple R-squared:  0.6419,    Adjusted R-squared:  0.6368
F-statistic: 125.1 on 9 and 628 DF,  p-value: < 2.2e-16


--------------------------------------------------------------
```

Model Selection

Since there are multiple model types and possibly numerous models for each type explored, we have compiled a list of all the associated results from cross-validation into a single table for easy viewing. Nonetheless, in the results table (see appendix) note that the R-squared value is used to choose the best model should we intend to compare across the different Salary response types, i.e. raw and log transformed. The reason for this decision is because the RMSE metric does not appropriately translate for different response units. Given such, all models were trained using the R-squared metric in anticipation of future comparison for model selection. In the results table it is clear to see that the best performing model based on the R-squared metric is the RanFor with the mtry hyper-parameter tuned to 11 and an R-squared value of 0.604.

Testing the Final Model:

Using the final model on the test data as the most recent season's data (2021), we got test results which appear to be consistent with our literature review. Unexpectedly, the R-squared value was superior to those of any of the previous models from cross-validation as well as its own though the RMSE was consistent with our expectations.

Closing Remarks:

Looking at many of the models we explored as well the associated cross-validation R-squared values, we an see that the average proportion of variance explained is approximately 0.554. Such a finding would seem to indicate performance is roughly half the story when it comes to explaining NBA player's salaries. As a consequence, such models can certainly be useful but there is additional information that will be required to truly model the associated Salaries of an NBA player. Such information can range from data containing the player popularity and publicity to player health history which may affect ticket sales revenue and performance during certain seasons respectively. We were also quite pleased to see that the hybrid model approach was as effective as one of the random forest models in terms of R-squared value and superior to both of the linear and log-linear model. Be that as it may, we still think that the log-linear model could be further improved through the use of weights, but failed to find a weighting scheme which offered significant improvement during the project time-frame. If such a weighting scheme was found, it may be safe to say that the descriptive capability of such a model would offer favorably higher interpretability than our final model. There is also the opportunity to use each of the model types in a more rigorous exploration of hybrid models where the final model selected could contain multiple clusters with a different model type per each cluster.

References

Abdurahmanmaarouf. “NBA Salary Prediction.” Kaggle, Kaggle, 13 Oct. 2020, https://www.kaggle.com/code/abdurahmanmaarouf/nba-salary-prediction/notebook.

Gaines, Cork. “The NBA Is the Highest-Paying Sports League in the World.” Business Insider, Business Insider, 20 May 2015, https://www.businessinsider.com/sports-leagues-top-salaries-2015-5.

Papadaki, Ioanna & Tsagris, Michail. (2022). Are NBA Players’ Salaries in Accordance with Their Performance on Court?. 10.1007/978-3-030-85254-2_25.

Appendix

Salary: A fixed regular payment made to NBA players based on their contracts.

BPM: Box Plus/Minus (available since the 1973-74 season in the NBA); a box score estimate of the points per 100 possessions that a player contributed above a league-average player, translated to an average team.

AST: Assists

WS/48: Win Shares Per 48 Minutes (available since the 1951-52 season in the NBA); an estimate of the number of wins contributed by the player per 48 minutes (league average is approximately 0.100).

VORP: Value Over Replacement Player (available since the 1973-74 season in the NBA); a box score estimate of the points per 100 TEAM possessions that a player contributed above a replacement-level (-2.0) player, translated to an average team and prorated to an 82-game season. Multiply by 2.70 to convert to wins over replacement.

DRB: Defensive Rebounds (available since the 1973-74 season in the NBA)

G: Games

STL: Steals (available since the 1973-74 season in the NBA)

GS: Games Started (available since the 1982 season)

BLK: Blocks (available since the 1973-74 season in the NBA)

FT%: Free Throw Percentage

Season: The NBA year that the stats were recorded for a particular player.

3P%: 3-Point Field Goal Percentage (available since the 1979-80 season in the NBA)

Age: How old a particular player is during the NBA season

NOTE: The feature descriptions in the appendix section are as listed on https://www.basketball-reference.com/about/glossary.html

Table 3: Results of Cross-Validation for all Models

Model	hyper-parameter	value	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
Lin	intercept	1	5008962.339	0.479	3688123.368	223326.955	0.033	97513.569
LogLin	intercept	1	0.429	0.441	0.327	0.009	0.031	0.005
RanFor	mtry	2	4499872.494	0.591	3034661.557	286770.462	0.032	141736.659
RanFor	mtry	5	4391806.340	0.603	2925076.415	262565.763	0.034	112443.219
RanFor	mtry	8	4379606.861	0.604	2904240.278	258805.405	0.036	109383.548
RanFor	mtry	11	4376132.434	0.604	2900772.593	243891.472	0.036	98420.723
LogRanFor	mtry	2	0.387	0.545	0.290	0.010	0.040	0.008
LogRanFor	mtry	5	0.384	0.552	0.284	0.012	0.040	0.010
LogRanFor	mtry	8	0.384	0.553	0.284	0.014	0.042	0.010
LogRanFor	mtry	11	0.385	0.552	0.283	0.014	0.042	0.010
KNN	k.neighbors	1	5828355.501	0.416	3509251.720	259375.413	0.053	143617.396
KNN	k.neighbors	2	5140219.597	0.484	3282331.039	318535.612	0.031	191587.813
KNN	k.neighbors	3	4876552.562	0.519	3176948.614	292429.164	0.036	187595.628
KNN	k.neighbors	4	4707207.281	0.544	3093106.765	263006.975	0.031	176677.496
KNN	k.neighbors	5	4681166.999	0.548	3087287.049	255068.321	0.030	162494.046
KNN	k.neighbors	6	4607781.528	0.560	3051601.678	251428.049	0.033	154801.244
KNN	k.neighbors	7	4571668.815	0.567	3050628.152	233024.297	0.036	138697.666
KNN	k.neighbors	8	4543285.835	0.572	3049190.768	207286.786	0.032	123622.748
KNN	k.neighbors	9	4539315.843	0.573	3051614.660	210266.658	0.031	118096.907
KNN	k.neighbors	10	4554947.754	0.570	3075517.059	188500.869	0.031	107638.991
KNN	k.neighbors	11	4558221.488	0.570	3081440.420	183062.722	0.034	101111.387
KNN	k.neighbors	12	4557571.740	0.571	3081139.803	196107.980	0.033	108807.827
KNN	k.neighbors	13	4544839.532	0.574	3082588.858	192006.195	0.035	110905.229
KNN	k.neighbors	14	4542991.577	0.575	3083801.864	194335.805	0.032	117393.968
KNN	k.neighbors	15	4532555.585	0.577	3077591.568	201267.230	0.032	120856.206
KNN	k.neighbors	16	4532307.791	0.578	3073854.724	213402.842	0.032	122582.807
KNN	k.neighbors	17	4528712.888	0.579	3074446.800	210219.387	0.031	115594.838
KNN	k.neighbors	18	4540683.107	0.577	3082567.783	216442.576	0.030	119259.571
KNN	k.neighbors	19	4552869.096	0.575	3092487.857	221808.705	0.029	116514.054
KNN	k.neighbors	20	4552726.790	0.575	3095593.163	217658.969	0.028	116470.195
Hybrid	k.clusters	3	4678923.496	0.546	3386767.126	396461.623	0.060	279846.854

Table 4: Test Results of Final Model

Model	Hyper-Paramter	Value	RMSE	R-Squared	Adjusted R-Squared
RanFor	mtry	11	5232483	0.672	0.664