FIFA player overalls 2022 (CYO Project)

Isaac Cyrman Casafont

3/30/2021

```
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: knitr
```

```
if(!require(pdftools)) install.packages("pdftools", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: pdftools
```

```
## Warning: package 'pdftools' was built under R version 4.0.4
```

```
## Using poppler version 0.73.0
```

```
if(!require(latexpdf)) install.packages("latexpdf", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: latexpdf
```

```
library(knitr)
library(pdftools)
library(latexpdf)
```

```
#Install all the required packages:
if(!require(dslabs)) install.packages("dslabs", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dslabs
```

```
## Warning: package 'dslabs' was built under R version 4.0.4
```

```
if(!require(broom)) install.packages("broom", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: broom
```

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(ggpubr)) install.packages("ggpubr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: ggpubr
```

```
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: corrplot

## Warning: package 'corrplot' was built under R version 4.0.4

## corrplot 0.84 loaded
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
#load all the required packages:
library(dslabs)
library(broom)
library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(ggpubr)
library(corrplot)
library(caret)
```

Introduction:

The video game **FIFA** is one of the most played and recognized games around the world, currently this game has 30.4 million active players and much of this support for the game is due to the fact that many of its players are fans of one of the biggest sports in the world, which is soccer. In this game you can play soccer matches with any team, in any stadium, in any competition and most importantly with anyone who wants it. In addition to that, this game has the option of being able to start a career as a technical director or as a player of a team. Also, you can play online with friends or others players, with a predetermined or created team, etc.

In this work, we will focus on being technical directors and we will manage the database with all existing players and their information in the FIFA 2021 game. Much of the information that will be used is the following: **Personal, club and player statistics data**. Within this information, we can find data such as: **Player overall, height, weight, club, team position, statistics, ID, date of birth, market value, wages, among many others.**. Before we focus on the purpose of the project, we will make a brief description of the data. The initial data set has **18944 obs. and 106 variables**, where we will only focus on the players who play on the field (**NON-GOALKEPEERS IN THE DATA SET**) and with this initial data set, elaborate three data sets with specific selected information, in consequence, our final data sets has **PlayerInfo: (16861 obs. of 11 variables)**, **PlayerClubInfo: (15959 obs. of 11 variables)** and **PlayerStats: (16861 obs. of 43 variables)**.

The purpose of this project is to create predictions for the player overalls in order to know and anticipate their overalls for the next FIFA game, which will be the "FIFA 2022". These predictions will be build, based on the current statistics and personal information of the player such as: age, international reputation, weak foot, etc,. Continuing with the purpose of the project, the following processes can be found in order to achieve the proposed objective: **DOWNLOAD, UNZIP AND READ DATA PROCESS, DATA CLEANING, FILTERING AND SELECTING PROCESS, DATA ANALYSIS PROCESS and MODELING PROCESS**. Finally, as a general objective in this project, it is to demonstrate our abilities when **creating, analyzing, shaping, visualizing, presenting** and above all, being precise when **constructing and evaluating** the data for predictive models.

IMPORTANT: In this report, code will be hidden only for the data analysis process, this is because the report had too many pages. Also, some results were hidden.

```r
########################################
# DOWNLOAD, UNZIP AND READ DATA PROCESS #
########################################

temp <- tempfile()
download.file("https://github.com/Isaacj59/CYO-project/raw/main/FIFA_21.zip",temp)
FIFA2021 <- read_csv(unz(temp, "players_21.csv"))
```

```
##
## -- Column specification --------------------------------------------------------
```

```
## cols(
##   .default = col_double(),
##   player_url = col_character(),
##   short_name = col_character(),
##   long_name = col_character(),
##   dob = col_date(format = ""),
##   nationality = col_character(),
##   club_name = col_character(),
##   league_name = col_character(),
##   player_positions = col_character(),
##   preferred_foot = col_character(),
##   work_rate = col_character(),
##   body_type = col_character(),
##   real_face = col_character(),
##   player_tags = col_character(),
##   team_position = col_character(),
##   loaned_from = col_character(),
##   joined = col_date(format = ""),
##   nation_position = col_character(),
##   player_traits = col_character(),
##   defending_marking = col_logical(),
##   ls = col_character()
##   # ... with 25 more columns
## )
## i Use `spec()` for the full column specifications.
```

```
unlink(temp)
```

DATA CLEANING, FILTERING AND SELECTING PROCESS:

In this process, we created three data sets (**PlayerInfo, PlayerCLubInfo and PlayerStats**) with specific selected columns for future data analysis and modeling work. Also, for this process, we filtered the data by player positions in order to only have field players in the data sets and cleaned them omitting and replacing NA values.

```
#################################
# DATA CLEANING, FILTERING AND  #
#       SELECTING PROCESS       #
#################################
```

PlayerInfo data set: This data set was create to save the information of the player.

```
PlayerInfo<- FIFA2021 %>% filter(!player_positions=="GK") %>% select(sofifa_id, short_name, age, dob, na

NA_Values_PlayerInfo<-colSums(is.na(PlayerInfo==TRUE))
NA_Values_PlayerInfo
```

```
##        sofifa_id       short_name              age              dob
##                0                0                0                0
##      nationality       height_cm       weight_kg player_positions
##                0                0                0                0
##        wage_eur       value_eur          overall
##                0                0                0
```

PlayerCLubInfo data set: This data set was create to save the information of the player and club.

```
PlayerCLubInfo<- FIFA2021 %>% filter(!player_positions=="GK") %>% select(sofifa_id,short_name, club_name

PlayerClubInfo<-PlayerCLubInfo %>% mutate(club_name = as.character(PlayerCLubInfo$club_name),
                          long_name = as.character(PlayerCLubInfo$short_name),
                          league_name = as.character(PlayerCLubInfo$league_name),
                          sofifa_id= as.numeric(PlayerCLubInfo$sofifa_id),
                          league_rank= as.numeric(PlayerCLubInfo$league_rank),
                          value_eur= as.numeric(PlayerCLubInfo$value_eur),
                          wage_eur= as.numeric(PlayerCLubInfo$wage_eur),
                          release_clause_eur= as.numeric(PlayerCLubInfo$release_clause_eur))

PlayerCLubInfo<-na.omit(PlayerCLubInfo)

NA_Values_PlayerClubInfo<-colSums(is.na(PlayerCLubInfo==TRUE))
NA_Values_PlayerClubInfo
```

```
##          sofifa_id         short_name          club_name        league_name
##                  0                  0                  0                  0
##        league_rank      team_position          value_eur          wage_eur
##                  0                  0                  0                  0
## release_clause_eur   player_positions            overall
##                  0                  0                  0
```

PlayerStats data set: This data set was create to save the general information of the player and player stats.

```
PlayerStats<- FIFA2021 %>% filter(!player_positions=="GK") %>%  select(sofifa_id,short_name,potential,w
                                  defending, physic, attacking_crossing, attacking_finishing, attacking
                                  movement_sprint_speed, movement_agility, movement_reactions, movement
                                  mentality_aggression, mentality_interceptions, mentality_positioning,
                                  defending_sliding_tackle,skill_curve,skill_moves,skill_dribbling,skil

PlayerStats<-replace(PlayerStats, is.na(PlayerStats==TRUE), 0)

NA_Values_PlayerStats<-colSums(is.na(PlayerStats==TRUE))
NA_Values_PlayerStats
```

```
##                  sofifa_id                 short_name
##                          0                          0
##                  potential                  work_rate
##                          0                          0
##   international_reputation                    overall
##                          0                          0
##                        age                  weak_foot
##                          0                          0
##                skill_moves                       pace
##                          0                          0
##                   shooting                    passing
##                          0                          0
##                   dribbling                  defending
##                          0                          0
```

```
##                     physic         attacking_crossing
##                          0                          0
##        attacking_finishing attacking_heading_accuracy
##                          0                          0
##    attacking_short_passing          attacking_volleys
##                          0                          0
##      movement_acceleration      movement_sprint_speed
##                          0                          0
##           movement_agility         movement_reactions
##                          0                          0
##           movement_balance           power_shot_power
##                          0                          0
##              power_jumping              power_stamina
##                          0                          0
##             power_strength           power_long_shots
##                          0                          0
##       mentality_aggression    mentality_interceptions
##                          0                          0
##      mentality_positioning        mentality_composure
##                          0                          0
##           mentality_vision        mentality_penalties
##                          0                          0
##  defending_standing_tackle   defending_sliding_tackle
##                          0                          0
##                skill_curve            skill_dribbling
##                          0                          0
##          skill_fk_accuracy         skill_long_passing
##                          0                          0
##         skill_ball_control
##                          0
```

DATA ANALYSIS PROCESS:

In this process, we applied our management and visualization skills in the previously cleaned data sets that were created in the **DATA CLEANING, FILTERING and SELECTING PROCESS**. In the same, we elaborated bar plots and tables in order to achieve a deeper comprehension of the data. (This process can be considered crucial for the modeling process).

```
#########################
# DATA ANALYSIS PROCESS #
#########################
```

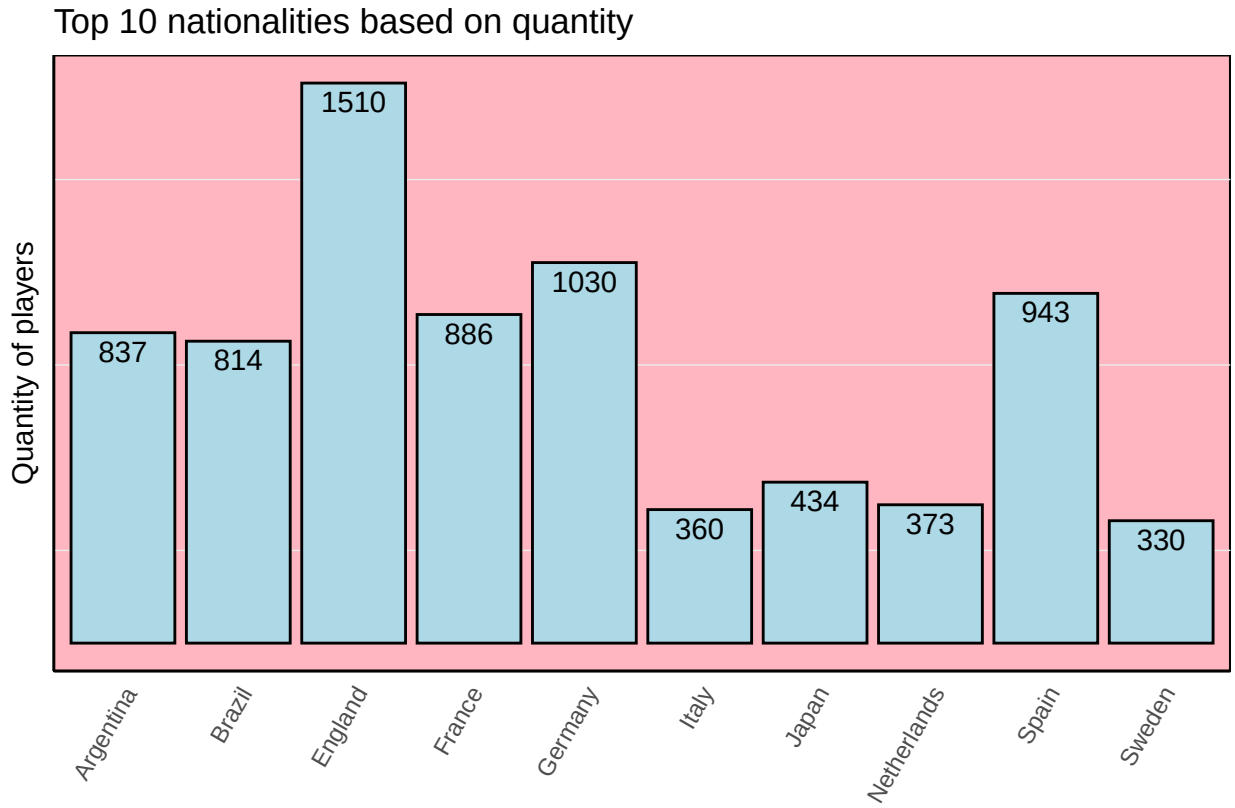
Top ten nationalities based on quantity:

As you can notice, in the following table and plot for the top ten nationalities based on quantity. The nationality with more players in the data set is **England** with **1510** professional soccer players.

```
Quantity_Players<- PlayerInfo %>% summarise(PlayerId=sofifa_id) %>% count()

Quantity_Nationality<- PlayerInfo %>% group_by(nationality) %>% count()
Nationalities_list<-Quantity_Nationality[1]
top10_nationality_quantity<-Quantity_Nationality[order(-Quantity_Nationality$n),][1:10,]

(top10_nationality_quantity) %>% knitr::kable()
```

nationality	n
England	1510
Germany	1026
Spain	943
France	886
Argentina	837
Brazil	814
Japan	434
Netherlands	373
Italy	360
Sweden	330

```
ggplot(data=top10_nationality_quantity, aes(x=nationality, y=n)) +
  geom_bar(stat="identity", fill="light blue",color="black") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.background = element_rect(fill = "light pink",
        colour = "black", size = 0.5, linetype = "solid"), axis.line = element_line(colour = "black")) +
        geom_text(aes(label = signif(n, digits = 3)), nudge_y = -50,nudge_x = -0) +
        theme(axis.text.x = element_text(angle = 60, hjust = 1),
        axis.text.y = element_blank()) +
        ggtitle("Top 10 nationalities based on quantity") + xlab("") + ylab("Quantity of players")
```



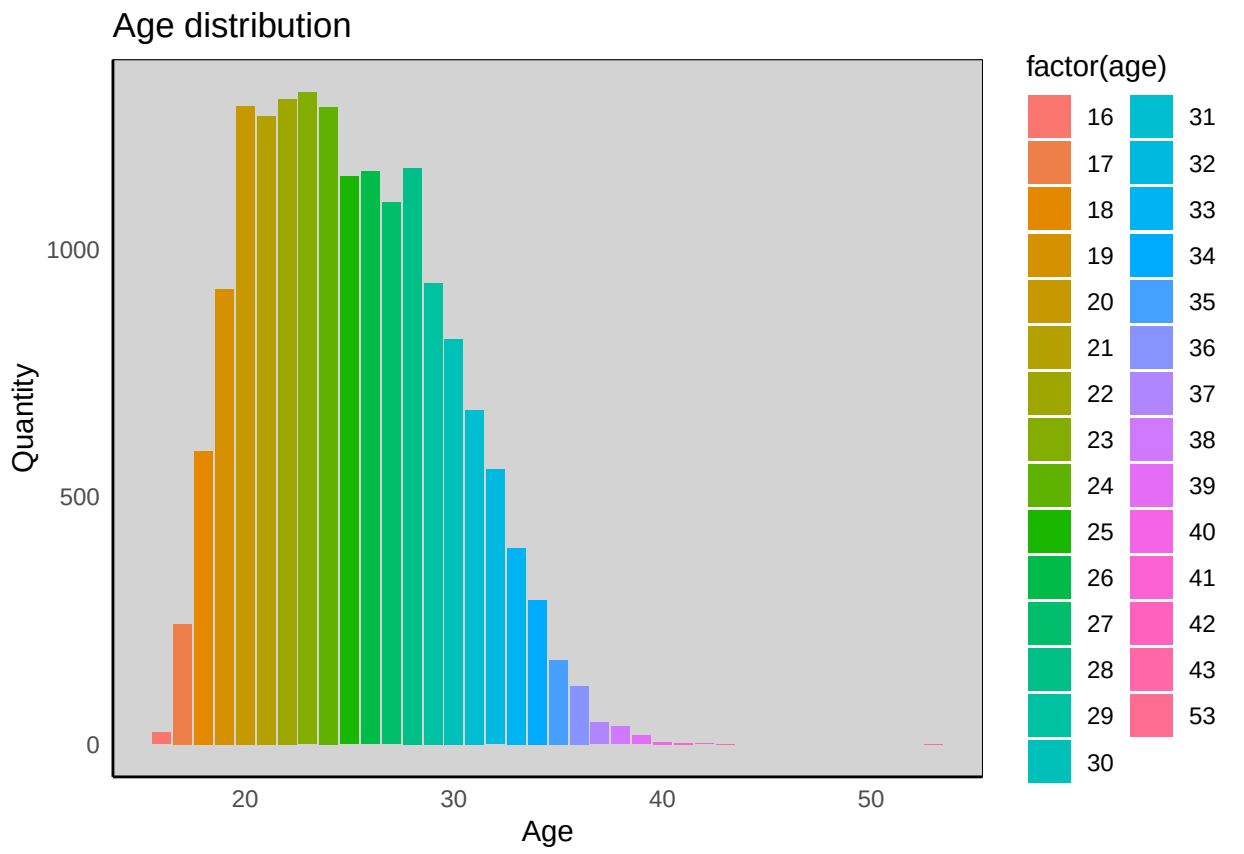
Age distribution:

In the following table and plot below, we can see that the more recurrent players age in the data set is **23 years** with **1316** appearances and the less recurrent is share between **43 years** and **53 years** with **1** appearances.

```
Quantity_Age<- PlayerInfo %>% group_by(age) %>% count()
Quantity_Age %>% knitr::kable()
```

age	n
16	24
17	243
18	592
19	919
20	1288
21	1268
22	1302
23	1316
24	1286
25	1146
26	1156
27	1094
28	1163
29	931
30	818
31	675
32	555
33	396
34	291
35	170
36	118
37	44
38	36
39	18
40	5
41	3
42	2
43	1
53	1

```
ggplot(Quantity_Age, aes(age, n)) +
  geom_col(aes(fill = factor(age))) +theme_minimal() +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
  panel.background = element_rect(fill = "light gray",
  colour = "black",size = 0.2, linetype = "solid"), axis.line = element_line(colour = "black")) +
  ggtitle("Age distribution") + xlab("Age") + ylab("Quantity")
```



Top ten older and youngest players:

If we take a look into the tables below, we can found the top ten older and youngest players in the data set. In the top ten older players table, the first position its for **K. Miura** with an age of **53 years**, meanwhile, for the top ten youngest players table, there isn't any position because of all them count with an age of **16 years**.

```
Ages_all_players<-PlayerInfo %>% group_by(age) %>% summarise(Name=short_name)
```

```
## `summarise()` has grouped output by 'age'. You can override using the `.groups` argument.
```

```
Top10_Players_Older<-Ages_all_players[order(-Ages_all_players$age),][1:10,]
(Top10_Players_Older) %>% knitr::kable()
```

age	Name
53	K. Miura
43	H. Sulaimani
42	Hilton
42	S. Nakamura
41	Lee Dong Gook
41	D. Bulman
41	K. Ellison
40	J. Cáceres
40	Nino
40	S. Aquino

```
Top10_Player_Youngest<-Ages_all_players[order(Ages_all_players$age),][1:10,]
(Top10_Player_Youngest) %>% knitr::kable()
```

age	Name
16	R. Cherki
16	A. Karabec
16	U. Bertelli
16	D. Hoyo-Kowalski
16	L. Gourna-Douath
16	O. Babuscu
16	M. Tanlongo
16	O. Beyaz
16	W. Faghir
16	A. Descotte

Top ten players with best wage:

In the following table and plot, we can appreciate the top ten players with best wage in the data set. The player with the best wage is the FC Barcelona player **L. Messi** (best known as **Lionel Messi**) with **€560,000** in a month of work.

```
Players_wage_value<- PlayerInfo %>% select(short_name,wage_eur,value_eur)
Wage_Players<- Players_wage_value %>% group_by(wage_eur) %>% summarize(Name=short_name)
```

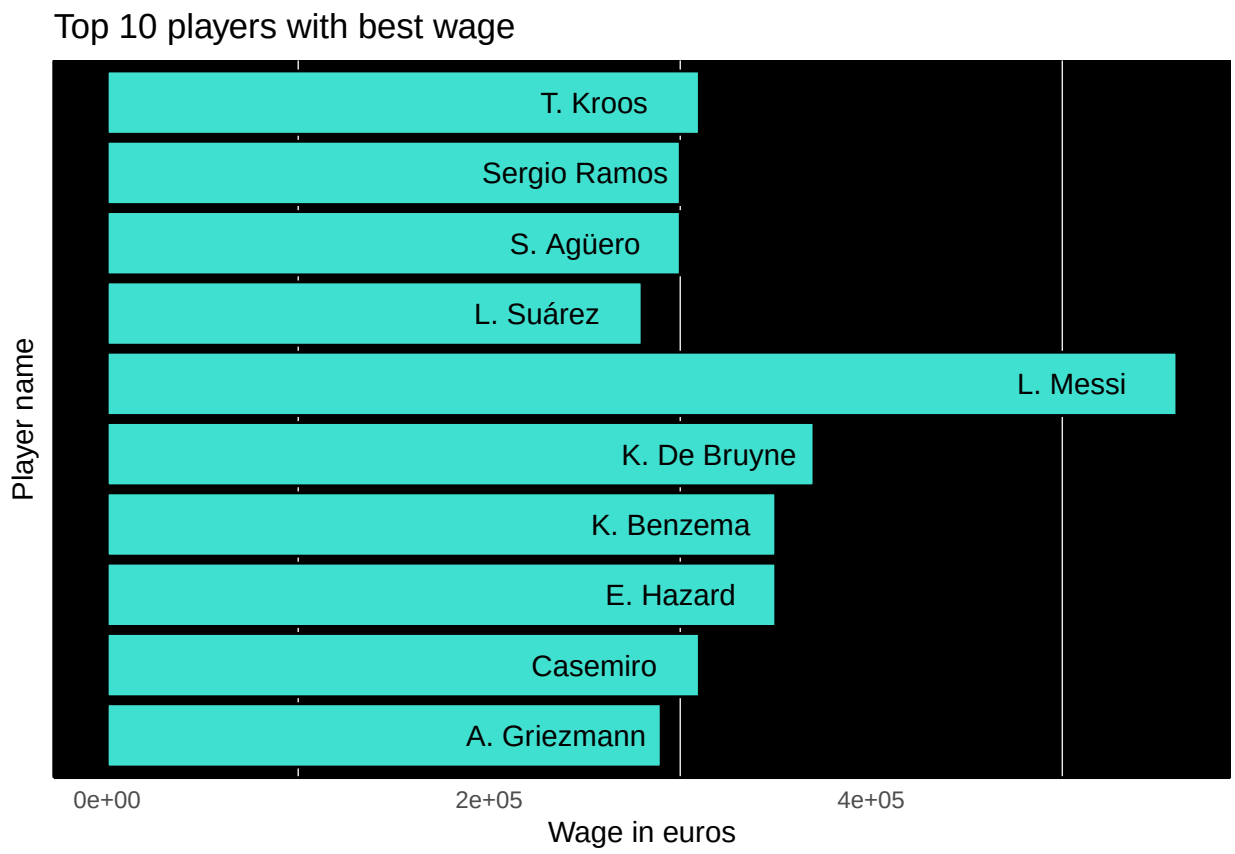
```
## `summarise()` has grouped output by 'wage_eur'. You can override using the `.groups` argument.
```

```
Top10_best_paid<-Wage_Players[order(-Wage_Players$wage_eur),][1:10,]
(Top10_best_paid) %>% knitr::kable()
```

wage_eur	Name
560000	L. Messi
370000	K. De Bruyne
350000	K. Benzema
350000	E. Hazard
310000	Casemiro
310000	T. Kroos
300000	S. Agüero
300000	Sergio Ramos
290000	A. Griezmann
280000	L. Suárez

```
ggplot(Top10_best_paid, aes(x=wage_eur, y=Name)) +
  geom_bar(stat="identity", fill="turquoise",colour="black") +
  geom_text(aes(label = Name,wage_eur), nudge_y = 0,nudge_x = -55000) +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
  panel.background = element_rect(fill = "black",
  colour = "black", size = 0.5, linetype = "solid"),
```

```
  axis.line = element_line(colour = "black")) +
  theme(axis.text.y = element_blank()) +
  ggtitle("Top 10 players with best wage") + xlab("Wage in euros") + ylab("Player name")
```



Top ten more valuable players:

In the following table and plot, we can appreciate the top ten more valuable players in the data set. The player with the best wage is the Paris Saint German player **K. Mbappe** with a high value of **€105,500,000**.

```
Value_Players<- Players_wage_value %>% group_by(value_eur) %>% summarize(Name=short_name)
```

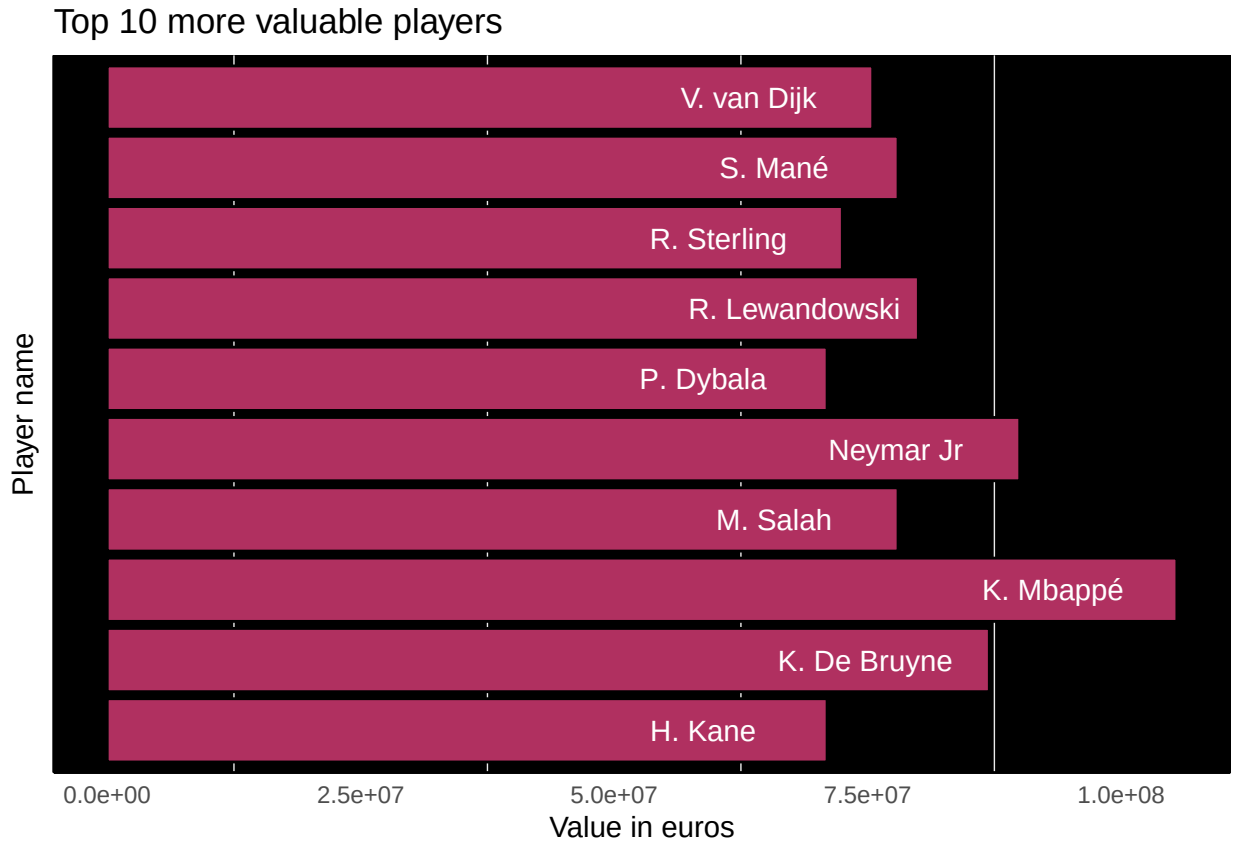
```
## `summarise()` has grouped output by 'value_eur'. You can override using the `.groups` argument.
```

```
Top10_more_valuable<-Value_Players[order(-Value_Players$value_eur),][1:10,]
Top10_more_valuable %>% knitr::kable()
```

value_eur	Name
105500000	K. Mbappé
90000000	Neymar Jr
87000000	K. De Bruyne
80000000	R. Lewandowski
78000000	S. Mané
78000000	M. Salah
75500000	V. van Dijk

value_eur	Name
72500000	R. Sterling
71000000	H. Kane
71000000	P. Dybala

```
ggplot(Top10_more_valuable, aes(x=value_eur, y=Name)) +
  geom_bar(stat="identity", fill="maroon",colour="black") +
  geom_text(aes(label = Name), nudge_y = 0,nudge_x = -12220000,color="white") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.background = element_rect(fill = "black",
        colour = "black", size = 0.5, linetype = "solid"),
        axis.line = element_line(colour = "black")) +
        theme(axis.text.y = element_blank()) +
        ggtitle("Top 10 more valuable players") +
        xlab("Value in euros") + ylab("Player name")
```



Top ten highest and heaviest players:

If we look at the tables below, we can notice that the highest player in the data set is **A. Ba** with **203 cm** and the heaviest player in the data set is **A. Akinfenwa** with **110 kg**.

```
Height_Players<-PlayerInfo %>% group_by(height_cm) %>% summarise(Name=short_name)
```

```
## `summarise()` has grouped output by 'height_cm'. You can override using the `.groups` argument.
```

```
Top10_Tallest<-Height_Players[order(-Height_Players$height_cm),][1:10,]
Top10_Tallest %>% knitr::kable()
```

height_cm	Name
203	A. Ba
202	S. Maierhofer
201	P. Onuachu
201	H. Veerman
201	A. Vukotic
201	A. Sjöberg
201	S. Makienok
200	T. Chorý
200	K. Sidibé
200	S. Kalajdžic

```
Weight_Players<-PlayerInfo %>% group_by(weight_kg) %>% summarise(Name=short_name)
```

```
## `summarise()` has grouped output by 'weight_kg'. You can override using the `.groups` argument.
```

```
Top10_Heavier<-Weight_Players[order(-Weight_Players$weight_kg),][1:10,]
Top10_Heavier %>% knitr::kable()
```

weight_kg	Name
110	A. Akinfenwa
104	O. Oularé
101	W. Morgan
101	T. Chorý
100	H. Maguire
100	D. Dike
100	R. Greenidge
100	Ricardo Santos
99	T. Petrášek
98	J. Beauguel

Linear regression model for players height and weight:

In the result and plot below, we can appreciate a high correlation between the players height and weight of **0.7354**, which means that this two variables are related and depend on each other.

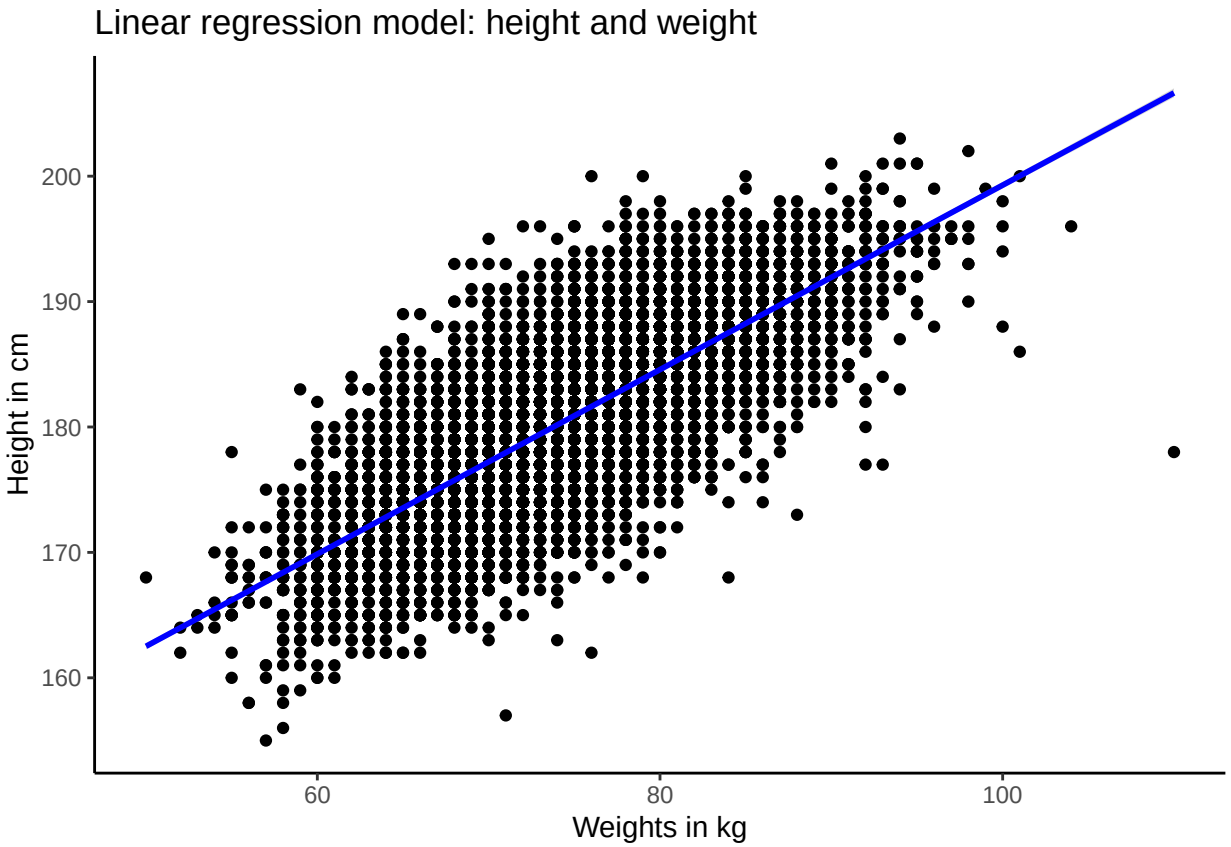
```
CorWH<-PlayerInfo %>% select(weight_kg,height_cm)

lm(height_cm~weight_kg,data=PlayerInfo)
```

```
##
## Call:
## lm(formula = height_cm ~ weight_kg, data = PlayerInfo)
##
## Coefficients:
```

```
## (Intercept)    weight_kg
##    125.7446       0.7354
```

```
ggplot(CorWH, aes(x = weight_kg, y = height_cm)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))+
        ggtitle("Linear regression model: height and weight") +
        xlab("Weights in kg") + ylab("Height in cm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Top ten clubs with the highest salaries:

As we can see in the top ten clubs with the highest salaries table and plot. The professional soccer club **Real Madrid** have the best paying in comparison to the other clubs in the data set with a mean of **€166,259.26** in month.

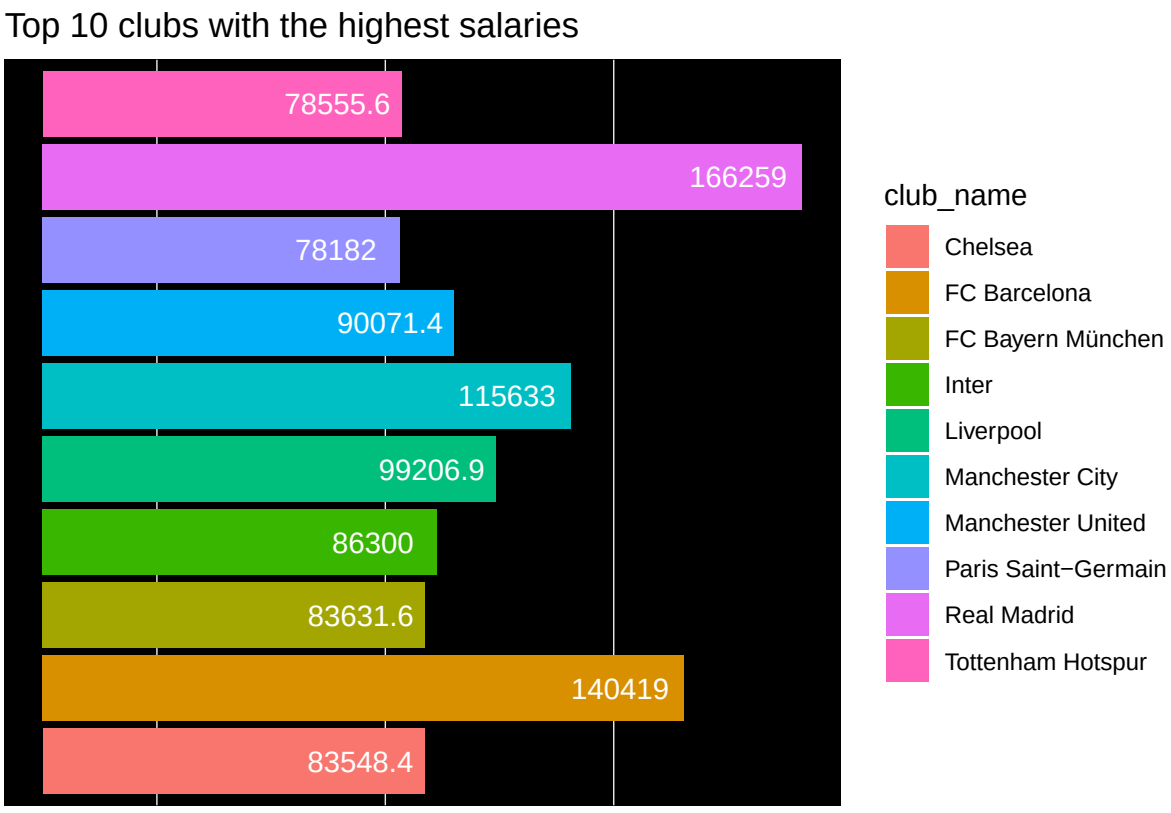
```
Mean_Top10_club_wage<- PlayerCLubInfo %>% group_by(club_name) %>% summarise(Mean=mean(wage_eur)) %>% to
```

```
## Selecting by Mean
```

```
(Mean_Top10_club_wage) %>% knitr::kable()
```

club_name	Mean
Chelsea	83548.39
FC Barcelona	140419.35
FC Bayern München	83631.58
Inter	86300.00
Liverpool	99206.90
Manchester City	115633.33
Manchester United	90071.43
Paris Saint-Germain	78182.00
Real Madrid	166259.26
Tottenham Hotspur	78555.56

```
ggplot(Mean_Top10_club_wage, aes(x=Mean, y=club_name)) +
  geom_col(aes(fill = club_name)) + geom_text(aes(label = signif(Mean)), nudge_y = 0,nudge_x = -14000,c
  theme_minimal() + theme(panel.grid.major = element_blank(),
  panel.background = element_rect(fill = "black",
  colour = "black", size = 0.5, linetype = "solid")) +
  theme(axis.text.x = element_blank(),
  axis.text.y = element_blank()) +
  ggtitle("Top 10 clubs with the highest salaries") +
  xlab("Club wague in euros based on players wages") + ylab("")
```



Top ten more valuable clubs:

As we can see in the top ten more valuable clubs table and plot. The professional soccer club that can be consider as the "more valuable club" is the **FC Bayern München** with a mean value of **€29,694,737**.

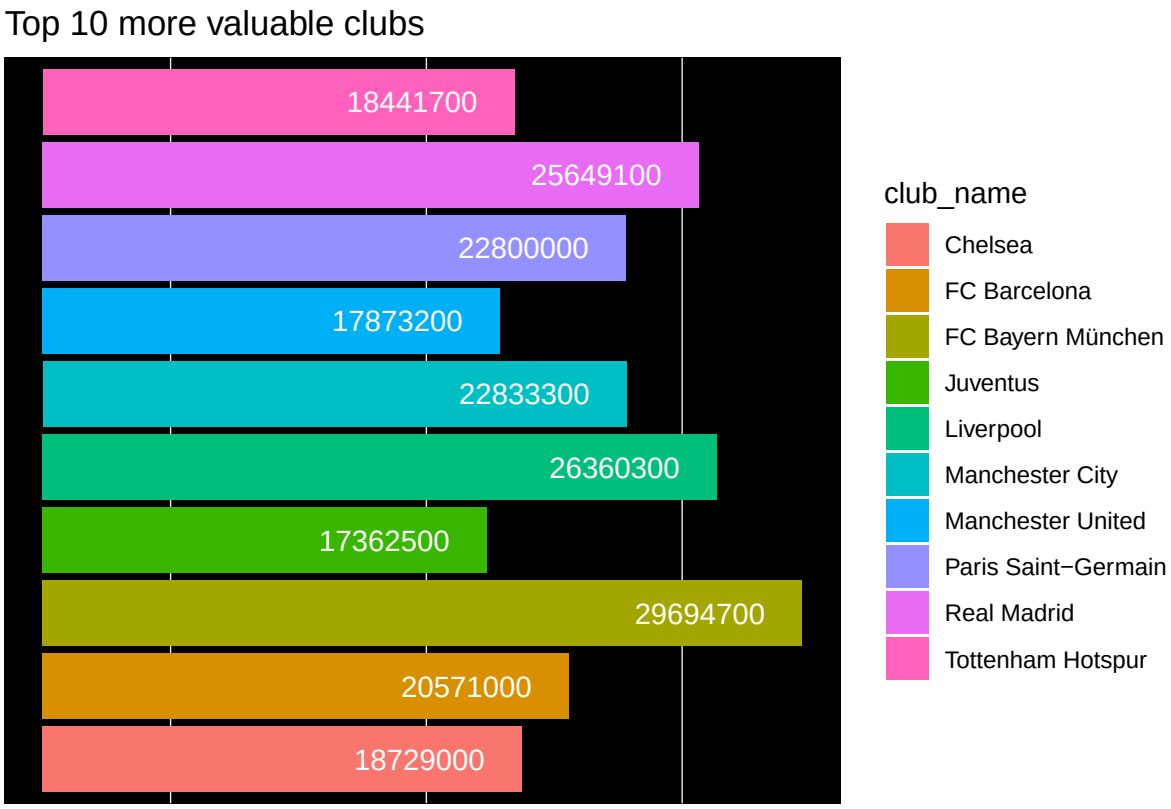
```
Mean_Top10_club_value<- PlayerCLubInfo %>% group_by(club_name) %>% summarise(Mean=mean(value_eur)) %>%
```

```
## Selecting by Mean
```

```
(Mean_Top10_club_value) %>% knitr::kable()
```

club_name	Mean
Chelsea	18729032
FC Barcelona	20570968
FC Bayern München	29694737
Juventus	17362500
Liverpool	26360345
Manchester City	22833333
Manchester United	17873214
Paris Saint-Germain	22800000
Real Madrid	25649074
Tottenham Hotspur	18441667

```
ggplot(Mean_Top10_club_value, aes(x=Mean, y=club_name)) +
  geom_col(aes(fill = club_name)) +
  geom_text(aes(label = signif(Mean)), nudge_y = 0,nudge_x = -4000000, color="white") +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.background = element_rect(fill = "black",
        colour = "black", size = 0.5, linetype = "solid")) +
        theme(axis.text.x = element_blank(),
        axis.text.y = element_blank()) +
        ggtitle("Top 10 more valuable clubs") + xlab("Club value in euros based on players values") + yl
```



Club value in euros based on players values

Overall distribution:

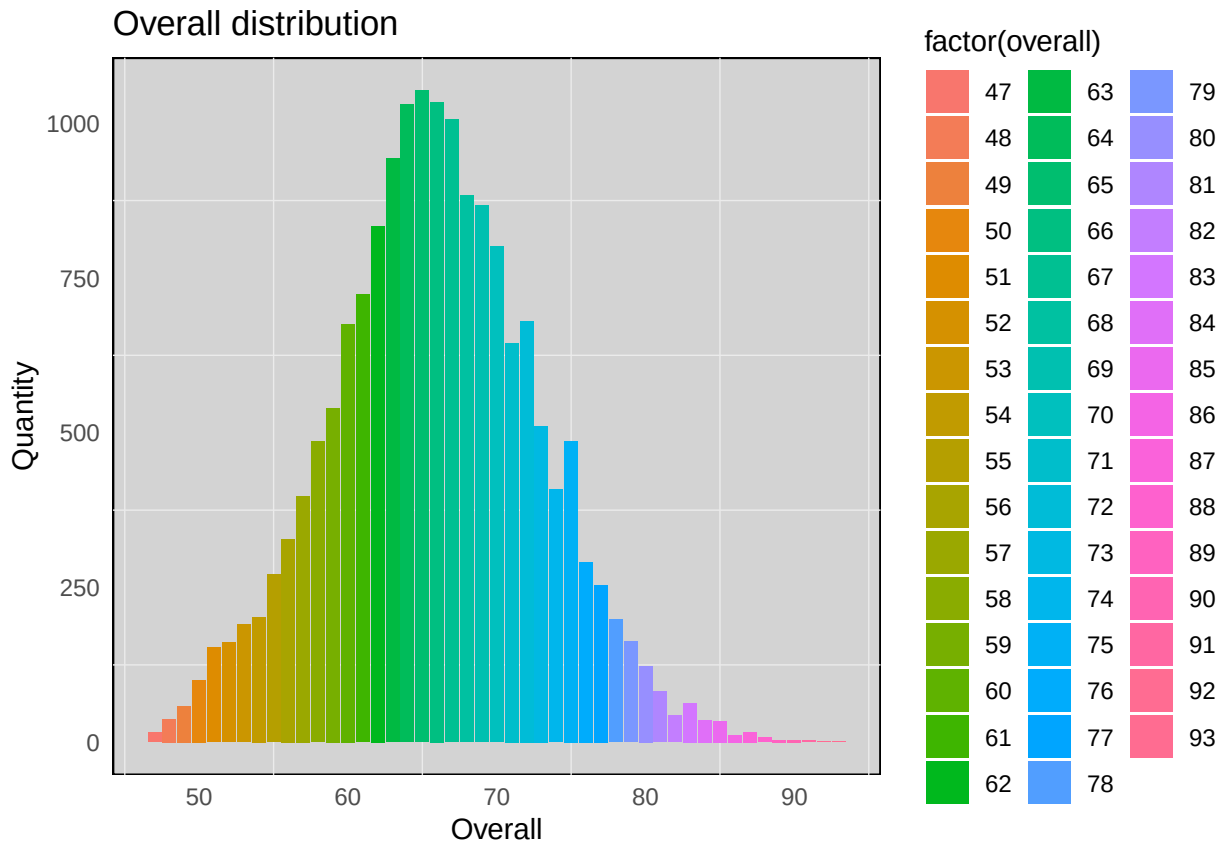
In the following table and plot below, we can see that the more recurrent players overall in the data set is **65** with **1053** appearances and the less recurrent is **92** and **93** with **1** appearances.

```
Quantity_Overalls<- PlayerStats %>% group_by(overall) %>% count()
(Quantity_Overalls) %>% knitr::kable()
```

overall	n
47	16
48	38
49	59
50	101
51	153
52	161
53	190
54	202
55	271
56	328
57	398
58	486
59	540
60	676
61	723
62	834
63	943

overall	n
64	1030
65	1053
66	1034
67	1006
68	883
69	867
70	801
71	645
72	680
73	511
74	408
75	487
76	291
77	254
78	198
79	163
80	123
81	82
82	43
83	63
84	36
85	34
86	12
87	17
88	8
89	4
90	4
91	3
92	1
93	1

```
ggplot(Quantity_Overalls, aes(overall, n)) +
  geom_col(aes(fill = factor(overall))) +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.background = element_rect(fill = "light grey",
        colour = "black", size = 0.5, linetype = "solid")) +
        ggtitle("Overall distribution") + xlab("Overall") + ylab("Quantity")
```



Distribution of players work rate:

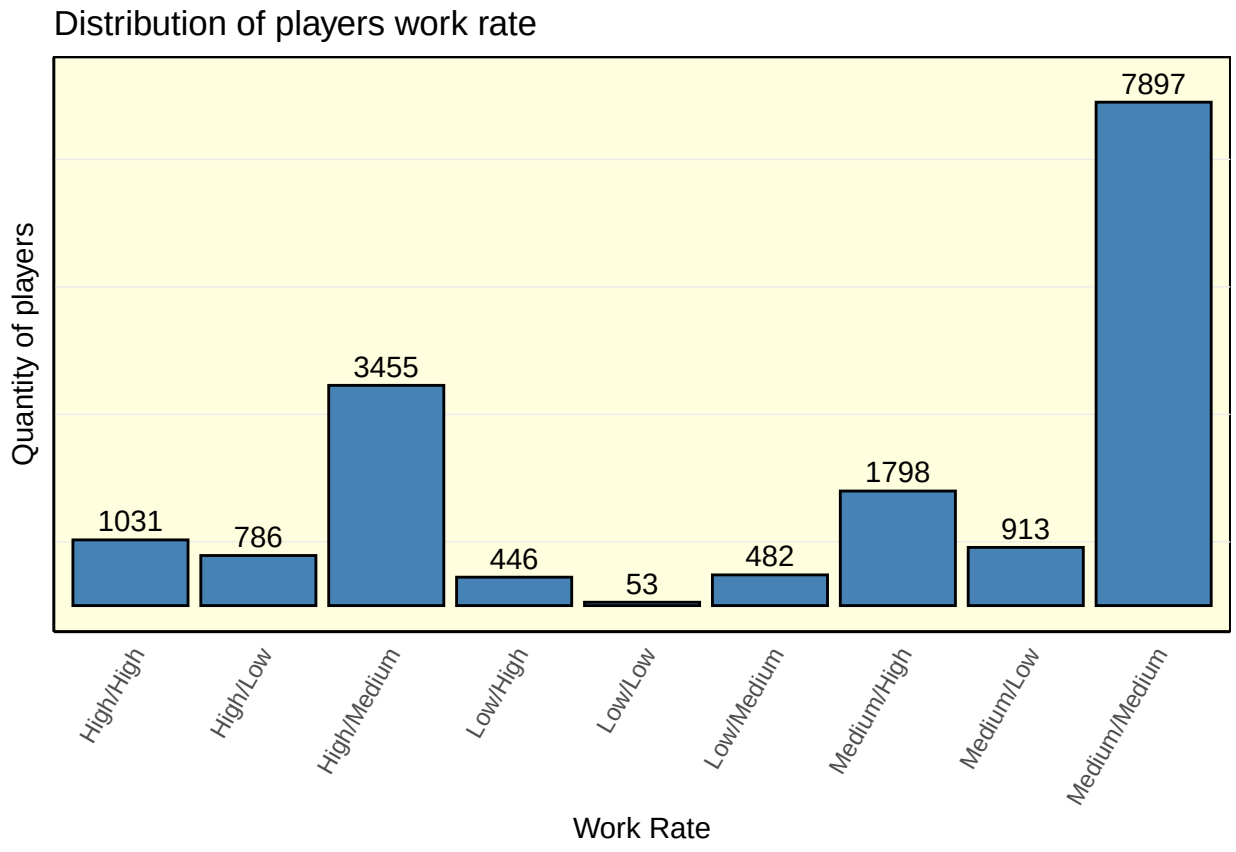
As we can notice in the table and plot below, most of the soccer players in the data set have a work rate of **Medium/Medium** with **7897** appearances. This work rate can be related to the most recurrent overall which is **65** and with this we can conclude that the more the player work, the more overall the player will have.

```
Quantity_WorkRate<- PlayerStats %>% group_by(work_rate) %>% count()
Quantity_WorkRate %>% knitr::kable()
```

work_rate	n
High/High	1031
High/Low	786
High/Medium	3455
Low/High	446
Low/Low	53
Low/Medium	482
Medium/High	1798
Medium/Low	913
Medium/Medium	7897

```
ggplot(Quantity_WorkRate, aes(x=work_rate, y=n)) +
  geom_bar(stat="identity", fill="steel blue",colour="black") +
  geom_text(aes(label = signif(n)), nudge_y = 300,nudge_x = 0,color="black") +
```

```
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.background = element_rect(fill = "light yellow",
        colour = "black", size = 0.5, linetype = "solid"),
        axis.line = element_line(colour = "black"))+
        theme(axis.text.x = element_text(angle = 60, hjust = 1),axis.text.y = element_blank())+
        ggtitle("Distribution of players work rate") + xlab("Work Rate") + ylab("Quantity of players")
```



Top ten players in terms of overall:

In the table below, we can notice that the player with the highest overall is **L. Messi** with an overall of **93**. This explain us the reason why this player receive the highest wage in the data set and have a extremely high market value.

```
Players_overall_name<- PlayerStats %>% group_by(overall) %>% summarise(Name=short_name)
```

```
## `summarise()` has grouped output by 'overall'. You can override using the `.groups` argument.
```

```
Top10_Players<- Players_overall_name[order(-Players_overall_name$overall),][1:10,]
(Top10_Players) %>% knitr::kable()
```

overall	Name
93	L. Messi
92	Cristiano Ronaldo

overall	Name
91	R. Lewandowski
91	Neymar Jr
91	K. De Bruyne
90	K. Mbappé
90	V. van Dijk
90	S. Mané
90	M. Salah
89	S. Agüero

Linear regression model for players age and players overall:

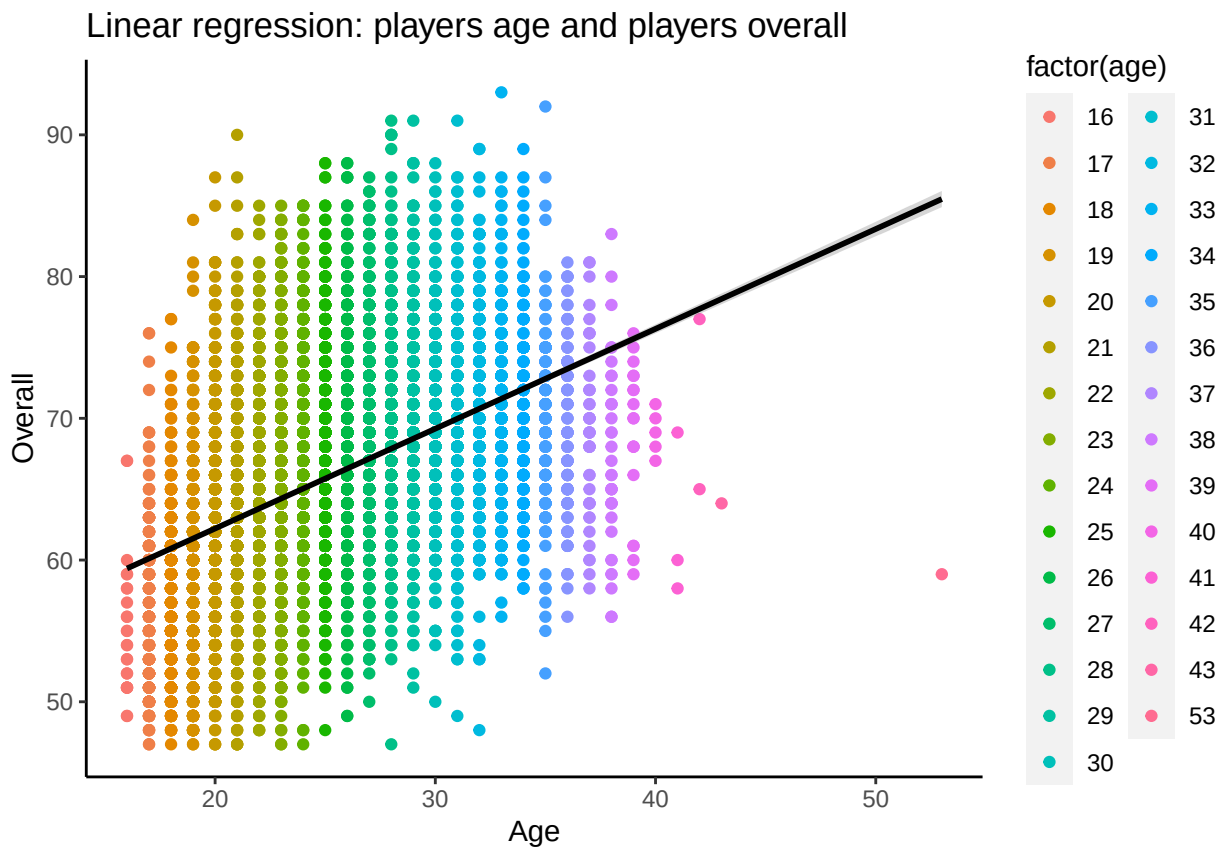
In the result and plot below, we can appreciate a high correlation between the players age and players overall of **0.70**, which means that this two variables are related and depend on each other. After interpreting the result and plot below, we can notice that most of the younger and older players have a lower overall in comparison to the players that can't be considered as old or young players.

```
lm(overall ~ age, data= FIFA2021)
```

```
##
## Call:
## lm(formula = overall ~ age, data = FIFA2021)
##
## Coefficients:
## (Intercept)          age
##     48.0718       0.6979
```

```
ggplot(PlayerStats, aes(x = age, y = overall, colour=factor(age))) +
  geom_point() +
  geom_smooth(method = "lm", col = "black") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))+
        ggtitle("Linear regression: players age and players overall") + xlab("Age") + ylab("Overall")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Quantity of players in every club:

Here we can see an example of this data set with ten clubs. For every clubs should be different the quantity of players (this depends on every league rules).

```
Quantity_PLayers_Clubs<- PlayerCLubInfo %>% group_by(club_name) %>% count()
(Quantity_PLayers_Clubs)[1:10,] %>% knitr::kable()
```

club_name	n
1. FC Heidenheim 1846	25
1. FC Kaiserslautern	25
1. FC Köln	26
1. FC Magdeburg	23
1. FC Nürnberg	24
1. FC Saarbrücken	24
1. FC Union Berlin	23
1. FSV Mainz 05	29
Aalborg BK	21
Aalesunds FK	23

```
Clubs_names<-Quantity_PLayers_Clubs[1]
```

League rank distribution:

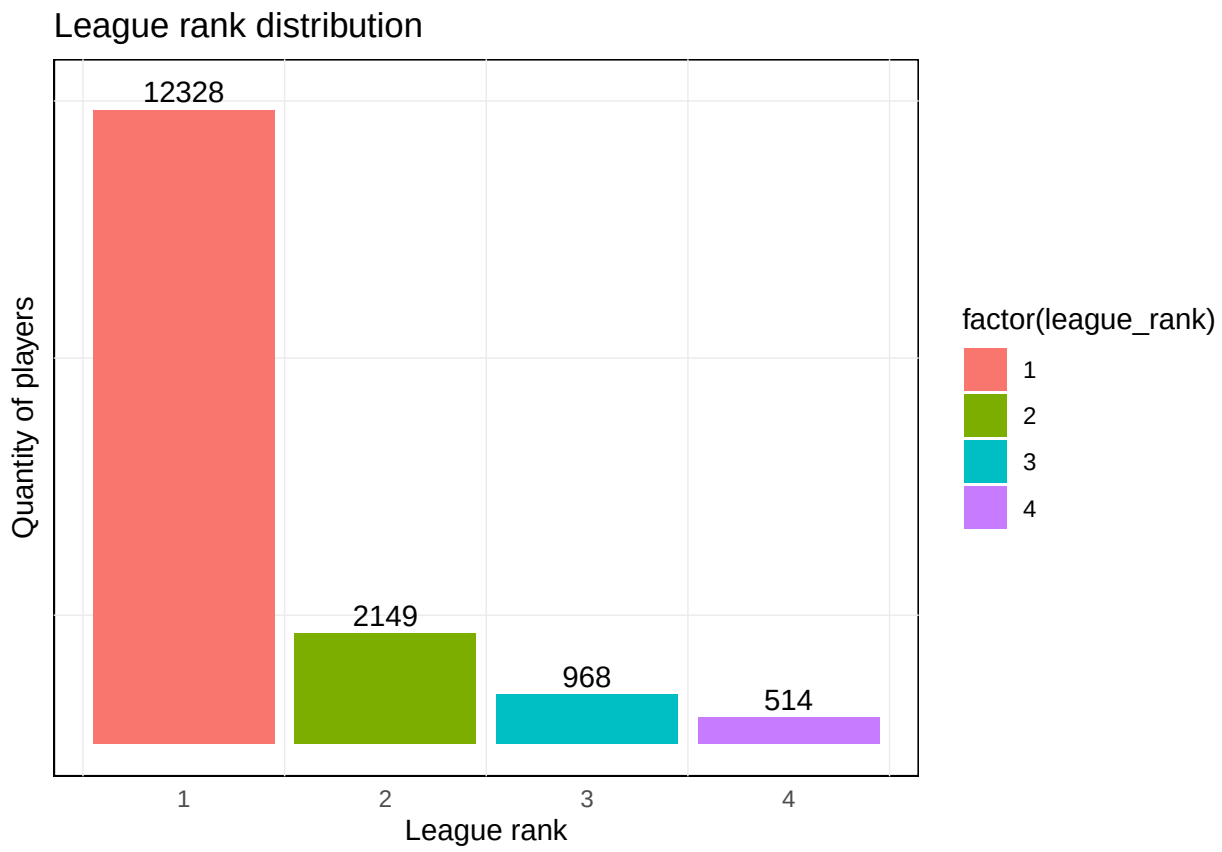
In the following plot and table below, we can appreciate that most of the players (**12328 out of 16861**) in

the data set play in the first league rank. This league rank can be considered as the better league rank in the data set because it counts with the best salaries and more valuable clubs.

```
Quantity_LeagueRanks<-PlayerCLubInfo %>% group_by(league_rank) %>% count()
Quantity_LeagueRanks %>% knitr::kable()
```

league_rank	n
1	12328
2	2149
3	968
4	514

```
ggplot(Quantity_LeagueRanks,aes(league_rank,n)) + geom_col(aes(fill=factor(league_rank))) +
  geom_text(aes(label = signif(n)), nudge_y = 350,nudge_x = 0) +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.background = element_rect(fill = "white",
        colour = "black", size = 0.5, linetype = "solid")) +
        theme(axis.text.y = element_blank(),) +
        ggtitle("League rank distribution") + xlab("League rank") + ylab("Quantity of players")
```



Distribution of players positions:

In the table below, we can appreciate the distribution of all the players positions in the data set.

```
Quantity_TeamPosition<- PlayerCLubInfo %>% group_by(team_position) %>% count()
(Quantity_TeamPosition) %>% knitr::kable()
```

team_position	n
CAM	275
CB	122
CDM	177
CF	13
CM	72
LAM	24
LB	522
LCB	655
LCM	413
LDM	241
LF	12
LM	390
LS	217
LW	160
LWB	75
RAM	25
RB	533
RCB	661
RCM	406
RDM	241
RES	2641
RF	11
RM	407
RS	211
RW	159
RWB	74
ST	423
SUB	6799

Top ten leagues based on players quantity:

As we can see in table below, most of the players play in the MLS (USA Major League Soccer) league. This league counts with **604 players**.

```
Quantity_leagues<- PlayerCLubInfo %>% summarise(LeagueName=league_name) %>% count()

Quantity_Players_Leagues<- PlayerCLubInfo %>% group_by(league_name) %>% count()
Leagues_name<- Quantity_Players_Leagues[1]

Top10_Leagues<- Quantity_Players_Leagues[order(-Quantity_Players_Leagues$n),][1:10,]
Top10_Leagues %>% knitr::kable()
```

league_name	n
USA Major League Soccer	604
English League Championship	603
English Premier League	577
Argentina Primera División	560

league_name	n
Italian Serie A	545
Spain Primera Division	541
French Ligue 1	525
English League One	521
English League Two	514
Turkish Süper Lig	504

Top ten clubs based on overall:

In the following plot and table, we can appreciate the top ten clubs in overall terms. In the first position, we can see the professional soccer club **Real Madrid** with a mean overall of **80.62963**.

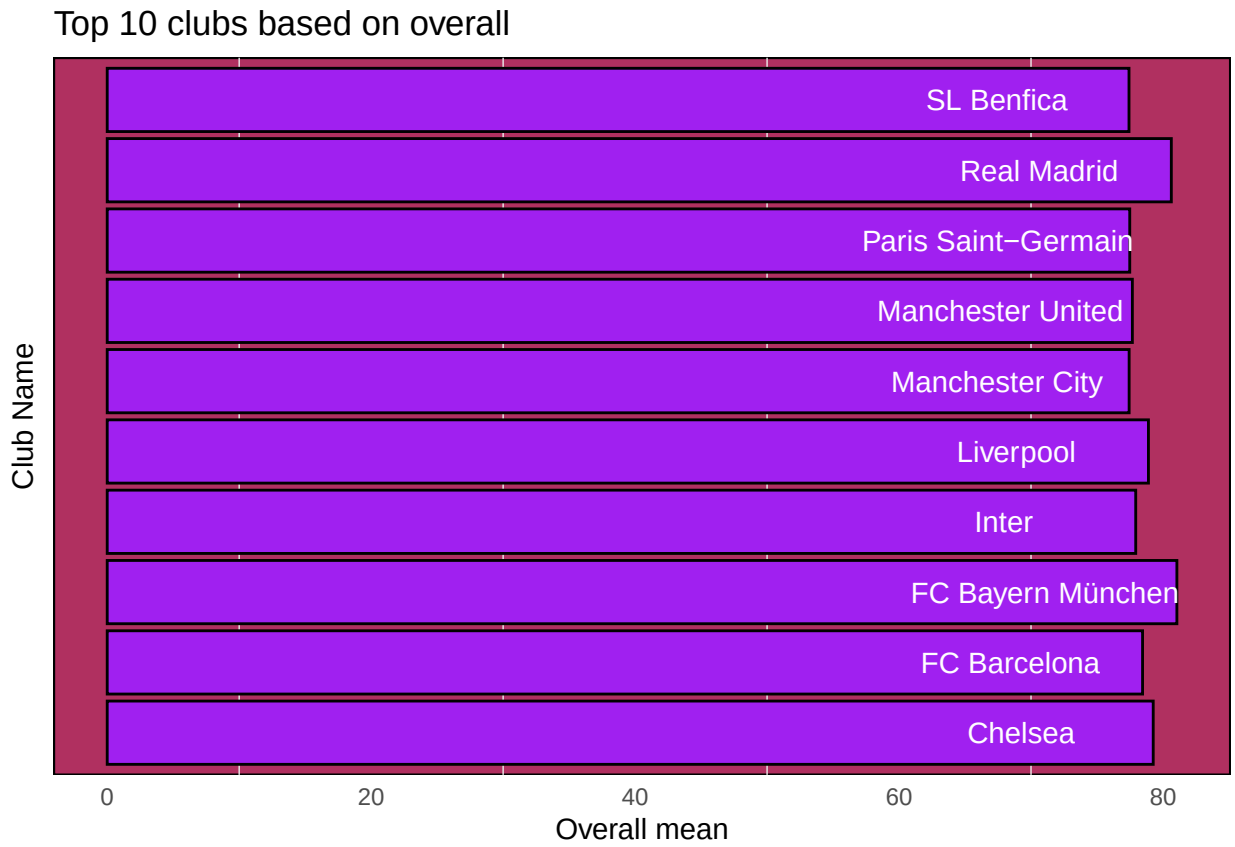
```
Mean_Top10_clubs<- PlayerCLubInfo %>% group_by(club_name) %>% summarise(Mean=mean(overall)) %>% top_n(1
```

```
## Selecting by Mean
```

```
Mean_Top10_clubs %>% knitr::kable()
```

club_name	Mean
Chelsea	79.25806
FC Barcelona	78.45161
FC Bayern München	81.05263
Inter	77.93333
Liverpool	78.89655
Manchester City	77.43333
Manchester United	77.67857
Paris Saint-Germain	77.48000
Real Madrid	80.62963
SL Benfica	77.42308

```
ggplot(Mean_Top10_clubs, aes(x=Mean, y=club_name)) +
  geom_bar(stat="identity", fill="purple",colour="black")  +
  theme_minimal() + theme(panel.grid.major = element_blank(),
  panel.background = element_rect(fill = "maroon",
  colour = "black", size = 0.5, linetype = "solid")) +
  geom_text(aes(y = club_name,label = club_name), nudge_y = 0,nudge_x = -10, color = "white") +
  theme(axis.text.y = element_blank()) +
        ggtitle("Top 10 clubs based on overall") + xlab("Overall mean") + ylab("Club Name")
```



Top ten leagues with the highest salaries:

If we take a look into the top ten leagues with the highest salaries table and plot, we can notice that the league with the best paying is the **English Premier League** with a monthly pay mean of **€53018.198**.

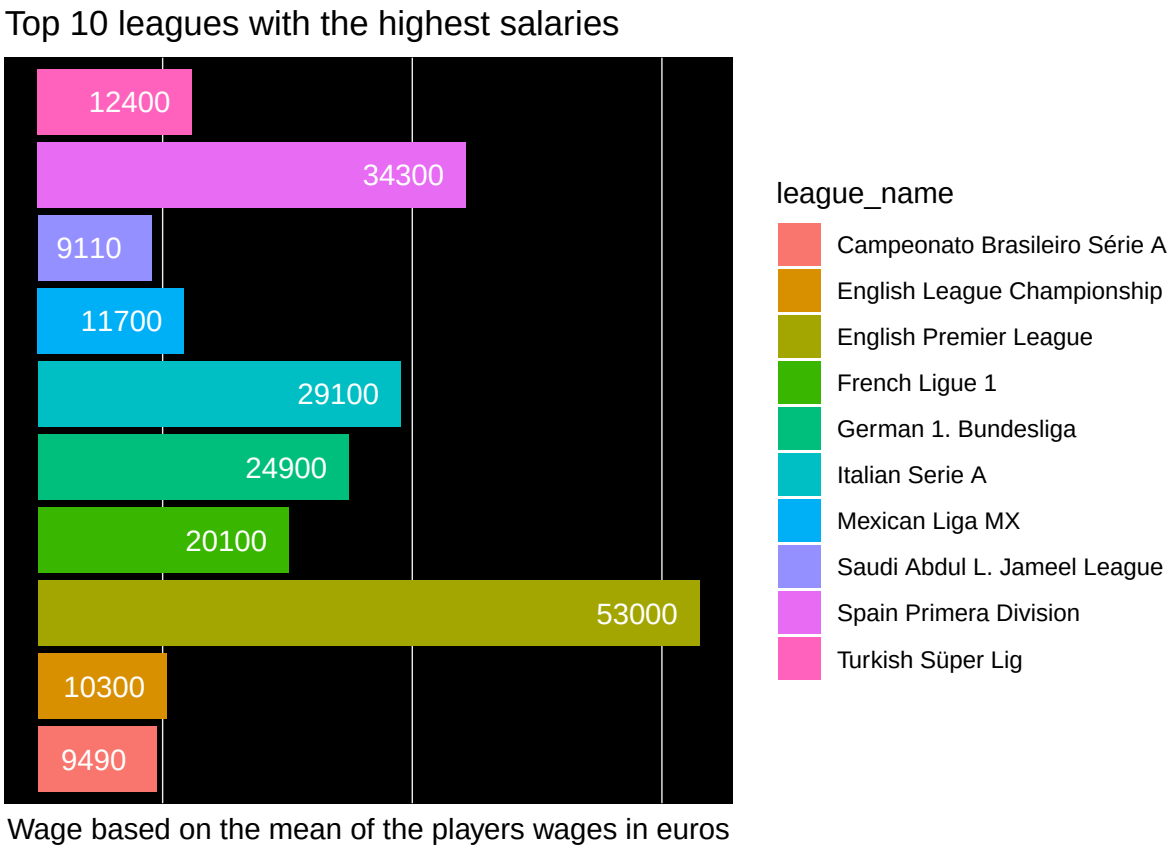
```
Mean_Top10_leagues_wage<- PlayerCLubInfo %>% group_by(league_name) %>% summarise(Mean=mean(wage_eur)) %
```

```
## Selecting by Mean
```

```
Mean_Top10_leagues_wage %>% knitr::kable()
```

league_name	Mean
Campeonato Brasileiro Série A	9486.154
English League Championship	10322.554
English Premier League	53018.198
French Ligue 1	20101.143
German 1. Bundesliga	24900.640
Italian Serie A	29074.862
Mexican Liga MX	11716.870
Saudi Abdul L. Jameel League	9105.820
Spain Primera Division	34293.253
Turkish Süper Lig	12351.587

```
ggplot(Mean_Top10_leagues_wage, aes(x=Mean, y=league_name)) +
  geom_col(aes(fill = league_name))  +  theme_minimal() +
  theme(panel.grid.major = element_blank(),
  panel.background = element_rect(fill = "black",
  colour = "black", size = 0.5, linetype = "solid")) +
  geom_text(aes(label = signif(Mean, digits = 3)), nudge_y = 0,nudge_x = -5000, color = "white") +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank()) +
        ggtitle("Top 10 leagues with the highest salaries") + xlab("Wage based on the mean of the playe
```



Top ten more valuable leagues:

In the top ten more valuable leagues table and plot, we can appreciate that the more valuable league is the **English Premier League** with a mean value of **€10,084,749**.

```
Mean_Top10_leagues_value<- PlayerCLubInfo %>% group_by(league_name) %>% summarise(Mean=mean(value_eur))
```

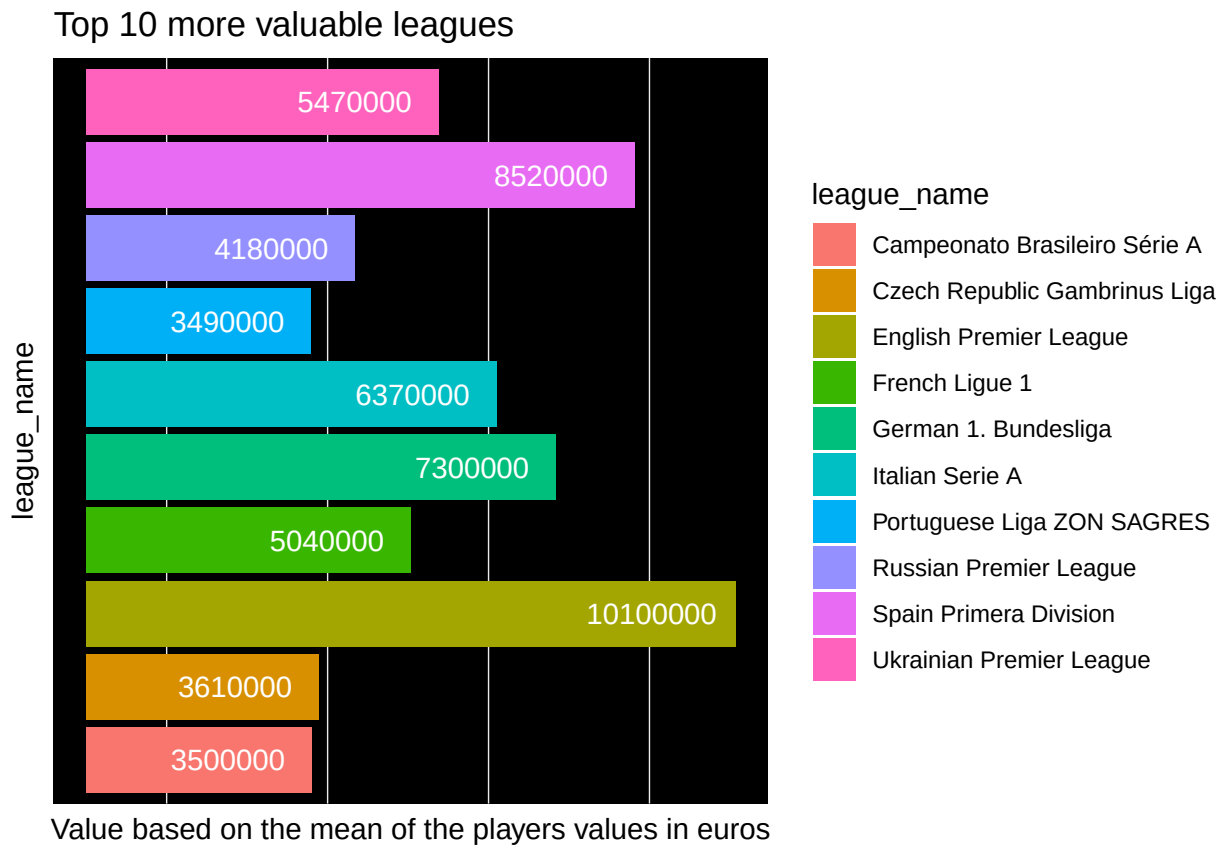
```
## Selecting by Mean
```

```
Mean_Top10_leagues_value %>% knitr::kable()
```

league_name	Mean
Campeonato Brasileiro Série A	3502292
Czech Republic Gambrinus Liga	3612857

league_name	Mean
English Premier League	10084749
French Ligue 1	5041219
German 1. Bundesliga	7295981
Italian Serie A	6370266
Portuguese Liga ZON SAGRES	3494452
Russian Premier League	4177365
Spain Primera Division	8522357
Ukrainian Premier League	5468627

```
ggplot(Mean_Top10_leagues_value, aes(x=Mean, y=league_name)) +
  geom_col(aes(fill = league_name))  +  theme_minimal() + theme(panel.grid.major = element_blank(),
  panel.background = element_rect(fill = "black", colour = "black", size = 0.5, linetype = "solid")) +
  geom_text(aes(label = signif(Mean, digits = 3)), nudge_y = 0,nudge_x = -1300000, color = "white") +
  theme(axis.text.x = element_blank(),
  axis.text.y = element_blank(),) + ggtitle("Top 10 more valuable leagues") + xlab("Value based on the m
```



CREATE FIFA SET (TRAINING SET) AND VALIDATION SET (TEST SET):

In this process, we created a training and validation set in order to start making our modeling process for the FIFA 2022 overalls predictions.

```
#######################################
# CREATE FIFA SET (TRAINING SET) AND #
#      VALIDATION SET (TEST SET)      #
```

```
#####################################

#Validation set will be 40% of FIFA2021 data.
set.seed(123, sample.kind="Rounding")
```

```
## Warning in set.seed(123, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index <- createDataPartition(y = PlayerStats$overall, times = 1, p = 0.40, list = FALSE)
FIFA <- PlayerStats[-test_index,]
validation_set <- PlayerStats[test_index,]

#Here we have to make sure sofifa_id in validation set are also in FIFA set.
validation <- validation_set %>%
  semi_join(validation_set, by = "sofifa_id")

#Add rows removed from validation set back into FIFA set
removed <- anti_join(validation_set, validation)
```

```
## Joining, by = c("sofifa_id", "short_name", "potential", "work_rate", "international_reputation", "ov
```

```
FIFA <- rbind(FIFA, removed)

#Remove non-wanted data sets
rm(test_index, validation_set, removed)
```

```
NA_Values_FIFA<-colSums(is.na(FIFA==TRUE))
NA_Values_FIFA %>% knitr::kable()
```

	x
sofifa_id	0
short_name	0
potential	0
work_rate	0
international_reputation	0
overall	0
age	0
weak_foot	0
skill_moves	0
pace	0
shooting	0
passing	0
dribbling	0
defending	0
physic	0
attacking_crossing	0
attacking_finishing	0
attacking_heading_accuracy	0
attacking_short_passing	0
attacking_volleys	0
movement_acceleration	0

	x
movement_sprint_speed	0
movement_agility	0
movement_reactions	0
movement_balance	0
power_shot_power	0
power_jumping	0
power_stamina	0
power_strength	0
power_long_shots	0
mentality_aggression	0
mentality_interceptions	0
mentality_positioning	0
mentality_composure	0
mentality_vision	0
mentality_penalties	0
defending_standing_tackle	0
defending_sliding_tackle	0
skill_curve	0
skill_dribbling	0
skill_fk_accuracy	0
skill_long_passing	0
skill_ball_control	0

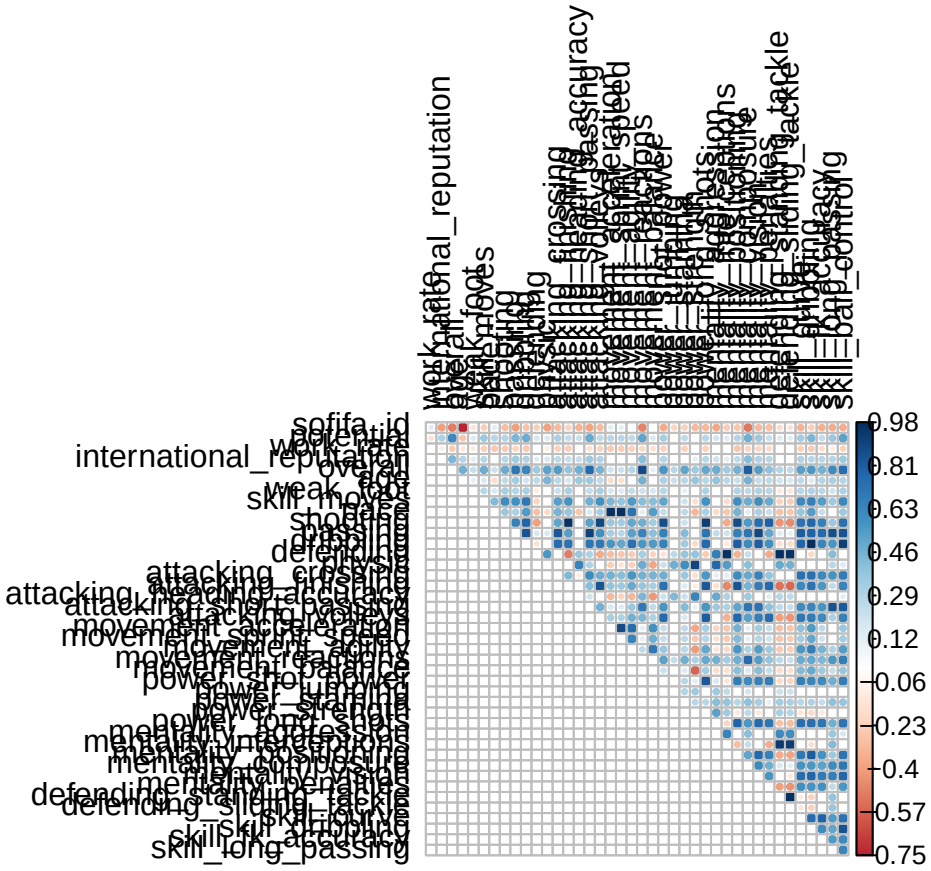
```
NA_Values_Validation<-colSums(is.na(validation==TRUE))
NA_Values_Validation
```

```
##                  sofifa_id                 short_name
##                          0                          0
##                  potential                  work_rate
##                          0                          0
##   international_reputation                    overall
##                          0                          0
##                        age                  weak_foot
##                          0                          0
##                skill_moves                       pace
##                          0                          0
##                   shooting                    passing
##                          0                          0
##                  dribbling                  defending
##                          0                          0
##                     physic         attacking_crossing
##                          0                          0
##        attacking_finishing attacking_heading_accuracy
##                          0                          0
##    attacking_short_passing          attacking_volleys
##                          0                          0
##      movement_acceleration      movement_sprint_speed
##                          0                          0
##           movement_agility         movement_reactions
##                          0                          0
##           movement_balance           power_shot_power
```

```
##                           0                           0
##              power_jumping              power_stamina
##                           0                           0
##             power_strength           power_long_shots
##                           0                           0
##       mentality_aggression    mentality_interceptions
##                           0                           0
##      mentality_positioning        mentality_composure
##                           0                           0
##           mentality_vision        mentality_penalties
##                           0                           0
##  defending_standing_tackle   defending_sliding_tackle
##                           0                           0
##                skill_curve            skill_dribbling
##                           0                           0
##          skill_fk_accuracy         skill_long_passing
##                           0                           0
##         skill_ball_control
##                           0
```

Before starting the modeling process, we have to analyze our variables. The most common and accurate form to evaluate our variables was creating a function that make correlations for all of them. After having the results in a plot, we had to choose the variables with the best results in order to build more precise models.

```
Correlation <- function(data=FIFA,sig=0.1){
  FIFA_Correlation <- FIFA %>% mutate_if(is.character, as.factor)
  FIFA_Correlation <- FIFA_Correlation %>% mutate_if(is.factor, as.numeric)
  corr <- cor(FIFA_Correlation)
  corr[lower.tri(corr,diag=TRUE)] <- NA
  corr[corr == 1] <- NA
  corr <- as.data.frame(as.table(corr))
  corr <- na.omit(corr)
  corr <- subset(corr, abs(Freq) > sig)
  corr <- corr[order(-abs(corr$Freq)),]
  matrix_corr <- reshape2::acast(corr, Var1~Var2, value.var="Freq")
  corrplot(matrix_corr, is.corr=FALSE, tl.col="black", na.label=" ")
}

Correlation()
```



MODELING PROCESS:

In this process, we trained and built different types of models in order to create predictions for the FIFA 2022 players overall. The purpose of this process was to achieve the most accurate predictions using the RMSE as our evaluation metric and demonstrate data visualization, analytics, training and precision skills. All models were retrieved from the caret package.

```
######################
#  MODELING PROCESS  #
######################
```

For the **Linear Regression** model, we did predictions based on the method "lm" and created plots in order to visualize the results. This model can be considered one of the most common for predictive models.

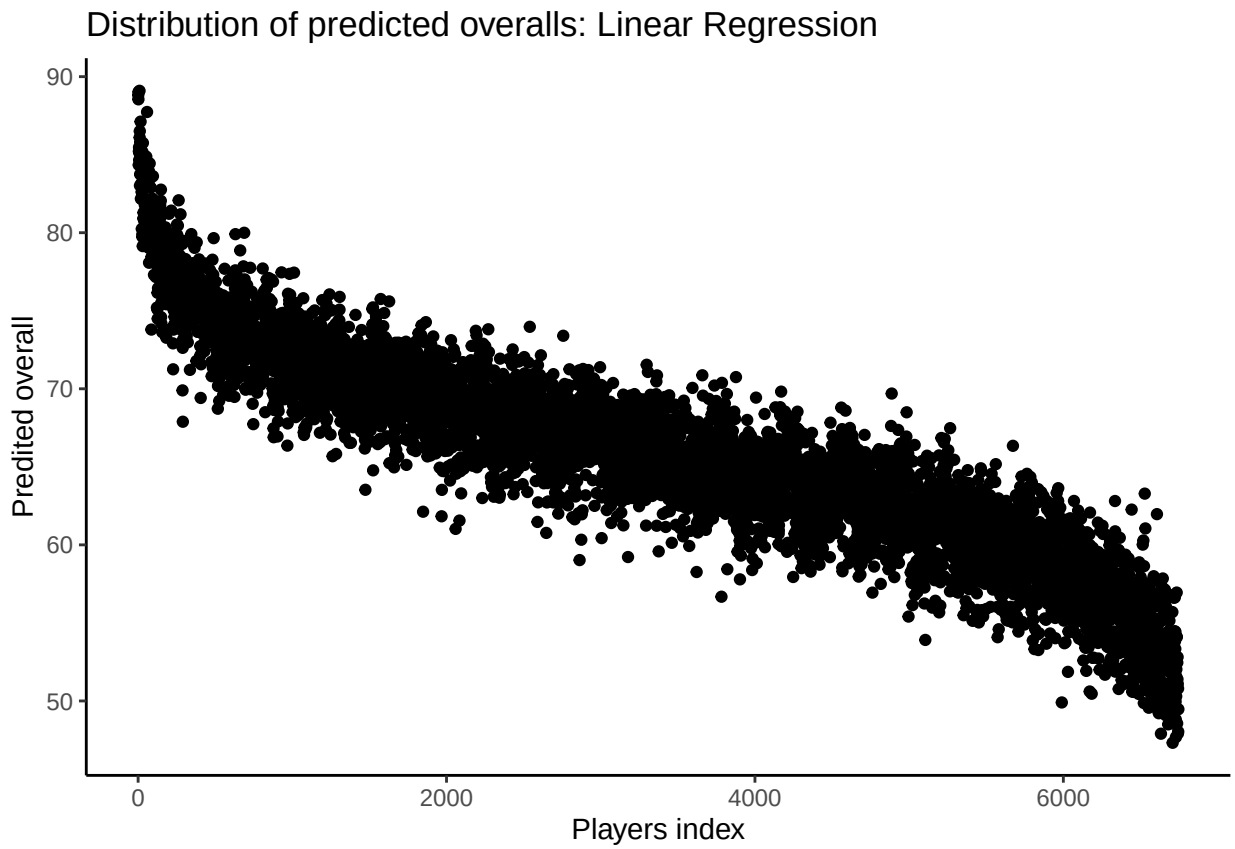
Formula used: $Y = a + bX + e$

```
############################
# LINEAR REGRESSION MODEL #
############################

#We are going to create a linear regression model with the function lm().
#Use overall as the variable that will be predicted and use the rest of the variables as contributors f
#Add our train data for the model.
LinearRegression_Model<-lm(overall~  age + international_reputation+ weak_foot + skill_moves + pace + sh
                           defending + physic + attacking_crossing + attacking_finishing + attacking_headi
                           movement_sprint_speed + movement_agility + movement_reactions + movement_balance
                           power_strength + power_long_shots + mentality_aggression + mentality_interceptio
```

```
                        mentality_penalties + defending_standing_tackle + defending_sliding_tackle+ skil
                        + skill_ball_control,
                        #We are going to use FIFA data set for training the data.
                        data=FIFA)

#Show an executive summary of the model.
Summary<-summary(LinearRegression_Model)
```

After we made the predictions for the **Linear Regression** model, we elaborated a plot in order to visualize the distribution of predicted overalls for all the players in the training set.

```
#We are going to create a new data set, use predict() function, add our model and evaluate in the test
#This will create predictions for our linear regression model.
Prediction_Linear_Regression<-predict(LinearRegression_Model, validation,interval="confidence")

#We are going to create a new data set, use cbind() function and generate an outcome with test set and
Outcome_linear_regression<-cbind(validation, Prediction_Linear_Regression)

#Plot the predictions - this one will show us the distribution of predicted overalls.
ggplot(Outcome_linear_regression,aes(x=1:6746,y=fit)) +geom_point() +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
  ggtitle("Distribution of predicted overalls: Linear Regression") + ylab("Predited overall")+ xlab("Pl
```



```r
#We are going to use confint() function in order to see the confidence intervals.
Confint_values<-confint(LinearRegression_Model)
```

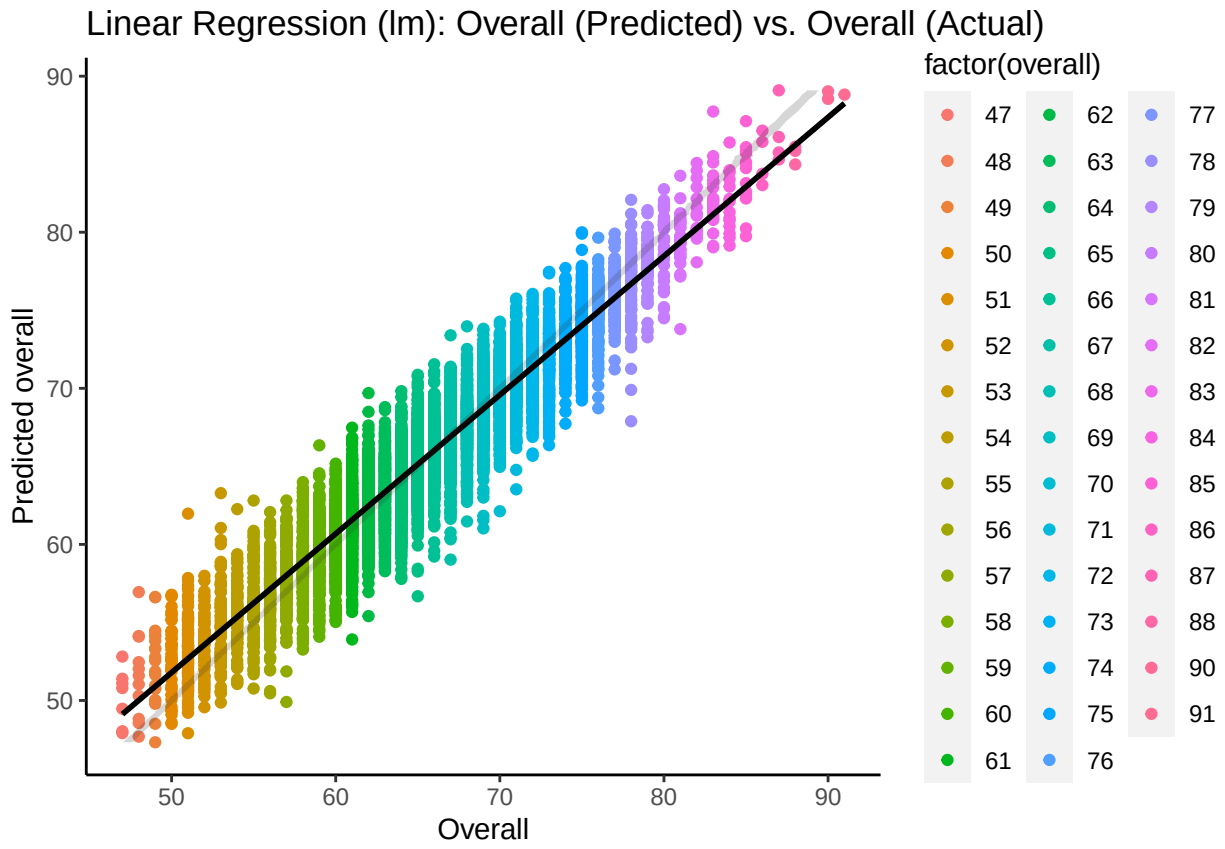
In the following plot, we can see a linear regression graph for our **Linear Regression** model. This plot show us the relationship between the predicted overall and the actual overall. In this plot we can see a high correlation/relationship with a value of **0.8894** in this two variables. However, this graph and value can be improved depending on the prediction made for each model.

```r
#Here we are going to create, evaluated and plot a linear regression model for the following two variab
#overall(actual values) and fit(predicted values).
lm(fit~overall,data=Outcome_linear_regression)
```

```
##
## Call:
## lm(formula = fit ~ overall, data = Outcome_linear_regression)
##
## Coefficients:
## (Intercept)      overall
##      7.3219       0.8894
```

```r
ggplot(Outcome_linear_regression, aes(x = overall, y =fit, xmin = lwr, xmax = upr,color = factor(overall
  geom_point() +
  geom_smooth(method = "lm", col = "black") + geom_ribbon(alpha = 0.2, color = FALSE) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))+
        ggtitle("Linear Regression (lm): Overall (Predicted) vs. Overall (Actual)") + xlab("Overall") +
```

```
## `geom_smooth()` using formula 'y ~ x'
```



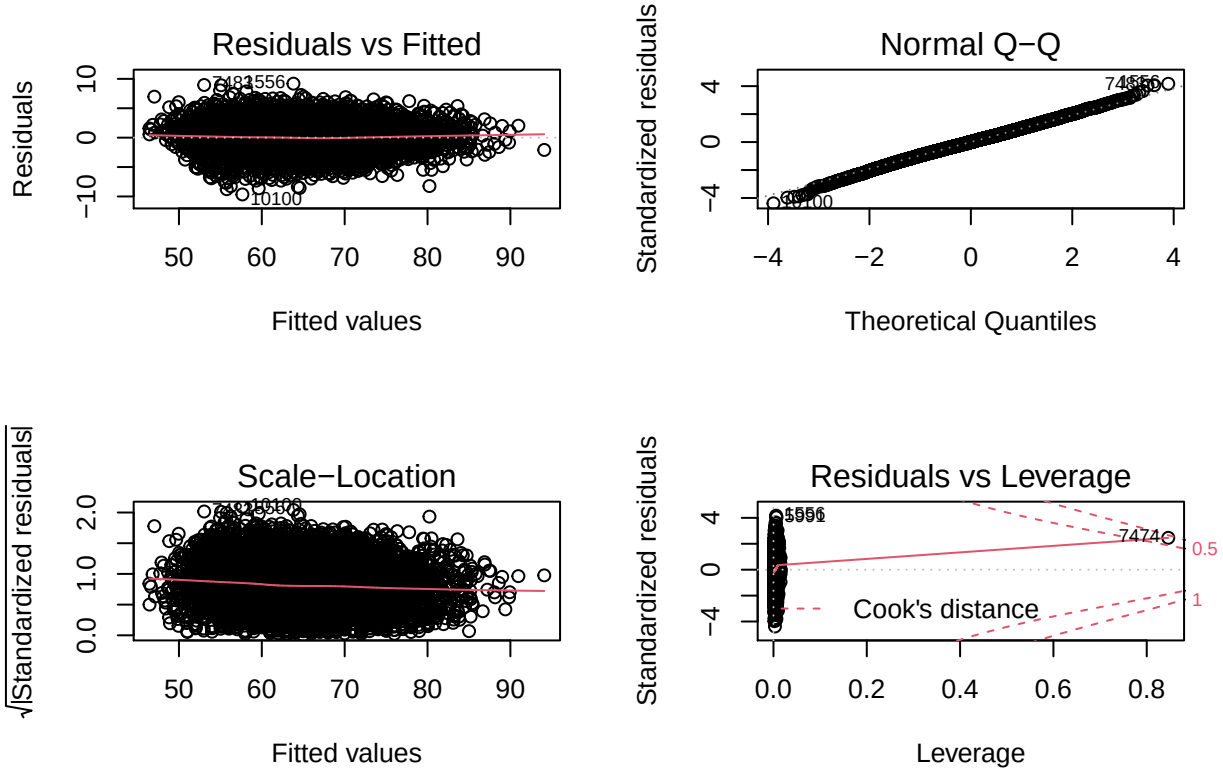

```
#We are going to evaluate our linear regression model with the RMSE as our metric.
Linear_regression_RMSE<-RMSE(validation$overall,Prediction_Linear_Regression)
#Show the RMSE.
Linear_regression_RMSE %>% knitr::kable()
```

x
2.240636

In the following plot, we can see the main four graphs for the **Linear Regression** model.

Graphs created: Residuals vs. Fitted, Normal Q-Q, Scale-Location and Residuals vs. Leverage.

```
#We are going to show the main four graphs of our model.
par(mfrow=c(2,2))
plot(LinearRegression_Model, which=1) #Residuals vs. Fitted
plot(LinearRegression_Model, which=2) #Normal Q-Q
plot(LinearRegression_Model, which=3) #Scale-Location
plot(LinearRegression_Model, which=5) #Residuals vs. Leverage
```



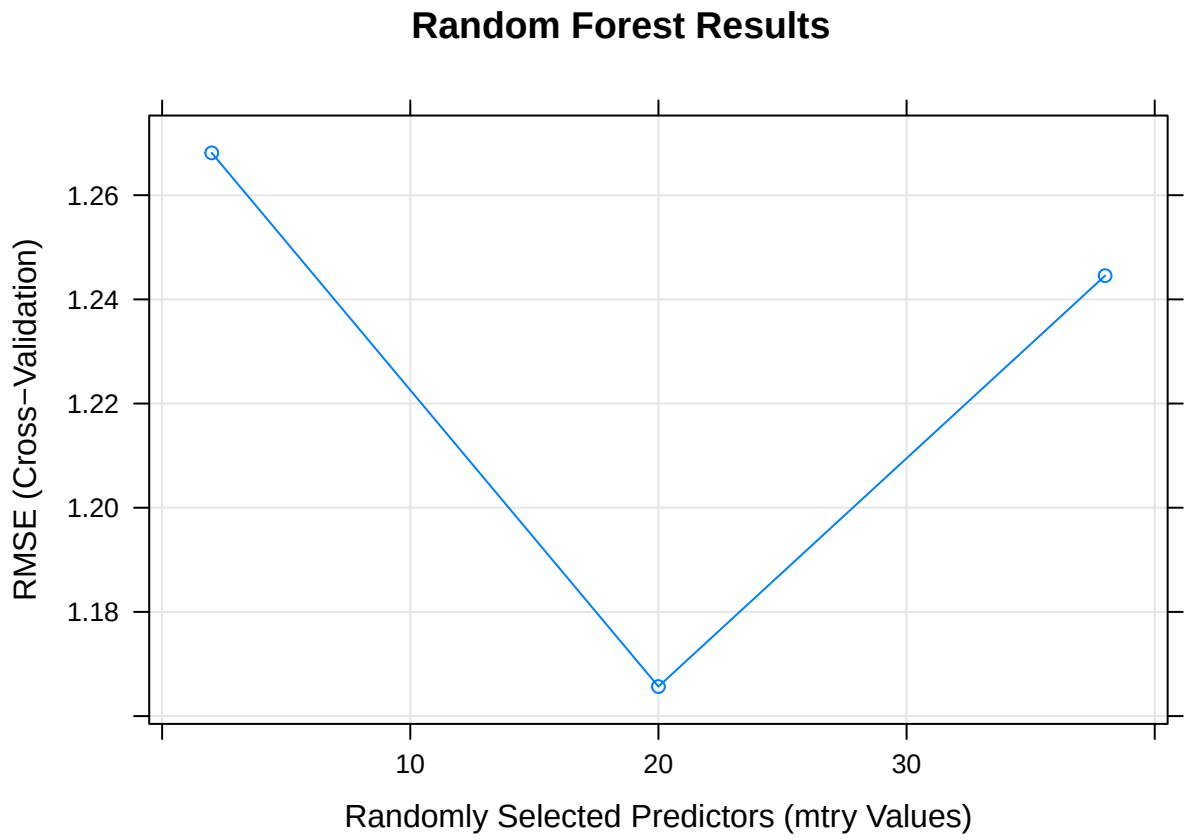
For the **Random Forest** model, we have done predictions based on the method "rf" and created plots in order to visualize the results. This model can be considered one of the most common and accurate for predictive models and is define as an ensemble learning method for classification.

```
########################
# RANDOM FOREST MODEL #
########################

#We are going create a trControl with the best parameters for the Random Forest model.
trControlRF <- trainControl(method = "cv",
                            number = 10,
                            search = "grid")

#We are going to create a Random Forest model with the function train().
#Use overall as the variable that will be predicted and use the rest of the variables as contributors f
#Add our train data, train control parameter define above, our metric and method for the model.
RandomForest_Model <- train(overall~  age + international_reputation+ weak_foot + skill_moves + pace + s
                            defending + physic + attacking_crossing + attacking_finishing + attacking_headi
                            movement_sprint_speed + movement_agility + movement_reactions + movement_balanc
                            power_strength + power_long_shots + mentality_aggression + mentality_interceptio
                            mentality_penalties + defending_standing_tackle + defending_sliding_tackle+ skil
                            + skill_ball_control,
                            data = FIFA,
                            method = "rf",
                            metric="RMSE",
                            trControl = trControlRF)
```

```r
#show the results.
show(RandomForest_Model)
```

```
## Random Forest
##
## 10115 samples
##    38 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9103, 9102, 9104, 9104, 9104, 9105, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##    2    1.268120  0.9716293  0.9616607
##   20    1.165671  0.9727149  0.8846786
##   38    1.244548  0.9683474  0.9367475
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 20.
```

```r
#Plot the results.
plot(RandomForest_Model,xlab="Randomly Selected Predictors (mtry Values)", main="Random Forest Results")
```

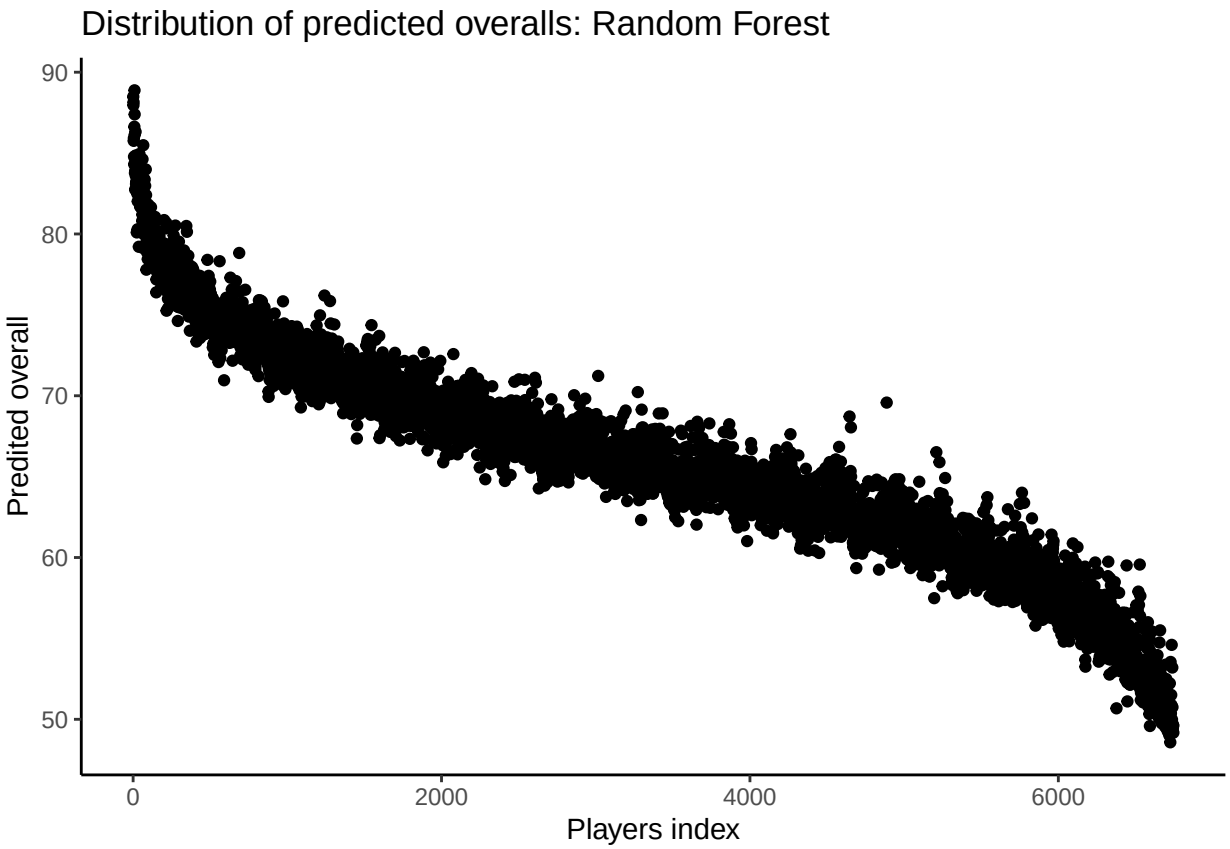


```
#We are going to create a new data set and store the best tune obtained in the model.
Best_mtry<-RandomForest_Model$bestTune$mtry
#We are going to create a new data set and store the best RMSE obtained in the model.
Best_RMSE<-min(RandomForest_Model$results$RMSE)
#show the best RMSE result.
(Best_RMSE) %>% knitr::kable()
```

x
1.165671

After we made the predictions for the **Random Forest** model, we elaborated a plot in order to visualize the distribution of predicted overalls for all the players in the training set.

```
#We are going to create a new data set, use predict() function, add our model and evaluate in the test
#This will create predictions for our random forest model.
Prediction_RandomForest<-predict(RandomForest_Model, validation,interval="prediction")

#We are going to create a new data set, use cbind() function and generate an outcome with test set and
Outcome_RandomForest<-cbind(validation, Prediction_RandomForest)

#Plot the predictions - this one will show us the distribution of predicted overalls.
ggplot(Outcome_RandomForest,aes(x=1:6746,y=Prediction_RandomForest)) +geom_point() +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
  ggtitle("Distribution of predicted overalls: Random Forest") + ylab("Predited overall")+ xlab("Players
```



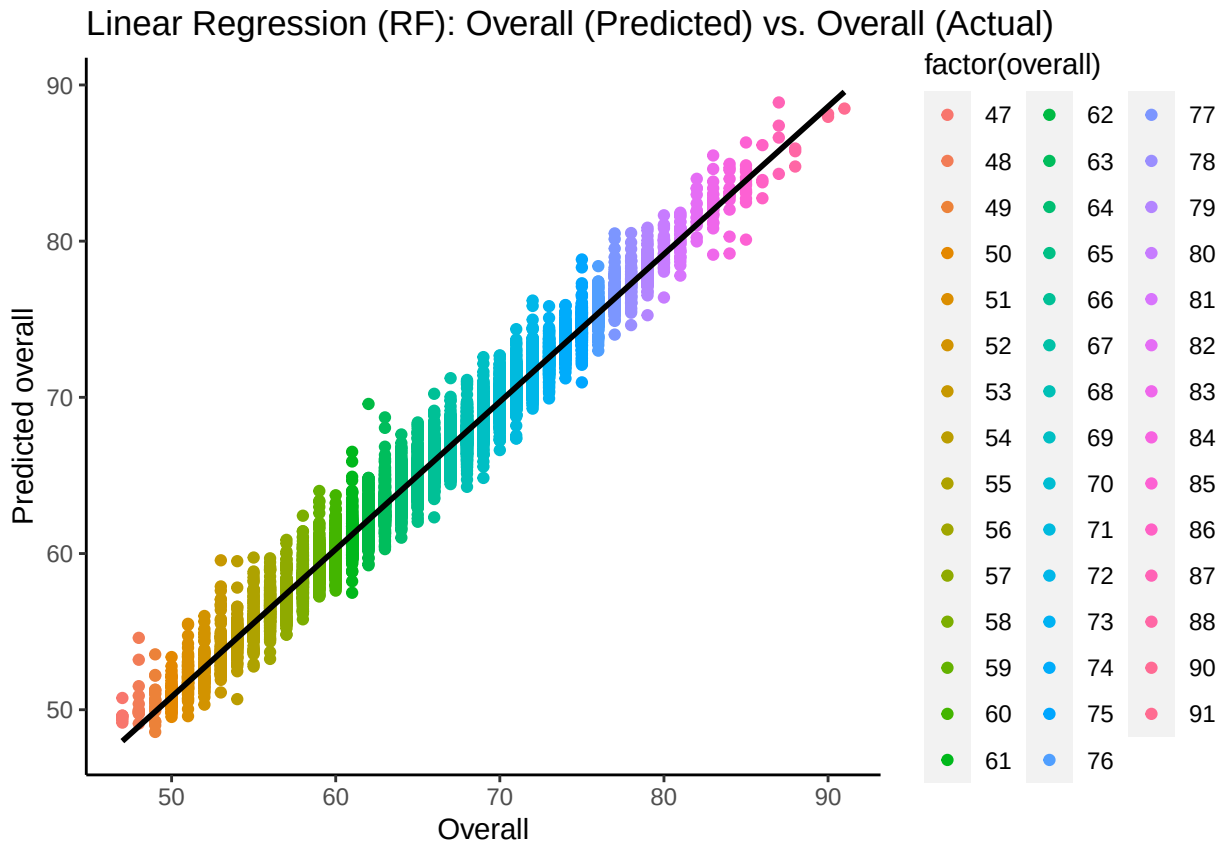
In the following plot, we can see a linear regression graph for our **Random Forest** model. This plot show us the relationship between the predicted overall and the actual overall. In this plot we can see an almost perfect correlation/relationship with a value of **0.9447** in this two variables. The reason why this model got a better correlation in comparison to the linear regression model was because the predictions for this model were more precise.

```
#Here we are going to create, evaluated and plot a linear regression model (RF) for the following two v
#overall(actual values) and fit(predicted values).
lm(Prediction_RandomForest~overall,data=Outcome_RandomForest)
```

```
##
## Call:
## lm(formula = Prediction_RandomForest ~ overall, data = Outcome_RandomForest)
##
## Coefficients:
## (Intercept)      overall
##      3.5871       0.9447
```

```
ggplot(Outcome_RandomForest, aes(x = overall, y = Prediction_RandomForest,color = factor(overall))) +
  geom_point() +
  geom_smooth(method = "lm", col = "black") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))+
        ggtitle("Linear Regression (RF): Overall (Predicted) vs. Overall (Actual)") + xlab("Overall") +
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
#We are going to evaluate our Random Forest model with the RMSE as our metric.
RF_RMSE<-RMSE(validation$overall,Prediction_RandomForest)
#Show the RMSE.
RF_RMSE %>% knitr::kable()
```

x
1.14096

For the **(Knn) K-nearest neighbors** model, we have done predictions based on the method "knn" and created plots in order to visualize the results. This model can be considered one of the most common for predictive models but is not more accurate than the random forest model. This model can be defined as a supervised machine learning algorithm that can be used to solve both classification and regression tasks.

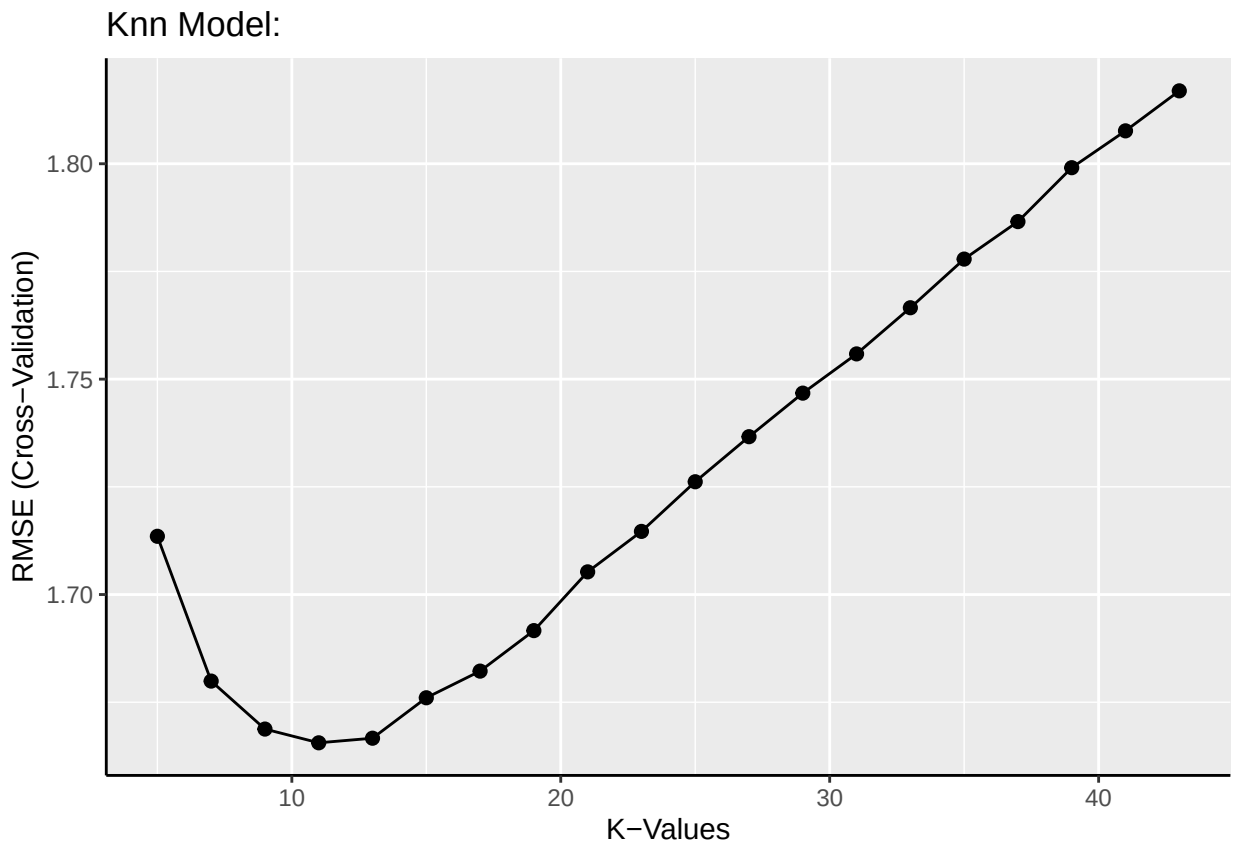
```
##############
# KNN MODEL #
##############

#We are going create a trControl with the best parameters for the Knn model.
trControlKnn <- trainControl(method = "cv",
                             number = 10,
                             search = "grid")

#We are going to create a Knn model with the function train().
```

```r
#Use overall as the variable that will be predicted and use the rest of the variables as contributors f
#Add our train data, train control parameter define above, tuneLength parameter, our metric and method
Knn_Model <- train(overall~  age + international_reputation+ weak_foot + skill_moves + pace + shooting +
                        defending + physic + attacking_crossing + attacking_finishing + attacking_headi
                        movement_sprint_speed + movement_agility + movement_reactions + movement_balanc
                        power_strength + power_long_shots + mentality_aggression + mentality_interceptio
                        mentality_penalties + defending_standing_tackle + defending_sliding_tackle+ skil
                        + skill_ball_control,
                        data = FIFA,
                        method = "knn",
                        metric="RMSE",
                        tuneLength = 20,
                        trControl = trControlKnn)
```

```r
#show the results.
show(Knn_Model)
```

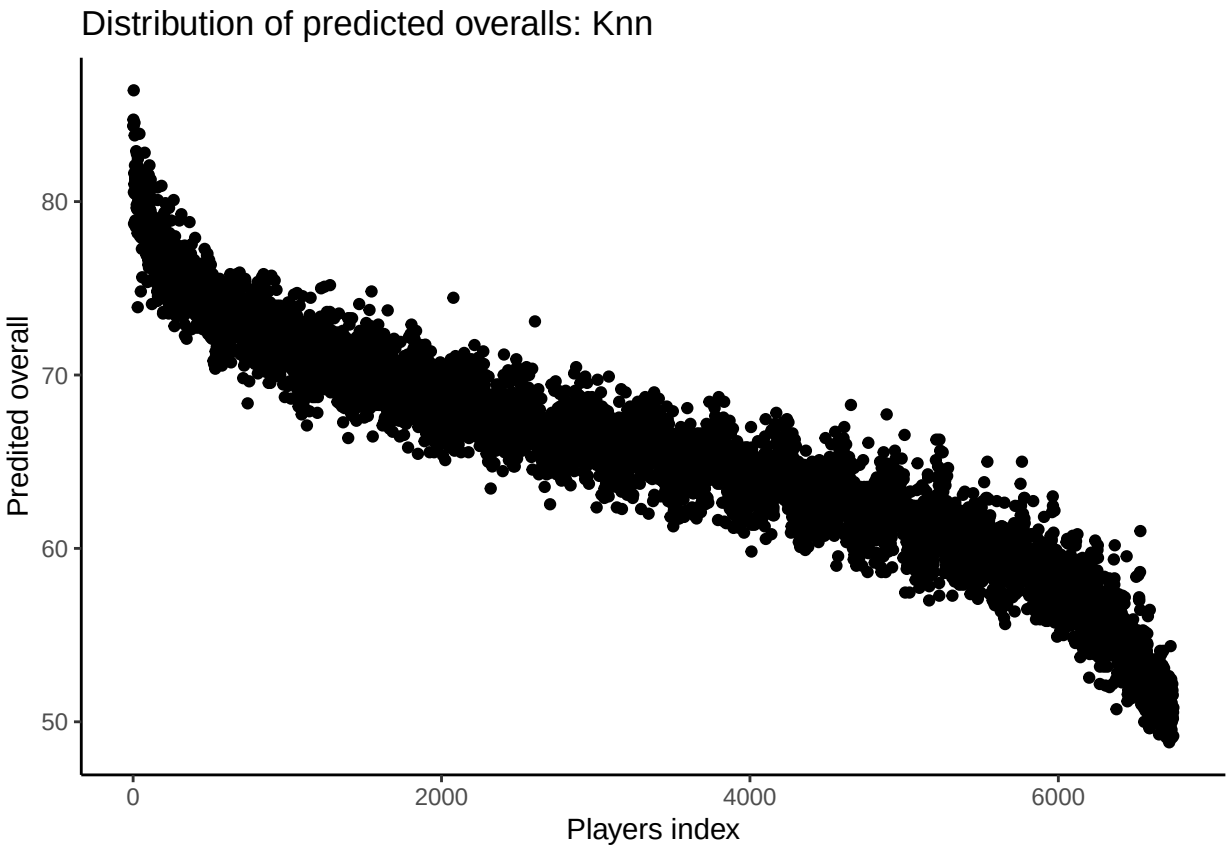
```
## k-Nearest Neighbors
##
## 10115 samples
##    38 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 9105, 9103, 9104, 9103, 9103, 9105, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
##    5  1.713521  0.9441665  1.334563
##    7  1.679907  0.9475465  1.306158
##    9  1.668770  0.9496132  1.297946
##   11  1.665579  0.9507730  1.289880
##   13  1.666648  0.9514611  1.289381
##   15  1.676034  0.9514207  1.292446
##   17  1.682223  0.9515792  1.297111
##   19  1.691626  0.9514586  1.304457
##   21  1.705263  0.9511615  1.313350
##   23  1.714656  0.9510088  1.320147
##   25  1.726180  0.9506663  1.328733
##   27  1.736641  0.9504521  1.336610
##   29  1.746748  0.9501983  1.343150
##   31  1.755858  0.9499960  1.350069
##   33  1.766573  0.9496953  1.358895
##   35  1.777869  0.9493380  1.367318
##   37  1.786572  0.9491557  1.372746
##   39  1.799085  0.9487060  1.381453
##   41  1.807636  0.9484921  1.387764
##   43  1.816925  0.9482497  1.394178
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 11.
```

```r
#Plot the results - RMSE vs. K-values.
ggplot(Knn_Model,aes(x=k,y=Knn_Model$results$RMSE)) + geom_point(size=2) +
                      theme(axis.line = element_line(colour = "black")) +
                      ggtitle("Knn Model:")  + xlab("K-Values")
```



After we made the predictions for the **(Knn) K-nearest neighbors** model, we elaborated a plot in order to visualize the distribution of predicted overalls for all the players in the training set.

```r
#We are going to create a new data set, use predict() function, add our model and evaluate in the test
#This will create predictions for our Knn model.
Prediction_Knn<-predict(Knn_Model, validation,interval="prediction")

#We are going to create a new data set, use cbind() function and generate an outcome with test set and
Outcome_Knn<-cbind(validation, Prediction_Knn)

#Plot the predictions - this one will show us the distribution of predicted overalls.
ggplot(Outcome_Knn,aes(x=1:6746,y=Prediction_Knn)) +geom_point() +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
  ggtitle("Distribution of predicted overalls: Knn") + ylab("Predited overall")+ xlab("Players index")
```



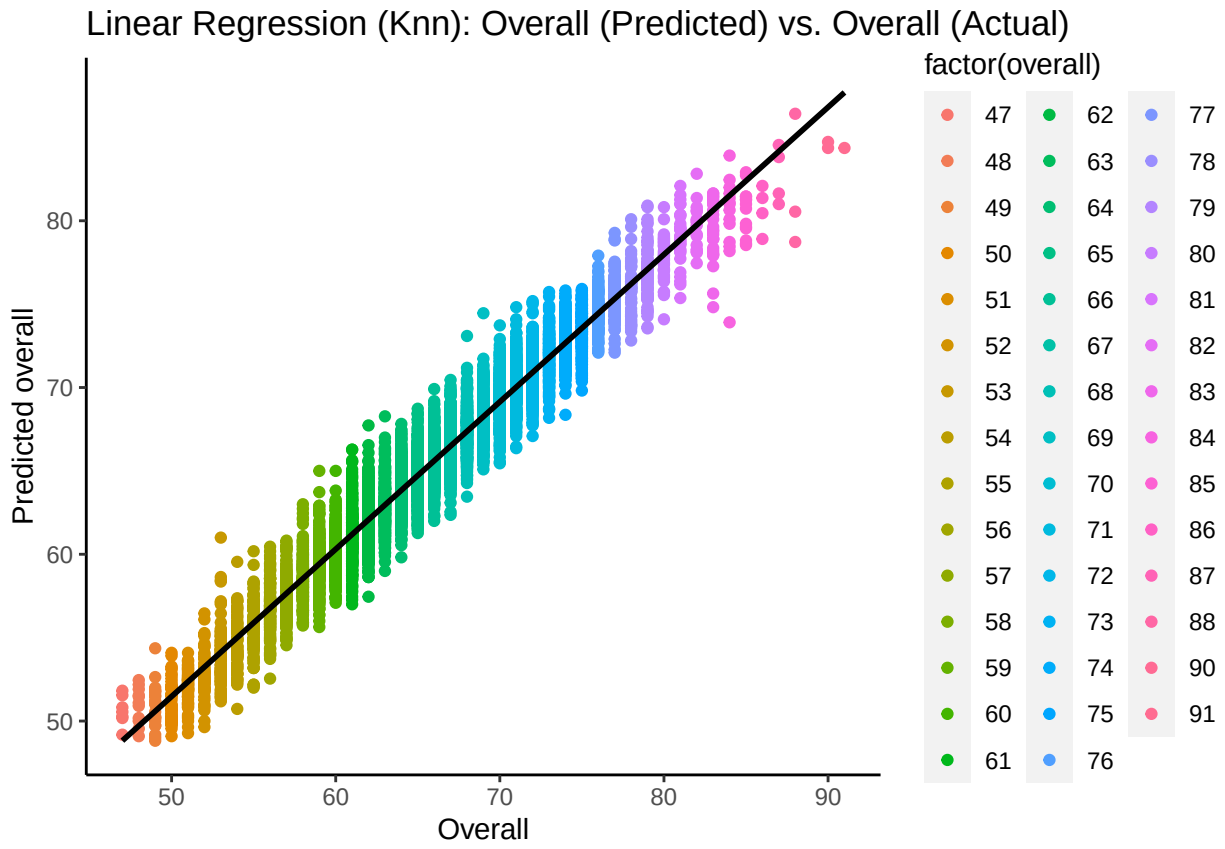
In the following plot, we can see a linear regression graph for our **(Knn) K-nearest neighbors** model. This plot show us the relationship between the predicted overall and the actual overall. In this plot we can see a high correlation/relationship with a value of **0.8835** in this two variables.

```
#Here we are going to create, evaluated and plot a linear regression model (Knn) for the following two
#overall(actual values) and fit (predicted values).
lm(Prediction_Knn~overall,data=Outcome_Knn)
```

```
##
## Call:
## lm(formula = Prediction_Knn ~ overall, data = Outcome_Knn)
##
## Coefficients:
## (Intercept)      overall
##      7.2991       0.8835
```

```
ggplot(Outcome_Knn, aes(x = overall, y = Prediction_Knn,color = factor(overall))) +
  geom_point() +
  geom_smooth(method = "lm", col = "black") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))+
        ggtitle("Linear Regression (Knn): Overall (Predicted) vs. Overall (Actual)") + xlab("Overall")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
#We are going to evaluate our Knn model with the RMSE as our metric.
Knn_RMSE<-RMSE(validation$overall,Prediction_Knn)
#Show the RMSE.
(Knn_RMSE) %>% knitr::kable()
```

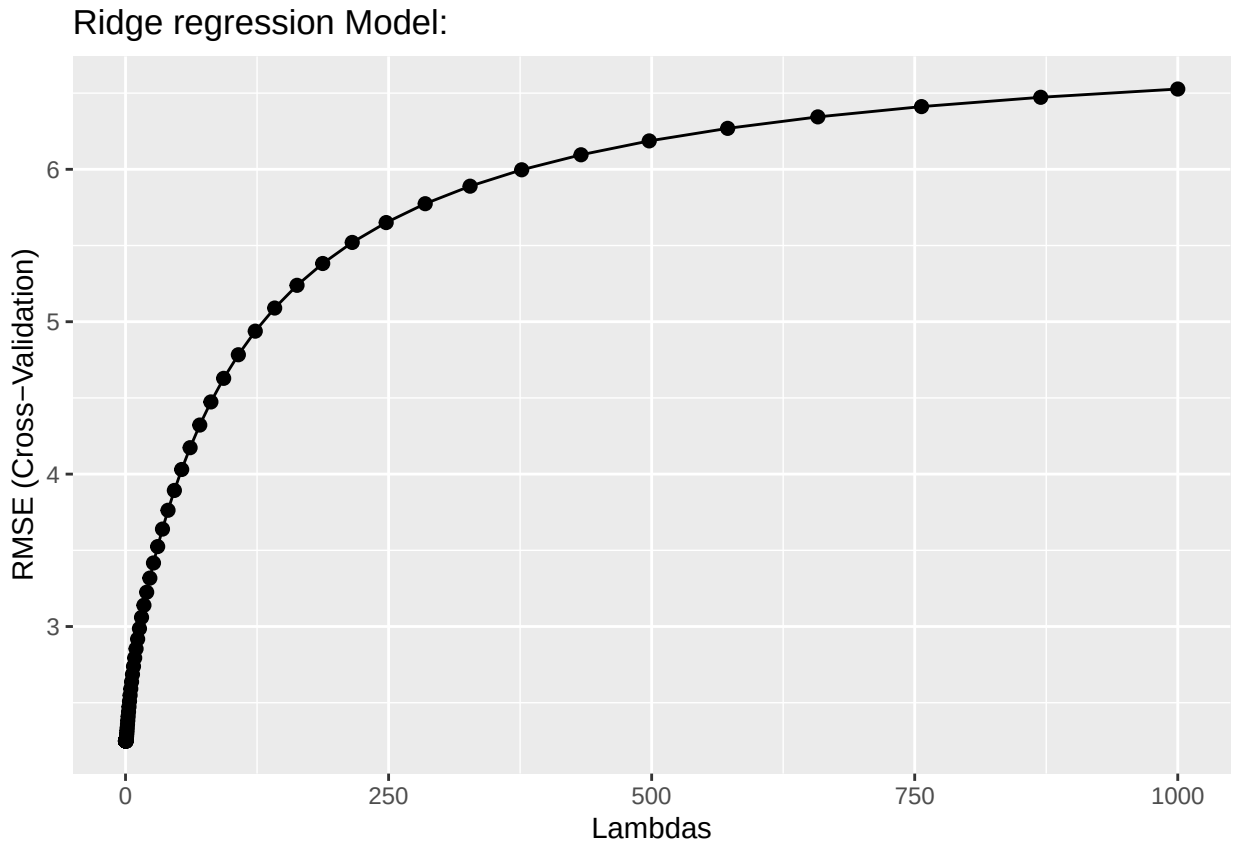
x
1.652653

For the **Ridge Regression** model, we have done predictions based on the method "glm" and created plots in order to visualize the results. This is a model tuning method that is used to analyze any data that suffers from multicollinearity. Also, this model can be considered common for predicting models.

```
############################
# RIDGE REGRESSION MODEL #
############################

#We are going create a trControl with the best parameters for the Ridge Regression model.
trControlRigde <- trainControl("cv", number = 10)

#We are going to create a lambdas with the best parameters for the Ridge Regression model.
lambda <- 10^seq(-3, 3, length = 100)

#We are going to create a Ridge Regression model with the function train().
```

```r
#Use overall as the variable that will be predicted and use the rest of the variables as contributors f
#Add our train data, train control parameter define above, tuneGrid parameter, our metric and method fo
Ridge_Model<-train(overall~  age + international_reputation+ weak_foot + skill_moves + pace + shooting +
                           defending + physic + attacking_crossing + attacking_finishing + attacking_
                           movement_sprint_speed + movement_agility + movement_reactions + movement_ba
                           power_strength + power_long_shots + mentality_aggression + mentality_intere
                           mentality_penalties + defending_standing_tackle + defending_sliding_tackle+
                           + skill_ball_control,
                           data=FIFA,
                           method="glmnet",
                           trControl=trControlRigde,
                           tuneGrid = expand.grid(alpha = 0, lambda = lambda))
```

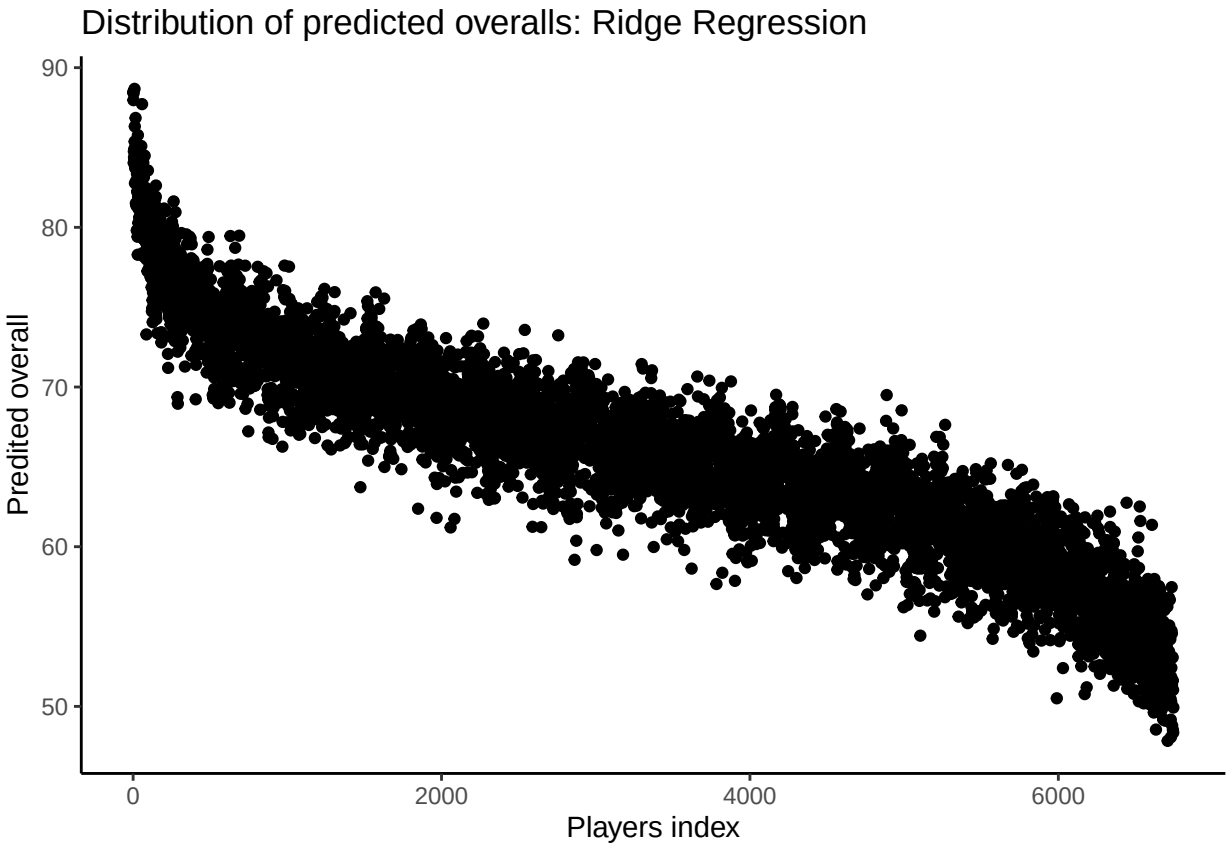
```r
#Show the results.
ridge_results<-Ridge_Model
#Plot the results - RMSE vs. Lambdas.
ggplot(Ridge_Model,aes(x=lambda,y=Ridge_Model$results$RMSE)) + geom_point(size=2) +
       ggtitle("Ridge regression Model:") + xlab("Lambdas")
```



After we made the predictions for the **Ridge Regression** model, we elaborated a plot in order to visualize the distribution of predicted overalls for all the players in the training set.

```r
#We are going to create a new data set, use predict() function, add our model and evaluate in the test
#This will create predictions for our Ridge Regression model.
Prediction_Ridge<-predict(Ridge_Model, validation,interval="prediction")
```

```r
#We are going to create a new data set, use cbind() function and generate an outcome with test set and
Outcome_Ridge<-cbind(validation, Prediction_Ridge)

#Plot the predictions - this one will show us the distribution of predicted overalls.
ggplot(Outcome_Ridge,aes(x=1:6746,y=Prediction_Ridge)) +geom_point() +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
  ggtitle("Distribution of predicted overalls: Ridge Regression") + ylab("Predited overall")+ xlab("Play
```



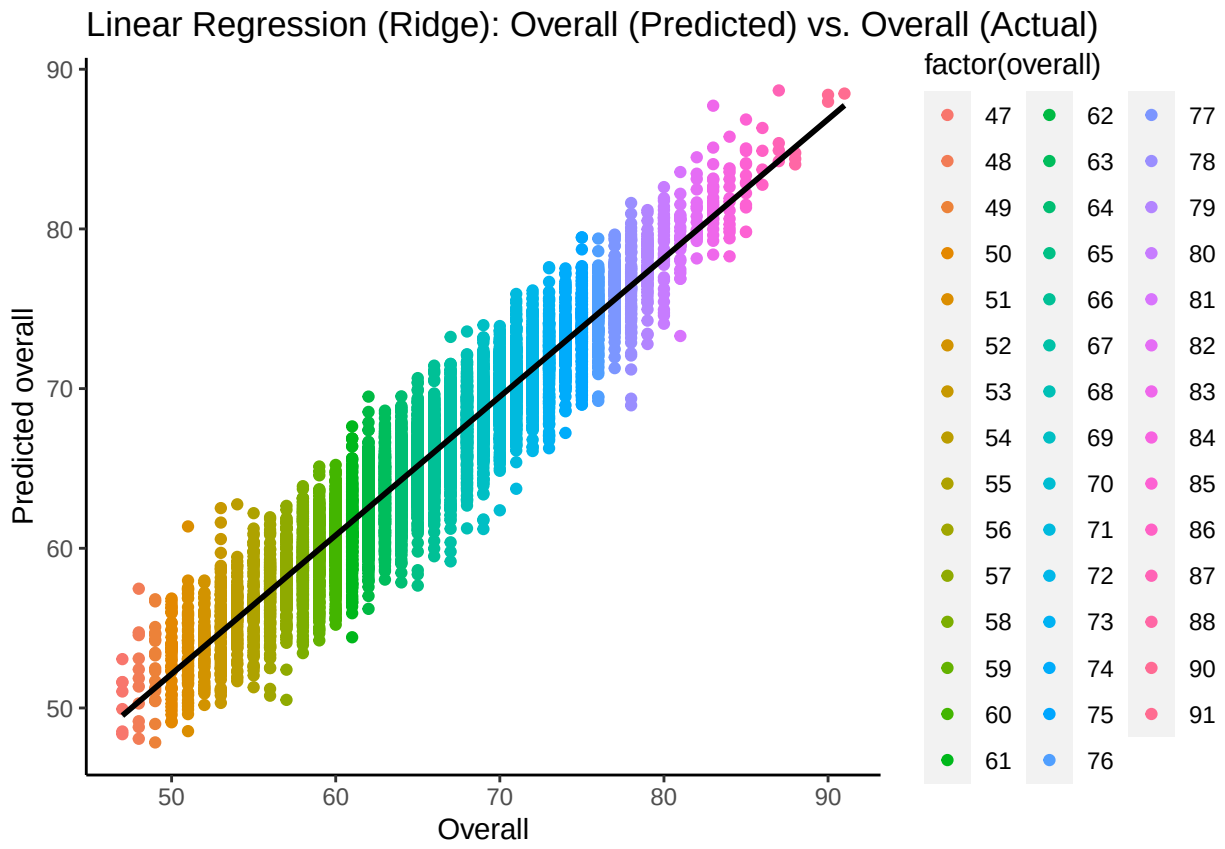
In the following plot, we can see a linear regression graph for our **Ridge Regression** model. This plot show us the relationship between the predicted overall and the actual overall. In this plot we can see a high correlation/relationship with a value of **0.8685** in this two variables.

```r
#Here we are going to create, evaluated and plot a linear regression model (Ridge) for the following tw
#overall(actual values) and fit (predicted values).
lm(Prediction_Ridge~overall,data=Outcome_Ridge)
```

```
##
## Call:
## lm(formula = Prediction_Ridge ~ overall, data = Outcome_Ridge)
##
## Coefficients:
## (Intercept)      overall
##      8.7028       0.8685
```

```
ggplot(Outcome_Ridge, aes(x = overall, y = Prediction_Ridge,color = factor(overall))) +
  geom_point() +
  geom_smooth(method = "lm", col = "black") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))+
        ggtitle("Linear Regression (Ridge): Overall (Predicted) vs. Overall (Actual)") + xlab("Overall")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



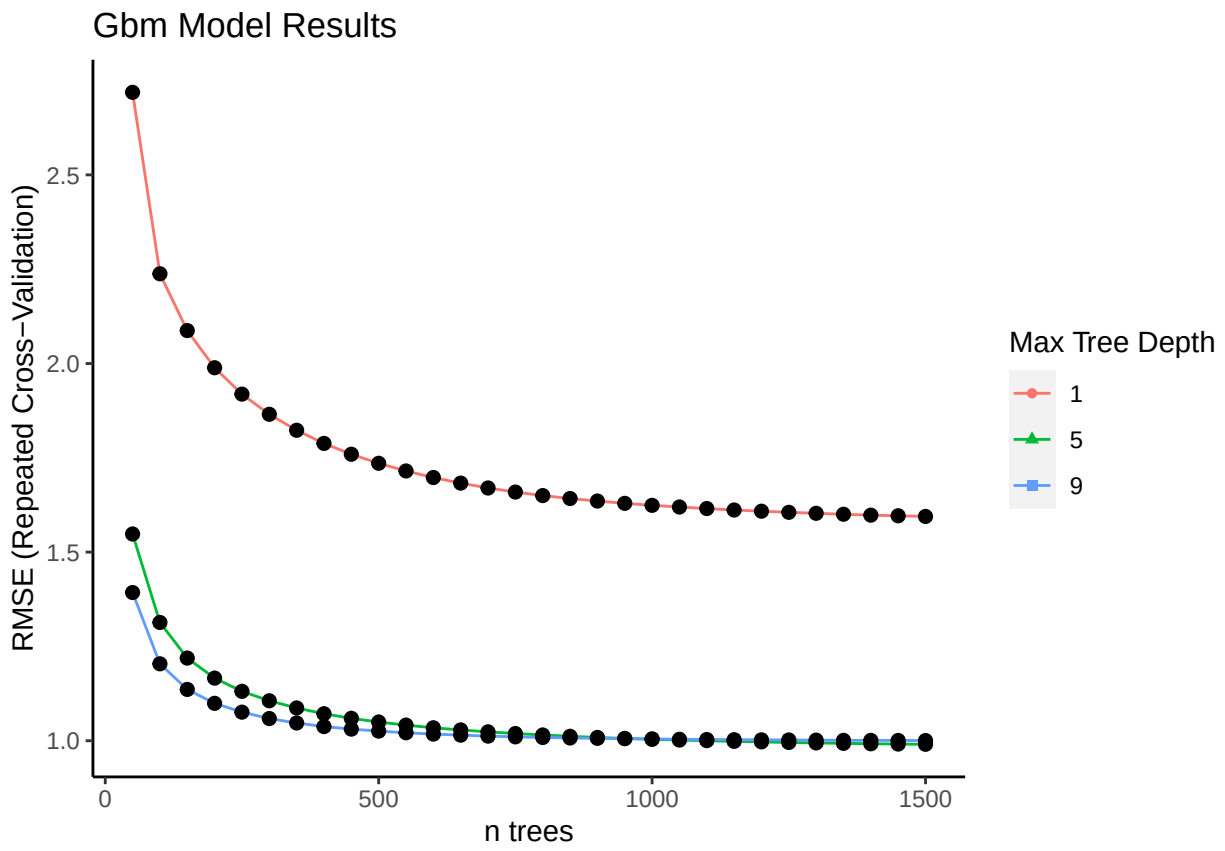

```
#We are going to evaluate our Ridge Regression model with the RMSE as our metric.
Ridge_RMSE<-RMSE(validation$overall,Prediction_Ridge)
#Show the RMSE.
(Ridge_RMSE) %>% knitr::kable()
```

x
2.254696

For the **GBM (Gradient Boosting Machine)** model, we have done predictions based on the method "gbm" and created plots in order to visualize the results. This model combines the predictions from multiple decision trees to generate the final predictions. Also, this model can be considered as one of the hardest for predicting models because much of the parameters must be done correctly. However, can be a great option if it's built in the right form.

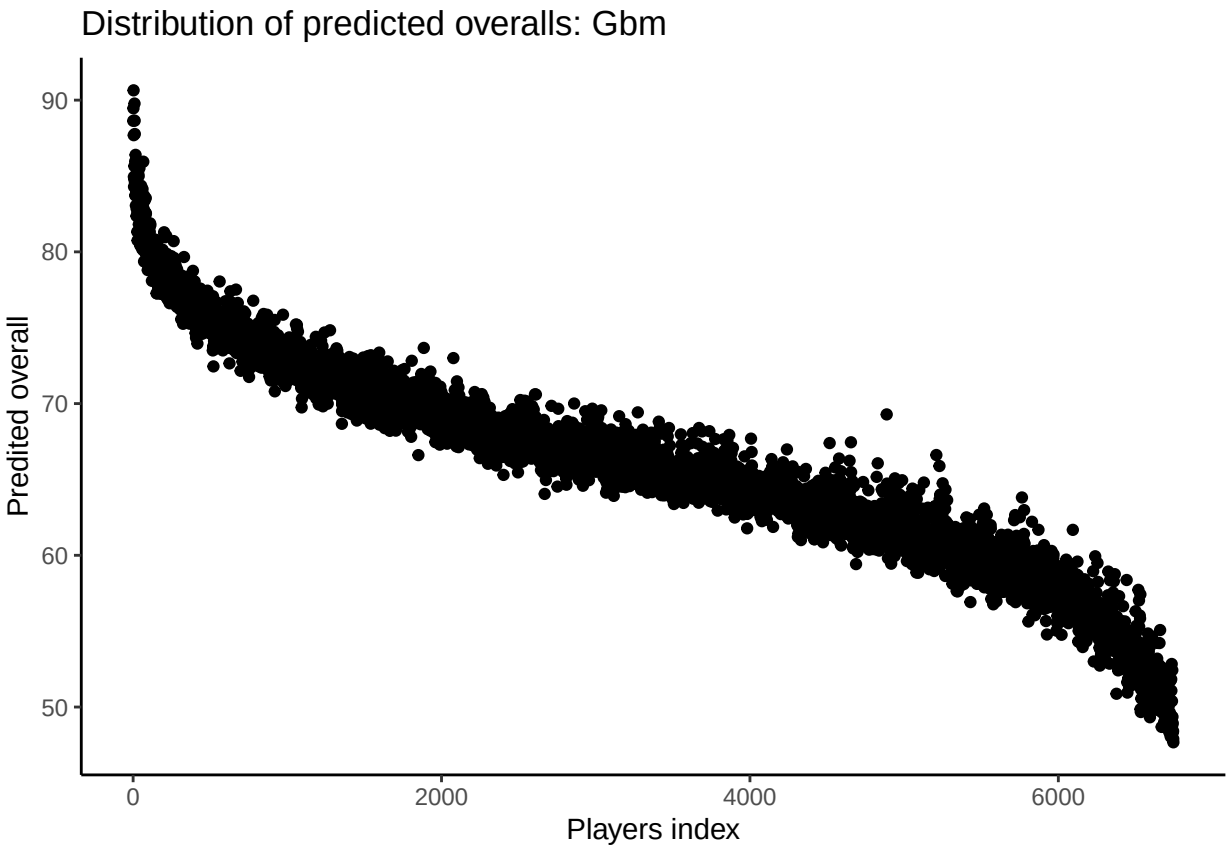
```
#############
# GBM MODEL #
#############

#We are going create a trControl with the best parameters for the Gbm model.
trControlGBM<- trainControl(method = "repeatedcv",
                            number = 10,
                            repeats=10)

#We are going create a Grid with the best parameters for the Gbm model.
GridGBM <-  expand.grid(interaction.depth = c(1, 5, 9),
                        n.trees = (1:30)*50,
                        shrinkage = 0.1,
                        n.minobsinnode = 20)

#We are going to set seed in 825 in order to generate a sequence of random numbers.
set.seed(825)
#We are going to create a Gbm model with the function train().
#Use overall as the variable that will be predicted and use the rest of the variables as contributors f
#Add our train data, train control parameter define above, grid parameter, our metric and method for th
GBM_Model <- train(overall~  age + international_reputation+ weak_foot + skill_moves + pace + shooting +
                   defending + physic + attacking_crossing + attacking_finishing + attacking_heading_ac
                   movement_sprint_speed + movement_agility + movement_reactions + movement_balance + po
                   power_strength + power_long_shots + mentality_aggression + mentality_interceptions +
                   mentality_penalties + defending_standing_tackle + defending_sliding_tackle+ skill_cu
                  + skill_ball_control, data = FIFA,
                   method = "gbm",
                   trControl = trControlGBM,
                   ## This last option is actually one
                   ## for gbm() that passes through
                   verbose = FALSE,
                   tuneGrid = GridGBM)
```

```
#Show the results.
gbm_model<-GBM_Model
#Plot the results - RMSE vs. n trees.
ggplot(GBM_Model,aes(n.trees,GBM_Model$results$RMSE)) + geom_point(size=2) +
  theme(axis.line = element_line(colour = "black"),
        panel.background = element_rect(fill = "white")) +
        ggtitle("Gbm Model Results") + xlab("n trees")
```



After we made the predictions for the **(GBM) Gradient Boosting Machine** model, we elaborated a plot in order to visualize the distribution of predicted overalls for all the players in the training set.

```
#We are going to create a new data set, use predict() function, add our model and evaluate in the test
#This will create predictions for our Gbm model.
Prediction_gbm<-predict(GBM_Model, validation)

#We are going to create a new data set, use cbind() function and generate an outcome with test set and
Outcome_gbm<-cbind(validation, Prediction_gbm)

#Plot the predictions - this one will show us the distribution of predicted overalls.
ggplot(Outcome_gbm,aes(x=1:6746,y=Prediction_gbm)) +geom_point() +
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black")) +
  ggtitle("Distribution of predicted overalls: Gbm") + ylab("Predited overall")+ xlab("Players index")
```



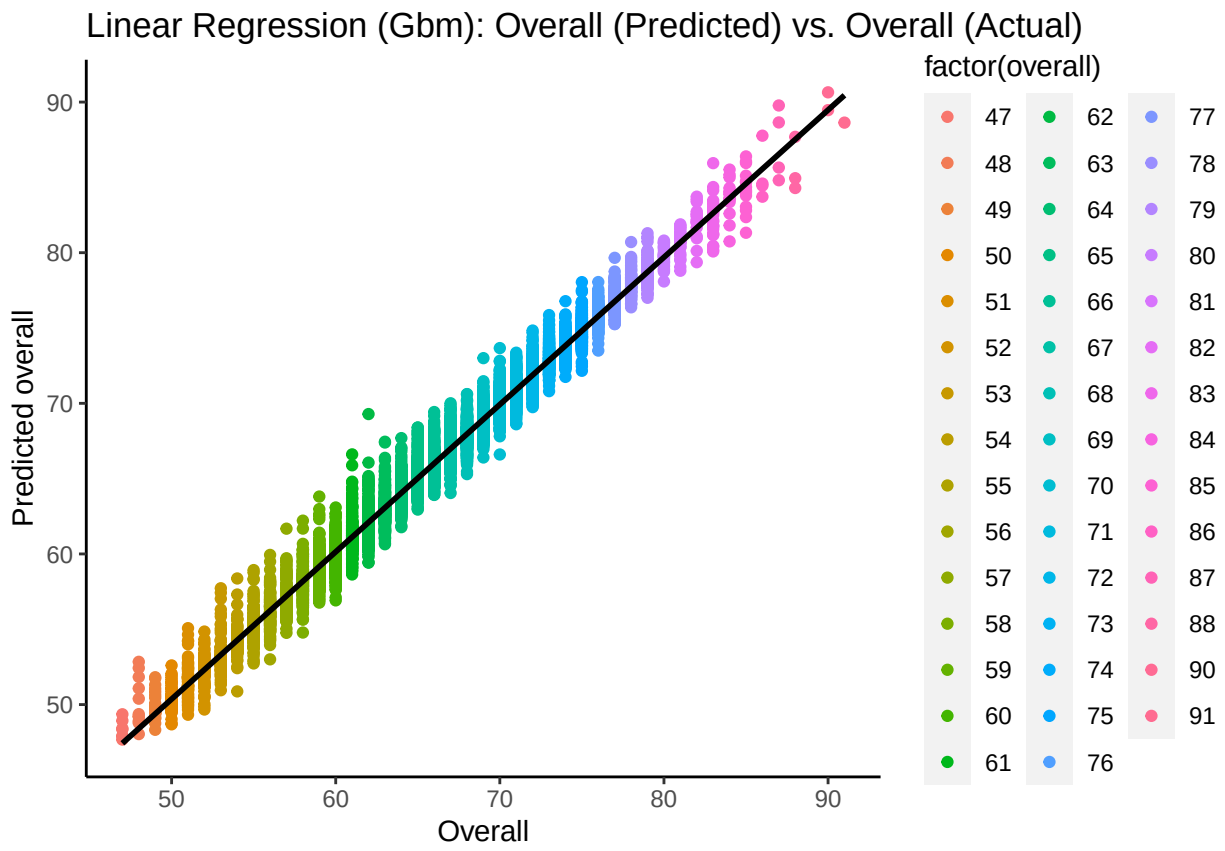
In the following plot, we can see a linear regression graph for our **(GBM) Gradient Boosting Machine** model. This plot show us the relationship between the predicted overall and the actual overall. In this plot we can see an almost perfect correlation/relationship with a value of **0.9778** in this two variables.

```
#Here we are going to create, evaluated and plot a linear regression model (Gbm) for the following two
#overall(actual values) and fit (predicted values).
lm(Prediction_gbm~overall,data=Outcome_gbm)
```

```
##
## Call:
## lm(formula = Prediction_gbm ~ overall, data = Outcome_gbm)
##
## Coefficients:
## (Intercept)      overall
##      1.4742       0.9778
```

```
ggplot(Outcome_gbm, aes(x = overall, y = Prediction_gbm,color = factor(overall))) +
  geom_point() +
  geom_smooth(method = "lm", col = "black") +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank(), axis.line = element_line(colour = "black"))+
        ggtitle("Linear Regression (Gbm): Overall (Predicted) vs. Overall (Actual)") + xlab("Overall")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
#We are going to evaluate our Gbm model with the RMSE as our metric.
GBM_RMSE<-RMSE(validation$overall,Prediction_gbm)
#Show the RMSE.
(GBM_RMSE) %>% knitr::kable()
```

x
0.9756678

Results and interpretation for the models:

- The "Linear regression" model evaluated in the validation set gave us a RMSE of **2.24**. The performance of this model can be considered poor and not very precise.
- The "Random Forest" model evaluated in the validation set gave us a RMSE of **1.14**. The performance of this model can be considered very good and a lot more precise in comparison to the linear regression model.
- The "K-nearest neighbors" model evaluated in the validation set gave us a RMSE of **1.65**. The performance of this model can be considered good and more precise than the linear regression model. However, the random forest model realized a better performance than this model.
- The "Ridge regression" model evaluated in the validation set gave us a RMSE of **2.25**. The performance of this model can be considered poor and not very precise, actually this model has the worst RMSE of all the models.

- The "Gradient Boosting Machine" model evaluated in the validation set gave us a RMSE of **0.98**. The performance of this model can be considered great and very precise. In fact, this model has surpass by a lot the other models RMSE results.

```
#We are going to create a new data set, add models name and RMSE results of all the models into a data
RESULTS<- data.frame(
  Model = c("Linear Regression","Random Forest","Knn", "Ridge Regression","Gbm"),
  RMSE= c( Linear_regression_RMSE, RF_RMSE, Knn_RMSE, Ridge_RMSE, GBM_RMSE))

RESULTS %>% knitr::kable()
```

Model	RMSE
Linear Regression	2.2406356
Random Forest	1.1409596
Knn	1.6526533
Ridge Regression	2.2546961
Gbm	0.9756678

Conclusions:

-Based on the results obtained, the **(GBM) Gradient Boosting Machine model**, which was more tuned and worked, achieved a lower RMSE than the others models. In consequence, we can conclude that, the more tuned the model is, the more exact results will be.

-After comparing the linear regression graphs for all the models, it was possible to observe that the more accurate the prediction is, the more **correlation/relationship** will exist for the players overall.

-After analyzing the results, it is possible to conclude that the **"GBM (Gradient Boosting Machine)" model** was the one that gave us the lowest RMSE, which allowed us to determine that this model was the most accurate and precise in terms of predictions.

-Finally, in the process of creating predictions for the FIFA 2022 players overall, a limitation was found, this limitation was a column named **"defending marking"** that had NA VALUES, this column was very important because it could have contribute a lot in the process of building and evaluating the predictions. However, predictions were good despite the fact that this limitation make our models be less accurate. For future work purposes, I recommend to fill the columns with NA values and omit irrelevant information like, for example: players traits, body type, real face, etc.

Bibliography:

-Panchotia, R. (2020, August 5). Predictive Modelling Using Linear Regression - The Startup. Medium. https://medium.com/swlh/predictive-modelling-using-linear-regression-e0e399dc4745#:%7E:text=Linear%20regression%20is%20one%20of,given%20predictor%20variable(s).

-Harrison, O. (2019, July 14). Machine Learning Basics with the K-Nearest Neighbors Algorithm. Medium. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761#:%7E:text=KNN%20works%20by%20finding%20the,in%20the%20case%20of%20regression).

-Donges, N. (2020, September 3). A complete guide to the random forest algorithm. Built In. https://builtin.com/data-science/random-forest-algorithm

-S. (2021, February 6). Ridge Regression: Simple Definition. Statistics How To. https://www.statisticshowto.com/ridge-regression/

-Irizarry, R. A. (2021, 20 february). Introduction to Data Science. https://rafalab.github.io/dsbook/. https://rafalab.github.io/dsbook/

-Murphy, R. (2019, September 12). FIFA player ratings explained: How are the card number & stats decided? Goal.Com. https://www.goal.com/en-ae/news/fifa-player-ratings-explained-how-are-the-card-number-stats/1hszd2fgr7wgf1n2b2yjdpgynu

–zev@zevross.com. (2018, October 2). Predictive modeling and machine learning in R with the caret package. Technical Tidbits From Spatial Analysis & Data Science. http://zevross.com/blog/2017/09/19/predictive-modeling-and-machine-learning-in-r-with-the-caret-package/