Domain: Sport

Datasets:

1. FIFA 18 dataset: https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset
2. FIFA 19 dataset: https://www.kaggle.com/karangadiya/fifa19
3. FIFA 20 dataset: https://www.kaggle.com/stefanoleone992/fifa-20-complete-player- dataset

Problem Statement:

Trying to understand the players population in the FIFA games. To understand the features/variable contributing towards the overall of the player and build a predictive model that could estimate of the overall of the player with accuracy.

Also, build a model to predict the value of the player with accuracy and least numbers of variables.

And build a model to recommend the most similar players to a particular player.

Three datasets are being used here, all three for different reasons.

- FIFA 19 dataset is being used for exploration, visualization, build predictive model and the recommendation model.
- FIFA 18 data set is used only as training the data along with FIFA 19 data set.
- FIFA 20 dataset is only being used as the test dataset.

I have tried to make the project modular as such the data cleaning for each dataset is done in a different script and are later sourced to the main script. In the same way all the predictive models built are saved and then loaded to the main script. Scripts for the models are presented separately.

1. # Initial Data Cleaning (Using Excel & R)
    a. Removing unnecessary fields (Using R)

```
## Removing unnecessary features from the data set
player_db18 <- dplyr::select(player_db18, -c(Sr.Nos., Special, CAM, CB, CDM, CF, CM, LAM, LB, LCB, LCM, LDM, LF, LM,
                                             LS, LW, LWB, RAM, RB, RCB, RCM, RDM, RF, RM, RS, RW, RWB, ST))
```

```
## Removing unnecessary features from the dataset
player_db19 <- dplyr::select(player_db19, -c(i..Sr.Nos., Special, LS, ST, RS, LW, LF, CF, RF, RW, RW, LAM, CAM, RAM, LM,
                                             LCM, CM, RCM, RM, LWB, LDM, CDM, RDM, RWB, LB, LCB, CB, RCB, RB))
```

```
## Removing unnecessary features from the dataset
player_db20 <- dplyr::select(player_db20, -c(long_name, dob, Height, Weight, Prefered.Position, Preferred.Foot,
                                             Weak.Foot, skill.Moves, Work.Rate, Body.Type, Real.Face, Release.Clause,
                                             player_tags, Jersey.Number, Loaned.From, Joined, Contract.Valid.Until,
                                             nation_position, nation_jersey_number, pace, shooting, passing, dribbling,
                                             defending, physic, gk_diving, gk_handling, gk_kicking, gk_reflexes,
                                             gk_speed, gk_positioning, player_traits, ls, st, rs, lw, lf, cf, rf, rw,
                                             lam, cam, ram, lm, lcm, cm, rcm, rm, lwb, ldm, cdm, rdm, rwb, lb, lcb, cb,
                                             rcb, rb))
```

    b. Simplifying the position into Goal Keeper (GK), Right Back (RB), Centre Back (CB), Left Back (LB), Defensive Midfielder (DM), Centre Midfielders (CM), Wingers & Forwards. (Using Excel)

CAM <- CM	LCB <- CB	RAM <- CM	GK <- GK	LS <- FORWARD	RS <- FORWARD
CB <- CB	LCM <- CM	RB <- RB	LAM <- CM	LW <- WINGER	ST <- FORWARD
CDM <- CD	LDM <- DM	RCB <- CB	LB <- LB	LWB <- LB	
CF <- FORWARD	LF <- FORWARD	RCM <- CM	RW <- WINGER	RWB <- RB	
CM <- CM	LM <- WINGER	RDM <- DM	RF <- FORWARD	RM <- WINGER	

    c. Dealing with NAs. (Using R)

```
## Dealing NAs
player_db18[is.na(player_db18)] <- 0
```

```
## Dealing with NAs
player_db20[is.na(player_db20)] <- 0
```

```
## Dealing NAs
player_db19$Value[is.na(player_db19$Value)] <- 0
player_db19$Wage[is.na(player_db19$Wage)] <- 0
player_db19$International.Reputation[is.na(player_db19$International.Reputation)] <- 0
player_db19$Weak.Foot[is.na(player_db19$Weak.Foot)] <- 0
player_db19$Skill.Moves[is.na(player_db19$Skill.Moves)] <- 0
player_db19$Jersey.Number[is.na(player_db19$Jersey.Number)] <- 0
player_db19$Crossing[is.na(player_db19$Crossing)] <- 0
player_db19$Finishing[is.na(player_db19$Finishing)] <- 0
player_db19$HeadingAccuracy[is.na(player_db19$HeadingAccuracy)] <- 0
player_db19$ShortPassing[is.na(player_db19$ShortPassing)] <- 0
player_db19$Volleys[is.na(player_db19$Volleys)] <- 0
player_db19$Dribbling[is.na(player_db19$Dribbling)] <- 0
player_db19$Curve[is.na(player_db19$Curve)] <- 0
player_db19$FKAccuracy[is.na(player_db19$FKAccuracy)] <- 0
player_db19$LongPassing[is.na(player_db19$LongPassing)] <- 0
player_db19$BallControl[is.na(player_db19$BallControl)] <- 0
player_db19$Acceleration[is.na(player_db19$Acceleration)] <- 0
player_db19$SprintSpeed[is.na(player_db19$SprintSpeed)] <- 0
player_db19$Agility[is.na(player_db19$Agility)] <- 0
player_db19$Reactions[is.na(player_db19$Reactions)] <- 0
player_db19$Balance[is.na(player_db19$Balance)] <- 0
player_db19$ShotPower[is.na(player_db19$ShotPower)] <- 0
player_db19$Jumping[is.na(player_db19$Jumping)] <- 0
player_db19$Stamina[is.na(player_db19$Stamina)] <- 0
player_db19$Strength[is.na(player_db19$Strength)] <- 0
player_db19$LongShots[is.na(player_db19$LongShots)] <- 0
player_db19$Aggression[is.na(player_db19$Aggression)] <- 0
player_db19$Interceptions[is.na(player_db19$Interceptions)] <- 0
player_db19$Positioning[is.na(player_db19$Positioning)] <- 0
player_db19$Vision[is.na(player_db19$Vision)] <- 0
player_db19$Penalties[is.na(player_db19$Penalties)] <- 0
player_db19$Composure[is.na(player_db19$Composure)] <- 0
player_db19$Marking[is.na(player_db19$Marking)] <- 0
player_db19$StandingTackle[is.na(player_db19$StandingTackle)] <- 0
player_db19$SlidingTackle[is.na(player_db19$SlidingTackle)] <- 0
player_db19$GKDiving[is.na(player_db19$GKDiving)] <- 0
player_db19$GKHandling[is.na(player_db19$GKHandling)] <- 0
player_db19$GKKicking[is.na(player_db19$GKKicking)] <- 0
player_db19$GKPositioning[is.na(player_db19$GKPositioning)] <- 0
player_db19$GKReflexes[is.na(player_db19$GKReflexes)] <- 0
player_db19$Release.Clause[is.na(player_db19$Release.Clause)] <- 0
```

d. Changing joined date and contract valid date to time stamps. (Using R)

```
## Adding date stamps to the joined and contract end date
player_db19$JoinedDate <- mdy(player_db19$Joined)
head(player_db19$JoinedDate)
player_db19$ContractEndDate  <- as.Date(player_db19$Contract.Valid.Until, format = c("%Y"))
head(player_db19$ContractEndDate)
player_db19 <- dplyr::select(player_db19, -c(Joined, Contract.Valid.Until))
player_db19$JoinedDate[is.na(player_db19$JoinedDate)] <- "2010-01-01"
player_db19$ContractEndDate[is.na(player_db19$ContractEndDate)] <- "2020-12-31"
```

```
> ## Adding date stamps to the joined and contract end date
> player_db19$JoinedDate <- mdy(player_db19$Joined)
> head(player_db19$JoinedDate)
[1] "2004-07-01" "2018-07-10" "2017-08-03" "2011-07-01" "2015-08-30" "2012-07-01"
> player_db19$ContractEndDate  <- as.Date(player_db19$Contract.Valid.Until, format = c("%Y"))
> head(player_db19$ContractEndDate)
[1] "2021-06-13" "2022-06-13" "2022-06-13" "2020-06-13" "2023-06-13" "2020-06-13"
> player_db19 <- dplyr::select(player_db19, -c(Joined, Contract.Valid.Until))
> player_db19$JoinedDate[is.na(player_db19$JoinedDate)] <- "2010-01-01"
> player_db19$ContractEndDate[is.na(player_db19$ContractEndDate)] <- "2020-12-31"
>
```

e. Removing units from weight feature. (Using R)

```
## Removing lbs tag from weight factor
head(player_db19$Weight)
player_db19$Wght <- substr(player_db19$Weight, 1, nchar(player_db19$Weight)-3)
head(player_db19$Wght)
player_db19$Weight <- NULL
player_db19$Wght <- as.numeric(player_db19$Wght)
player_db19$Wght <- conv_unit(player_db19$Wght, "lbs", "kg")
player_db19$Wght <- set_units(player_db19$Wght, kg)
sum(is.na(player_db19$Wght))
player_db19$Wght[is.na(player_db19$Wght)] <- 0
```

```
> ## Removing lbs tag from weight factor
> head(player_db19$Weight)
[1] "159lbs" "183lbs" "150lbs" "168lbs" "154lbs" "163lbs"
> player_db19$Wght <- substr(player_db19$Weight, 1, nchar(player_db19$Weight)-3)
> head(player_db19$Wght)
[1] "159" "183" "150" "168" "154" "163"
> player_db19$Weight <- NULL
> player_db19$Wght <- as.numeric(player_db19$Wght)
> player_db19$Wght <- conv_unit(player_db19$Wght, "lbs", "kg")
> player_db19$Wght <- set_units(player_db19$Wght, kg)
> sum(is.na(player_db19$Wght))
[1] 48
> player_db19$Wght[is.na(player_db19$Wght)] <- 0
```

f. Changing Value, Wages and Release Clause from human form to real values. (Using Excel (LEFT and LEN function))

g. Trimming data & changing the data type. (Using R)

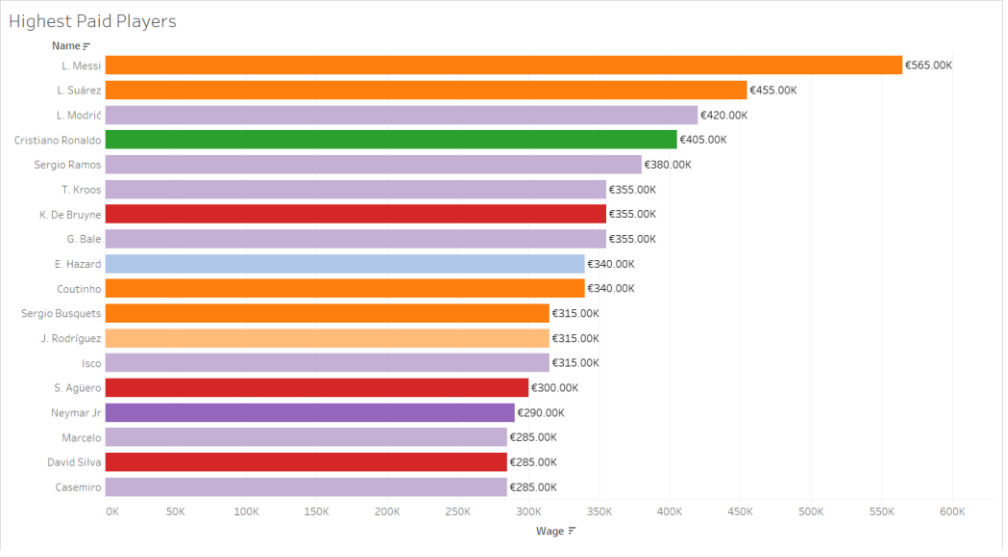
```
player_db18$Crossing <- strtrim(player_db18$Crossing, 2)
player_db18$Finishing <- strtrim(player_db18$Finishing, 2)
player_db18$HeadingAccuracy <- strtrim(player_db18$HeadingAccuracy, 2)
player_db18$ShortPassing <- strtrim(player_db18$ShortPassing, 2)
player_db18$Volleys <- strtrim(player_db18$Volleys, 2)
player_db18$Dribbling <- strtrim(player_db18$Dribbling, 2)
player_db18$Curve <- strtrim(player_db18$Curve, 2)
player_db18$FKAccuracy <- strtrim(player_db18$FKAccuracy, 2)
player_db18$LongPassing <- strtrim(player_db18$LongPassing, 2)
player_db18$BallControl <- strtrim(player_db18$BallControl, 2)
player_db18$Acceleration <- strtrim(player_db18$Acceleration, 2)
player_db18$SprintSpeed <- strtrim(player_db18$SprintSpeed, 2)
player_db18$Agility <- strtrim(player_db18$Agility, 2)
player_db18$Reactions <- strtrim(player_db18$Reactions, 2)
player_db18$Balance <- strtrim(player_db18$Balance, 2)
player_db18$ShotPower <- strtrim(player_db18$ShotPower, 2)
player_db18$Jumping <- strtrim(player_db18$Jumping, 2)
player_db18$Stamina <- strtrim(player_db18$Stamina, 2)
player_db18$Strength <- strtrim(player_db18$Strength, 2)
player_db18$LongShots <- strtrim(player_db18$LongShots, 2)
player_db18$Aggression <- strtrim(player_db18$Aggression, 2)
player_db18$Interceptions <- strtrim(player_db18$Interceptions, 2)
player_db18$Positioning <- strtrim(player_db18$Positioning, 2)
player_db18$Vision <- strtrim(player_db18$Vision, 2)
player_db18$Penalties <- strtrim(player_db18$Penalties, 2)
player_db18$Composure <- strtrim(player_db18$Composure, 2)
player_db18$Marking <- strtrim(player_db18$Marking, 2)
player_db18$StandingTackle <- strtrim(player_db18$StandingTackle, 2)
player_db18$SlidingTackle <- strtrim(player_db18$SlidingTackle, 2)
player_db18$GKDiving <- strtrim(player_db18$GKDiving, 2)
player_db18$GKHandling <- strtrim(player_db18$GKHandling, 2)
player_db18$GKKicking <- strtrim(player_db18$GKKicking, 2)
player_db18$GKPositioning <- strtrim(player_db18$GKPositioning, 2)
player_db18$GKReflexes <- strtrim(player_db18$GKReflexes, 2)
```

```r
player_db18$Crossing <- as.numeric(player_db18$Crossing)
player_db18$Finishing <- as.numeric(player_db18$Finishing)
player_db18$HeadingAccuracy <- as.numeric(player_db18$HeadingAccuracy)
player_db18$ShortPassing <- as.numeric(player_db18$ShortPassing)
player_db18$Volleys <- as.numeric(player_db18$Volleys)
player_db18$Dribbling <- as.numeric(player_db18$Dribbling)
player_db18$Curve <- as.numeric(player_db18$Curve)
player_db18$FKAccuracy <- as.numeric(player_db18$FKAccuracy)
player_db18$LongPassing <- as.numeric(player_db18$LongPassing)
player_db18$BallControl <- as.numeric(player_db18$BallControl)
player_db18$Acceleration <- as.numeric(player_db18$Acceleration)
player_db18$SprintSpeed <- as.numeric(player_db18$SprintSpeed)
player_db18$Agility <- as.numeric(player_db18$Agility)
player_db18$Reactions <- as.numeric(player_db18$Reactions)
player_db18$Balance <- as.numeric(player_db18$Balance)
player_db18$ShotPower <- as.numeric(player_db18$ShotPower)
player_db18$Jumping <- as.numeric(player_db18$Jumping)
player_db18$Stamina <- as.numeric(player_db18$Stamina)
player_db18$Strength <- as.numeric(player_db18$Strength)
player_db18$LongShots <- as.numeric(player_db18$LongShots)
player_db18$Aggression <- as.numeric(player_db18$Aggression)
player_db18$Interceptions <- as.numeric(player_db18$Interceptions)
player_db18$Positioning <- as.numeric(player_db18$Positioning)
player_db18$Vision <- as.numeric(player_db18$Vision)
player_db18$Penalties <- as.numeric(player_db18$Penalties)
player_db18$Composure <- as.numeric(player_db18$Composure)
player_db18$Marking <- as.numeric(player_db18$Marking)
player_db18$StandingTackle <- as.numeric(player_db18$StandingTackle)
player_db18$SlidingTackle <- as.numeric(player_db18$SlidingTackle)
player_db18$GKDiving <- as.numeric(player_db18$GKDiving)
player_db18$GKHandling <- as.numeric(player_db18$GKHandling)
player_db18$GKKicking <- as.numeric(player_db18$GKKicking)
player_db18$GKPositioning <- as.numeric(player_db18$GKPositioning)
player_db18$GKReflexes <- as.numeric(player_db18$GKReflexes)
player_db18$International.Reputation <- as.numeric(player_db18$International.Reputation)
```

```r
player_db20$Wage <- as.numeric(player_db20$Wage)
player_db20$Value <- as.numeric(player_db20$Value)
player_db20$Potential <- as.numeric(player_db20$Potential)
player_db20$Overall <- as.numeric(player_db20$Overall)
player_db20$Age <- as.numeric(player_db20$Age)
player_db20$ID <- as.numeric(player_db20$ID)
player_db20$Crossing <- as.numeric(player_db20$Crossing)
player_db20$Finishing <- as.numeric(player_db20$Finishing)
player_db20$HeadingAccuracy <- as.numeric(player_db20$HeadingAccuracy)
player_db20$ShortPassing <- as.numeric(player_db20$ShortPassing)
player_db20$Volleys <- as.numeric(player_db20$Volleys)
player_db20$Dribbling <- as.numeric(player_db20$Dribbling)
player_db20$Curve <- as.numeric(player_db20$Curve)
player_db20$FKAccuracy <- as.numeric(player_db20$FKAccuracy)
player_db20$LongPassing <- as.numeric(player_db20$LongPassing)
player_db20$BallControl <- as.numeric(player_db20$BallControl)
player_db20$Acceleration <- as.numeric(player_db20$Acceleration)
player_db20$SprintSpeed <- as.numeric(player_db20$SprintSpeed)
player_db20$Agility <- as.numeric(player_db20$Agility)
player_db20$Reactions <- as.numeric(player_db20$Reactions)
player_db20$Balance <- as.numeric(player_db20$Balance)
player_db20$ShotPower <- as.numeric(player_db20$ShotPower)
player_db20$Jumping <- as.numeric(player_db20$Jumping)
player_db20$Stamina <- as.numeric(player_db20$Stamina)
player_db20$Strength <- as.numeric(player_db20$Strength)
player_db20$LongShots <- as.numeric(player_db20$LongShots)
player_db20$Aggression <- as.numeric(player_db20$Aggression)
player_db20$Interceptions <- as.numeric(player_db20$Interceptions)
player_db20$Positioning <- as.numeric(player_db20$Positioning)
player_db20$Vision <- as.numeric(player_db20$Vision)
player_db20$Penalties <- as.numeric(player_db20$Penalties)
player_db20$Composure <- as.numeric(player_db20$Composure)
player_db20$Marking <- as.numeric(player_db20$Marking)
player_db20$StandingTackle <- as.numeric(player_db20$StandingTackle)
player_db20$SlidingTackle <- as.numeric(player_db20$SlidingTackle)
player_db20$GKDiving <- as.numeric(player_db20$GKDiving)
player_db20$GKHandling <- as.numeric(player_db20$GKHandling)
player_db20$GKKicking <- as.numeric(player_db20$GKKicking)
player_db20$GKPositioning <- as.numeric(player_db20$GKPositioning)
player_db20$GKReflexes <- as.numeric(player_db20$GKReflexes)
player_db20$Position <- as.factor(player_db20$Position)
player_db20$International.Reputation <- as.numeric(player_db20$International.Reputation)
```

2. Exploratory Analysis, answering various questions to learn the characteristics of the population using cleaned FIFA 19 dataset.

    a. Highest Paid Players



Top 18 highest paid players are shown in the graph & and the colors are defined to show the club of the player.

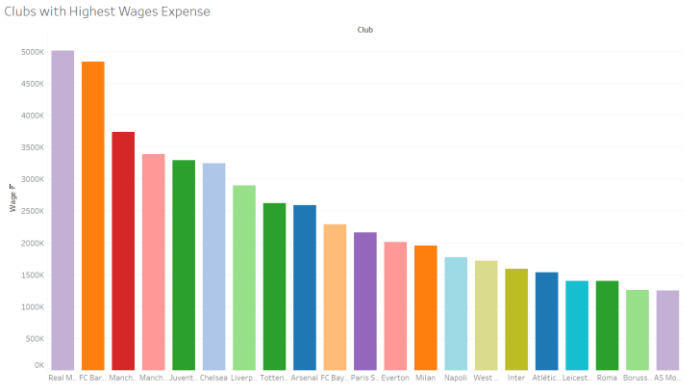
Lionel Messi is the highest paid player of the game, followed by Luis Suarez and Luka Modric.

Next, we see Cristiano Ronaldo, now it might seem weird as the Ronaldo's overall is same as Messi's only different being age, they both also play roughly the same position. There might be case to be made that the Barcelona the club for which Messi plays is generally considered to be richer that Juventus.

But I think age is the bigger factor here, I will be analyzing it further.

Also there seems to be a domination of orange (Barcelona) and pink (Real Madrid), so I would like to analyze the total wage expenditure of these clubs.

b. Clubs with Highest Wages Expenditure

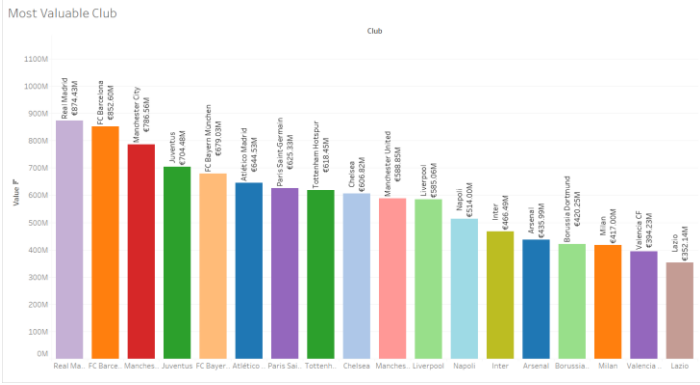


As we saw in the Highest Paid Players graph, it was visible that most of the players there were from Barcelona and Real Madrid. From our current visualization it is again clear that these clubs spend highest for wages.

If we ignore these two clubs from the graph, we can see that there is an incremental increase in wage expenditure. And there seems to be a lot higher increase from third to second place.

Maybe it could have something to do with the value of the player these clubs have. As it can naturally be assumed that the higher valued players would demand higher wages.

c. Most Valuable Club



Value of the player here is determined by the summing the values of all their player.

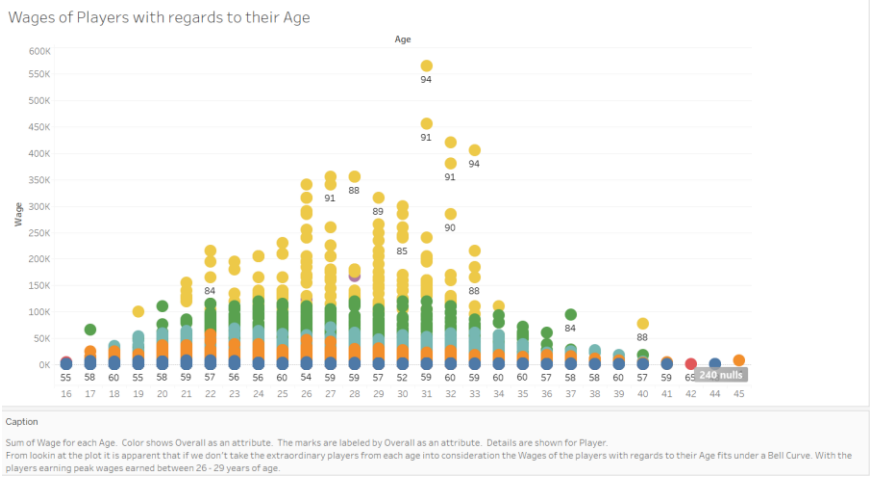
Here it is abundantly clear why the wages of Real Madrid and Barcelona were high; they have the most valuable squad in the game.

Also, almost all the clubs here keep the same place as the graph for highest wage expense. Expect that of Manchester United which had the fourth highest wage expenditure but has the tenth most valuable squad. Maybe they lacking in the negotiations or this might be because of their poor performance run in the last half decade or so.

On the other hand, we have FC Bayern who have the fifth most valuable club but are tenth highest in wages. These could be because of the near domination in their national league.

If we think along the same line, it could be argued that the reason of high value squad and high wage expense for Real Madrid and Barcelona could be result of their year on year competition for the national league title and their rivalry. This could be resulting in their near domination of top two spot of their league.

d. Wages of Players with regards to their Age



Wages of all the players distributed by age.

We can see an apparent bell curve in the distribution, there are also outliers apparent.

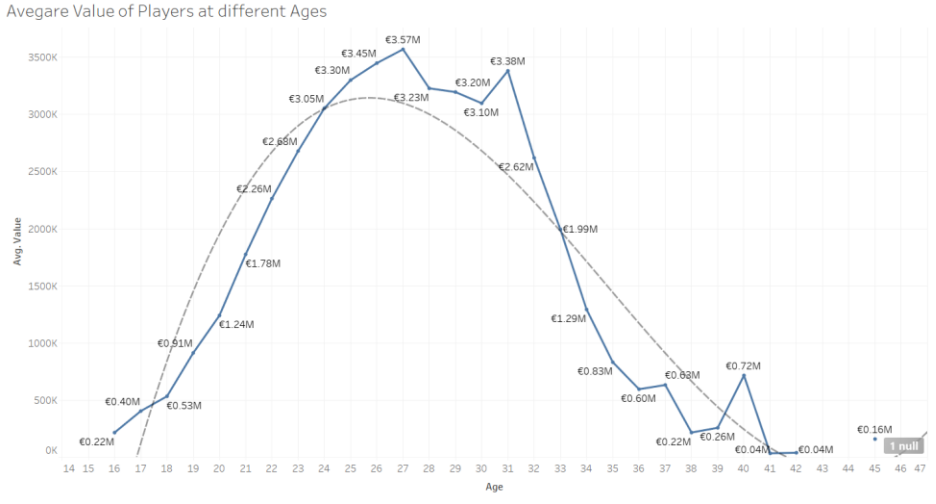
It can be hypothesized that the player earns most when they are bet 25 – 30 years old.
There are of course exceptions to this theory but those can easily be considered outliers.

It is often said that the footballers are at their peak between 27 – 29. That would mean they are growing before that. That could be the reason that for the growth of the wages as the player are reaching 25-27 and once, they reach their peak the wages stagnate and soon after the wages

This shouldn’t be a surprise to anyone as football is a very physical game and the of course as the players grow they learn, gain more experience but once, they have passed their physical they would be very experience but their body would not be able to the same things they would have done I their prime.

This can of course be negated, the best example for this would be Cristiano Ronaldo.
Even at the age of 33 he is still earning more then most players in their prime.
This because he has adapted his playing style as he has grown older, relying less on his pace and more on his understanding of the game and efficient shorter runs.

e. Average Value of Players at different Ages



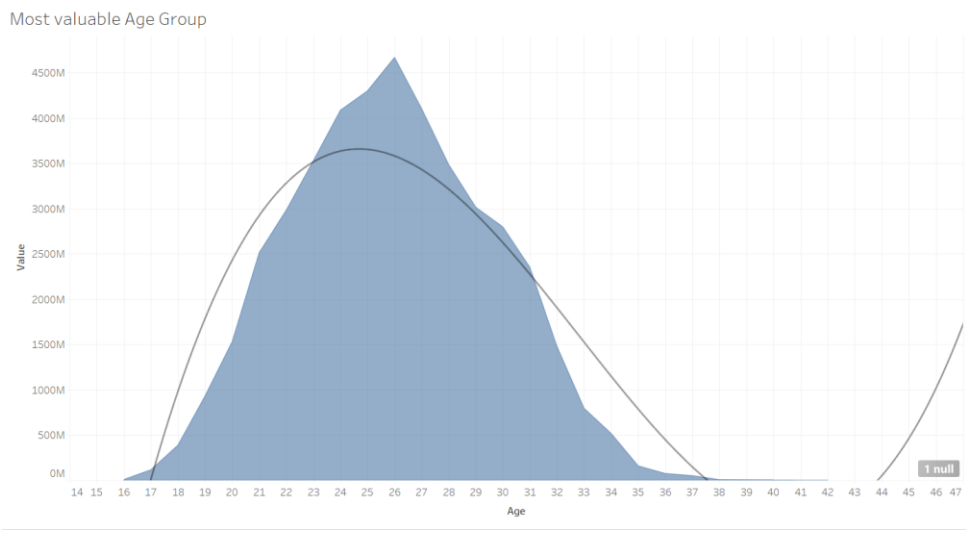
As we discussed earlier players reach their peak around 27-29 years of age. Which is also apparent from this visualization.

We can also see that the players value increase as they grow and reach their peak and then gradually fall off.

It is also apparent that the players in the game are reaching their peak at 27.

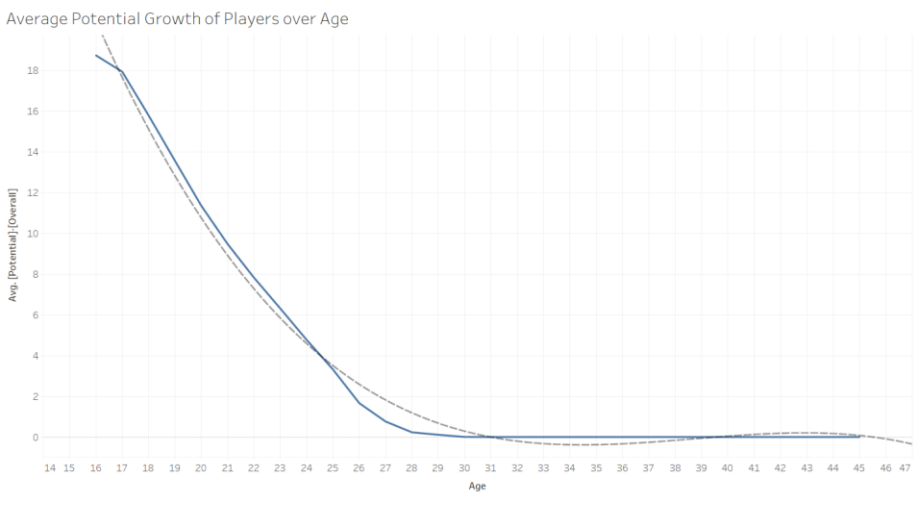
All of this also confirmed by the trend line fitted, which shows a steep increase in value of the player, stabilization for a short while and then a gradual decline.

f. Most valuable Age Group



For this visualization value of all the players in at a particular age is summed.

g. Average Potential Growth of Players over Age

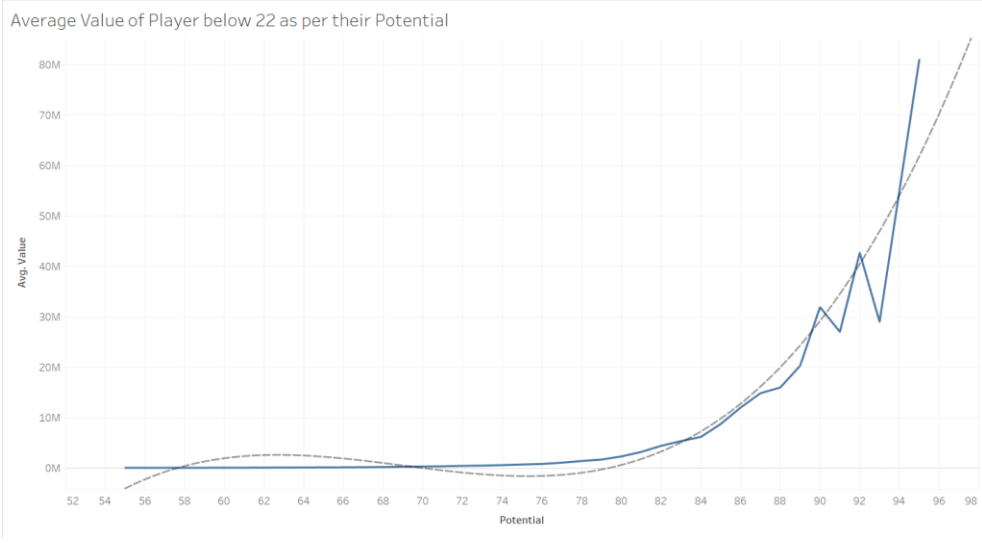


No surprises here the as it is abundantly clear that the younger the player the more growth potential he has.

But that is not why I wanted to visualize this. I wanted to visualize this because it confirms our hypothesis that the players reach their peak at 27 -29.

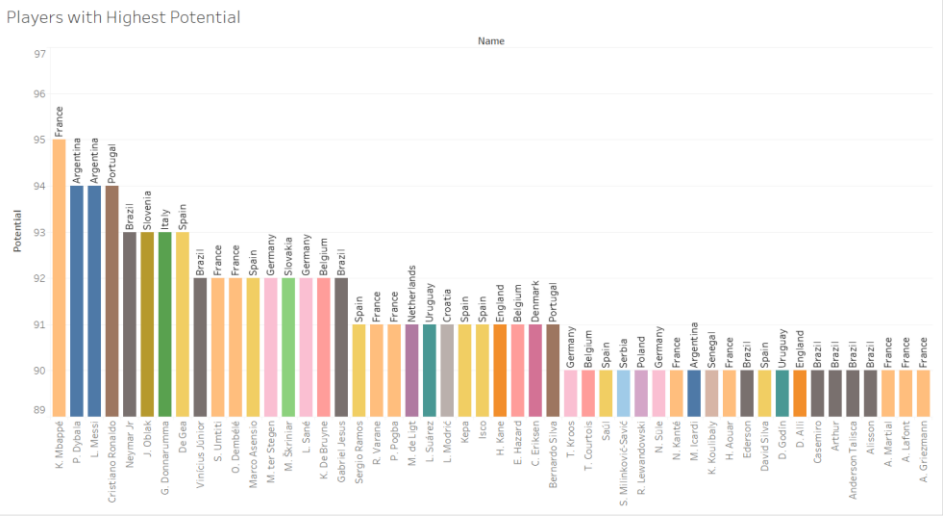
As that is when they stop growing. After that point the player only gathers more and more experience.

h. Average Value of Player below the age of 22 as per their Potential

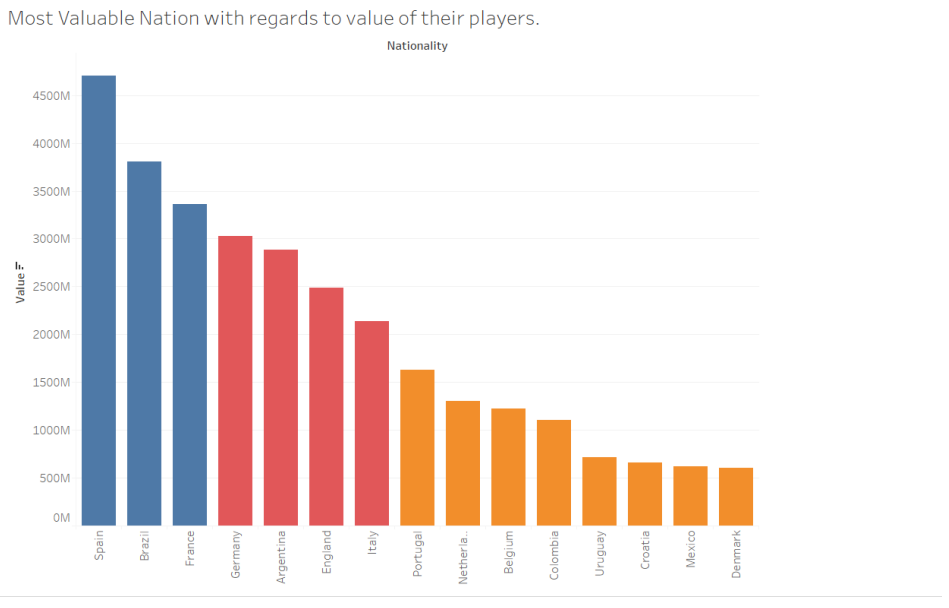


What I am try to visualize here is that if the potential of the player below the age of 22 is ascertained to be above 80 there would be exponential increase in his value.

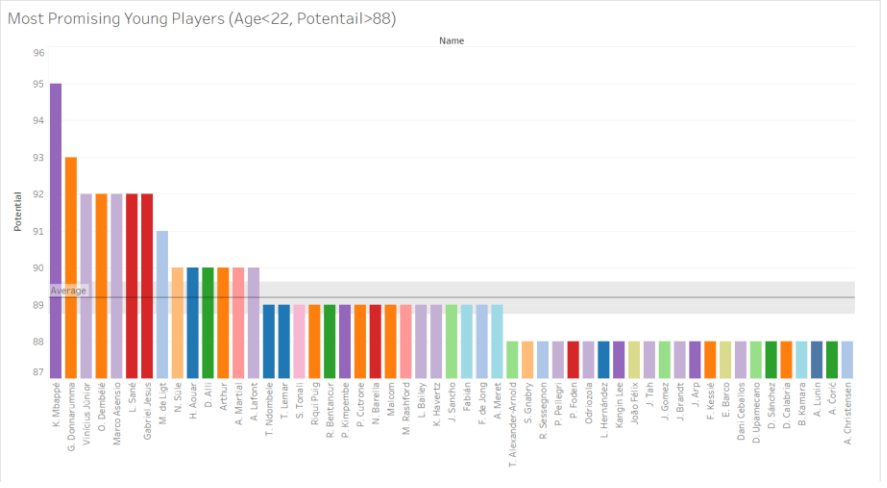
i. Players with Highest Potential



j. Most Valuable Nation with regards to value of their players

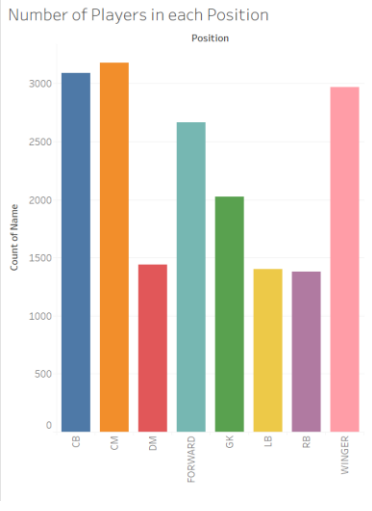


k. Most Promising Young Players (Age<22, Potential>88)

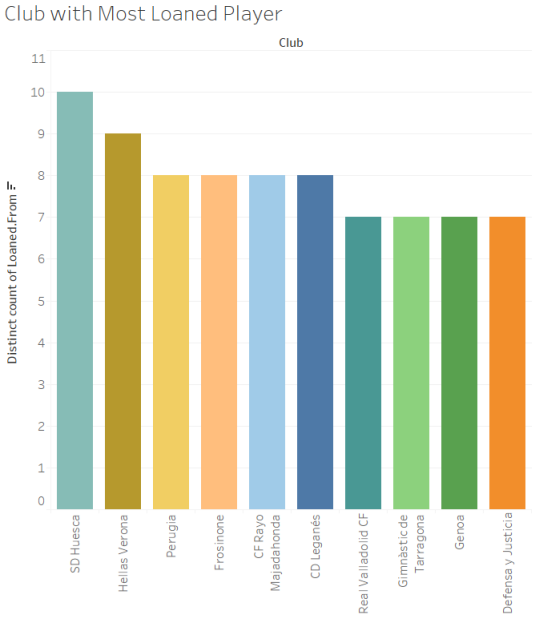


When you are playing career mode in the game who is young and has a high potential.
Here we have all the players below the age of 22 and potential higher than 88.

l. Number of Players in each Position

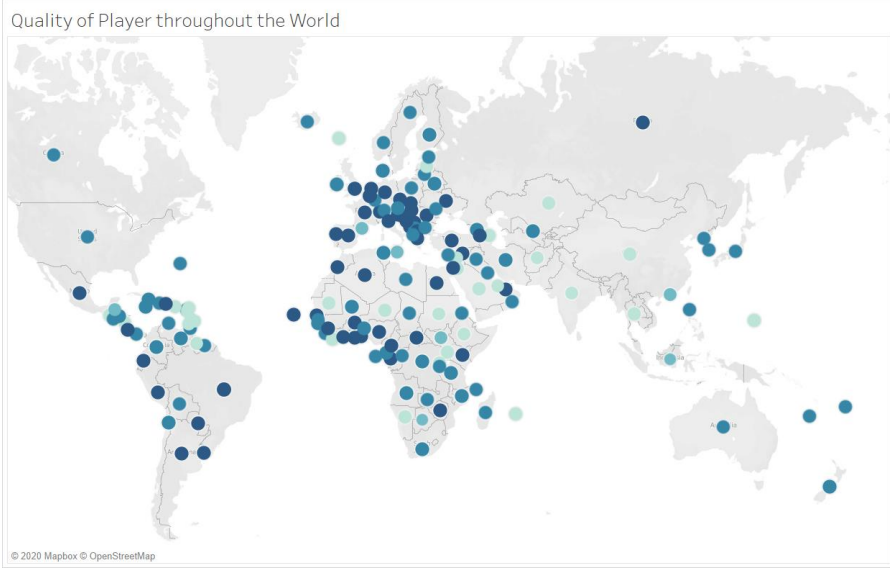


m. Club with Most Loaned Player



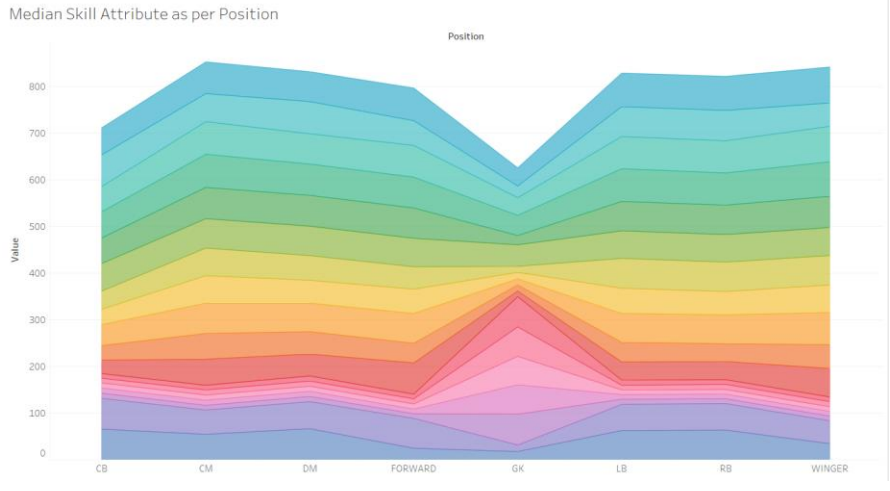
Not all clubs are equal in wealth and as such some prefer to buy the players they need while the others have to loan the players. This graph shows the clubs with most players got on loan.

n. Quality of Player throughout the World



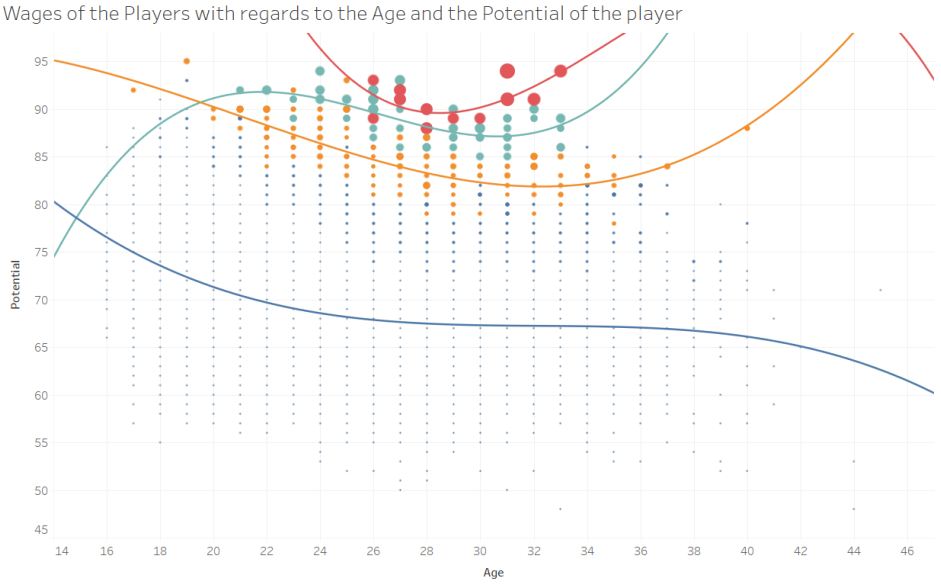
Here, median overall of the player has been taken for every nation. To determine the quality of players each nation have. Darker the point the better the quality of players.

o. Median Skill Attribute as per Position



Medians skills required for each position.

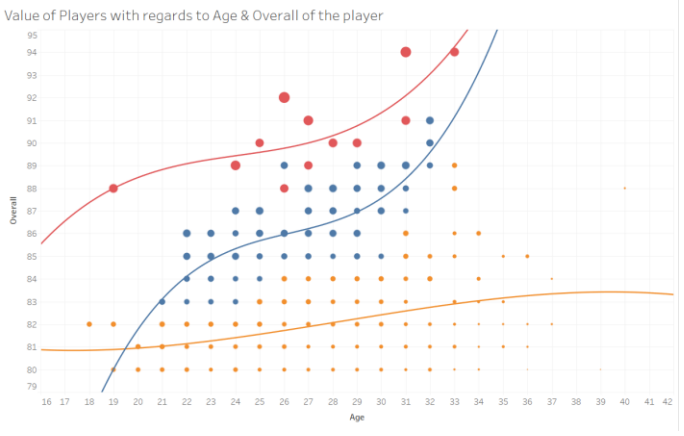
p. Wages of the Players with regards to the Age and the Potential of the player



Visualized is the wages of the players with regards to the age and potential of the player.

We can see four distinct clusters, or career path players could have.

q. Value of Players with regards to Age & Overall of the player



Similar to above this is the value of the players with regards to Overall and Age of the player.

There are three different clusters.
I like to think of them as;

- Average players
- Good players
- Extraordinary players

3. Finally, we get to building various ML Model to predict the Overall & Value of the players.
   I have built multiple models of each type to ascertain the best model for the job. We will start with the model to predict the overall of the players and then move on to models to predict the value of the players. I have used Multi-Linear Regression, Regression Tree & Neural Networks. Let's get started,

   a. Building models to predict the Overall of the Player

      - Multi-Linear Regression Model
        Creating the first model to identify the necessary variable for the model;

```
## Multi-Linear Regression Model for Predicting Overall
lm1 <- lm(Overall ~ Age + Potential + Position + Crossing + Finishing + HeadingAccuracy + ShortPassing +
            Volleys + Dribbling + Curve + FKAccuracy + LongPassing + BallControl + Acceleration + SprintSpeed +
            Agility + Reactions + Balance + ShotPower + Jumping + Stamina + Strength + LongShots + Aggression +
            Interceptions + Positioning + Vision + Penalties + Composure + Marking + StandingTackle +
            SlidingTackle + GKDiving + GKHandling + GKKicking + GKPositioning + GKReflexes, player_db)
summary(lm1)
```

```
> summary(lm1)

Call:
lm(formula = Overall ~ Age + Potential + Position + Crossing +
    Finishing + HeadingAccuracy + ShortPassing + Volleys + Dribbling +
    Curve + FKAccuracy + LongPassing + BallControl + Acceleration +
    SprintSpeed + Agility + Reactions + Balance + ShotPower +
    Jumping + Stamina + Strength + LongShots + Aggression + Interceptions +
    Positioning + Vision + Penalties + Composure + Marking +
    StandingTackle + SlidingTackle + GKDiving + GKHandling +
    GKKicking + GKPositioning + GKReflexes, data = player_db)

Residuals:
     Min       1Q   Median       3Q      Max
-25.5525  -1.1105   0.1368   1.2550   7.7327

Coefficients:
                    Estimate  Std. Error t value            Pr(>|t|)
(Intercept)        8.0514055   0.3487466  23.087 < 0.0000000000000002 ***
Age                0.5079829   0.0038879 130.656 < 0.0000000000000002 ***
Potential          0.5323927   0.0029422 180.949 < 0.0000000000000002 ***
PositionCB       -20.9477528   0.2871575 -72.949 < 0.0000000000000002 ***
PositionCM       -23.2098690   0.2908744 -79.793 < 0.0000000000000002 ***
PositionDM       -23.0397430   0.2926318 -78.733 < 0.0000000000000002 ***
PositionFORWARD  -22.7237981   0.2918280 -77.867 < 0.0000000000000002 ***
PositionGK       -21.9122694   0.3481804 -62.934 < 0.0000000000000002 ***
PositionLB       -22.5231855   0.2918440 -77.175 < 0.0000000000000002 ***
PositionRB       -22.4813121   0.2918303 -77.036 < 0.0000000000000002 ***
PositionWINGER   -22.3741363   0.2899391 -77.168 < 0.0000000000000002 ***
Crossing           0.0129694   0.0014735   8.802 < 0.0000000000000002 ***
Finishing          0.0148769   0.0016963   8.770 < 0.0000000000000002 ***
HeadingAccuracy    0.0343081   0.0014770  23.228 < 0.0000000000000002 ***
ShortPassing       0.0473184   0.0023925  19.778 < 0.0000000000000002 ***
Volleys            0.0026586   0.0014584   1.823             0.068317 .
Dribbling          0.0154163   0.0021025   7.332   0.0000000000002308 ***
Curve              0.0021397   0.0014391   1.487             0.137054
FKAccuracy         0.0039790   0.0012833   3.101             0.001933 **
LongPassing       -0.0043916   0.0017846  -2.461             0.013867 *
BallControl        0.0759510   0.0025459  29.832 < 0.0000000000000002 ***
Acceleration       0.0206975   0.0019705  10.503 < 0.0000000000000002 ***
SprintSpeed        0.0182212   0.0018408   9.899 < 0.0000000000000002 ***
Agility            0.0005042   0.0014771   0.341             0.732837
Reactions          0.1423474   0.0020521  69.367 < 0.0000000000000002 ***
Balance           -0.0193186   0.0013366 -14.453 < 0.0000000000000002 ***
ShotPower          0.0087459   0.0014774   5.920   0.0000000032516514 ***
Jumping            0.0008381   0.0010545   0.795             0.426782
Stamina            0.0327461   0.0012180  26.885 < 0.0000000000000002 ***
Strength           0.0194850   0.0012654  15.399 < 0.0000000000000002 ***
LongShots          0.0030692   0.0015786   1.944             0.051871 .
Aggression         0.0028618   0.0011142   2.568             0.010219 *
Interceptions      0.0033968   0.0015938   2.131             0.033070 *
Positioning       -0.0074743   0.0016186  -4.618   0.0000038934045944 ***
Vision            -0.0051248   0.0014592  -3.512             0.000445 ***
Penalties         -0.0106921   0.0013903  -7.690   0.0000000000000150 ***
Composure          0.0330748   0.0015384  21.500 < 0.0000000000000002 ***
Marking            0.0075537   0.0014479   5.217   0.0000001828169634 ***
StandingTackle     0.0178778   0.0023829   7.502   0.0000000000000641 ***
SlidingTackle     -0.0191860   0.0022312  -8.599 < 0.0000000000000002 ***
GKDiving           0.0415887   0.0030966  13.430 < 0.0000000000000002 ***
GKHandling         0.0405574   0.0031067  13.055 < 0.0000000000000002 ***
GKKicking          0.0213010   0.0028821   7.391   0.0000000000001490 ***
GKPositioning      0.0275882   0.0030062   9.177 < 0.0000000000000002 ***
GKReflexes         0.0493194   0.0030462  16.191 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.902 on 36143 degrees of freedom
Multiple R-squared:  0.9251,    Adjusted R-squared:  0.925
F-statistic: 1.015e+04 on 44 and 36143 DF,  p-value: < 0.00000000000000022
```

We have got good R-squared results and residual error is low as well.

Next, to optimise the model the unnecessary variables have been dropped.

```
lm2  <- lm(Overall ~ Age + Potential + Position + Crossing + Finishing + HeadingAccuracy + ShortPassing +
           Dribbling + BallControl + Acceleration + SprintSpeed + Reactions + Balance + ShotPower +
           Stamina + Strength + FKAccuracy + Vision + Penalties + Composure + Marking + StandingTackle +
           SlidingTackle + GKDiving + GKHandling + GKKicking + GKPositioning + GKReflexes, player_db)
save(lm2, file = "MultiLinearModelFinal.rda") #Finalizing LM2
```

```
Call:
lm(formula = Overall ~ Age + Potential + Position + Crossing +
    Finishing + HeadingAccuracy + ShortPassing + Dribbling +
    BallControl + Acceleration + SprintSpeed + Reactions + Balance +
    ShotPower + Stamina + Strength + FKAccuracy + Vision + Penalties +
    Composure + Marking + StandingTackle + SlidingTackle + GKDiving +
    GKHandling + GKKicking + GKPositioning + GKReflexes, data = player_db)

Residuals:
     Min       1Q   Median       3Q      Max
-25.5326  -1.1139   0.1398   1.2548   7.7397

Coefficients:
                  Estimate Std. Error t value             Pr(>|t|)
(Intercept)       7.934280   0.348312  22.779 < 0.0000000000000002 ***
Age               0.510240   0.003849 132.562 < 0.0000000000000002 ***
Potential         0.533221   0.002937 181.535 < 0.0000000000000002 ***
PositionCB      -20.943836   0.286191 -73.181 < 0.0000000000000002 ***
PositionCM      -23.239148   0.290402 -80.024 < 0.0000000000000002 ***
PositionDM      -23.050474   0.292168 -78.895 < 0.0000000000000002 ***
PositionFORWARD -22.753428   0.290932 -78.209 < 0.0000000000000002 ***
PositionGK      -21.898935   0.347876 -62.950 < 0.0000000000000002 ***
PositionLB      -22.552446   0.291090 -77.476 < 0.0000000000000002 ***
PositionRB      -22.508799   0.291068 -77.332 < 0.0000000000000002 ***
PositionWINGER  -22.408782   0.289197 -77.486 < 0.0000000000000002 ***
Crossing          0.012383   0.001416   8.746 < 0.0000000000000002 ***
Finishing         0.014131   0.001494   9.456 < 0.0000000000000002 ***
HeadingAccuracy   0.034774   0.001418  24.529 < 0.0000000000000002 ***
ShortPassing      0.044343   0.002022  21.927 < 0.0000000000000002 ***
Dribbling         0.014444   0.002059   7.015   0.0000000000023458 ***
BallControl       0.075845   0.002536  29.907 < 0.0000000000000002 ***
Acceleration      0.020967   0.001924  10.897 < 0.0000000000000002 ***
SprintSpeed       0.018081   0.001830   9.880 < 0.0000000000000002 ***
Reactions         0.142865   0.002007  71.195 < 0.0000000000000002 ***
Balance          -0.018561   0.001239 -14.983 < 0.0000000000000002 ***
ShotPower         0.010864   0.001303   8.337 < 0.0000000000000002 ***
Stamina           0.033111   0.001187  27.901 < 0.0000000000000002 ***
Strength          0.020092   0.001234  16.281 < 0.0000000000000002 ***
FKAccuracy        0.005471   0.001141   4.796   0.0000016263714068 ***
Vision           -0.006229   0.001412  -4.412   0.0000102559511745 ***
Penalties        -0.010526   0.001374  -7.660   0.0000000000000191 ***
Composure         0.033363   0.001530  21.812 < 0.0000000000000002 ***
Marking           0.008170   0.001426   5.730   0.0000000101312316 ***
StandingTackle    0.019937   0.002261   8.819 < 0.0000000000000002 ***
SlidingTackle    -0.018538   0.002208  -8.396 < 0.0000000000000002 ***
GKDiving          0.041556   0.003095  13.426 < 0.0000000000000002 ***
GKHandling        0.040569   0.003106  13.062 < 0.0000000000000002 ***
GKKicking         0.021302   0.002883   7.390   0.0000000000001505 ***
GKPositioning     0.027417   0.003006   9.122 < 0.0000000000000002 ***
GKReflexes        0.049455   0.003046  16.237 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.903 on 36152 degrees of freedom
Multiple R-squared:  0.925,     Adjusted R-squared:  0.925
F-statistic: 1.275e+04 on 35 and 36152 DF,  p-value: < 0.00000000000000022
```

R-squared has not decreased much and neither has residual error increased a lot.

Testing the model;

```
## Loading the MultiLinear Model for predicting Overall
load(file = "D:/Amity/Capstone Project/Prediction Models/MultiLinear Model - Overall/MultiLinearModelFinal.rda")
summary(lm2)
Overall.Predicted.MLM <- predict(lm2, player_db20, probability = T)
Overall.Predicted.MLM <- round(Overall.Predicted.MLM)
aucMLM <- roc(player_db20$Overall, Overall.Predicted.MLM, plot = T, col = "cyan")
RMSE(Overall.Predicted.MLM, player_db20$Overall)
View(Overall.Predicted.MLM)
```

Root-Mean-Error for this model is 1.83379, which is acceptable but we must look at other models built to determine the best from the lot.

- Regression Tree Model

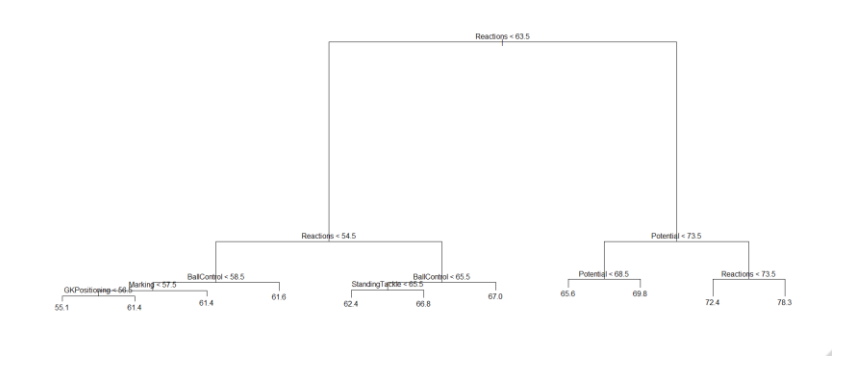
  I have created only one Regression Tree Model using the same variables we used in the last model.

```
## regression tree model
set.seed(1)
rt <- tree(Overall ~ Age + Potential + Position + Crossing + Finishing + HeadingAccuracy + ShortPassing +
             Volleys + Dribbling + Curve + FKAccuracy + LongPassing + BallControl + Acceleration + SprintSpeed +
             Agility + Reactions + Balance + ShotPower + Jumping + Stamina + Strength + LongShots + Aggression +
             Interceptions + Positioning + Vision + Penalties + Composure + Marking + StandingTackle +
             SlidingTackle + GKDiving + GKHandling + GKKicking + GKPositioning + GKReflexes, player_db)
summary(rt)
plot(rt)
text(rt, pretty = 0)
cv.tree(rt)
rtmodel <- prune.tree(rt, best = 11)
plot(rtmodel)
text(rtmodel, pretty = 0)
save(rtmodel, file = "RegressionTree Model.rda")
```

  I have pruned the tree for optimization.

  Let’s, test the model and have a look at how we have done.

```
## Loading the Regression Tree Model for preticting Overall
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Overall/RegressionTree Model.rda")
par(xpd = NA) # to prevent text clipping
plot(rtmodel)
text(rtmodel, digits = 3)
Overall.Predicted.RTM <- predict(rtmodel, player_db20)
Overall.Predicted.RTM <- round(Overall.Predicted.RTM)
RMSE(Overall.Predicted.RTM, player_db20$Overall)
```



  The Root-Mean-Squared-Error of this model is 3.146225 which is a lot higher than the Multi Linear Model (lm2) and thus lm2 seems like a better model till now.

- Artificial Neural Network Model

  Before, we start with the last model built for predicting overall of the players, we must standardize the data as it helps greatly with the reducing the model building time also, I have seen better accuracy using standardized data.

```
MinOvr <- min(player_db$Overall)
MinAge <- min(player_db$Age)
MinPot <- min(player_db$Potential)
MinCro <- min(player_db$Crossing)
MinFin <- min(player_db$Finishing)
MinHAc <- min(player_db$HeadingAccuracy)
MinSps <- min(player_db$ShortPassing)
MinDri <- min(player_db$Dribbling)
MinBCt <- min(player_db$BallControl)
MinAcc <- min(player_db$Acceleration)
MinSSp <- min(player_db$SprintSpeed)
MinRec <- min(player_db$Reactions)
MinBal <- min(player_db$Balance)
MinSPw <- min(player_db$ShotPower)
MinSta <- min(player_db$Stamina)
MinStr <- min(player_db$Strength)
MinFKA <- min(player_db$FKAccuracy)
MinVis <- min(player_db$Vision)
MinPen <- min(player_db$Penalties)
MinCom <- min(player_db$Composure)
MinMar <- min(player_db$Marking)
MinSTk <- min(player_db$StandingTackle)
MinSlT <- min(player_db$SlidingTackle)
MinGKD <- min(player_db$GKDiving)
MinGKH <- min(player_db$GKHandling)
MinGKK <- min(player_db$GKKicking)
MinGKP <- min(player_db$GKPositioning)
MinGKR <- min(player_db$GKReflexes)
MinVal <- min(player_db$Value)
MinInt <- min(player_db$International.Reputation)
```

```
MaxOvr <- max(player_db$Overall)
MaxAge <- max(player_db$Age)
MaxPot <- max(player_db$Potential)
MaxCro <- max(player_db$Crossing)
MaxFin <- max(player_db$Finishing)
MaxHAc <- max(player_db$HeadingAccuracy)
MaxSps <- max(player_db$ShortPassing)
MaxDri <- max(player_db$Dribbling)
MaxBCt <- max(player_db$BallControl)
MaxAcc <- max(player_db$Acceleration)
MaxSSp <- max(player_db$SprintSpeed)
MaxRec <- max(player_db$Reactions)
MaxBal <- max(player_db$Balance)
MaxSPw <- max(player_db$ShotPower)
MaxSta <- max(player_db$Stamina)
MaxStr <- max(player_db$Strength)
MaxFKA <- max(player_db$FKAccuracy)
MaxVis <- max(player_db$Vision)
MaxPen <- max(player_db$Penalties)
MaxCom <- max(player_db$Composure)
MaxMar <- max(player_db$Marking)
MaxSTk <- max(player_db$StandingTackle)
MaxSlT <- max(player_db$SlidingTackle)
MaxGKD <- max(player_db$GKDiving)
MaxGKH <- max(player_db$GKHandling)
MaxGKK <- max(player_db$GKKicking)
MaxGKP <- max(player_db$GKPositioning)
MaxGKR <- max(player_db$GKReflexes)
MaxVal <- max(player_db$Value)
MaxInt <- max(player_db$International.Reputation)
```

```
Overall <- scale(player_db$Overall, center = MinOvr, scale = MaxOvr - MinOvr)
Age <- scale(player_db$Age, center = MinAge, scale = MaxAge - MinAge)
Potential <- scale(player_db$Potential, center = MinPot, scale = MaxPot - MinPot)
Crossing <- scale(player_db$Crossing, center = MinCro, scale = MaxCro - MinCro)
Finishing <- scale(player_db$Finishing, center = MinFin, scale = MaxFin - MinFin)
HeadingAccuracy <- scale(player_db$HeadingAccuracy, center = MinHAc, scale = MaxHAc - MinHAc)
ShortPassing <- scale(player_db$ShortPassing, center = MinSps, scale = MaxSps - MinSps)
Dribbling <- scale(player_db$Dribbling, center = MinDri, scale = MaxDri - MinDri)
BallControl <- scale(player_db$BallControl, center = MinBCt, scale = MaxBCt - MinBCt)
Acceleration <- scale(player_db$Acceleration, center = MinAcc, scale = MaxAcc - MinAcc)
SprintSpeed <- scale(player_db$SprintSpeed, center = MinSSp, scale = MaxSSp - MinSSp)
Reaction <- scale(player_db$Reactions, center = MinRec, scale = MaxRec - MinRec)
Balance <- scale(player_db$Balance, center = MinBal, scale = MaxBal - MinBal)
ShotPower <- scale(player_db$ShotPower, center = MinSPw, scale = MaxSPw - MinSPw)
Stamina <- scale(player_db$Stamina, center = MinSta, scale = MaxSta - MinSta)
Strength <- scale(player_db$Strength, center = MinStr, scale = MaxStr - MinStr)
FKAccuracy <- scale(player_db$FKAccuracy, center = MinFKA, scale = MaxFKA - MinFKA)
Vision <- scale(player_db$Vision, center = MinVis, scale = MaxVis - MinVis)
Penalties <- scale(player_db$Penalties, center = MinPen, scale = MaxPen - MinPen)
Composure <- scale(player_db$Composure, center = MinCom, scale = MaxCom - MinCom)
Marking <- scale(player_db$Marking, center = MinMar, scale = MaxMar - MinMar)
StandingTackle <- scale(player_db$StandingTackle, center = MinSTk, scale = MaxSTk - MinSTk)
SlidingTackle <- scale(player_db$SlidingTackle, center = MinSlT, scale = MaxSlT - MinSlT)
GKDiving <- scale(player_db$GKDiving, center = MinGKD, scale = MaxGKD - MinGKD)
GKHandling <- scale(player_db$GKHandling, center = MinGKH, scale = MaxGKH - MinGKH)
GKKicking <- scale(player_db$GKKicking, center = MinGKK, scale = MaxGKK - MinGKK)
GKPositioning <- scale(player_db$GKPositioning, center = MinGKP, scale = MaxGKP - MinGKP)
GKReflexes <- scale(player_db$GKReflexes, center = MinGKR, scale = MaxGKR - MinGKR)
Value <- scale(player_db$Value, center = MinVal, scale = MaxVal - MinVal)
International.Reputation <- scale(player_db$International.Reputation, center = MinInt, scale = MaxInt - MinInt)
```

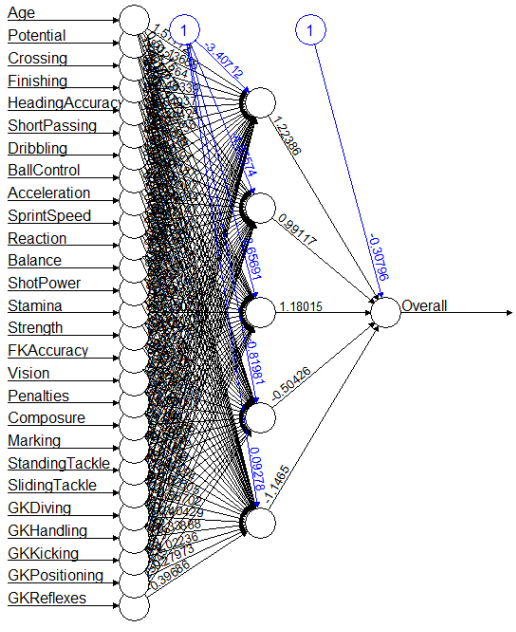
```
player_db_nn <- data.frame(Value, Overall, Age, Potential, Crossing, Finishing, HeadingAccuracy, ShortPassing,
                           Dribbling, BallControl, Acceleration, SprintSpeed, Reaction, Balance,  ShotPower,
                           Stamina, Strength, FKAccuracy, Vision, Penalties, Composure, Marking, StandingTackle,
                           SlidingTackle, GKDiving, GKHandling, GKKicking, GKPositioning, GKReflexes)
```

I am sure there are better ways to standardize the data. But this is the one I remembered at the time of doing this and since I had already done it, I didn't bother changing it when I remembered other better methods.

Now to building the model

```
## ANN for predicting Overall
NeuralNetwork <- neuralnet(Overall ~ Age + Potential + Crossing + Finishing + HeadingAccuracy + ShortPassing +
                             Dribbling + BallControl + Acceleration + SprintSpeed + Reaction + Balance +
                             ShotPower + Stamina + Strength + FKAccuracy + Vision + Penalties + Composure + Marking +
                             StandingTackle + SlidingTackle + GKDiving + GKHandling + GKKicking + GKPositioning +
                             GKReflexes,
                           data = player_db_nn,
                           hidden = 5,
                           linear.output = TRUE,
                           rep = 2,
                           stepmax = 1e+08,
                           threshold = .1)
save(NeuralNetwork, file = "NeuralNetworkOverall.rda")
```

```
## Loading the Neural Network for predicting Overall
load(file = "D:/Amity/Capstone Project/Prediction Models/NeuralNetwork Model - Overall/NeuralNetworkOverall.rda")
plot(NeuralNetwork)
Overall.Predicted.ANN <- predict(NeuralNetwork, player_db20_nn)
Overall.Predicted.ANN <- (Overall.Predicted.ANN *
                            (max(player_db$Overall) - min(player_db$Overall))) + min(player_db$Overall)
Overall.Predicted.ANN <-  round(Overall.Predicted.ANN)
View(Overall.Predicted.ANN)
View(player_db20$Overall)
aucANN <- roc(player_db20$Overall, Overall.Predicted.ANN, plot = T, col = "cyan")
RMSE(Overall.Predicted.ANN, player_db20$Overall)
```



We also need to standardize the test data for the model to be able to predict, that is done in script file "FIFA_20_DataCleaning.R".

Further after predicting the prediction results need to be rescaled to the scale of the original data to of any use to us.

RMSE is 2.133613 which more than it was for lm2 but is still acceptable.

b. Building models to predict the Value of the Player

While building the model for predicting the Value of the Player there were many difficulties to be faced.

First and foremost was that, I, for the life of couldn't come with a model which would give me any form of satisfying results.

There were of course many reasons for this, not least of which is that valuation of a player is very delicate and many factors are considered such as Age, Overall, Potential, International Reputation these are the once used by me to build the model but, there are many other factors to be considered which can are not quantifiable or that we do not have access to such as squad status, league the player is playing, image rights, adaptability, etc.

So, keeping all the above things in mind I am only presenting the models that were even slightly satisfactory.

I made multiple Regression Tree and Neural Network but to no avail. At the end I decided to segment the data based on the overall of the players. I created 4 Segments;

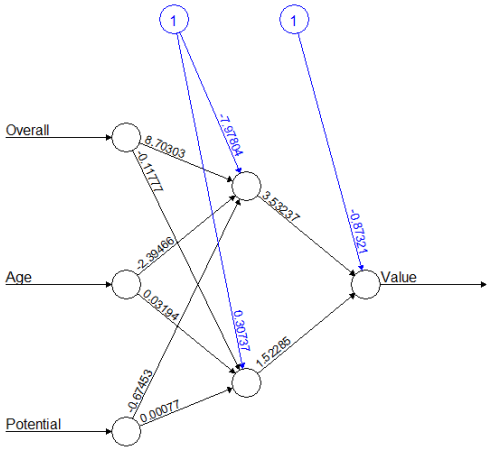
First <- Players with Overall more than or equal 80
Second <- Players with Overall less than 80 but more than or equal to 70
Third <- Players with Overall less than 70 but more than or equal to 60
Forth/Last <- Everyone else
Doing same to the test data as well.

This I thought would give the model a chance to understand the population with less fluctuation of the dependent variable. Which has worked out better than anything else I tried.

- Model 1 (NeuralNetworkValue)

```
## ANN for predicting Value of player
NeuralNetworkValue1 <- neuralnet(Value ~ Overall + Age + Potential,
                                 data = player_db_nn_V, hidden = 2,
                                 linear.output = T,
                                 rep = 2,
                                 stepmax = 1e+08,
                                 threshold = .1)
save(NeuralNetworkValue)
```

```
## Loading the Neural Network for predicting Value of the player
load(file = "D:/Amity/Capstone Project/Prediction Models/NeuralNetwork Model - Value/NeuralNetworkValue.rda")
plot(NeuralNetworkValue)
Value.Predicted.ANN.1 <- predict(NeuralNetworkValue, player_db20_nn)
Value.Predicted.ANN.1 <- (Value.Predicted.ANN.1 *
                          (max(player_db$Value) - min(player_db$Value))) + min(player_db$Value)
Value.Predicted.ANN.1 <- round(Value.Predicted.ANN.1)
View(Value.Predicted.ANN.1)
View(player_db20$Value)
aucANNVal <- roc(player_db20$Value, Value.Predicted.ANN.1, plot = T, col = "maroon")
RMSE(Value.Predicted.ANN.1, player_db20$Value)
```



RMSE <- 1737018

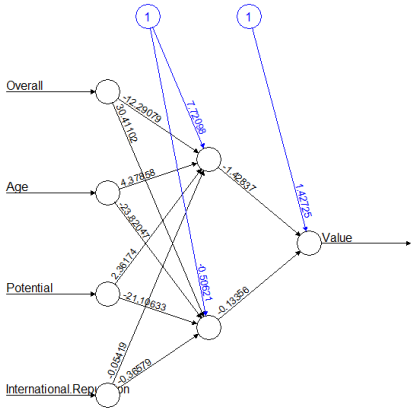
I have only used 3 variables here with scaled data, this was but a test run. A baseline of sorts.

RMSE is higher than I would prefer, but lest move forward to and have a look at how other models do.

- Model 2 (NeuralNetwork_Value)

```
NeuralNetwork_Value <- neuralnet(Value ~ Overall + Age + Potential + International.Reputation,
                                 data = player_db_nn_V, hidden = 2, linear.output = TRUE, rep = 2,
                                 stepmax = 1e+08, threshold = .1)
save(NeuralNetwork_Value, file = "NeuralNetwork_Value.rda")
```

```
load(file = "D:/Amity/Capstone Project/Prediction Models/NeuralNetwork Model - Value/NeuralNetwork_Value.rda")
plot(NeuralNetwork_Value)
Value.Predicted.ANN.3 <- predict(NeuralNetwork_Value, player_db20_nn)
Value.Predicted.ANN.3 <- (Value.Predicted.ANN.3 *
                          (max(player_db$Value) - min(player_db$Value))) + min(player_db$Value)
Value.Predicted.ANN.3 <- round(Value.Predicted.ANN.3)
View(Value.Predicted.ANN.3)
View(player_db20$Value)
RMSE(Value.Predicted.ANN.3, player_db20$Value)
```

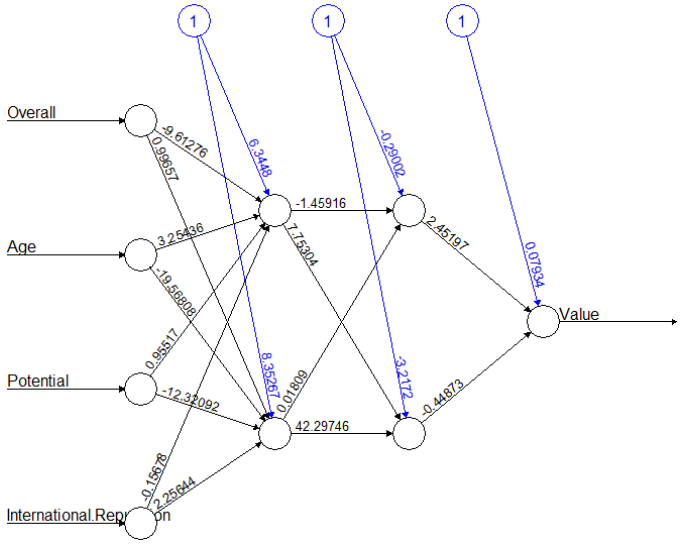


The complexity of the model has been increased, number of repetitions has also been increased but, the RMSE (1877101) has also increased for this model.

- Model 3 (NeuralNetworkValue1)

```
NeuralNetwork_Value1 <- neuralnet(Value ~ Overall + Age + Potential + International.Reputation,
                                  data = player_db_nn_V, hidden = c(2,2), linear.output = TRUE, rep = 2,
                                  stepmax = 1e+08, threshold = .1)
save(NeuralNetwork_Value1, file = "NeuralNetwork_Value1.rda")
```

```
load(file = "D:/Amity/Capstone Project/Prediction Models/NeuralNetwork Model - Value/NeuralNetwork_Value1.rda")
plot(NeuralNetwork_Value1)
Value.Predicted.ANN.4 <- predict(NeuralNetwork_Value1, player_db20_nn)
Value.Predicted.ANN.4 <- (Value.Predicted.ANN.4 *
                          (max(player_db$Value) - min(player_db$Value))) + min(player_db$Value)
Value.Predicted.ANN.4 <- round(Value.Predicted.ANN.4)
View(Value.Predicted.ANN.4)
View(player_db20$Value)
RMSE(Value.Predicted.ANN.4, player_db20$Value)
```



Error: 1.072467 Steps: 9897

RMSE <- 2193584

- Model 4 (rtvalue1)

```
## regresion tree model for predictiing value
set.seed(123)
rtvalue1 <- rpart(formula = Value ~ Overall + Age + Potential + Position + Crossing + Finishing +
                    HeadingAccuracy + ShortPassing + Volleys + Dribbling + Curve + FKAccuracy + LongPassing +
                    BallControl + Acceleration + SprintSpeed + Agility + Reactions + Balance + ShotPower +
                    Jumping + Stamina + Strength + LongShots + Aggression + Interceptions + Positioning +
                    Vision + Penalties + Composure + Marking + StandingTackle + SlidingTackle + GKDiving +
                    GKHandling + GKKicking + GKPositioning + GKReflexes,
                  data = player_db,
                  method = "anova",
                  control = list(cp = 0, xval = 10))
save(rtvalue1, file = "RegressionTreeValueModel1.rda")
```

```
## Loading the Regression Tree Model for predicting Value of the Player
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel1.rda")
rtvalmodpred1 <- predict(rtvalue1, player_db20)
rtvalmodpred1 <- round(rtvalmodpred1)
View(rtvalmodpred1)
RMSE(rtvalmodpred1, player_db20$Value)
```

RMSE <- 1018627

- Model 5 (baggedrtvalue)

```
baggedrtvalue <- bagging(formula = Value ~ Overall + Age + Potential + Position + Crossing + Finishing +
                           HeadingAccuracy + ShortPassing + Volleys + Dribbling + Curve + FKAccuracy + LongPassing +
                           BallControl + Acceleration + SprintSpeed + Agility + Reactions + Balance + ShotPower +
                           Jumping + Stamina + Strength + LongShots + Aggression + Interceptions + Positioning +
                           Vision + Penalties + Composure + Marking + StandingTackle + SlidingTackle + GKDiving +
                           GKHandling + GKKicking + GKPositioning + GKReflexes,
                         data = player_db,
                         coob = T)
save(baggedrtvalue, file = "RegressionTreeModelBagged.rda")
```

```
# Loading the Bagging Regression Tree Model
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeModelBagged.rda")
rtvalmodpred2 <- predict(baggedrtvalue, player_db20)
rtvalmodpred2 <- round(rtvalmodpred2)
View(rtvalmodpred2)
RMSE(rtvalmodpred2, player_db20$Value)
```

RMSE <- 1686963

- Model 6 (model) [Bagged Regression Tree Model]

```
ntree <- 30:100
rmse <- vector(mode = "numeric", length = length(ntree))
for(i in seq_along(ntree)) {
  set.seed(123)
  model <- bagging(formula = Value ~ Overall + Age + Potential + Position + Crossing + Finishing +
                     HeadingAccuracy + ShortPassing + Volleys + Dribbling + Curve + FKAccuracy + LongPassing +
                     BallControl + Acceleration + SprintSpeed + Agility + Reactions + Balance + ShotPower +
                     Jumping + Stamina + Strength + LongShots + Aggression + Interceptions + Positioning +
                     Vision + Penalties + Composure + Marking + StandingTackle + SlidingTackle + GKDiving +
                     GKHandling + GKKicking + GKPositioning + GKReflexes,
                   data = player_db,
                   coob = T,
                   nbagg = ntree[i])
  rmse[i] <- model$err
}
save(model, file = "RegressionTreeModelBagged2.rda")
```

```
#Loading the second Bagged Regression Tree Model
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeModelBagged2.rda")
rtvalmodpred3 <- predict(model, player_db20)
rtvalmodpred3 <- round(rtvalmodpred3)
View(rtvalmodpred3)
RMSE(rtvalmodpred3, player_db20$Value)
```

RMSE <- 1663853

- Model 7 [Neural Networks with Segmented Data]

  For this model again the data needs to be scaled, it is of course not ideal, but it does help in reducing model building time and the accuracy of the model does not suffer a lot. It is done in an act of balancing accuracy to compute time required.

```
## ANN for predicting Value of the player with dataset divided in segments for better predictions
# Data is divided as per the overall of the players as follows
# Players with Overall above 80 <- Segment 1
# Players with Overall more than equal to 70 but less than 80 <- Segment 2
# Players with Overall more than equal to 60 but less than 70 <- Segment 3
# And everyone else <- Segment 4
# Diffrent prediction model will be created for diffrent segments of the player
MinOvr1 <- min(segment1$Overall)
MinAge1 <- min(segment1$Age)
MinPot1 <- min(segment1$Potential)
MinVal1 <- min(segment1$Value)
MinInt1 <- min(segment1$International.Reputation)
MinOvr2 <- min(segment2$Overall)
MinAge2 <- min(segment2$Age)
MinPot2 <- min(segment2$Potential)
MinVal2 <- min(segment2$Value)
MinInt2 <- min(segment2$International.Reputation)
MinOvr3 <- min(segment3$Overall)
MinAge3 <- min(segment3$Age)
MinPot3 <- min(segment3$Potential)
MinVal3 <- min(segment3$Value)
MinInt3 <- min(segment3$International.Reputation)
MinOvr4 <- min(segment4$Overall)
MinAge4 <- min(segment4$Age)
MinPot4 <- min(segment4$Potential)
MinVal4 <- min(segment4$Value)
MinInt4 <- min(segment4$International.Reputation)

MaxOvr1 <- max(segment1$Overall)
MaxAge1 <- max(segment1$Age)
MaxPot1 <- max(segment1$Potential)
MaxVal1 <- max(segment1$Value)
MaxInt1 <- max(segment1$International.Reputation)
MaxOvr2 <- max(segment2$Overall)
MaxAge2 <- max(segment2$Age)
MaxPot2 <- max(segment2$Potential)
MaxVal2 <- max(segment2$Value)
MaxInt2 <- max(segment2$International.Reputation)
MaxOvr3 <- max(segment3$Overall)
MaxAge3 <- max(segment3$Age)
MaxPot3 <- max(segment3$Potential)
MaxVal3 <- max(segment3$Value)
MaxInt3 <- max(segment3$International.Reputation)
MaxOvr4 <- max(segment4$Overall)
MaxAge4 <- max(segment4$Age)
MaxPot4 <- max(segment4$Potential)
MaxVal4 <- max(segment4$Value)
MaxInt4 <- max(segment4$International.Reputation)
```

```
Overall1 <- scale(segment1$Overall, center = MinOvr1, scale = MaxOvr1 - MinOvr1)
Age1 <- scale(segment1$Age, center = MinAge1, scale = MaxAge1 - MinAge1)
Potential1 <- scale(segment1$Potential, center = MinPot1, scale = MaxPot1 - MinPot1)
Value1 <- scale(segment1$Value, center = MinVal1, scale = MaxVal1 - MinVal1)
International.Reputation1 <- scale(segment1$International.Reputation, center = MinInt1, scale = MaxInt1 - MinInt1)
Overall2 <- scale(segment2$Overall, center = MinOvr2, scale = MaxOvr2 - MinOvr2)
Age2 <- scale(segment2$Age, center = MinAge2, scale = MaxAge2 - MinAge2)
Potential2 <- scale(segment2$Potential, center = MinPot2, scale = MaxPot2 - MinPot2)
Value2 <- scale(segment2$Value, center = MinVal2, scale = MaxVal2 - MinVal2)
International.Reputation2 <- scale(segment2$International.Reputation, center = MinInt2, scale = MaxInt2 - MinInt2)
Overall3 <- scale(segment3$Overall, center = MinOvr3, scale = MaxOvr3 - MinOvr3)
Age3 <- scale(segment3$Age, center = MinAge3, scale = MaxAge3 - MinAge3)
Potential3 <- scale(segment3$Potential, center = MinPot3, scale = MaxPot3 - MinPot3)
Value3 <- scale(segment3$Value, center = MinVal3, scale = MaxVal3 - MinVal3)
International.Reputation3 <- scale(segment3$International.Reputation, center = MinInt3, scale = MaxInt3 - MinInt3)
Overall4 <- scale(segment4$Overall, center = MinOvr4, scale = MaxOvr4 - MinOvr4)
Age4 <- scale(segment4$Age, center = MinAge4, scale = MaxAge4 - MinAge4)
Potential4 <- scale(segment4$Potential, center = MinPot4, scale = MaxPot4 - MinPot4)
Value4 <- scale(segment4$Value, center = MinVal4, scale = MaxVal4 - MinVal4)
International.Reputation4 <- scale(segment4$International.Reputation, center = MinInt4, scale = MaxInt4 - MinInt4)

player_db_seg1 <- data.frame(Value1, Overall1, Potential1, Age1, International.Reputation1)
player_db_seg2 <- data.frame(Value2, Overall2, Potential2, Age2, International.Reputation2)
player_db_seg3 <- data.frame(Value3, Overall3, Potential3, Age3, International.Reputation3)
player_db_seg4 <- data.frame(Value4, Overall4, Potential4, Age4, International.Reputation4)
```

Once the data is scaled the we need to move on to building the models. Here, since the data is segmented, we need to will need to create separate models for each segment.

```
NeuralNetwork_Value_Seg1 <- neuralnet(Value1 ~ Overall1 + Age1 + Potential1 + International.Reputation1,
                                      data = player_db_seg1,
                                      hidden = 3,
                                      linear.output = T,
                                      rep = 2,
                                      stepmax = 1e+08,
                                      threshold = .1)
save(NeuralNetwork_Value_Seg1, file = "NeuralNetwork_Value_Seg1.rda")

NeuralNetwork_Value_Seg2 <- neuralnet(Value2 ~ Overall2 + Age2 + Potential2 + International.Reputation2,
                                      data = player_db_seg2,
                                      hidden = 3,
                                      linear.output = T,
                                      rep = 2,
                                      stepmax = 1e+08,
                                      threshold = .1)
save(NeuralNetwork_Value_Seg2, file = "NeuralNetwork_Value_Seg2.rda")

NeuralNetwork_Value_Seg3 <- neuralnet(Value3 ~ Overall3 + Age3 + Potential3 + International.Reputation3,
                                      data = player_db_seg3,
                                      hidden = 3,
                                      linear.output = T,
                                      rep = 2,
                                      stepmax = 1e+08,
                                      threshold = .1)
save(NeuralNetwork_Value_Seg3, file = "NeuralNetwork_Value_Seg3.rda")

NeuralNetwork_Value_Seg4 <- neuralnet(Value4 ~ Overall4 + Age4 + Potential4 + International.Reputation4,
                                      data = player_db_seg4,
                                      hidden = 3,
                                      linear.output = T,
                                      rep = 2,
                                      stepmax = 1e+08,
                                      threshold = .1)
save(NeuralNetwork_Value_Seg4, file = "NeuralNetwork_Value_Seg4.rda")
```

```
# Getting prediction for first segment
pred.seg1 <- predict(NeuralNetwork_Value_Seg1, test_seg1)
pred.seg1 <- (pred.seg1 *
                (max(segment1$Value) - min(segment1$Value))) + min(segment1$Value)
pred.seg1 <- round(pred.seg1)
((mean(testsegment1$Value - pred.seg1))/mean(testsegment1$Value))*100 ## % deviance in value
RMSE(pred.seg1, testsegment1$Value)

# Getting prediction for second segment
pred.seg2 <- predict(NeuralNetwork_Value_Seg2, test_seg2)
pred.seg2 <- (pred.seg2 *
                (max(segment2$Value) - min(segment2$Value))) + min(segment2$Value)
pred.seg2 <- round(pred.seg2)
((mean(testsegment2$Value - pred.seg2))/mean(testsegment2$Value))*100 ## % deviance in value
RMSE(pred.seg2, testsegment2$Value)

# Getting prediction for third segment
pred.seg3 <- predict(NeuralNetwork_Value_Seg3, test_seg3)
pred.seg3 <- (pred.seg3 *
                (max(segment3$Value) - min(segment3$Value))) + min(segment3$Value)
pred.seg3 <- round(pred.seg3)
((mean(testsegment3$Value - pred.seg3))/mean(testsegment3$Value))*100 ## % deviance in value
RMSE(pred.seg3, testsegment3$Value)

# Getting prediction for last segment
pred.seg4 <- predict(NeuralNetwork_Value_Seg4, test_seg4)
pred.seg4 <- (pred.seg4 *
                (max(segment4$Value) - min(segment4$Value))) + min(segment4$Value)
pred.seg4 <- round(pred.seg4)
((mean(testsegment4$Value - pred.seg4))/mean(testsegment4$Value))*100 ## % deviance in value
RMSE(pred.seg4, testsegment4$Value)
```

Deviance of various model predictions to their actual values are as follows;

For Segment 1 = 3.554717%RMSE = 4045410
For Segment 2 = 2.486% RMSE = 1566107
For Segment 3 = 4.587903%RMSE = 111627
For Segment 4 = 3.6521% RMSE = 28000

- Model 8 (rtvalue_seg1,2,3&4) [Regression Models with Segmented Data]

```
## Regresssion tree with segmented data
rtvalue_seg1 <- rpart(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                      data = segment1,
                      method = "anova",
                      control = list(cp = 0, xval = 1000))
save(rtvalue_seg1, file = "RegressionTreeValueModel_Seg1.rda")

rtvalue_seg2 <- rpart(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                      data = segment2,
                      method = "anova",
                      control = list(cp = 0, xval = 10))
save(rtvalue_seg2, file = "RegressionTreeValueModel_Seg2.rda")

rtvalue_seg3 <- rpart(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                      data = segment3,
                      method = "anova",
                      control = list(cp = 0, xval = 10))
save(rtvalue_seg3, file = "RegressionTreeValueModel_Seg3.rda")

rtvalue_seg4 <- rpart(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                      data = segment4,
                      method = "anova",
                      control = list(cp = 0, xval = 10))
save(rtvalue_seg4, file = "RegressionTreeValueModel_Seg4.rda")
```

```
## Regression Tree Model (Segmented Data)
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg1.rda")
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg2.rda")
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg3.rda")
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg4.rda")

# Prediction for first segment
pred.rt.seg1 <- predict(rtvalue_seg1, testsegment1)
((mean(testsegment1$Value - pred.rt.seg1))/mean(testsegment1$Value))*100
RMSE(pred.rt.seg1, testsegment1$Value)

# Prediction for second segment
pred.rt.seg2 <- predict(rtvalue_seg2, testsegment2)
((mean(testsegment2$Value - pred.rt.seg2))/mean(testsegment2$Value))*100
RMSE(pred.rt.seg2, testsegment2$Value)

# Prediction for third segment
pred.rt.seg3 <- predict(rtvalue_seg3, testsegment3)
((mean(testsegment3$Value - pred.rt.seg3))/mean(testsegment3$Value))*100
RMSE(pred.rt.seg3, testsegment3$Value)

# Prediction for the last segment
pred.rt.seg4 <- predict(rtvalue_seg4, testsegment4)
((mean(testsegment4$Value - pred.rt.seg4))/mean(testsegment4$Value))*100
RMSE(pred.rt.seg4, testsegment4$Value)
```

```
> ## Regression Tree Model (Segmented Data)
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg1.rda")
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg2.rda")
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg3.rda")
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTreeValueModel_Seg4.rda")
>
> # Prediction for first segment
> pred.rt.seg1 <- predict(rtvalue_seg1, testsegment1)
> ((mean(testsegment1$Value - pred.rt.seg1))/mean(testsegment1$Value))*100
[1] 2.979969
> RMSE(pred.rt.seg1, testsegment1$Value)
[1] 4687335
>
> # Prediction for second segment
> pred.rt.seg2 <- predict(rtvalue_seg2, testsegment2)
> ((mean(testsegment2$Value - pred.rt.seg2))/mean(testsegment2$Value))*100
[1] 2.990882
> RMSE(pred.rt.seg2, testsegment2$Value)
[1] 822291
>
> # Prediction for third segment
> pred.rt.seg3 <- predict(rtvalue_seg3, testsegment3)
> ((mean(testsegment3$Value - pred.rt.seg3))/mean(testsegment3$Value))*100
[1] 3.431882
> RMSE(pred.rt.seg3, testsegment3$Value)
[1] 85696.24
>
> # Prediction for the last segment
> pred.rt.seg4 <- predict(rtvalue_seg4, testsegment4)
> ((mean(testsegment4$Value - pred.rt.seg4))/mean(testsegment4$Value))*100
[1] 1.555361
> RMSE(pred.rt.seg4, testsegment4$Value)
[1] 12823.89
>
```

- Model 9 (model.seg1, 2, 3 & 4) [Bagged Regression Tree Model with Segmented Data]

```
ntree1 <- 100:2000
rmse1 <- vector(mode = "numeric", length = length(ntree1))
for(i in seq_along(ntree1)) {
  set.seed(1234)
  model.seg1 <- bagging(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                        data = segment1,
                        coob = T,
                        nbagg = ntree1[i])
  rmse1[i] <- model.seg1$err
}
save(model.seg1, file = "RegressionTree Bagging Seg 1.rda")

ntree2 <- 100:2000
rmse2 <- vector(mode = "numeric", length = length(ntree2))
for(i in seq_along(ntree2)) {
  set.seed(1234)
  model.seg2 <- bagging(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                        data = segment2,
                        coob = T,
                        nbagg = ntree2[i])
  rmse2[i] <- model.seg2$err
}
save(model.seg2, file = "RegressionTree Bagging Seg 2.rda")

ntree3 <- 100:2000
rmse3 <- vector(mode = "numeric", length = length(ntree3))
for(i in seq_along(ntree3)) {
  set.seed(1234)
  model.seg3 <- bagging(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                        data = segment3,
                        coob = T,
                        nbagg = ntree3[i])
  rmse3[i] <- model.seg3$err
}
save(model.seg3, file = "RegressionTree Bagging Seg 3.rda")

ntree4 <- 100:2000
rmse4 <- vector(mode = "numeric", length = length(ntree4))
for(i in seq_along(ntree4)) {
  set.seed(1234)
  model.seg4 <- bagging(formula = Value ~ Overall + Age + Potential + Position + International.Reputation,
                        data = segment4,
                        coob = T,
                        nbagg = ntree4[i])
  rmse4[i] <- model.seg4$err
}
save(model.seg4, file = "RegressionTree Bagging Seg 4.rda")
```

```
## Bagged Regression Tree (Segmented Data)
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 1.rda")
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 2.rda")
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 3.rda")
load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 4.rda")

# Prediction for first segment
bag.pred.rt.seg1 <- predict(model.seg1, testsegment1)
((mean(testsegment1$Value - bag.pred.rt.seg1))/mean(testsegment1$Value))*100
RMSE(bag.pred.rt.seg1, testsegment1$Value)

# Prediction for second segment
bag.pred.rt.seg2 <- predict(model.seg2, testsegment2)
((mean(testsegment2$Value - bag.pred.rt.seg2))/mean(testsegment2$Value))*100
RMSE(bag.pred.rt.seg2, testsegment2$Value)

# Prediction for third segment
bag.pred.rt.seg3 <- predict(model.seg3, testsegment3)
((mean(testsegment3$Value - bag.pred.rt.seg3))/mean(testsegment3$Value))*100
RMSE(bag.pred.rt.seg3, testsegment3$Value)

#Prediction for last segment
bag.pred.rt.seg4 <- predict(model.seg4, testsegment4)
((mean(testsegment4$Value - bag.pred.rt.seg4))/mean(testsegment4$Value))*100
RMSE(bag.pred.rt.seg4, testsegment4$Value)
```

```
> ## Bagged Regression Tree (Segmented Data)
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 1.rda")
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 2.rda")
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 3.rda")
> load(file = "D:/Amity/Capstone Project/Prediction Models/RegressionTree Model - Value/RegressionTree Bagging Seg 4.rda")
>
> # Prediction for first segment
> bag.pred.rt.seg1 <- predict(model.seg1, testsegment1)
> ((mean(testsegment1$Value - bag.pred.rt.seg1))/mean(testsegment1$Value))*100
[1] 2.842415
> RMSE(bag.pred.rt.seg1, testsegment1$Value)
[1] 4932448
>
> # Prediction for second segment
> bag.pred.rt.seg2 <- predict(model.seg2, testsegment2)
> ((mean(testsegment2$Value - bag.pred.rt.seg2))/mean(testsegment2$Value))*100
[1] 2.513906
> RMSE(bag.pred.rt.seg2, testsegment2$Value)
[1] 1239809
>
> # Prediction for third segment
> bag.pred.rt.seg3 <- predict(model.seg3, testsegment3)
> ((mean(testsegment3$Value - bag.pred.rt.seg3))/mean(testsegment3$Value))*100
[1] 3.137263
> RMSE(bag.pred.rt.seg3, testsegment3$Value)
[1] 144464.5
>
> #Prediction for last segment
> bag.pred.rt.seg4 <- predict(model.seg4, testsegment4)
> ((mean(testsegment4$Value - bag.pred.rt.seg4))/mean(testsegment4$Value))*100
[1] 2.874199
> RMSE(bag.pred.rt.seg4, testsegment4$Value)
[1] 20676.17
>
```

I ended up building 9 models to predict the value of the player to get a satisfying result and that came from Model 9 (model.seg1, 2, 3 & 4) [Bagged Regression Tree Model with Segmented Data].

It has been able to predict data from all 4 segments with least deviance, although that being said, these models take up about 3.5GB of space alone. I understand this might be a breaking point for many. In that case Model 8 (rtvalue_seg1,2,3&4) [Regression Models with Segmented Data] can be used as it takes up significantly less space and can predict close enough to Model 9.

4. Creating a Recommendation Model. (Using Python)
    a. Importing “pandas” & “numpy”

```
import pandas as pd
import numpy as np
```

    b. Importing data and dropping unnecessary fields.

```
data = pd.read_csv('data.csv')
```

```
## Dropping unecessaru fields

fld_drp = ['ID','Sr.Nos.','Weak Foot','Release Clause','Wage','Photo', 'Nationality', 'Flag',
                      'Club Logo', 'International Reputation',
                      'Body Type', 'Real Face','Jersey Number', 'Joined',
                      'Contract Valid Until','LS', 'ST', 'RS', 'LW', 'LF', 'CF', 'RF',
                      'RW','LAM', 'CAM', 'RAM', 'LM', 'LCM', 'CM', 'RCM', 'RM', 'LWB',
                      'LDM','CDM', 'RDM', 'RWB', 'LB', 'LCB','CB', 'RCB', 'RB']
```

```
data.drop(fld_drp, axis = 1, inplace = True)
```

    c. Importing necessary libraries for building the model & creating sub-data base necessary for the same.

```
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.decomposition import PCA
```

```
attribute = data.iloc[:, 15:43]
attribute['Skill Moves'] = data['Skill Moves']
workrate = data['Work Rate'].str.get_dummies(sep = '/ ')
position = data['Position'].str.get_dummies(sep = ' ')
attribute = pd.concat([attribute, workrate, position], axis = 1)
df = attribute
attribute = attribute.dropna()
df['Name'] = data['Name']
df['Overall'] = data['Overall']
df['Potential'] = data['Potential']
df['Age'] = data['Age']
df['Value'] = data['Value']
df['Position'] = data['Position']
df = df.dropna()
```

```
print(attribute.columns)
```

```
Index(['Finishing', 'HeadingAccuracy', 'ShortPassing', 'Volleys', 'Dribbling',
       'Curve', 'FKAccuracy', 'LongPassing', 'BallControl', 'Acceleration',
       'SprintSpeed', 'Agility', 'Reactions', 'Balance', 'ShotPower',
       'Jumping', 'Stamina', 'Strength', 'LongShots', 'Aggression',
       'Interceptions', 'Positioning', 'Vision', 'Penalties', 'Composure',
       'Marking', 'StandingTackle', 'SlidingTackle', 'Skill Moves', 'High',
       'Low', 'Medium', 'CB', 'CM', 'DM', 'FORWARD', 'GK', 'LB', 'RB',
       'WINGER'],
      dtype='object')
```

d. Creating the Model

```
scaler = StandardScaler()
s = scaler.fit_transform(attribute)
```

```
recommendor = NearestNeighbors(n_neighbors = 6, algorithm = "ball_tree")
recommendor.fit(s)
```

```
NearestNeighbors(algorithm='ball_tree', leaf_size=30, metric='minkowski',
                 metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                 radius=1.0)
```

```
p_index = recommendor.kneighbors(s)[1]
```

```
p_index
```

```
array([[    0,     5,    15,    23,   151,   403],
       [    1,   124,    87,   315,   461,   518],
       [    2,    65,    25,   157,    68,    94],
       ...,
       [18156, 18015, 17154, 17956, 17994, 17968],
       [18157, 17868, 18071, 17493, 17969, 18041],
       [18158, 18042, 18086, 18009, 18136, 18051]], dtype=int64)
```

e. The Recommendation Function

```
def g_index(x):
    return df[df['Name'] == x].index.tolist()[0]

def recommend(player):
    print("Players similar to {} are : ".format(player))
    index = g_index(player)
    for i in p_index[index][1:]:
        print(df.iloc[i]['Name'],
              df.iloc[i]['Age'],
              df.iloc[i]['Overall'],
              df.iloc[i]['Potential'],
              df.iloc[i]['Position'],
              df.iloc[i]['Value'])
```

f. Getting Recommendations

- L. Messi

```
recommend('L. Messi')
```

```
Players similar to L. Messi are :
E. Hazard 27 91 91 FORWARD 93000000.0
P. Dybala 24 89 94 FORWARD 89000000.0
S. Agüero 30 89 89 FORWARD 64500000.0
A. Gómez 30 84 84 FORWARD 30000000.0
F. Quagliarella 35 81 81 FORWARD 8000000.0
```

- K. De Bruyne

```
recommend('K. De Bruyne')
```

```
Players similar to K. De Bruyne are :
D. Alli 22 84 90 CM 42500000.0
Roberto Firmino 26 86 87 CM 53000000.0
L. Modrić 32 91 91 CM 67000000.0
A. Ramsey 27 82 83 CM 25000000.0
H. Ziyech 25 83 86 CM 32500000.0
```

- Cristiano Ronaldo

```
recommend('Cristiano Ronaldo')
```

```
Players similar to Cristiano Ronaldo are :
M. Depay 24 84 89 FORWARD 42000000.0
A. Lacazette 27 85 86 FORWARD 45000000.0
David Villa 36 82 82 FORWARD 8000000.0
T. Vaclík 29 80 81 GK 11500000.0
Antunes 31 80 80 LB 9500000.0
```

- Neymar Jr.

```
recommend('Neymar Jr')
```

```
Players similar to Neymar Jr are :
Douglas Costa 27 86 86 WINGER 46500000.0
K. Mbappé 19 88 95 WINGER 81000000.0
Ronaldo Cabrais 26 83 83 WINGER 28000000.0
M. Reus 29 86 86 WINGER 43500000.0
Y. Brahimi 28 85 85 WINGER 39000000.0
```

- De Gea

```
recommend('De Gea')
```

```
Players similar to De Gea are :
J. Pickford 24 83 88 GK 25000000.0
K. Schmeichel 31 84 84 GK 19000000.0
Ederson 24 86 90 GK 41500000.0
M. Neuer 32 89 89 GK 38000000.0
J. Guilavogui 27 79 80 DM 12000000.0
```

- Cesc Fabregas

```
recommend('Cesc Fàbregas')
```

```
Players similar to Cesc Fàbregas are :
Borja Valero 33 81 81 CM 11000000.0
João Moutinho 31 81 81 CM 15000000.0
M. Hamšík 30 87 87 CM 46500000.0
I. Gündoğan 27 84 84 CM 32000000.0
J. Pastore 29 82 82 CM 21500000.0
```

- T. Werner

```
recommend('T. Werner')
```

```
Players similar to T. Werner are :
K. Laimer 21 75 82 RB 8000000.0
I. Opara 29 75 76 CB 6000000.0
L. Bailey 20 81 89 WINGER 26500000.0
K. Huntelaar 34 77 77 FORWARD 4700000.0
M. Pucciarelli 27 74 75 FORWARD 6500000.0
```

Conclusion:

*All the file of this project can be found on (Github link) and the visualization can be found on (Tableau Public link)