ALY 6015

Intermediate Analytics

FINAL PROJECT REPORT

GROUP – A

- **Dimple Chowdhari**
- **Krishna Prakash Sankaramanchi**
- **Saloni Bhansali**
- **Sreelakshmi Sneha Mulukutla**

CONTENTS:

INTRODUCTION:

Problem Statement:

FIFA is followed by billions of fans worldwide. The revenue generated is not only from soccer matches but associated advertisement and merchandise as well. The key driving factor is how much a player is rated as it drives not only the success in matches but also the viewership each player brings to the clubs. There are various features which drive the value of the players. Each year there is a bidding war for soccer players who transfer between clubs and this is a widely monitored activity in sports fans around the world. This project will focus on predicting the value of the players using various regression methods to derive a model that explains the variance of our dataset to a greater extent.

Dataset:

The dataset for the group project is FIFA Dataset available on Kaggle.com which has 89 features like age, nationality, fitness and other characteristics of the players. The data also includes ratings of the players in multiple positions on the soccer field.

Variables in the dataset:

ID, Name, Age, Photo, Nationality, Flag, Overall, Potential, Club, Club Logo, Value, Wage, Special, Preferred Foot, International Reputation, Weak Foot, Skill Moves, Work Rate, Body Type, Real Face, Position, Jersey Number, Joined, Loaned From, Contract Valid Until, Height
Weight, Positions (from LS – RB), 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 and Release Clause.

Methods:

To predict a model for our dataset, the following data cleaning and regression methods are being used:

- MICE package, as a multiple imputation tool to replace the NAN values from our dataset
- Linear regression method
- Lasso Regression method
- PCA
- Decision Trees
- Gradient Boosted Method
- Random Forest Regression

DATA CLEANING:

i) We had NAN values in many of our columns, since the ratio of NAN values to the entire dataset is high, we have used MICE package to replace these unknown values with the values computed my MICE package. The missing values in the dataset are imputed using the MICE package as a model cannot perform well if there are missing values. The missing values can be corrected by either omitting the missing values or by replacing missing values by mean, median or mode value for the respective columns or the imputation technique using MICE package. It will simulate all possible data patterns and assign the best value out of it that fits the missing columns.

The following is the code and outputs describing the NAN values removal:

```
md.pattern(fifa[,c(12,13,16:18,23,18,55:89)])
sample1 = fifa[,c(12,13,16:18,23,55:89)]
colSums(is.na(sample1))
fifa1 = mice(sample1, m=5)
completeData = complete(fifa1,5)
fifa[,c(12,13,16:18,23,27,55:89)] = completeData
colSums(is.na(fifa))

## AFTER RUNNING MICE PACKAGE THE OUTPUT SHOWED NO NAN VALUES

> colSums(is.na(fifa))
                       X                       ID                     Name                      Age 
                       0                        0                        0                        0 
                   Photo              Nationality                     Flag                  Overall 
                       0                        0                        0                        0 
               Potential                     Club                Club.Logo                    Value 
                       0                        0                        0                        0 
                    Wage                  Special           Preferred.Foot International.Reputation 
                       0                        0                        0                        0 
               Weak.Foot              Skill.Moves                Work.Rate                Body.Type 
                       0                        0                        0                        0 
               Real.Face                 Position            Jersey.Number                   Joined 
                       0                        0                        0                        0 
             Loaned.From      Contract.Valid.Until                   Height                   Weight 
                       0                        0                        0                        0 
                      LS                       ST                       RS                       LW 
                       0                        0                        0                        0 
                      LF                       CF                       RF                       RW 
                       0                        0                        0                        0 
                     LAM                      CAM                      RAM                       LM 
                       0                        0                        0                        0 
                     LCM                       CM                      RCM                       RM 
                       0                        0                        0                        0 
                     LWB                      LDM                      CDM                      RDM 
                       0                        0                        0                        0 
                     RWB                       LB                      LCB                       CB 
                       0                        0                        0                        0 
                     RCB                       RB                 Crossing                Finishing 
                       0                        0                        0                        0 
```

```
  HeadingAccuracy      ShortPassing          Volleys        Dribbling
              0                0                0                0
          Curve          FKAccuracy        LongPassing      BallControl
              0                0                0                0
    Acceleration         SprintSpeed          Agility        Reactions
              0                0                0                0
        Balance           ShotPower          Jumping          Stamina
              0                0                0                0
       Strength           LongShots        Aggression    Interceptions
              0                0                0                0
    Positioning              Vision         Penalties        Composure
              0                0                0                0
        Marking       StandingTackle     SlidingTackle         GKDiving
              0                0                0                0
     GKHandling            GKKicking     GKPositioning       GKReflexes
              0                0                0                0
 Release.Clause          Position_1
              0                0
```

ii) The cell values contain alphanumeric variables as well as symbols for currency and height. These have been removed and converted to numeric values:

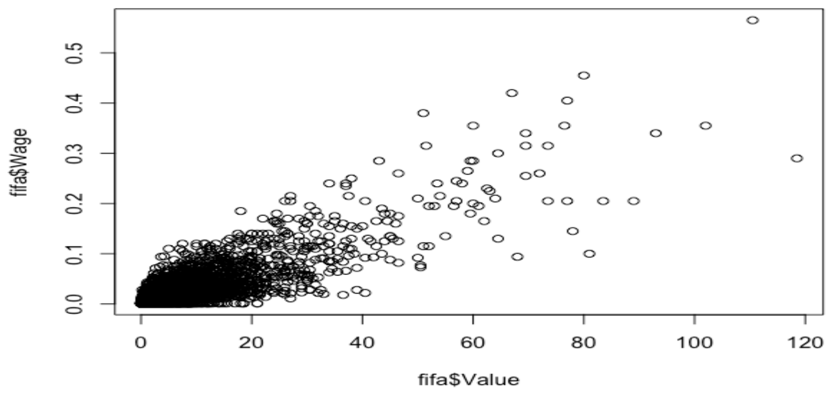
```
fifa$Weight = as.numeric(gsub("\\lbs", "", fifa$Weight))

fifa$Value <- ifelse(str_detect(fifa$Value, "M")==TRUE, as.numeric(gsub("[\\€M]", "",fifa$Value)),
 ifelse(str_detect(fifa$Value, "K")==TRUE, as.numeric(gsub("[\\€K]", "", fifa$Value))/1000,NA))

fifa$Release.Clause <- ifelse(str_detect(fifa$Release.Clause, "M")==TRUE,
as.numeric(gsub("[\\€M]", "",fifa$Release.Clause)), ifelse(str_detect(fifa$Release.Clause,
"K")==TRUE, as.numeric(gsub("[\\€K]", "", fifa$Release.Clause))/1000, NA))

fifa$Wage <- ifelse(str_detect(fifa$Wage, "M")==TRUE, as.numeric(gsub("[\\€M]", "",fifa$Wage)),
        ifelse(str_detect(fifa$Wage, "K")==TRUE, as.numeric(gsub("[\\€K]", "",
fifa$Wage))/1000, NA))
```

iv)We have also found certain values in our dataset that should be added to get our final values. Hence, we coded as follows to do a summation on those values for getting our final dataset after data cleaning.

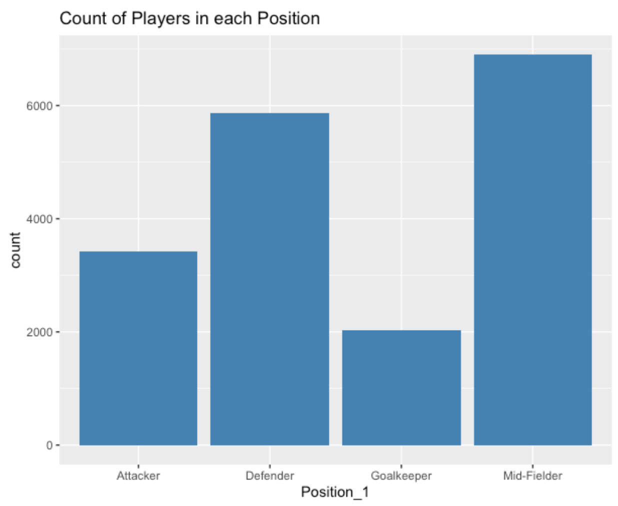
```
fifa[,27] = as.character(fifa[,27])

for(i in 1:18207){
   b = as.numeric(unlist(strsplit(fifa[i,27], "\\'")))
   fifa[i,27] = b[1]*30.48 + b[2]*2.54}

for(i in 29:54){
  fifa[,i] = as.character(fifa[,i])}


for(i in 1:18207) { for(j in 29:54){  fifa[i,j] = sum(as.numeric(unlist(strsplit(fifa[i,j], "\\+"))))  }}

for(i in 27:54){  fifa[,i] = as.numeric(fifa[,i])
```

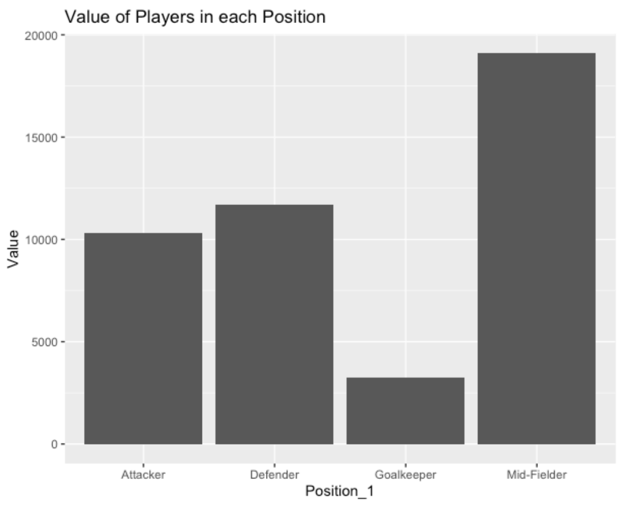
EXPLORATORY DATA ANALYSIS:

The next step is identifying the relationships between various variables. We wanted to see how our predictor variables vary with other independent variables for which we have performed the following EDA. The below graph represents how the dependent variable ‘Value’ varies with the independent variable ‘Wage’.



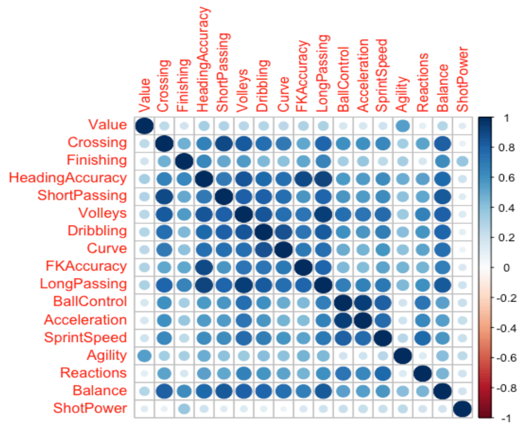
The bar graphs show that the highest number of players are mid-fielders and their value is also highest in the group. The other important feature is that the value of attacker and defenders is not significantly different, but the number of defenders is almost twice the attackers.

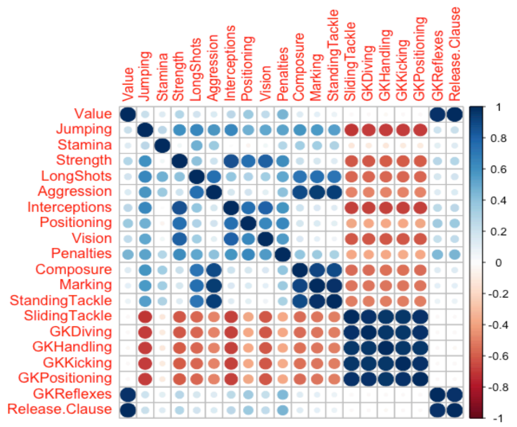




Plotting the correlation plots reveal that not all variables are closely related when mapping value of players with their skills in different areas. Agility, Goalkeeper reflexes as a skill is showing the highest correlation with the value of the players. But there are a lot of variables and it is not very clear which are the most distinctive and highly related variables in deciding the value of a player. This will be further improved by using the regularization.

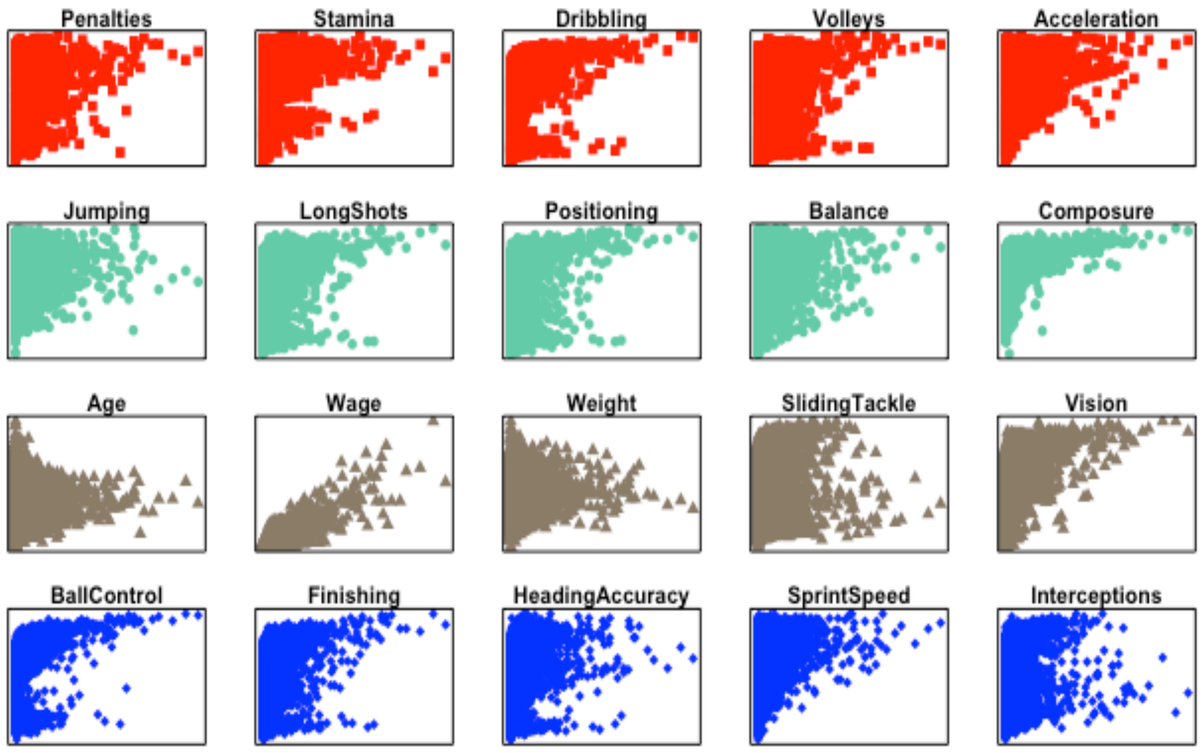
```
cor = cor(fifa[,c(12,55:70)])

cor1 = cor(fifa[,c(12,71:89)])

corrplot(cor, method = "circle")

corrplot(cor1, method = "circle")

plot(fifa$Value, fifa$Wage)

fifa$Position_1 = ifelse(fifa$Position == "CF" | fifa$Position == "LF" | fifa$Position == "LS" |
fifa$Position == "LW" | fifa$Position == "RF" | fifa$Position == "RS" | fifa$Position == "RW" |
fifa$Position == "ST", fifa$Position_1 <- "Attacker",
     ifelse(fifa$Position == "GK", fifa$Position_1 <- "Goalkeeper", ifelse(fifa$Position == "CB" |
fifa$Position == "LB" | fifa$Position == "LCB" | fifa$Position == "LWB" | fifa$Position == "RB" |
fifa$Position== "RCB" | fifa$Position == "RWB", fifa$Position_1 <- "Defender", fifa$Position_1 <-
"Mid-Fielder")))

ggplot(data = fifa, aes(Position_1)) + geom_bar(fill = "steelblue") + ggtitle("Count of Players in each
Position")

ggplot(data = fifa, aes(x=Position_1, y=Value)) + geom_bar(stat = "identity") + ggtitle("Value of
Players in each Position")
```





Performed more EDA to explore how the dependent variable "Value" varies with predictor variables. This will give us a clear picture on how we should choose a regression model and how we can eliminate certain predictor variables that are of least importance and can scatter away our model from answering our business problem. The following is the code representing scatter plot between dependent variable "Value" and few predictor variables.

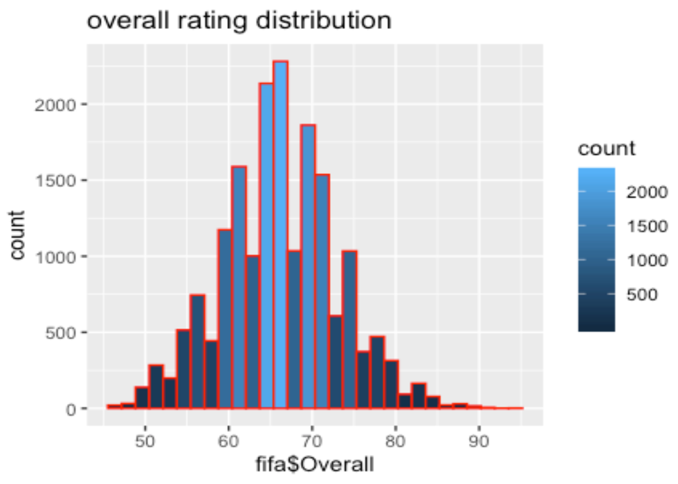
```
par(mfrow=c(1,1))
par(mfrow=c(4,5))
plot(fifa$Value,fifa$Penalties,main="Penalties",axes=FALSE, frame.plot=TRUE,cex.main=1, pch = 15, col = "red")
plot(fifa$Value,fifa$Stamina,main="Stamina",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 15, col = "red")
plot(fifa$Value,fifa$Dribbling,main="Dribbling",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 15, col = "red")
plot(fifa$Value,fifa$Volleys,main="Volleys",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 15, col = "red")
plot(fifa$Value,fifa$Acceleration,main="Acceleration",axes=FALSE,
frame.plot=TRUE,cex.main=1,pch = 15, col = "red")
plot(fifa$Value,fifa$Jumping,main="Jumping",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 16, col = "aquamarine3")
plot(fifa$Value,fifa$LongShots,main="LongShots",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 16, col = "aquamarine3")
plot(fifa$Value,fifa$Positioning,main="Positioning",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 16, col = "aquamarine3")
plot(fifa$Value,fifa$Balance,main="Balance",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 16, col = "aquamarine3")
plot(fifa$Value,fifa$Composure,main="Composure",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 16, col = "aquamarine3")
plot(fifa$Value,fifa$Age,main="Age",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 17, col = "bisque4")
plot(fifa$Value,fifa$Wage,main="Wage",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 17, col = "bisque4")
plot(fifa$Value,fifa$Weight,main="Weight",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 17, col = "bisque4")
plot(fifa$Value,fifa$SlidingTackle,main="SlidingTackle",axes=FALSE,
frame.plot=TRUE,cex.main=1,pch = 17, col = "bisque4")
plot(fifa$Value,fifa$Vision,main="Vision",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 17, col = "bisque4")
plot(fifa$Value,fifa$BallControl,main="BallControl",axes=FALSE,
frame.plot=TRUE,cex.main=1,pch = 18, col = "blue")
plot(fifa$Value,fifa$Finishing,main="Finishing",axes=FALSE, frame.plot=TRUE,cex.main=1,pch = 18, col = "blue")
plot(fifa$Value,fifa$HeadingAccuracy,main="HeadingAccuracy",axes=FALSE,
frame.plot=TRUE,cex.main=1,pch = 18, col = "blue")
plot(fifa$Value,fifa$SprintSpeed,main="SprintSpeed",axes=FALSE,
frame.plot=TRUE,cex.main=1,pch = 18, col = "blue")
plot(fifa$Value,fifa$Interceptions,main="Interceptions",axes=FALSE,
frame.plot=TRUE,cex.main=1,pch = 18, col = "blue")
```



The below graph represents the 'Overall' rating of the players with the highest rating assigned to a single player. We can see that the graph is normally distributed and the average range is 60-70.

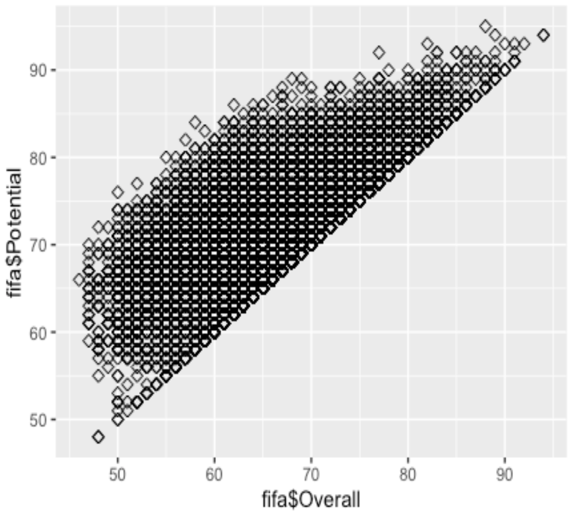
### Plot based Overall rating distribution:

g_overall <- ggplot(data = fifa, aes(fifa$Overall))
g_overall + geom_histogram(col="red", aes(fill = ..count..)) + ggtitle("overall rating distribution")



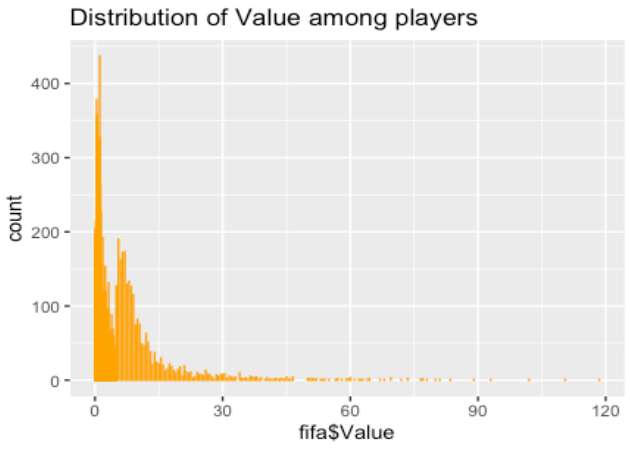
As a part of our effort to understand how the individual parameters are related to one another and how they vary, a plot has been used to substantiate our evidence of the relationship between variables Overall and Potential. This plot is primarily drawn to know if a player plays to his full potential has got a higher Overall rating or not. With our observation from the code, we have come to know that there is a linearity between Overall rating and potential as explained in following graph.

```
### PLot based Overall vs Potential:
g_overall2 <- ggplot(data = fifa, aes(x=fifa$Overall, y=fifa$Potential))+ geom_point(size=2, shape=23)
g_overall2
```



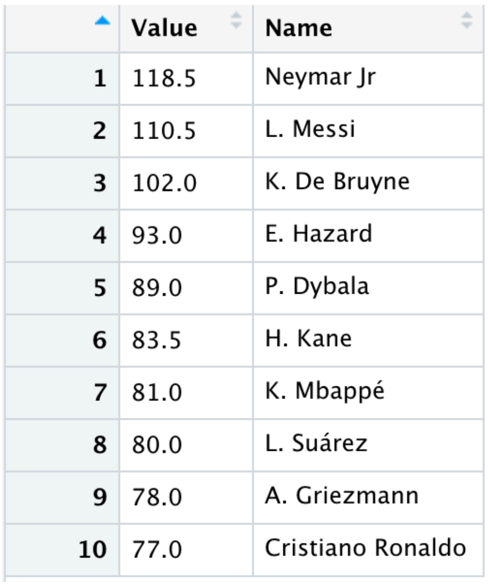
The density plot below shows that there are only 4-5 players with values higher than 90, while the average is in the range of 0-30. This plot also helps identify the outliers in the population.

```
### PLot based on distribution of value among players:
ggplot(data=fifa, aes(x = fifa$Value)) +
  geom_bar(col = "orange", aes(fill = ..count..)) + ggtitle("Distribution of Value among players")
```

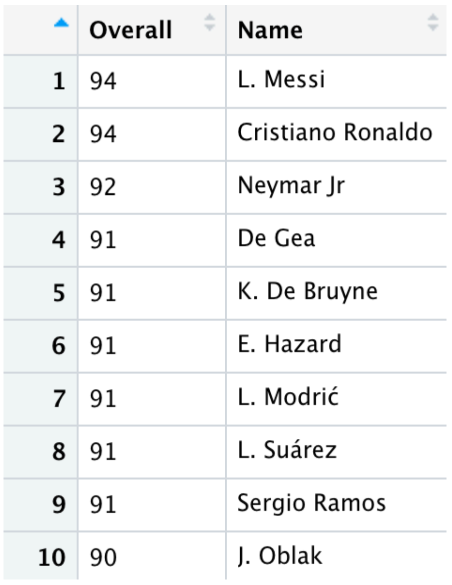


As part of our analysis we wanted to figure out Top 10 players by their Value and also by their Overall rating. We have written the following code to get our Top 10 Plyers tables as following:

```
######Top 10 players by value:

fifa_dist<-fifa %>%
  select(Value, Name) %>%
  arrange(desc(Value))
sliced_data_set<-fifa_dist%>%slice(1:10)
sliced_data_set

#######Top 10 players by overall rating

fifa_dist1<-fifa %>%
  select(Overall, Name) %>%
  arrange(desc(Overall))
sliced_data_set1<-fifa_dist1%>%slice(1:10)
sliced_data_set1
```



	Value	Name
1	118.5	Neymar Jr
2	110.5	L. Messi
3	102.0	K. De Bruyne
4	93.0	E. Hazard
5	89.0	P. Dybala
6	83.5	H. Kane
7	81.0	K. Mbappé
8	80.0	L. Suárez
9	78.0	A. Griezmann
10	77.0	Cristiano Ronaldo



	Overall	Name
1	94	L. Messi
2	94	Cristiano Ronaldo
3	92	Neymar Jr
4	91	De Gea
5	91	K. De Bruyne
6	91	E. Hazard
7	91	L. Modrić
8	91	L. Suárez
9	91	Sergio Ramos
10	90	J. Oblak

HYPOTHESIS TESTING:

As part of our preliminary statistical analysis, a Hypothesis test has been conducted for checking if attacker's value is more than value of defenders. Using the subset of the data for Attacker and Defender positions from main dataset and a sample of size 30, hypothesis test using t-test has been conducted.
Null hypothesis and alternative hypothesis are stated as follows:

Null Hypothesis: Average Value of Attackers is less than that of Defenders
Alternate Hypothesis: The average value of Attackers is greater than that of Defenders.

For confidence interval of 95%.

The p-value in the result turned out to be greater than confidence level. Hence, we fail to reject null hypothesis, but we do not have enough evidence to prove that attackers have greater value than defenders. Following is the code and outputs that explain our analysis.

```
attackers = subset(fifa$Value, fifa$Position_1 == 'Attacker')
defenders = subset(fifa$Value, fifa$Position_1 == 'Defender')
set.seed(123)
attack_sample = sample(attackers,30)
defend_sample = sample(defenders,30)
t.test(attack_sample, defend_sample, alternative= "greater")
```

```

        Welch Two Sample t-test

data:  attack_sample and defend_sample
t = -0.82099, df = 47.934, p-value = 0.7921
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -1.851658       Inf
sample estimates:
mean of x mean of y
 1.770833  2.379333
```

Performing Principal Component Analysis (PCA)

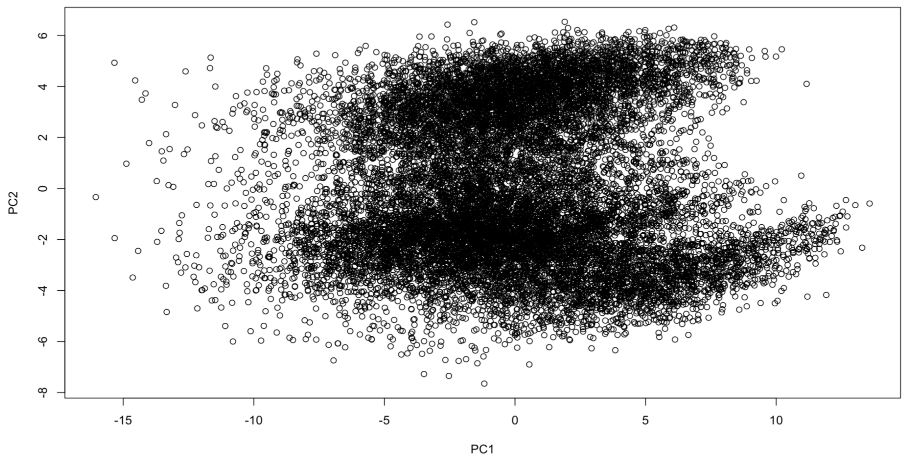
PCA is used when there is a very wide data with multiple variables. PCA draws the best line in k-dimensional dataset such that the distance from each point in the dataset, measured as ‘score’, to the best line is the minimum. The purpose is to look for clustering, outliers or time-based patterns in the score plots. In the below PCA analysis and model based on PCA analysis, PC1-PC4 explain 95% of the variability in dataset with a very high AIC and BIC. The plot between PC1 PC2 is overlapping and does not show any significant groups.

```
fifapca <- fifa %>% filter(!fifa$Position == 'GK')
fifapca <- fifapca %>% filter(!fifapca$Value == '0')
fifapca <- fifapca %>% filter(!fifapca$Height == '0')
fifapca <- fifapca %>% filter(!fifapca$Weight == '0')
fifapca <- data.frame(c(fifapca[4],fifapca[8],fifapca[12], fifapca[27:54]))
nrow(fifapca)
pcafifa <- prcomp(fifapca, center = TRUE, scale. = TRUE)
summary(pcafifa)
```

```
Importance of components:
                        PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
Standard deviation     4.9381 1.54235 1.33711 0.98689 0.94343 0.49446 0.47804 0.25055 0.19451 0.07008
Proportion of Variance 0.7866 0.07674 0.05767 0.03142 0.02871 0.00789 0.00737 0.00202 0.00122
0.00016
Cumulative Proportion  0.7866 0.86336 0.92103 0.95245 0.98116 0.98904 0.99642 0.99844 0.99966
0.99982
                        PC11    PC12    PC13    PC14    PC15      PC16      PC17      PC18      PC19
Standard deviation     0.04557 0.03790 0.02968 0.02632 0.02208 4.876e-15 3.775e-16 3.775e-16 3.775e-
16
Proportion of Variance 0.00007 0.00005 0.00003 0.00002 0.00002 0.000e+00 0.000e+00 0.000e+00
0.000e+00
Cumulative Proportion  0.99989 0.99993 0.99996 0.99998 1.00000 1.000e+00 1.000e+00 1.000e+00
1.000e+00
                         PC20      PC21      PC22      PC23      PC24      PC25      PC26      PC27
Standard deviation     3.775e-16 3.775e-16 3.775e-16 3.775e-16 3.775e-16 3.775e-16 3.775e-16 3.775e-16
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
1.000e+00
                         PC28      PC29      PC30      PC31
Standard deviation     3.775e-16 3.775e-16 3.775e-16 3.726e-16
Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00
Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00
```

```
Call:
lm(formula = fifapca$Value ~ pcafifa$x[, 1] + pcafifa$x[, 2] + 
    pcafifa$x[, 3] + pcafifa$x[, 4])

Residuals:
   Min     1Q Median     3Q    Max 
-8.011 -1.882 -0.284  1.468 61.353
```

```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     2.554545   0.025665   99.53   <2e-16 ***
pcafifa$x[, 1] -0.755706   0.006122 -123.44   <2e-16 ***
pcafifa$x[, 2] -0.088130   0.008513  -10.35   <2e-16 ***
pcafifa$x[, 3]  0.854557   0.018635   45.86   <2e-16 ***
pcafifa$x[, 4]  3.481337   0.026962  129.12   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.238 on 15913 degrees of freedom
Multiple R-squared:  0.6819,     Adjusted R-squared:  0.6819
F-statistic:  8530 on 4 and 15913 DF,  p-value: < 2.2e-16

>
> plot(pcafifa$x[,1:4])
> AIC(pcamodel)
[1] 82587.34
> BIC(pcamodel)
[1] 82633.4
```



Statistical Modeling:
Linear Regression:

Linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. The case of more than one explanatory variable is Multiple Linear Regression.

We are going to follow the Multiple Linear Regression technique to predict the Value of the Player using many explanatory variables.

As we have divided our explanatory variables to Positions Score and Skill Scores, it seems that there are too many explanatory variables. This calls for Feature Engineering which can reduce the variables to an extent.

Hence, we have combined the Positions Scores according the Soccer Terminology and reduced the 27 variables to 3 variables – "Forward", "Midfield" and "Defense". Later, dropped all other Position Score variables

```
test = fifa
```

Taking test to fifa so that we don't have to load from DB again and again

Merging the columns of positions to reduce the number of features (Feature Engineering)

```
test$Forward = rowMeans(fifa[,29:36])
test$Midfield = rowMeans(fifa[,37:44])
test$Defense = rowMeans(fifa[,45:54])
test = test[,-c(29:54)]
```

Goal Keepers have special skills in our data. Hence, we build two models. One for an outfield player (*without GK) and one for Goalkeeper (GK).

```
fifasample1 = subset(test, test$Position != 'GK')
fifasample2 = subset(test, test$Position == 'GK')
```

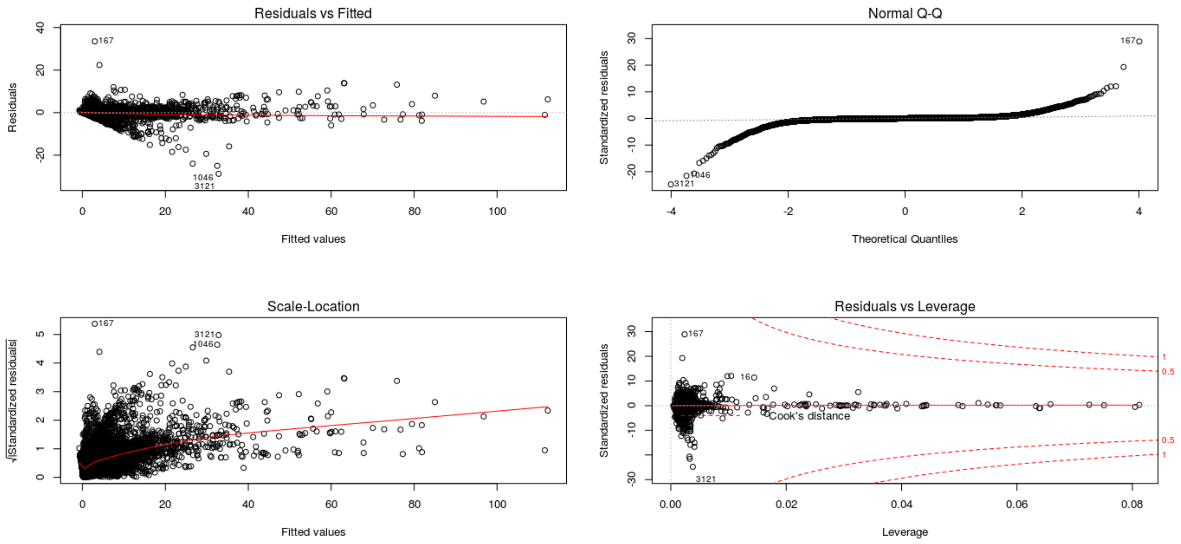
Linear Regression for an Outfield Player

```
firstreg1 = lm(Value ~ Release.Clause + Forward + Midfield + Defense +
Height+Weight+Crossing+Finishing+HeadingAccuracy+ShortPassing+Volleys+Dribbling+Curve+FKAccu
racy+LongPassing+BallControl+Acceleration+SprintSpeed+Agility+Reactions+Balance+ShotPower+Jump
ing+Stamina+Strength+LongShots+Aggression+Interceptions+Positioning+Vision+Penalties+Composure+
Marking+StandingTackle+SlidingTackle
, data = fifasample1)

> summary(firstreg1)

Call:
lm(formula = Value ~ Release.Clause + Forward + Midfield + Defense + Height + Weight + 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, data = fifasample1)

Residuals:
    Min      1Q  Median      3Q     Max
-28.747  -0.162   0.015   0.180  33.530

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2.4696847  0.5052329  -4.888 1.03e-06 ***
Release.Clause   0.4874566  0.0010176 479.034  < 2e-16 ***
Forward        -0.0037072  0.0297172  -0.125  0.90072
Midfield       -0.0043627  0.0355947  -0.123  0.90245
Defense         0.0144754  0.0149682   0.967  0.33352
Height          0.0004352  0.0026956   0.161  0.87175
Weight          0.0029971  0.0010627   2.820  0.00480 **
Crossing       -0.0005632  0.0012161  -0.463  0.64330
Finishing      -0.0032842  0.0029078  -1.129  0.25872
HeadingAccuracy  0.0015961  0.0020381   0.783  0.43355
ShortPassing     0.0014220  0.0034573   0.411  0.68086
Volleys         0.0060815  0.0013013   4.673 2.99e-06 ***
Dribbling      -0.0005583  0.0024225  -0.230  0.81772
Curve          -0.0029775  0.0012717  -2.341  0.01923 *
FKAccuracy       0.0036742  0.0011232   3.271  0.00107 **
LongPassing      0.0003062  0.0029598   0.103  0.91762
BallControl      0.0003964  0.0027408   0.145  0.88500
Acceleration     0.0035030  0.0019664   1.781  0.07486 .
SprintSpeed      0.0003036  0.0019115   0.159  0.87379
Agility        -0.0023061  0.0014598  -1.580  0.11419
Reactions       0.0143853  0.0019476   7.386 1.58e-13 ***
Balance         0.0026092  0.0014614   1.785  0.07420 .
ShotPower       0.0006825  0.0021597   0.316  0.75200
Jumping        -0.0002694  0.0009661  -0.279  0.78037
Stamina         0.0045740  0.0015025   3.044  0.00234 **
Strength        0.0001911  0.0016496   0.116  0.90777
LongShots      -0.0026992  0.0014945  -1.806  0.07091 .
Aggression     -0.0003914  0.0011369  -0.344  0.73064
Interceptions  -0.0019767  0.0021841  -0.905  0.36545
Positioning     0.0005726  0.0020436   0.280  0.77933
Vision          0.0036909  0.0032901   1.122  0.26196
Penalties       0.0004681  0.0012541   0.373  0.70895
Composure       0.0037562  0.0015946   2.356  0.01851 *
Marking         0.0013250  0.0018631   0.711  0.47697
StandingTackle  -0.0036313  0.0026643  -1.363  0.17293
SlidingTackle   -0.0041341  0.0024455  -1.690  0.09095 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.162 on 16146 degrees of freedom
Multiple R-squared:  0.9586,     Adjusted R-squared:  0.9586
F-statistic: 1.069e+04 on 35 and 16146 DF,  p-value: < 2.2e-16
```

We can see that our R-Squared is 95.86% which is considerably goo and p-value is very less. That signifies that our model is performing well.

```
> AIC(firstreg1)
[1] 50817.16
> BIC(firstreg1)
[1] 51101.75
```

This is the best AIC and BIC values for a set of models we have executed. We have optimized our model for the best R-Squared, AIC and BIC.
The below regression plots also confirm that the model is the best fit.



1. We can see that the Residuals vs Fitted values are completely random and there is no pattern and the red line is mostly a straight line. This is one of the factors explaining the model performed well.
2. The normal Q-Q plot which is a normal quantile-quantile plot has a diagonal shape which checks for homoscedasticity and the model is performing well.
3. The graph between Fitted values and the square root of Standardized Residuals also shows that the standardized residuals do not have any pattern.

Linear Regression for GK

For the Goalkeeper's model, we will only be considering those features which are relevant to the Goalkeeper. Let's check how the model performs.

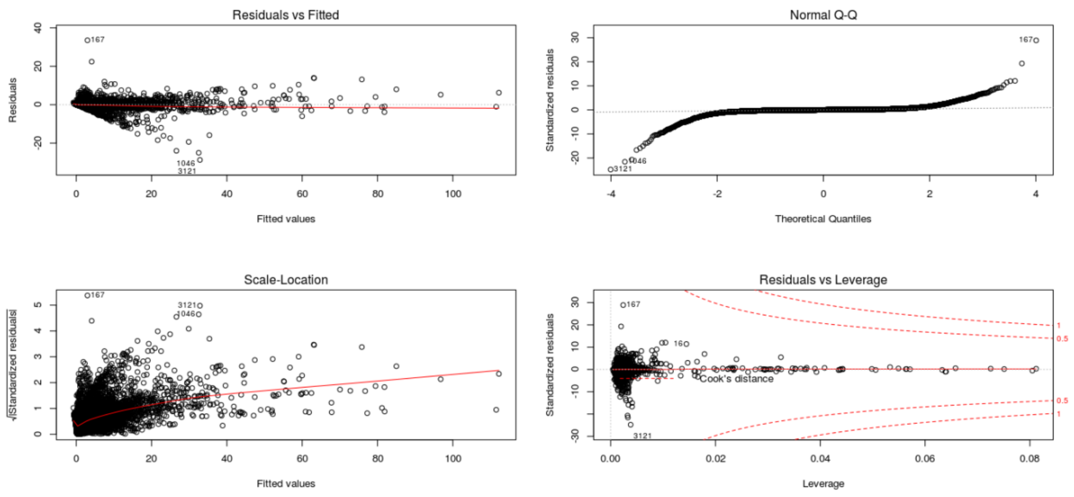
```
> summary(firstreg2)

Call:
lm(formula = Value ~ Release.Clause + Forward + Midfield + Defense + Height + Weight + GKDiving +
GKHandling + GKKicking + GKPositioning + GKReflexes, data = fifasample2)

Residuals:
    Min       1Q   Median      3Q      Max
-11.3909  -0.1177  -0.0027   0.0993   9.3218

Coefficients: (3 not defined because of singularities)
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    -2.3941439  0.6809793  -3.516 0.000448 ***
Release.Clause  0.4896005  0.0021705 225.572  < 2e-16 ***
Forward               NA         NA      NA       NA
```

```
Midfield              NA         NA      NA       NA
Defense               NA         NA      NA       NA
Height         0.0051637  0.0040568   1.273 0.203223
Weight        -0.0005091  0.0014426  -0.353 0.724184
GKDiving       0.0014517  0.0052695   0.275 0.782962
GKHandling     0.0111730  0.0046799   2.387 0.017057 *
GKKicking     -0.0004424  0.0032516  -0.136 0.891801
GKPositioning -0.0029496  0.0042347  -0.697 0.486174
GKReflexes     0.0158851  0.0049737   3.194 0.001426 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7259 on 2016 degrees of freedom
Multiple R-squared:  0.975,	Adjusted R-squared:  0.9749
F-statistic:  9821 on 8 and 2016 DF,  p-value: < 2.2e-16
```

We can see that our R-Squared is 97.49% which is better than the model for an outfield player and the p-value is very less. That signifies that our model is performing well. Let's check at AIC and BIC values of the model.

```
> AIC(firstreg2)
[1] 4460.31
> BIC(firstreg2)
[1] 4516.443
```

This is the best AIC and BIC values for a set of models we have executed. We have optimized our model for the best R-Squared, AIC and BIC.
The below regression plots also confirm that the model is the best fit.



1. We can see that the Residuals vs Fitted values are completely random and there is no pattern and the red line is significantly a straight line. This is one of the factors explaining the model performed well.
2. The normal Q-Q plot which is a normal quantile-quantile plot has a diagonal shape which checks for homoscedasticity and the model is performing well
3. The graph between Fitted values and the square root of Standardized Residuals also shows that the standardized residuals do not have any pattern.

Overall, the model performed well.

LASSO REGRESSION:

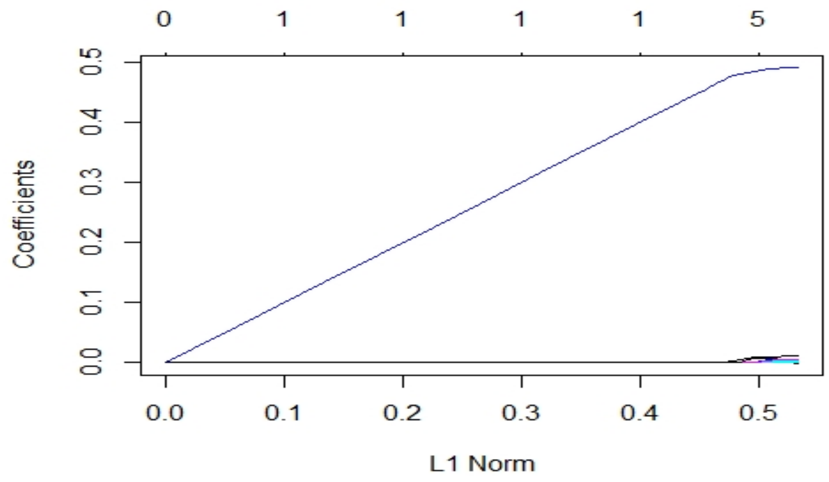
LASSO performs regularization and variable selection on a given model. It shrinks less relevant predictors to zero. Two lasso regression models were carried as the model acts different for an outfield player and for a Goalkeeper perhaps due to the varied skill set.

The first lasso regression model is excluding the goalkeeper but including the attackers, defenders and midfielder players. The first step is to convert the data into matrix to be able to perform lasso regression. Sub setting of the fifa dataset was done for predictors variables. The predictor variables for these football players are mainly comprised of the statistics of offensive, scoring and defensive. x1 are the independent variables while y1 is the dependent variable. The first plot indicates how much the coefficients are penalized for different values of lambda and at which stage each coefficient shrinks to zero. Majority of the variables shrink to zero at point L1 Norm 0.5. The cv.glmnet function gets the cross validation curve.

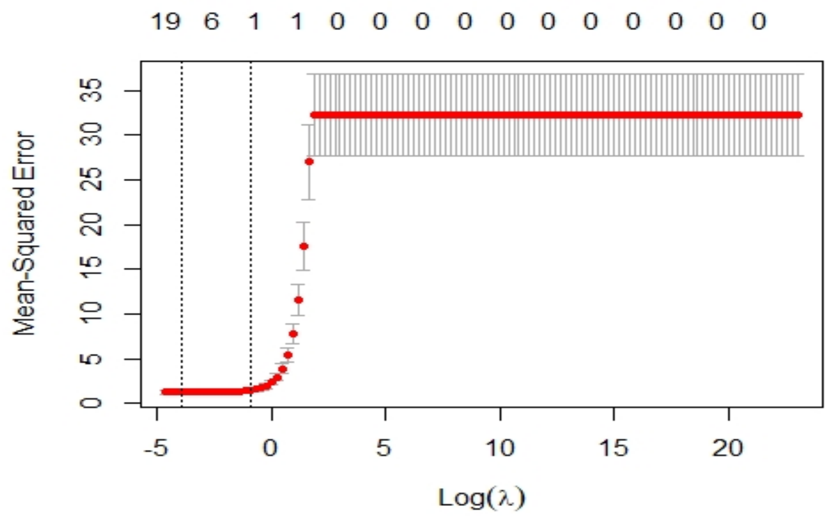
The value of lambda (cv_fit1$lambda.min) that minimizes the mean cross validation error is 0.01995262. The second plot indicates that the log of the optimal value of lambda (i.e. the one that minimizes the root mean square error) is approximately -5. The beta matrix results show that some coefficients have shrunk to zero. This indicates which predictors are important in explaining the variation in y such as release clause, midfield, defense etc.

```
#LASSO
###lasso regression excluding goalkeeper
library(glmnet)
x1 <- data.matrix(fifasample1[c(27:57,63,65:67)])#converting to matrix to be able to perform lasso
regression
y1 <- fifasample1$Value
lambdas <- 10^seq(10, -2, by = -.1)
fifafit1 <- glmnet(x1, y1, alpha = 1, lambda = lambdas, nlambda = 1000)
summary(fifafit1)
plot(fifafit1)
cv_fit1 <- cv.glmnet(x1, y1, alpha = 1, lambda = lambdas, nlambda = 1000)
plot(cv_fit1)
cv_fit1$lambda.min
fit_opt1 <- glmnet(x1, y1, alpha = 1, lambda = cv_fit1$lambda.min)
fit_opt1$beta

> x1 <- data.matrix(fifasample1[c(27:57,63,65:67)])#converting to matrix to be able to perform lasso
regression
> y1 <- fifasample1$Value
> lambdas <- 10^seq(10, -2, by = -.1)
> fifafit1 <- glmnet(x1, y1, alpha = 1, lambda = lambdas, nlambda = 1000)
> summary(fifafit1)
          Length Class     Mode
a0         121   -none-    numeric
beta      4235   dgCMatrix S4
df         121   -none-    numeric
dim          2   -none-    numeric
lambda     121   -none-    numeric
dev.ratio  121   -none-    numeric
nulldev      1   -none-    numeric
```

```
npasses      1   -none-    numeric
jerr         1   -none-    numeric
offset       1   -none-    logical
call         6   -none-    call
nobs         1   -none-    numeric
> plot(fifafit1)
```



```
> cv_fit1 <- cv.glmnet(x1, y1, alpha = 1, lambda = lambdas, nlambda = 1000)
> plot(cv_fit1)
```



```
> cv_fit1$lambda.min
[1] 0.01995262
> fit_opt1 <- glmnet(x1, y1, alpha = 1, lambda = cv_fit1$lambda.min)
> fit_opt1$beta
35 x 1 sparse Matrix of class "dgCMatrix"
```

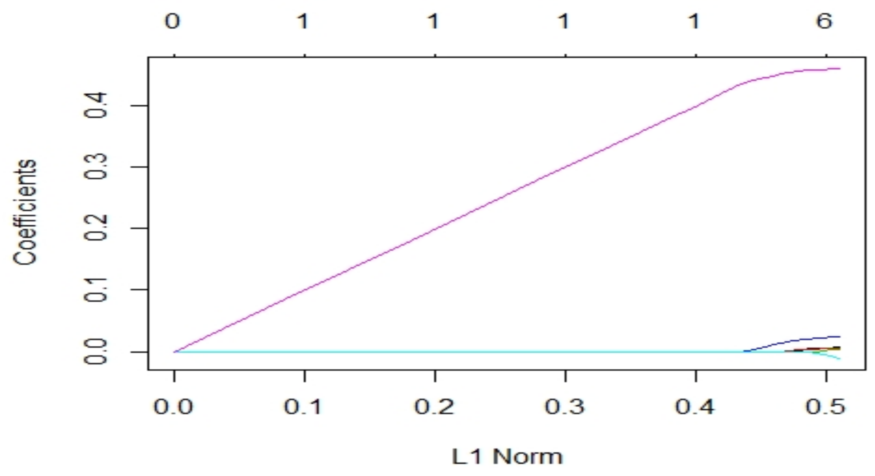
```
                 s0
Height         .
Weight         .
Crossing       .
Finishing      .
HeadingAccuracy 2.889844e-03
ShortPassing   .
Volleys        9.609315e-04
Dribbling      .
Curve          .
FKAccuracy     .
LongPassing    .
BallControl    .
Acceleration   .
SprintSpeed    .
Agility        .
Reactions      4.240428e-03
Balance        .
ShotPower      1.252029e-03
Jumping        .
Stamina        4.654275e-03
Strength       1.446235e-03
LongShots      .
Aggression     .
Interceptions  .
Positioning    .
Vision         .
Penalties      .
Composure      .
Marking        .
StandingTackle .
SlidingTackle  .
Release.Clause  4.899668e-01
Forward        2.573765e-03
Midfield       1.579592e-08
Defense        9.780223e-03
```

The second lasso regression model is for the goalkeepers. The predictor variables for the goalkeepers are mainly comprised of the statistics of goalkeeping. Alpha = 1 was used as it is the lasso penalty. The first plot indicates how much the coefficients are penalized for different values of lambda and at which stage each coefficient shrinks to zero. Majority of the variables shrink to zero at points between 0.4 and 0.5 of L1 Norm.

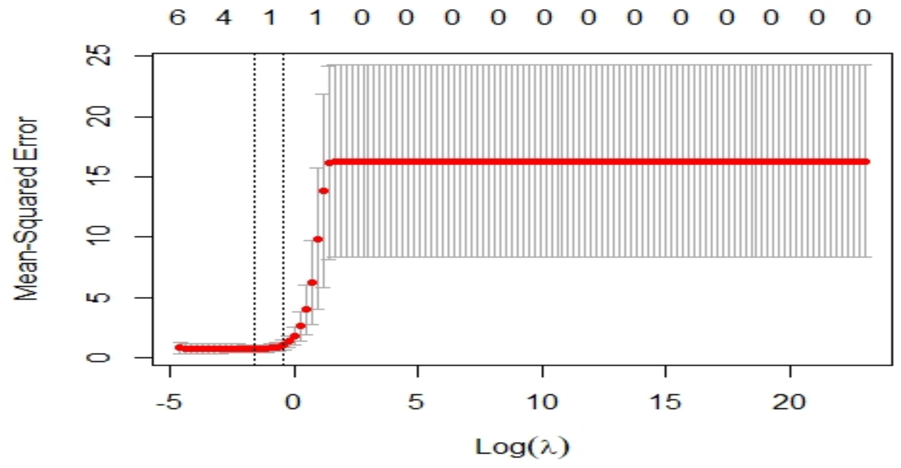
The value of lambda (cv_fit1$lambda.min) that minimizes the mean cross validation error is 0.1995262. The second plot indicates that the log of the optimal value of lambda (i.e. the one that minimizes the root mean square error) is approximately -2. The beta matrix results show that some coefficients have shrunk to zero. This indicates which predictors are important in explaining the variation in y and those are release clause and goalkeeper handling. This is an effective way to narrow down on important predictors when there are many variables.

```
###lasso regression for goalkeeper
x2 <- data.matrix(fifasample2[c(27:28,58:63)])#converting to matrix to be able to perform lasso regression
y2 <- fifasample2$Value
lambdas <- 10^seq(10, -2, by = -.1)
fifafit2 <- glmnet(x2, y2, alpha = 1, lambda = lambdas, nlambda = 1000)
summary(fifafit2)
plot(fifafit2)
cv_fit2 <- cv.glmnet(x2, y2, alpha = 1, lambda = lambdas, nlambda = 1000)
plot(cv_fit2)
cv_fit2$lambda.min
fit_opt2 <- glmnet(x2, y2, alpha = 1, lambda = cv_fit2$lambda.min)
fit_opt2$beta
```

```
###lasso regression for goalkeeper
> x2 <- data.matrix(fifasample2[c(27:28,58:63)])#converting to matrix to be able to perform lasso regression
> y2 <- fifasample2$Value
> lambdas <- 10^seq(10, -2, by = -.1)
> fifafit2 <- glmnet(x2, y2, alpha = 1, lambda = lambdas, nlambda = 1000)
> summary(fifafit2)
          Length Class     Mode
a0        121    -none-    numeric
beta      968    dgCMatrix S4
df        121    -none-    numeric
dim         2    -none-    numeric
lambda    121    -none-    numeric
dev.ratio 121    -none-    numeric
nulldev     1    -none-    numeric
npasses     1    -none-    numeric
jerr        1    -none-    numeric
offset      1    -none-    logical
call        6    -none-    call
nobs        1    -none-    numeric
> plot(fifafit2)
```



```
> cv_fit2 <- cv.glmnet(x2, y2, alpha = 1, lambda = lambdas, nlambda = 1000)
> plot(cv_fit2)
```



```
> cv_fit2$lambda.min
[1] 0.1995262
> fit_opt2 <- glmnet(x2, y2, alpha = 1, lambda = cv_fit2$lambda.min)
> fit_opt2$beta
8 x 1 sparse Matrix of class "dgCMatrix"
                       s0
Height          .
Weight          .
GKDiving        .
GKHandling      0.007331206
GKKicking       .
GKPositioning   .
GKReflexes      .
Release.Clause 0.444685362
```

Decision Trees

Decision Tree for an Outfield Player

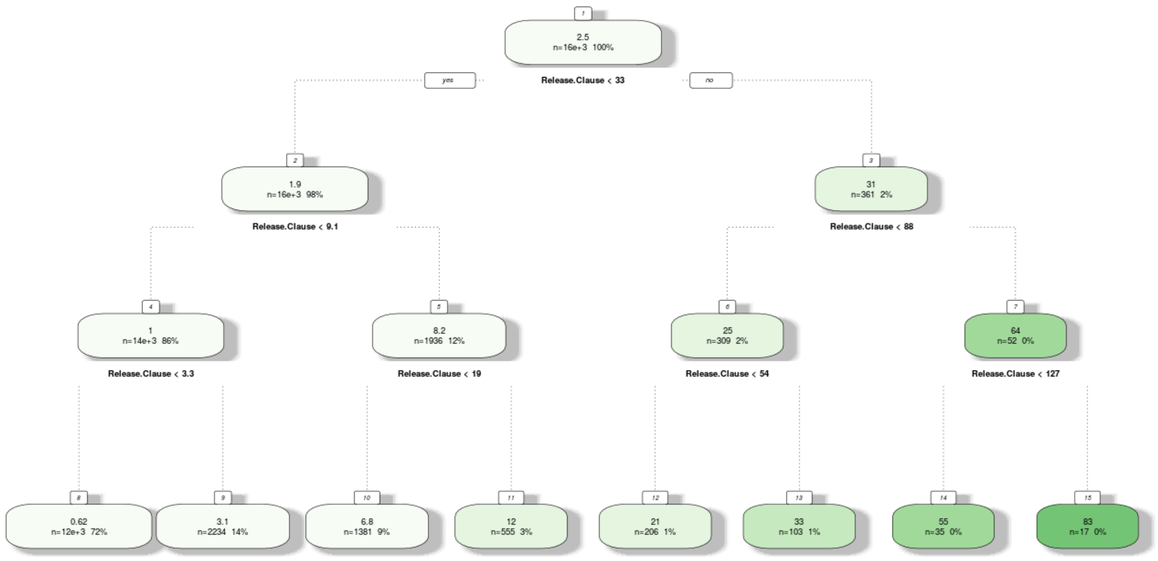
We have considered the same formula as we considered for the Linear Model.

```
> summary(decreg1)
Call:
rpart(formula = Value ~ Release.Clause + Forward + Midfield +
    Defense + Height + Weight + 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, data = fifasample1, method = "anova")
  n= 16182

          CP nsplit  rel error     xerror        xstd
1 0.55025874      0 1.00000000 1.00006109 0.069079225
2 0.16845920      1 0.44974126 0.45583057 0.029706343
3 0.13154345      2 0.28128206 0.28371989 0.029193512
4 0.02165754      3 0.14973861 0.15416197 0.010052485
5 0.02029296      4 0.12808107 0.13139082 0.010130827
6 0.01990554      5 0.10778810 0.11692914 0.010014068
7 0.01752770      6 0.08788257 0.09397012 0.009587661
8 0.01000000      7 0.07035487 0.07620231 0.005634210

Variable importance
Release.Clause       Midfield        Forward      Reactions    BallControl
            41             15             14             11             11
  ShortPassing      Dribbling      Finishing
             5              2              1

Node number 1: 16182 observations,    complexity param=0.5502587
  mean=2.539604, MSE=32.56899
  left son=2 (15821 obs) right son=3 (361 obs)
  Primary splits:
      Release.Clause < 33.1    to the left,  improve=0.5502587, (0 missing)
      Midfield       < 77.6875 to the left,  improve=0.4163648, (0 missing)
      Reactions      < 78.5    to the left,  improve=0.4040149, (0 missing)
      Forward        < 77.8125 to the left,  improve=0.4003221, (0 missing)
      BallControl    < 80.5    to the left,  improve=0.3896946, (0 missing)
  Surrogate splits:
      Midfield     < 79.5625 to the left,  agree=0.986, adj=0.368, (0 split)
      Forward      < 78.9375 to the left,  agree=0.985, adj=0.341, (0 split)
      Reactions    < 81.5    to the left,  agree=0.984, adj=0.280, (0 split)
      BallControl  < 82.5    to the left,  agree=0.984, adj=0.271, (0 split)
      ShortPassing < 82.5    to the left,  agree=0.982, adj=0.197, (0 split)

Node number 2: 15821 observations,    complexity param=0.1684592
```

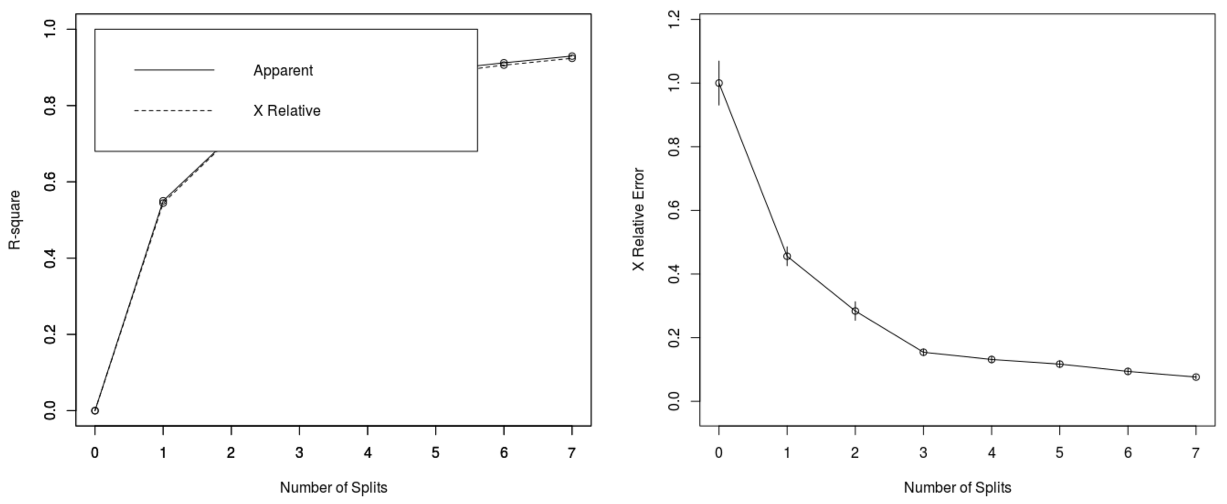
```
  mean=1.900131, MSE=8.320877
  left son=4 (13885 obs) right son=5 (1936 obs)
  Primary splits:
      Release.Clause < 9.05    to the left,  improve=0.6744164, (0 missing)
      Midfield       < 70.3125 to the left,  improve=0.4236089, (0 missing)
      Forward        < 70.9375 to the left,  improve=0.4214766, (0 missing)
      BallControl    < 73.5    to the left,  improve=0.4152495, (0 missing)
      Reactions      < 69.5    to the left,  improve=0.4046865, (0 missing)
  Surrogate splits:
      Forward     < 72.6875 to the left,  agree=0.919, adj=0.337, (0 split)
      Midfield    < 72.1875 to the left,  agree=0.919, adj=0.336, (0 split)
      BallControl < 74.5    to the left,  agree=0.915, adj=0.307, (0 split)
      Reactions   < 73.5    to the left,  agree=0.909, adj=0.253, (0 split)
      Dribbling   < 75.5    to the left,  agree=0.906, adj=0.234, (0 split)

Node number 3: 361 observations,    complexity param=0.1315434
  mean=30.56482, MSE=291.9201
  left son=6 (309 obs) right son=7 (52 obs)
  Primary splits:
      Release.Clause < 88.3    to the left,  improve=0.6578616, (0 missing)
      Forward        < 83.5625 to the left,  improve=0.4943440, (0 missing)
      Midfield       < 85.875  to the left,  improve=0.4435321, (0 missing)
      Reactions      < 88.5    to the left,  improve=0.3799705, (0 missing)
      BallControl    < 89.5    to the left,  improve=0.3798929, (0 missing)
  Surrogate splits:
      Forward     < 84.4375 to the left,  agree=0.917, adj=0.423, (0 split)
      Midfield    < 85.875  to the left,  agree=0.909, adj=0.365, (0 split)
      Reactions   < 88.5    to the left,  agree=0.898, adj=0.288, (0 split)
      BallControl < 89.5    to the left,  agree=0.889, adj=0.231, (0 split)
      Finishing   < 87.5    to the left,  agree=0.886, adj=0.212, (0 split)

Node number 4: 13885 observations,    complexity param=0.02165754
  mean=1.015569, MSE=1.425941
  left son=8 (11651 obs) right son=9 (2234 obs)
  Primary splits:
      Release.Clause < 3.25    to the left,  improve=0.5764985, (0 missing)
      Midfield       < 68.3125 to the left,  improve=0.3015632, (0 missing)
      Forward        < 68.1875 to the left,  improve=0.3009584, (0 missing)
      BallControl    < 68.5    to the left,  improve=0.2789015, (0 missing)
      Reactions      < 65.5    to the left,  improve=0.2673016, (0 missing)
  Surrogate splits:
      Forward     < 69.3125 to the left,  agree=0.880, adj=0.254, (0 split)
      Midfield    < 68.9375 to the left,  agree=0.880, adj=0.254, (0 split)
      BallControl < 71.5    to the left,  agree=0.870, adj=0.194, (0 split)
      Defense     < 70.15   to the left,  agree=0.868, adj=0.177, (0 split)
      Reactions   < 69.5    to the left,  agree=0.858, adj=0.117, (0 split)

Node number 5: 1936 observations,    complexity param=0.01990554
  mean=8.244215, MSE=11.91225
  left son=10 (1381 obs) right son=11 (555 obs)
  Primary splits:
      Release.Clause < 18.95   to the left,  improve=0.4548949, (0 missing)
      BallControl    < 77.5    to the left,  improve=0.1956996, (0 missing)
      Midfield       < 75.6875 to the left,  improve=0.1955383, (0 missing)
      Forward        < 74.6875 to the left,  improve=0.1874178, (0 missing)
      Reactions      < 73.5    to the left,  improve=0.1774117, (0 missing)
  Surrogate splits:
      Midfield    < 76.4375 to the left,  agree=0.755, adj=0.146, (0 split)
      Forward     < 76.6875 to the left,  agree=0.755, adj=0.144, (0 split)
      BallControl < 79.5    to the left,  agree=0.750, adj=0.128, (0 split)
      Defense     < 77.15   to the left,  agree=0.741, adj=0.095, (0 split)
      Reactions   < 77.5    to the left,  agree=0.741, adj=0.095, (0 split)

Node number 6: 309 observations,    complexity param=0.02029296
  mean=24.87994, MSE=70.18723
  left son=12 (206 obs) right son=13 (103 obs)
  Primary splits:
      Release.Clause < 54.35   to the left,  improve=0.4931344, (0 missing)
      Midfield       < 82.25   to the left,  improve=0.2620596, (0 missing)
      Reactions      < 82.5    to the left,  improve=0.2615993, (0 missing)
      Forward        < 81.9375 to the left,  improve=0.2129651, (0 missing)
      Composure      < 78.5    to the left,  improve=0.1774965, (0 missing)
  Surrogate splits:
      Midfield  < 82.6875 to the left,  agree=0.735, adj=0.204, (0 split)
      Forward   < 81.4375 to the left,  agree=0.731, adj=0.194, (0 split)
      Dribbling < 87.5    to the left,  agree=0.706, adj=0.117, (0 split)
      Reactions < 82.5    to the left,  agree=0.706, adj=0.117, (0 split)
      Defense   < 83.05   to the left,  agree=0.689, adj=0.068, (0 split)

Node number 7: 52 observations,    complexity param=0.0175277
  mean=64.34615, MSE=276.3033
  left son=14 (35 obs) right son=15 (17 obs)
  Primary splits:
      Release.Clause < 126.75  to the left,  improve=0.6429424, (0 missing)
      Forward        < 86.8125 to the left,  improve=0.5435158, (0 missing)
      Midfield       < 86.5625 to the left,  improve=0.4835494, (0 missing)
      BallControl    < 89.5    to the left,  improve=0.4571575, (0 missing)
      Reactions      < 87.5    to the left,  improve=0.3990598, (0 missing)
  Surrogate splits:
      Midfield    < 86.5625 to the left,  agree=0.923, adj=0.765, (0 split)
      Forward     < 86.8125 to the left,  agree=0.865, adj=0.588, (0 split)
      BallControl < 89.5    to the left,  agree=0.846, adj=0.529, (0 split)
      Reactions   < 87.5    to the left,  agree=0.827, adj=0.471, (0 split)
      LongShots   < 81.5    to the left,  agree=0.827, adj=0.471, (0 split)

Node number 8: 11651 observations
  mean=0.6185516, MSE=0.2828392

Node number 9: 2234 observations
  mean=3.086139, MSE=2.278259

Node number 10: 1381 observations
```

```
  mean=6.768501, MSE=4.797122

Node number 11: 555 observations
  mean=11.91622, MSE=10.71433

Node number 12: 206 observations
  mean=20.7199, MSE=26.26208

Node number 13: 103 observations
  mean=33.2, MSE=54.20233

Node number 14: 35 observations
  mean=55.05714, MSE=47.62531

Node number 15: 17 observations
  mean=83.47059, MSE=203.7197
```

We can see that there are total 15 Nodes and the details of all the 15 Nodes. Release Clause has the highest variable importance.



That is the information regarding 15 nodes
For 7 nodes, it is being decided regarding Release Clause. That suggests the importance of the Release Clause variable.



We can see that for the number of splits, the R-Squared is more than 90% which is a good sign indicating the good performance of the model.

Using the model to make predictions:

```
> fifasampletest = test[sample(nrow(test), 1),]
> yhat = predict(decreg1, fifasampletest)
> actual = fifasampletest$Value
> rmse = sqrt(mean((yhat-actual)^2))
> rmse
[1] 0.3985516
```

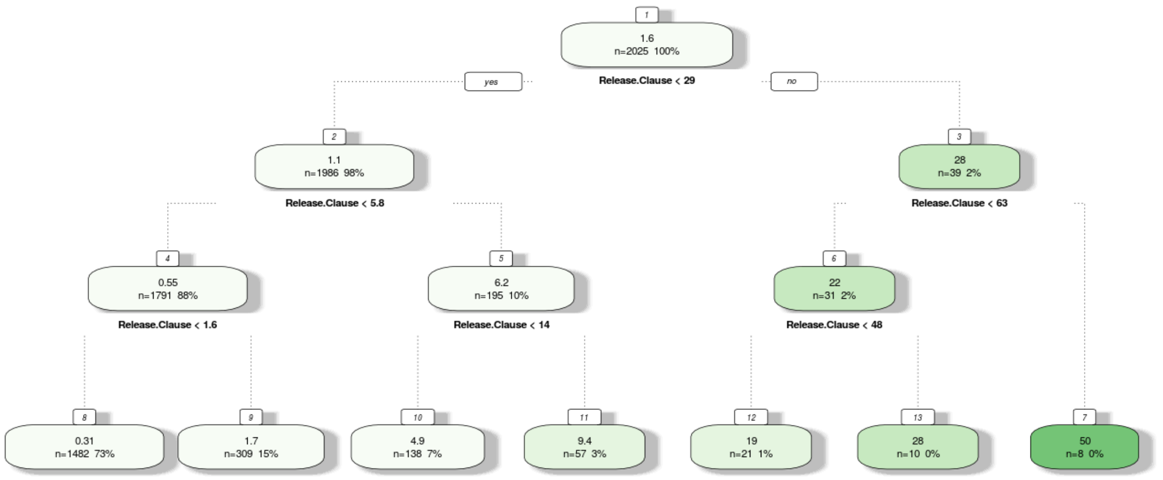
The Root Mean square is 0.398
y-hat (predicted) value is 0.61
The actual value is 0.22 which is actually close. Hence, our model is performing good.

Decision Tree for a Goalkeeper

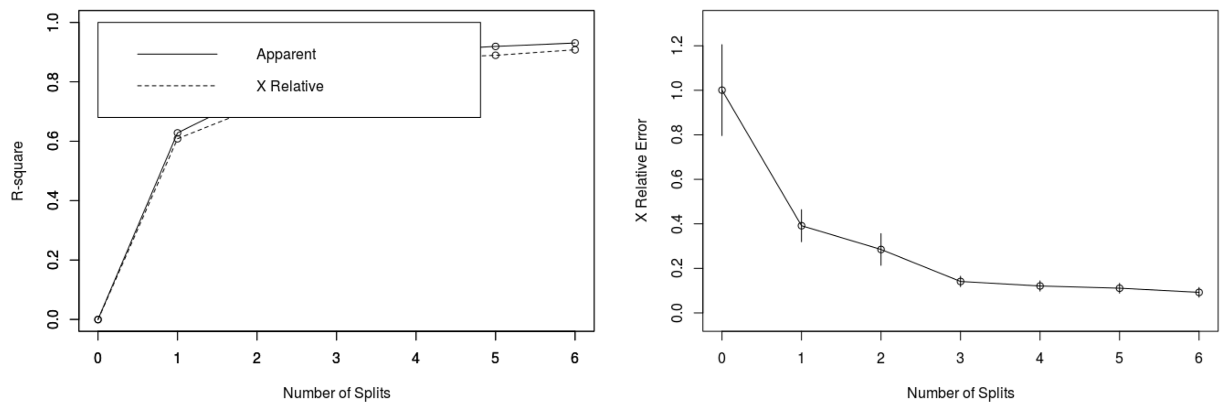
We are considering those features which we have considered for GK model in our linear regression

```
> printcp(decreg2)

Regression tree:
rpart(formula = Value ~ Release.Clause + Forward + Midfield +
    Defense + Height + Weight + GKDiving + GKHandling + GKKicking +
    GKPositioning + GKReflexes, data = fifasample2, method = "anova")

Variables actually used in tree construction:
[1] Release.Clause

Root node error: 42462/2025 = 20.969
```

```
n= 2025

        CP nsplit rel error   xerror     xstd
1 0.628145      0  1.000000 1.000779 0.204309
2 0.132213      1  0.371855 0.391642 0.071963
3 0.124565      2  0.239643 0.284834 0.071232
4 0.019748      3  0.115078 0.141125 0.023503
5 0.014477      4  0.095330 0.120861 0.023003
6 0.011636      5  0.080853 0.110753 0.022147
7 0.010000      6  0.069216 0.092236 0.021736
```



There are 13 nodes in total. In this model as well, the Release Clause is the dominant variable.



Unlike the model for an outfield player, there is a difference of R-Squared for Apparent and X-Relative after the first split. But the overall R-Squared is more than 90% which suggests a good model.

```
> fifasampletest1 = subset(test, test$Position == "GK")
> yhat1 = predict(decreg2, fifasampletest1)
> actual = fifasampletest1$Value
> rmse = sqrt(mean((yhat1-actual)^2))
> rmse
[1] 1.20474
```

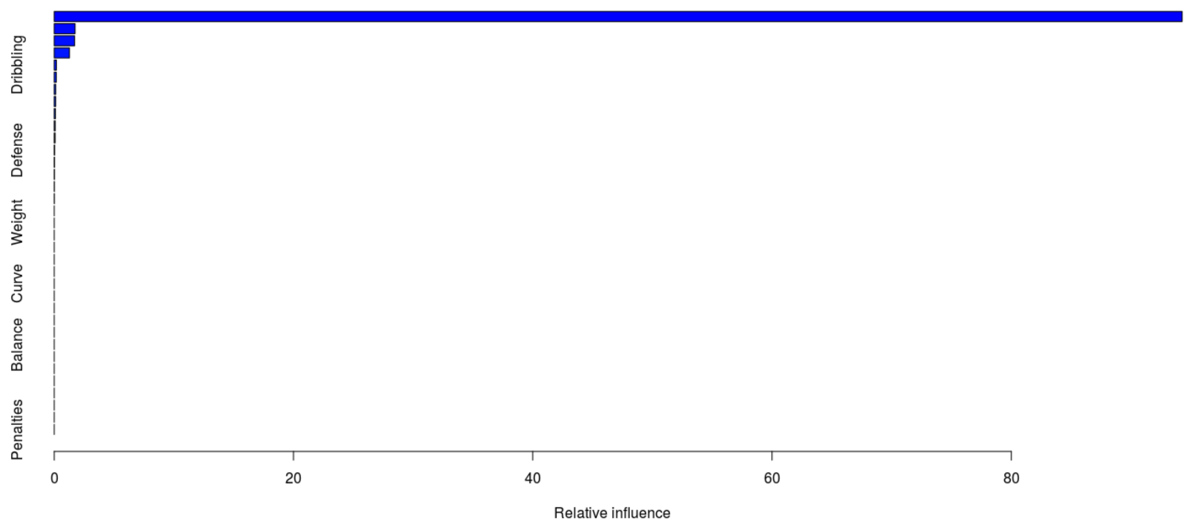
The Root Mean Squared Error is 1.2 which is fairly good.

Gradient Boosted Machine

Trying GBM model to check if there will be increase in performance of the model.

```
decreg2 = gbm(Value ~ Release.Clause + Forward + Midfield + Defense +
Height+Weight+Crossing+Finishing+HeadingAccuracy+ShortPassing+Volleys+Dribbling+Curve+FK
Accuracy+LongPassing+BallControl+Acceleration+SprintSpeed+Agility+Reactions+Balance+ShotPo
wer+Jumping+Stamina+Strength+LongShots+Aggression+Interceptions+Positioning+Vision+Penalti
es+Composure+Marking+StandingTackle+SlidingTackle
            , data = fifasample1)

> summary(decreg2)
                            var     rel.inf
Release.Clause   Release.Clause 94.27945712
Reactions             Reactions  1.74420677
Forward                 Forward  1.70240878
Midfield               Midfield  1.27552557
Dribbling             Dribbling  0.18711824
Composure             Composure  0.17636302
FKAccuracy           FKAccuracy  0.12740510
BallControl         BallControl  0.12333683
StandingTackle   StandingTackle  0.10068215
Vision                   Vision  0.08207886
Marking                 Marking  0.07511738
Defense                 Defense  0.04058103
Positioning         Positioning  0.03050667
LongShots             LongShots  0.02459721
Volleys                 Volleys  0.01629868
SlidingTackle     SlidingTackle  0.01431661
Height                   Height  0.00000000
Weight                   Weight  0.00000000
Crossing               Crossing  0.00000000
Finishing             Finishing  0.00000000
HeadingAccuracy HeadingAccuracy  0.00000000
ShortPassing       ShortPassing  0.00000000
Curve                     Curve  0.00000000
LongPassing         LongPassing  0.00000000
Acceleration       Acceleration  0.00000000
SprintSpeed         SprintSpeed  0.00000000
Agility                 Agility  0.00000000
Balance                 Balance  0.00000000
ShotPower             ShotPower  0.00000000
Jumping                 Jumping  0.00000000
Stamina                 Stamina  0.00000000
Strength               Strength  0.00000000
Aggression           Aggression  0.00000000
Interceptions     Interceptions  0.00000000
Penalties             Penalties  0.00000000
```

This shows what is the relative influence of each feature is to the model. 94.27% of the model's influence is from Release Clause, followed by Reactions, Forward and Midfield. Let's see that in a graphical way.



The long bar represents the Release Clause and is showing the relative influence of various variables.

Random Forest Regression:

As part of our model designing, we have chosen Random Forest regression. Random Forest algorithm is a combination of numerous decision trees. The reason behind it is a combination of decision trees and bagging algorithms that reduces the single tree's prediction variance of decision tree and in turn helps in improving performance of our prediction. Each decision tree of the forest is trained of subset of the bootstrapped dataset. The main advantage of random forest is that it will run the regression on the datasets that are left out after creation of decision trees. It makes use of every decision tree and runs regression to get high performance of prediction.

For random forest regression we have divided the data into train and test dataset and later we have divided the regression into two parts, one part containing goalkeeper dataset and other being data without goalkeeper dataset. This way we have performed the regression as follows:

```
####Random_Forest_regressor:

library(randomForest)
set.seed(300)
train_dataset <- sample_frac(fifasample1, 0.7)
index_value<- as.numeric(rownames(train_dataset))
test_dataset <- fifasample1[-index_value, ]   #splitting data into training and test data set :

which( colnames(fifasample1)=="Release.Clause" )# without release.clause
```

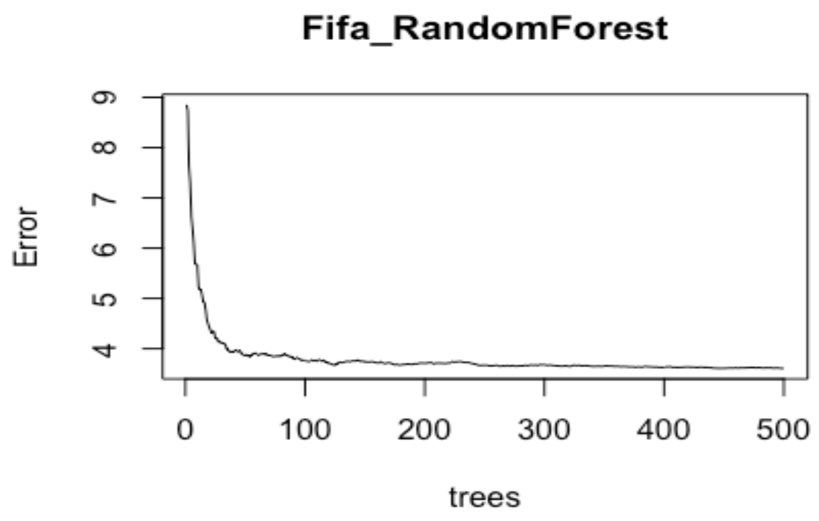
Random Forest regression on Dataset without Goalkeeper

```
library(randomForest)
set.seed(300)
Fifa_RandomForest<-randomForest(Value~Forward + Midfield + Defense + Height+Weight+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,data=train_dataset)
plot(Fifa_RandomForest)#how error decreases upon increasing number of trees-->80%
Fifa_RandomForest
```

As we can see from the output below that only 84.79% variance has been explained by our model.

```
Call:
 randomForest(formula = Value ~ Forward + Midfield + Defense +      Height + Weight + 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, data = train_dataset)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 11

          Mean of squared residuals: 3.610567
                    % Var explained: 84.79
```

The below plot shows how error decreases with increase in number of trees.



Evaluating model performance using multiple folds

```
##Evaluating model_performance :with multiple folds:
set.seed(1007)
library(caret)
NR53 <-  trainControl(method="repeatedcv", number=5, repeats=3)
Train_model53 <- train(Value ~ Forward + Midfield + Defense + Height +Weight +Crossing +Finishing
+HeadingAccuracy +ShortPassing +Volleys +Dribbling +Curve +FKAccurac y+LongPassing
+BallControl +Acceleration +SprintSpeed +Agility +Reactions +Balance +ShotPower +Jumping
+Stamina +Strength+LongShots +Aggression +Interceptions +Positioning +Vision +Penalties
+Composure +Marking +StandingTackle +SlidingTackle , data=train_dataset, method="rf", trControl
= NR53)
plot(Train_model53$finalModel)
Train_model53$finalModel$forest
post_prediction<-predict(Train_model53, test_dataset)
plot(post_prediction)
```

Using multiple folds we could increase the efficiency upto 85%.

```
> Train_model53$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 17

          Mean of squared residuals: 3.533683
```

```
                % Var explained: 85.11
```

Random Forest regression on Dataset without Goal keeper including more predictors

```
#Random_Forest regression without GK and with Release Clause+age,wage,reputation:-->92.75%
library(randomForest)
set.seed(375)
Fifa_RandomForest1<-
randomForest(Value~Age+International.Reputation+Overall+Wage+Release.Clause+ Forward + Midfield + Defense +
Height+Weight+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,data=train_dataset)
plot(Fifa_RandomForest1)#how error decreases upon increasing number of trees
Fifa_RandomForest1

###Evaluating model_performance :with multiple folds:-->94.01%(adding release claues age, wage , international rep.)
set.seed(375)
Train_model53_new<-
train(Value~Age+International.Reputation+Overall+Wage+Release.Clause+Forward + Midfield + Defense +
Height+Weight+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, data=train_dataset, method="rf", trControl = NR53)
plot(Train_model53_new$finalModel)
Train_model53_new$finalModel$forest
post_prediction_new<-predict(Train_model53_new, test_dataset)
```

We can see from the below output on how the performance increased when we increased the predictor variables. The performance increase up to 94.88%

```
> Fifa_RandomForest1

Call:
 randomForest(formula = Value ~ Age + International.Reputation +     Overall + Wage + Release.Clause + Forward + Midfield + Defense +      Height + Weight + 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, data = train_dataset)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 13

          Mean of squared residuals: 1.215829
                    % Var explained: 94.88
```

With multiple folds we could achieve more performance up to 95.99%.

```
> Train_model53_new$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 20

          Mean of squared residuals: 0.9510581
                    % Var explained: 95.99
```

Random Forest regression on Dataset with Goalkeeper data including more predictors and Evaluating model performance using multiple folds

```
#Random_Forest regression with GK and with Release Clause+age,wage etc.:-->94% var explained

library(randomForest)
set.seed(445)
Fifa_RandomForest_GK<-
randomForest(Value~Age+International.Reputation+Overall+Wage+Release.Clause + Forward + Midfield + Defense + Height+Weight+GKDiving+GKHandling+GKKicking+GKPositioning+GKReflexes, data= train_dataset)
plot(Fifa_RandomForest_GK)#how error decreases upon increasing number of trees
Fifa_RandomForest_GK

###Evaluating model_performance with multiple folds: for increasing efficiency:--->95.21%
set.seed(575)
Train_model53_GK<-train(Value~Age+International.Reputation+Overall+Wage+Release.Clause + Forward + Midfield + Defense + Height+Weight+GKDiving+GKHandling+GKKicking+GKPositioning+GKReflexes, data= train_dataset,method="rf", trControl = NR53)
Train_model53_GK$finalModel
plot(Train_model53_GK$finalModel)
Train_model53_GK$finalModel$forest
post_prediction_GK<-predict(Train_model53_GK, test_dataset)
```

We can see from the below output on how the performance increased when we increased the predictor variables for Goalkeeper. The performance increase up to 95.32 % and upon multiple folds it increased up to 96.5%.

```
> Fifa_RandomForest_GK

Call:
 randomForest(formula = Value ~ Age + International.Reputation +     Overall + Wage + Release.Clause + Forward + Midfield + Defense +     Height + Weight + GKDiving + GKHandling + GKKicking + GKPositioning +     GKReflexes, data = train_dataset)
```

```
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 5

          Mean of squared residuals: 1.110088
                    % Var explained: 95.32

> Train_model53_GK$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 8

          Mean of squared residuals: 0.831767
                    % Var explained: 96.5
```

CONCLUSION:

- We have predicted the Value of a Soccer Player using multiple Regression Techniques and able to achieve an accuracy of ~96%
- Out of which each Regression Technique has its own importance in conveying one information or the other. Linear Regression tells us how each Feature is affecting the Value of a player while LASSO Regression tells us what factors/variables are not necessary in the model.
- Decision Trees, Random Forests and Gradient Boosted Machine Regression Models tell us in what way the decision of Value is being considered for the player.

REFERENCES:

- BLUMAN, A. L. L. A. N. (2018). *Elementary Statistics: a brief version*. Place of publication not identified: MCGRAW-HILL EDUCATION.
- Rdatamining.com. (2020). *RDataMining.com: R and Data Mining*. [online] Available at: http://www.rdatamining.com/ [Accessed 3 Feb. 2020].
- Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.
- Oladipupo, T. (2010). Types of Machine Learning Algorithms. *New Advances in Machine Learning*. doi: 10.5772/9385.
- Little, M. A. (2019). Statistical machine learning. *Machine Learning for Signal Processing*, 149–186. doi: 10.1093/oso/9780198714934.003.0006.
- Other Classification Algorithms. (2012). *Machine Learning in Image Steganalysis*, 197–215. doi: 10.1002/9781118437957.ch12