NBA Points Prediction

Programming Language: R

“

**Project Description:**

The objective of this project is to analyze and predict points scored by NBA players. The project is to predict missing points and infer dependency between points and other attributes which are responsible in point prediction for professional basketball team owners using historical data. It includes determining the factors resulting in high points to the players, how much a player can score for the team given the player's record. For the implementation, we aim at using regression algorithms to analyze the player’s data. The project designed will contain data actual vs predicted and plotting results from implementation of linear regression and random forest algorithms.

**Language used: R**

Brief description about R:

R is an environment and language for statistical data processing and graphical representation. It is like the S language and environment which was created at Bell Laboratories. R can be considered as a different implementation of the S language. R gives a wide collection of statistical approaches for example, classical statistical tests, time-series analysis, linear and nonlinear modelling, time-series analysis, classification and graphical methods, and is an exceptionally extensible language.

Features of R:

1.) Analytics & Statistics Features in R

* Basics: Complex arithmetic, cross products, hyperbolic functions, matrix, etc.
* Basic Statistics: Mean, variance, median, cross-tabulations, correlations, etc.
* Probability distributions: Beta, Binomial, chi-square, F distribution, Weibull, etc.
* Big Data Analytics: Revolution R Enterprise.
* Machine learning: Cluster analysis, tree structures, neural networks, etc.
* Optimization and Mathematical Programming: nonlinear programming, etc.
* Signal Processing: Convolutions, FFT, Kalman filtering, wavelets, etc.
* Simulation and Random number generation: Super-Duper, SF Mersenne twister.
* Statistical Modeling: ANOVA, Kernel density estimation, Gaussian models, etc.
* Statistical Tests: Bartlett test of homogeneity of variance, Pairwise t-tests, etc.

2.) Graphics & Visualization Features of R:

* static graphics: Basic plots, graphic maps, projection maps, social network graphs, splines, Trellis plots, etc.
* Dynamic graphics: Animated graphics and movies, motion charts, interactive graphics.
* Devices and Formats: bitmap, BMP, Jpeg, aster graphics, SVG. TIFF, etc.

3.)Programming Features in R:

* Input/output: text, .csv, binary URLs, XML, MySQL, ODBC, Oracle, etc.
* Object-oriented programming: C, Java, Perl, Python, parallel programming, etc.
* Distributed computing: Amazon EC2 compatibility.
* Included R Packages: base, compiler, datasets, etc.; MASS, cluster, lattice, etc.

**Why R ?**

- R is free, open-source software distributed and maintained by R-project.

- Source code is available under the Free Software Foundation’s GNU General Public License.

-R provides data prediction algorithms in our project we used linear regression and random forest

algorithms which are in-built in R.

- R support beautiful and unique data visualizations to present multidimensional data in multi-panel charts, 3D graphs.

- R reads data from huge datasets(nba dataset used in the project) and gives result with high speed and accuracy.

**The R environment**

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It has an effective data handling and storage facility,

· A suite of operators for calculations on arrays, in particular matrices.

· A large, coherent, integrated collection of intermediate tools for data analysis.

· Graphical facilities for data analysis and display either directly at the computer or on hardcopy.

**R and statistics**

R is extensively used as a statistics system. It is an environment within which many classical and modern statistical techniques have been implemented. A few of these are built into the base R environment, but many are supplied as packages. There are about 25 packages supplied with R (called “standard” and “recommended” packages) and many more are available through the CRAN family.

In S a statistical analysis is normally done as a series of steps, with intermediate results being stored in objects. While SAS and SPSS give large output from a regression or discriminant analysis, R will give minimal output and store the results in a fit object for subsequent interrogation by further R functions.

**Major concepts used:**

Multiple Linear Regression

Random Forest

**Multiple Linear Regression:**

Multiple linear regression is an approach to model the relationship between two or more predictor variables and a response variable by fitting a linear equation to observational data. Each value of the predictor variable *x* is associated with a value of the dependent variable *y*. The population regression line for *p* explanatory variables *x*1, *x*2, ... , *x*p is defined to be y = 0 + 1*x*1 + 2*x*2 + ... + p*x*p. This line describes how the mean response y changes with the predictor variables. The observed values for *y* vary about their means y and are assumed to have the same standard deviation . The fitted values *b0*, *b1*, ..., *bp* estimate the parameters 0, 1, ..., p of the population regression line.

Since the observed values for *y* vary about their means y, the multiple regression model includes a term for this variation. In words, the model is expressed as DATA = FIT + RESIDUAL, where the "FIT" term represents the expression 0 + 1*x*1 + 2*x*2 + ... p*x*p. The "RESIDUAL" term represents the deviations of the observed values *y* from their means y, which are normally distributed with mean 0 and variance . The notation for the model deviations is .

Formally, the model for multiple linear regression, given *n* observations, is

*y*i = 0 + 1*x*i1 + 2*x*i2 + ... p*x*ip + i for *i* = 1,2, ... *n*.

In the least-squares model, the best-fitting line for the observed data is calculated by minimizing the sum of the squares of the vertical deviations from each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0). Because the deviations are first squared, then summed, there are no cancellations between positive and negative values. The least-squares estimates *b0*, *b1*, ... *bp* are usually computed by statistical software.

The values fit by the equation *b0* + *b1xi1* + ... + *bpxip* are denoted *i*, and the residuals *ei* are equal to *yi - i*, the difference between the observed and fitted values. The sum of the residuals is equal to zero.

The variance ² may be estimated by *s*² = , also known as the mean-squared error (or MSE).

The estimate of the standard error *s* is the square root of the MSE.

**Random Forest:**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

In the our project we used mean prediction(regression) random forest for computation.

Random Forest regression model used in our project is below:

require(randomForest)

nba\_rf <- nba[,-1]

train\_rf <- subset(nba\_rf, sample == TRUE)

test\_rf <- subset(nba\_rf, sample == FALSE)

set.seed(77)

rf\_final <-randomForest(PTS~. ,data=train\_rf, ntree=30, na.action=na.exclude, importance=T, proximity=T)

For number of trees 30 we get Mean of squared residuals: 1.232365 and % Var explained: 96.99.

**Technical documentation:**

Programming languages used: R language.

**Tools and environments:**

Any R interpreter (preferably R Studio)

**Input DataSet:**

Input dataset contains nba players data with the following columns:

* PLAYER : Name of the player.
* TEAM : Team name.
* AGE : Age of the player.
* GP : Number of games played by the player
* W : Number of wins of the team out of GP.
* L : Number of losses of the team out of GP.
* MIN : Minutes played
* FGA : Field goals attempted by the player
* FG% : Field goals percentage of the player.
* 3PM : Three-point field goals made.
* 3PA : Three-point field goals attempted.
* 3P% : Three-point field goals percentage.
* FTM : Field goals made.
* FTA : Field goals attempted.
* FT% : Field goals percentage.
* OREB : Offensive rebounds.
* DREB : Defensive rebounds.
* REB : Number of rebounds.
* AST : Number of assists.
* TOV : Number of turnovers.
* STL : Number of steals.
* BLK : Number of blocks.
* PF : Personal Fouls.
* DD2 : Double doubles.
* TD3 : Triple doubles.
* PTS : Points scored by the player. ”

**CODE:**

###NBA Statistics###

#Import the file

nba <- read.csv("C:/Users/lalit/Desktop/nba\_scrape\_data\_2016-17\_3.csv")

View(nba)

#We can see from the structure that all are mostly numeric variables, except team and player names

str(nba)

#We want to predict the points scored, and we start modelling for it

#We first split into test and training data

require(caTools)

set.seed(77)

sample = sample.split(nba$PTS, SplitRatio = .75)

train = subset(nba, sample == TRUE)

test = subset(nba, sample == FALSE)

##Models##

#Model 1: Multiple Linear Regression

mlr\_model <- step(lm(PTS~. -PLAYER - TEAM, data = train),direction="both")

#We s=check for summary to see the final important variables

summary(mlr\_model)

#Therefore, the final set is:

mlr\_new <- lm(formula = PTS ~ AGE + GP + FGA + X3PM + FTM +

+ TOV + STL + BLK + DD2 + TD3 + IMPACT, data = train)

#Let's predict the PTS in test data now

test\_predicted <- abs(predict(mlr\_new, newdata = test))

test\_actual <- test$PTS

#We finally check the RMSE and RMSLE values

rmse(test\_actual,test\_predicted)

rmsle(test\_actual,test\_predicted)

#The values are pretty close to zero, which means we have a very accurate model!

#We save these predicted values for the test data(random split) and plot

values<-cbind(test$PLAYER,as.data.frame(test\_actual), as.data.frame(test\_predicted))

View(values)

plot(test\_actual, test\_predicted)

##Model 2: Random Forest Trees##

require(randomForest)

nba\_rf <- nba[,-1]

train\_rf <- subset(nba\_rf, sample == TRUE)

test\_rf <- subset(nba\_rf, sample == FALSE)

#We run with an increasing number of trees until the error falls down no longer.

set.seed(77)#Reproducibility!

rf1 <-randomForest(PTS~. , data=train\_rf, ntree=10, na.action=na.exclude, importance=T,

proximity=T)

print(rf1)

rf2 <-randomForest(PTS~. , data=train\_rf, ntree=20, na.action=na.exclude, importance=T,

proximity=T)

print(rf2)

rf3 <-randomForest(PTS~. , data=train\_rf, ntree=30, na.action=na.exclude, importance=T, proximity=T)

print(rf3)

#Thus we see that 30 are best!

rf\_final <-randomForest(PTS~. , data=train\_rf, ntree=30, na.action=na.exclude, importance=T, proximity=T)

print(rf\_final)

##We fixed on 30 trees, we now tune for m which is the number of random variables in each tree

mtry <- tuneRF(train\_rf[,-7], train\_rf$PTS, ntreeTry=30,

stepFactor=1.5, improve=0.01, trace=TRUE, plot=TRUE)

#Thus, m = 12 is best for each tree in the random forest model. We now build the final model

rf\_climax <-randomForest(PTS~. , data=train\_rf, ntree=30, na.action=na.exclude, importance=T, proximity=T, mtry = 12)

print(rf\_climax)

#Let's plot the important variables

varImpPlot(rf\_climax,main="Points: Dependent Variables",col.main="red")

#Time to predict!

rf\_predicted <- predict(rf\_climax, newdata = test\_rf)

rf\_actual <- test$PTS

#RMSE and RMSLE

require(Metrics)

rmse(rf\_actual, rf\_predicted)

rmsle(rf\_actual, rf\_predicted)

#We can see that linear regression was a better model, probably because there is high linear relationship between PTS scored and the other variables

#Table!

values\_rf<-cbind(test$PLAYER,as.data.frame(rf\_actual), as.data.frame(rf\_predicted))

View(values\_rf)

plot(test\_actual, test\_predicted)

“

**Results:**

Model Multiple Linear Regression:

lm(formula = PTS ~ AGE + GP + FGA + FG. + X3PM + X3PA + X3P. +

FTM + FT. + AST + TOV + BLK + DD2 + TD3 + IMPACT, data = train)

Residuals:

For Multiple Linear Regression:

Min 1Q Median 3Q Max

-1.81169 -0.24405 -0.00837 0.20123 2.44858

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.021903 0.215390 -4.744 2.97e-06 \*\*\*

AGE -0.008817 0.005416 -1.628 0.10434

GP 0.002810 0.001260 2.231 0.02630 \*

FGA 0.967538 0.013509 71.619 < 2e-16 \*\*\*

FG. 0.037261 0.002937 12.687 < 2e-16 \*\*\*

X3PM 2.596280 0.205047 12.662 < 2e-16 \*\*\*

X3PA -0.775552 0.081213 -9.550 < 2e-16 \*\*\*

X3P. -0.004487 0.002005 -2.238 0.02582 \*

FTM 1.027095 0.033139 30.993 < 2e-16 \*\*\*

FT. -0.004444 0.001511 -2.941 0.00347 \*\*

AST -0.045045 0.026259 -1.715 0.08710 .

TOV -0.157953 0.081980 -1.927 0.05477 .

BLK 0.141300 0.073234 1.929 0.05443 .

DD2 0.024800 0.003321 7.468 5.75e-13 \*\*\*

TD3 -0.022205 0.011240 -1.976 0.04893 \*

IMPACT 0.043761 0.009600 4.558 6.98e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4461 on 376 degrees of freedom

Multiple R-squared: 0.9953, Adjusted R-squared: 0.9952

F-statistic: 5353 on 15 and 376 DF, p-value: < 2.2e-16

Take the final important variables into consideration and build the model.

mlr\_new <- lm(formula = PTS ~ AGE + GP + FGA + X3PM + FTM +

+ TOV + STL + BLK + DD2 + TD3 + IMPACT, data = train)

Predict the PTS using test data

test\_predicted <- abs(predict(mlr\_new, newdata = test))

test\_actual <- test$PTS

Checking the RMSE values

rmse(test\_actual,test\_predicted)

0.5925928

rmsle(test\_actual,test\_predicted)

0.1325086

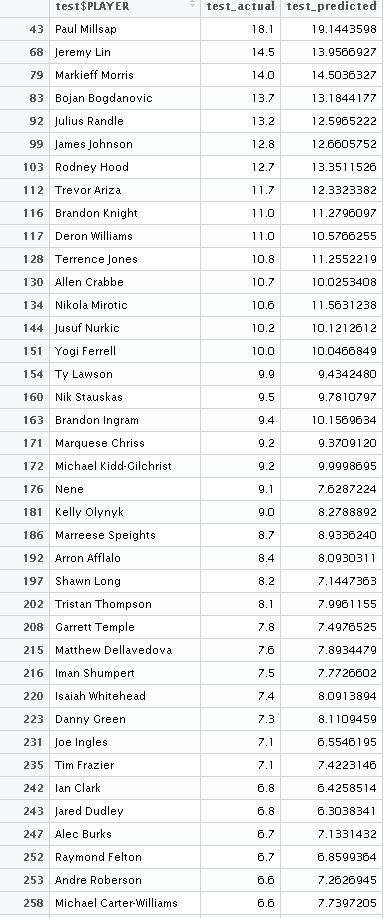


Table 1. Predicted vs Actual using Multiple Linear Regression.

For Random Forest Algorithm:

randomForest(formula = PTS ~ ., data = train\_rf, ntree = 30, importance = T, proximity = T, mtry = 18, na.action = na.exclude)

Type of random forest: regression

Number of trees: 30

No. of variables tried at each split: 18

Mean of squared residuals: 1.232365

% Var explained: 96.99

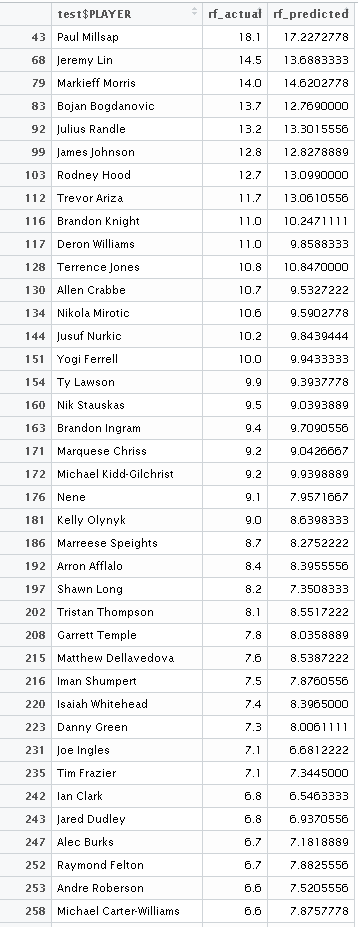
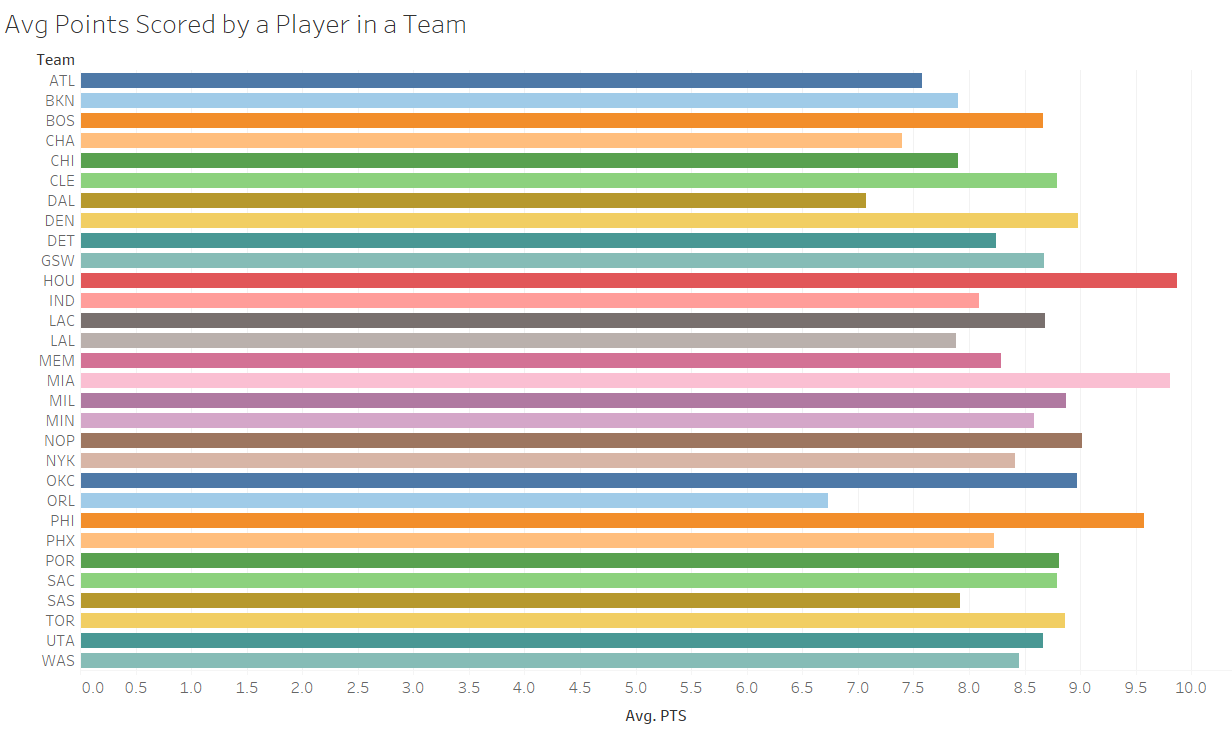


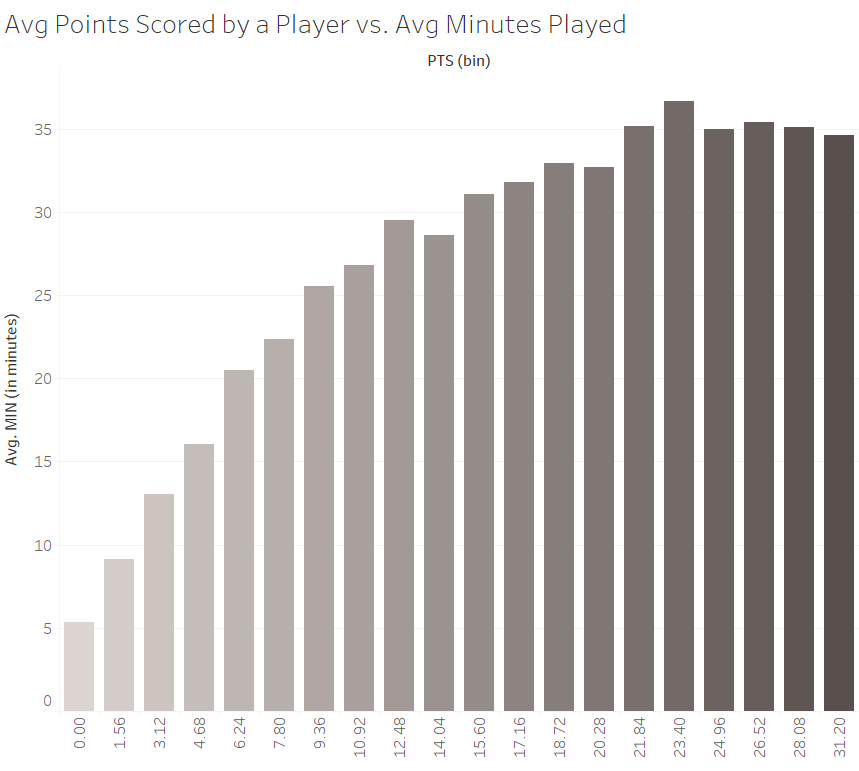
Table 2. Predicted vs Actual using Random Forest Regression.

**Graphical Representation:**

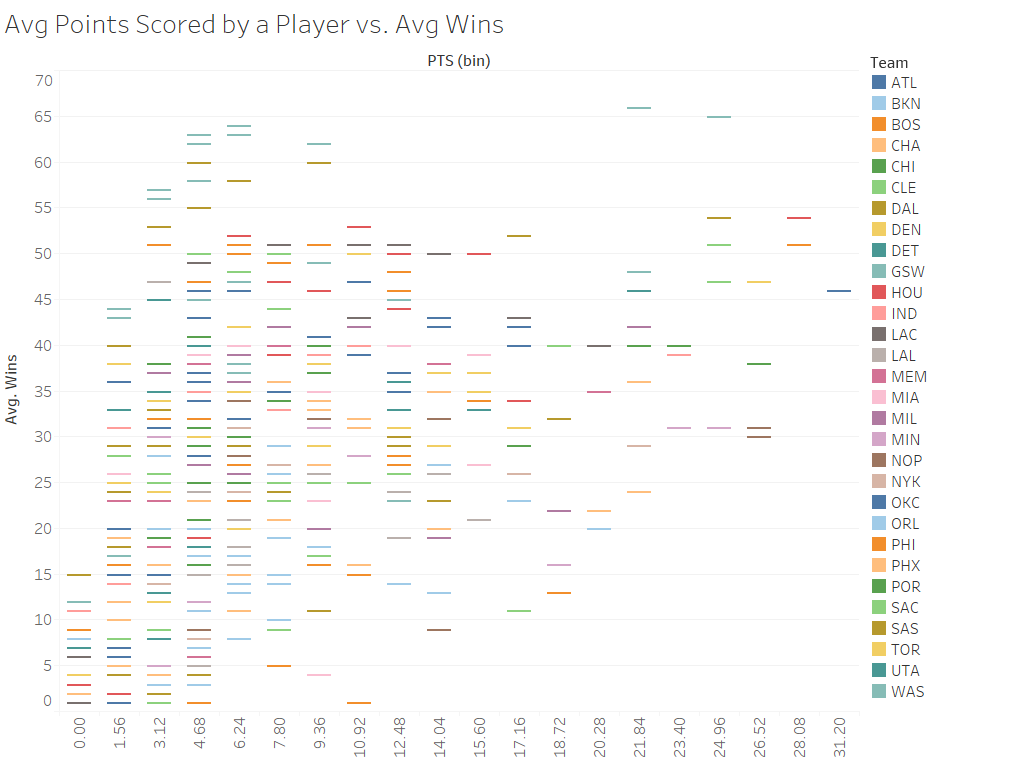
Data Exploration Graphs:



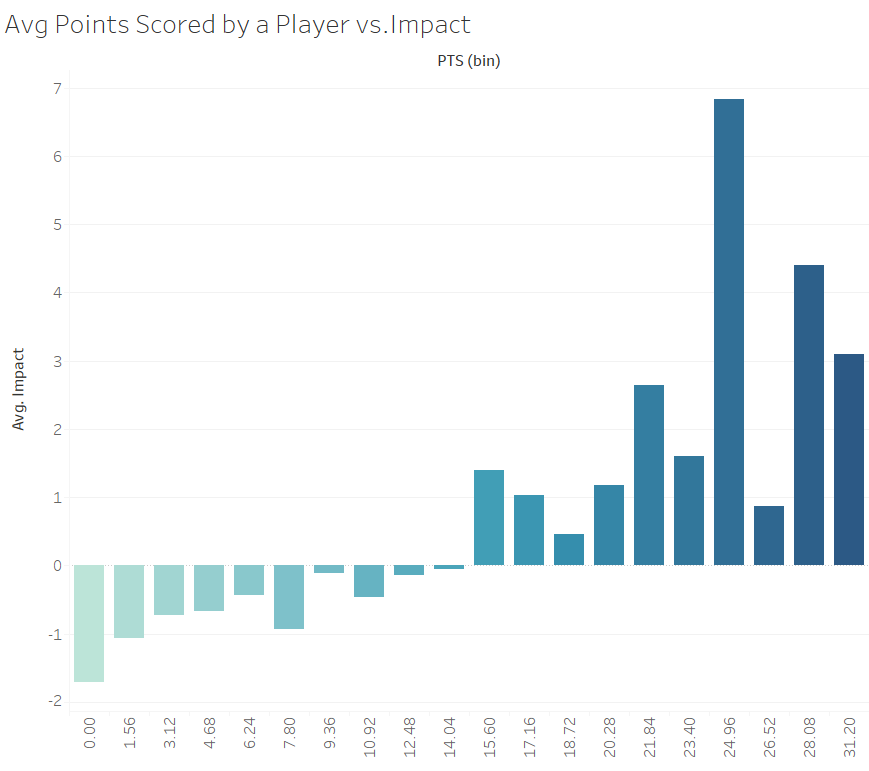
The above graph represents average points scored by players, team wise..



From the above graph, it can be noticed that Players who play longer of the match duration tend to be high scorers.

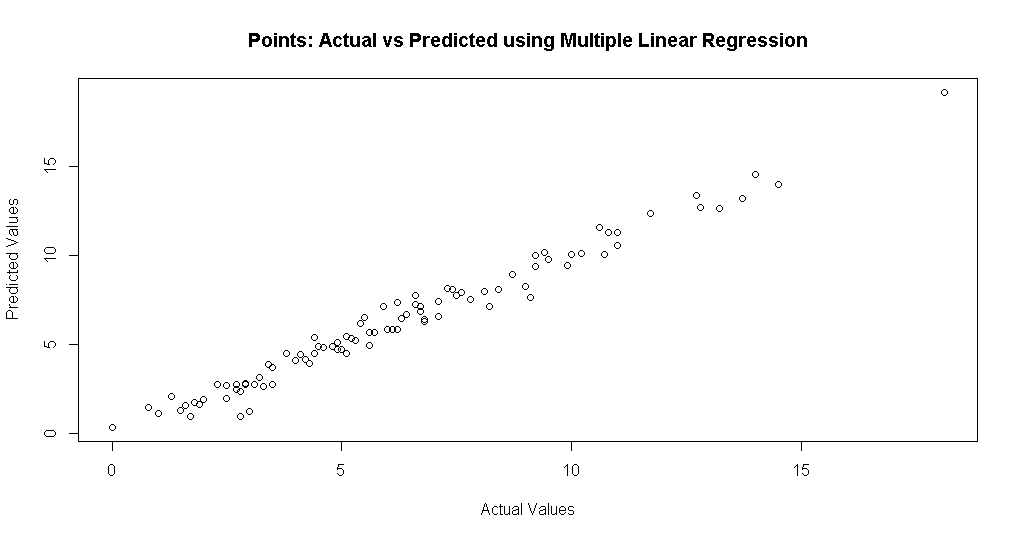


The above graph represents average points scored by a player team wise vs the player wins.



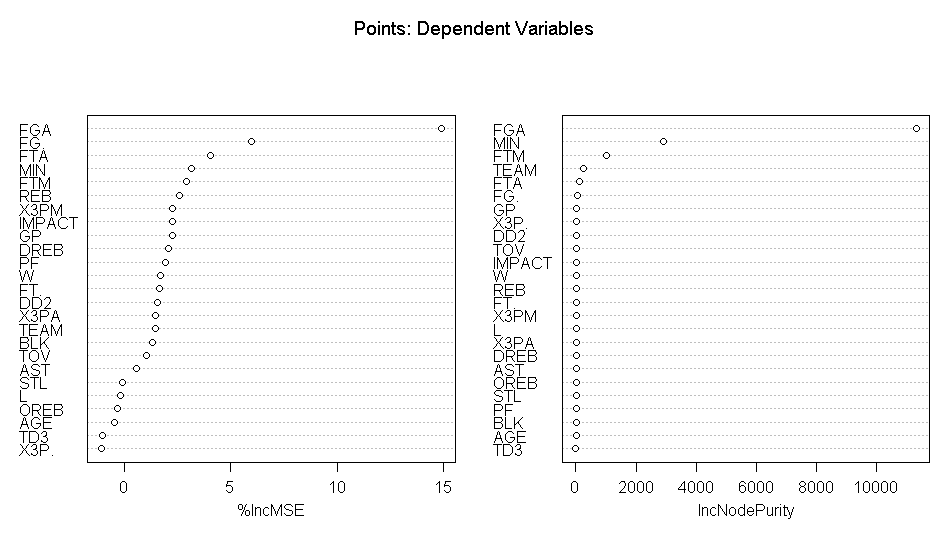
From the above graph, it can be noticed that Players with high scoring average have better impact to the Team than the low scorers.

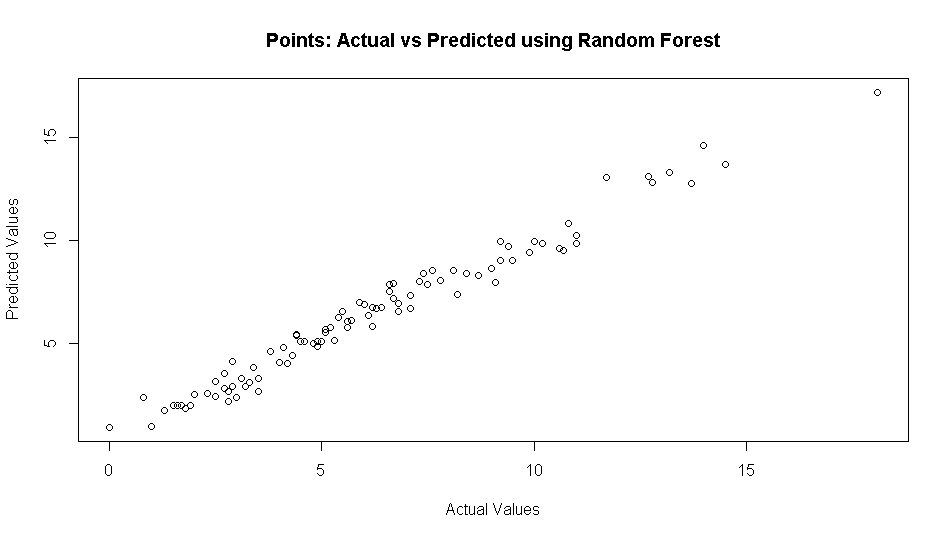
Multiple Regression:



Random Forest:

Finding the Dependent Variables:





**Conclusion:**

The points of basketball players are predicted by using two models first one multiple linear regression and second random forest algorithm.Output table 1 shows data with linear regression and table 2 with random forest algorithm.From the outputs it is clearly observed that multiple linear regression model was more efficient and accurate than random forest algorithm.The data can be used to predict the point of the player in future.

STEPS TO EXECUTE:

Execute the following line in R interpreter.

· Source(“<absolute path to the file>”)

· Change the data path value in the code to the data set location.

REFERENCES:

- <http://stats.nba.com/players/traditional/>

- <https://cran.r-project.org/web/packages/randomForest/randomForest.pdf>

- <http://www.statmethods.net/stats/regression.html>

- <https://www.linkedin.com/pulse/statistical-programming-features-r-thiensi-le>

- <https://en.wikipedia.org/wiki/R_(programming_language)>

- <https://www.tableau.com/trial/tableau-software>

”