Application of Machine Learning in Cricket

Tanvi

May 5, 2018

In the final post of the series, we will make use of Machine Learning techniques for complex analysis. Before explaining how to implement this in R, we would like to emphasize the importance of machine learning in sports especially cricket.

Machine Learning(ML) has brought about a positive change in most of the fields. It can also be applied in sports like cricket.ML can improve the performance and accuracy of players and develop better strategies for the upcoming games. This can be done by predicting the runs scored by a player or the team, the wickets that can be taken and finally predicting the final result of the match. It is always important to select the correct variables so that the prediction is accurate. In the post we will focus on applying ML techniques to the dataset in R. As we are dealing the batting dataset of Virat Kohli and MS Dhoni, we will be dealing with the following variables:

- `Runs` which indicates the runs scored by the player.
- `BF` which indicates the balls faced by the player.
- `Mins` which indicates the minutes spent at the crease by the player.

Before proceeding with the complex analysis it is important that we extract and clean the data.

```
library(cricketr)
virat <- getPlayerDataOD(253802,dir=".", file="kohliOd",type="batting")
dhoni <- getPlayerDataOD(28081,dir=".", file="dhoniOd",type="batting")
```

First, we have imported the `cricketr` library.Then, we have made use of `getPlayerDataOD` function available in cricketr package to get the data of Virat Kohli and MS Dhoni.The `profile number` of the player, `directory` for file, `file` name and the `type` of data required need to be passed as parameters to the function.

```
virat<- virat[virat$Runs != "DNB",]
virat <- virat[virat$Runs != "TDNB",]
virat$Runs <- as.numeric(gsub("\\*","",virat$Runs))
virat$Runs[is.na(virat$Runs)] <- 0
virat<- virat[complete.cases(virat),]
dhoni<- dhoni[dhoni$Runs != "DNB",]
dhoni <- dhoni[dhoni$Runs != "TDNB",]
dhoni$Runs <- as.numeric(gsub("\\*","",dhoni$Runs))
dhoni$Runs[is.na(dhoni$Runs)] <- 0
dhoni<- dhoni[complete.cases(dhoni),]
```

The above steps are required for cleaning the data. They remove the rows having the following:

- `DNB` which indicates the player Did Not Bat.
- `TDNB` which indicates the Team Did Not Bat.
- `*` which indicates that the player was not out. This is replaced with the runs scored by the player.

If you face any problem with these steps then I recommend you to read the previous post of the series **link** for a better understanding. Now, we finally proceed with the application of Machine Learning in R.

Holt-Winters Model for predicting the runs that can be scored

The `batsmanPerfForecast` function is available in *cricketr* package.This function is used for predicting the performance of the player in terms of runs. The parameters that need to be passed are:

- `file` which indicates the file name.Here, **kohliOd** and **dhoniOd** are the file names of Virat and Dhoni respectively.
- `name` which indicates the name of the player whose performance needs to be predicted.

The working of the function is explained below :

Based on the total number of rows available in the dataset the training and the testing data are divided in 90:10 ratio. The model used here to fit the training data is **Holt-Winters Model**. This model is used for predicting the runs by minimizing the squared error of prediction. It works on the basic principle of time series forecasting. Time series forecasting makes use of previous month's or day's or year's data to forecast the value for future. The starting month and the corresponding year and the ending month and the corresponding year (which is at 0.9 of the length of the rows) is the range for which the runs are taken for the training dataset. The similar procedure is followed for the rest of the 10% runs for creating a testing set. Then the Holt-Winters Model is trained using this data and the data forecasted is stored separately. This model is mostly employed in those situations that show a pattern repeating over some time i.e`- a trend, a season or both. The usual forecasting methods require the weighted average of the data. Whereas this model forecasts by associating the data with exponential weighted moving average. This method is also called Tripe exponential smoothing because of its three parameters:

1. Alpha parameter: This estimates the level component at the current point of time.
2. Beta parameter: This estimates the trend component at the current point of time. This can be seen by the slope.
3. Gamma parameter: This estimates the seasonal component at the current point of time.

The above three parameters have a value between 0 and 1. If the values are close to zero then it means that the recent data points have been given less weight in comparison to the old data points while making the predictions.

After all the data has been generated it is plotted as seen below. The main features of the plot are :

- The training data is plotted as a line in magenta color.
- The testing data is plotted as a line in red color.
- The forecasted data is plotted as a line in blue color.

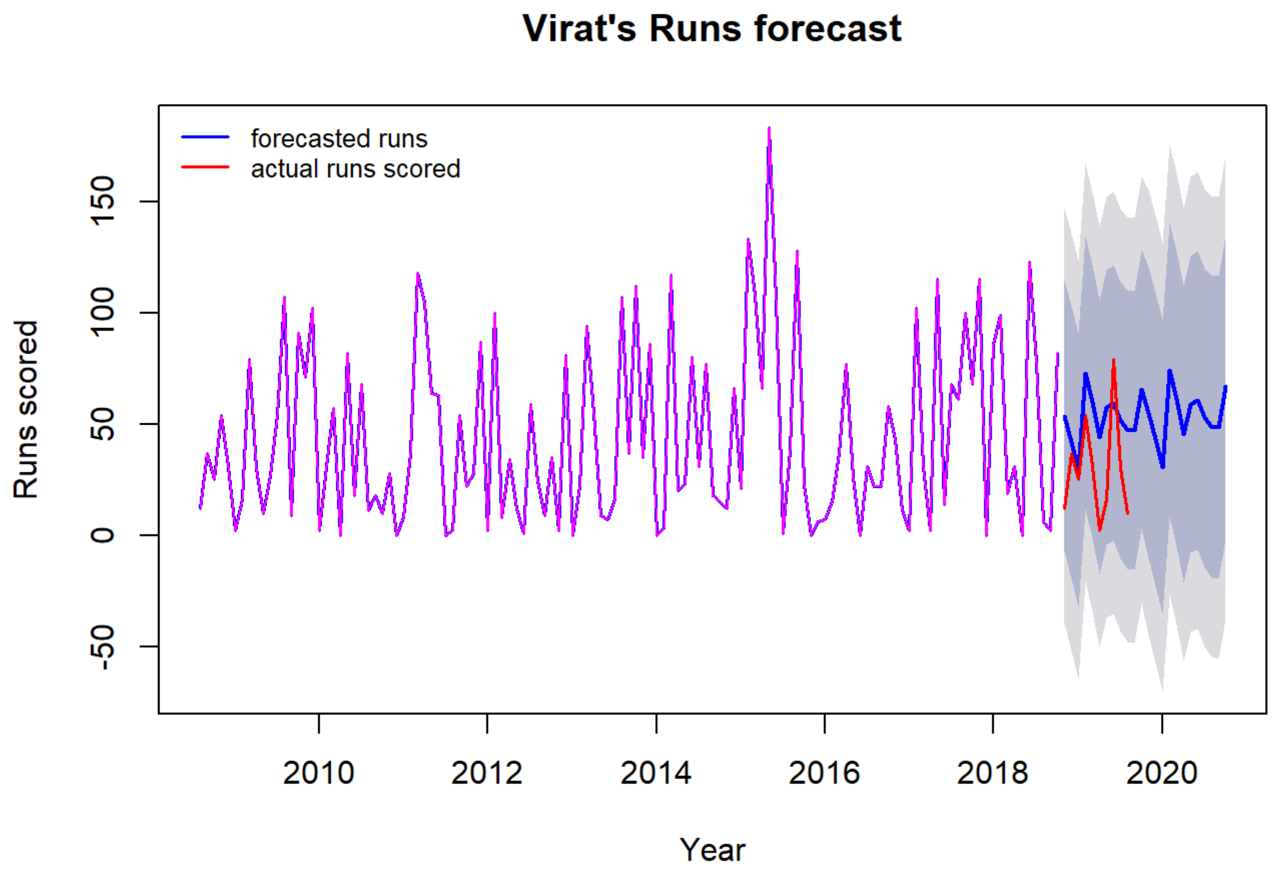
```
batsmanPerfForecast("kohliOd","Virat Kohli")
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = ts.train)
##
## Smoothing parameters:
##  alpha: 0.05384833
##  beta : 0.04209129
##  gamma: 0.2435666
##
## Coefficients:
##            [,1]
## a    63.4022533
## b     0.1097253
## s1   -9.8859838
## s2  -21.7516381
## s3  -34.3349029
## s4    9.2153305
## s5   -4.0305193
## s6  -19.9673518
## s7   -6.5597776
## s8   -4.8182749
## s9  -12.8262453
## s10 -17.1249001
## s11 -17.4730934
## s12   0.8230625
```

From above we can observe the following:

- The value of alpha is 0.071108 which is very low indicating that value of the level component at the current point of time relies less on recent and more on the past data.
- The value of beta is 0.03685509 which is very low indicating that value of the trend component at the current point of time relies less on recent and more on the past data. From the plot below you can observe that slope doesn't a lot.
- The value of gamma is 0.2512596 which is low indicating that value of the seasonal component at the current point of time relies less on recent and more on the past data.

The list of coefficients specifies the value of the level, trend and seasonal component for each month.



As you can see above the actual runs scored by Virat that is in red color and the runs predicted that is in blue color are quite accurate.

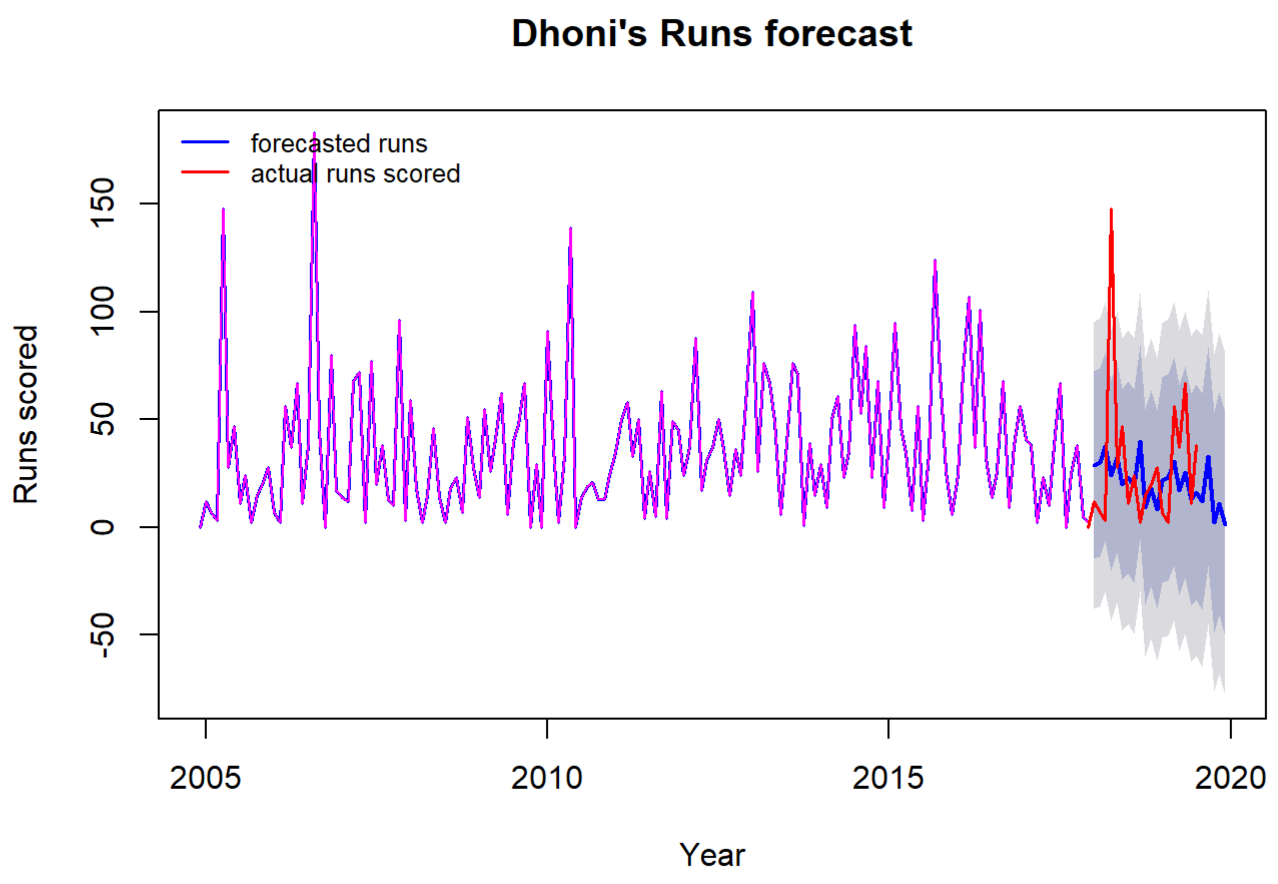
```
batsmanPerfForecast("dhoniOd","MS Dhoni")
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = ts.train)
##
## Smoothing parameters:
##  alpha: 0.07910944
##  beta : 0.04652701
##  gamma: 0.1864698
##
## Coefficients:
##             [,1]
## a    35.47960276
## b    -0.58020908
## s1   -6.09757273
## s2   -4.19217303
## s3    3.90887821
## s4   -9.09986160
## s5   -0.09244239
## s6  -12.07936198
## s7   -8.11811905
## s8  -11.73499610
## s9    9.43919220
## s10 -20.42279862
## s11 -11.08814756
## s12 -20.20560151
```

From above we can observe the following:

- The value of alpha is 0.06946173 which is very low indicating that value of the level component at the current point of time relies less on recent and more on the past data.
- The value of beta is 0.04646374 which is very low indicating that value of the trend component at the current point of time relies less on recent and more on the past data. From the plot below you can observe that slope doesn't a lot.
- The value of gamma is 0.2026552 which is low indicating that value of the seasonal component at the current point of time relies less on recent and more on the past data.

The list of coefficients specifies the value of the level, trend and seasonal component for each month.



As you can see above the actual runs scored by Dhoni that is in red color and the runs predicted that is in blue color are very different.

K Means for predicting player's runs scoring likelihood in %

Next we have used the `batsmanRunsLikelihood` function available in *cricketr* package.This function is used for predicting the runs likelihood in terms of percentage, based on the balls faced and the minutes spent at the crease of Kohli. The parameters that need to be passed are:

- `file` which indicates the file name.Here, **kohliOd** and **dhoniOd** are the file names of Virat and Dhoni respectively.
- `name` which indicates the name of the player whose performance needs to be predicted.

The working of the function is explained below :

The function makes use of K Means Clustering technique. It is an unsupervised technique used for grouping similar instances together in a cluster. The number of the clusters to be formed needs to specified. Here the number of clusters formed is three. The `kmeans` function is used for getting the three centroids of the three clusters. The below plot is a 3D scatterplot having the centroid of the clusters plotted. The three centroids are colored in red, blue and black in order. The three axes are as follows:

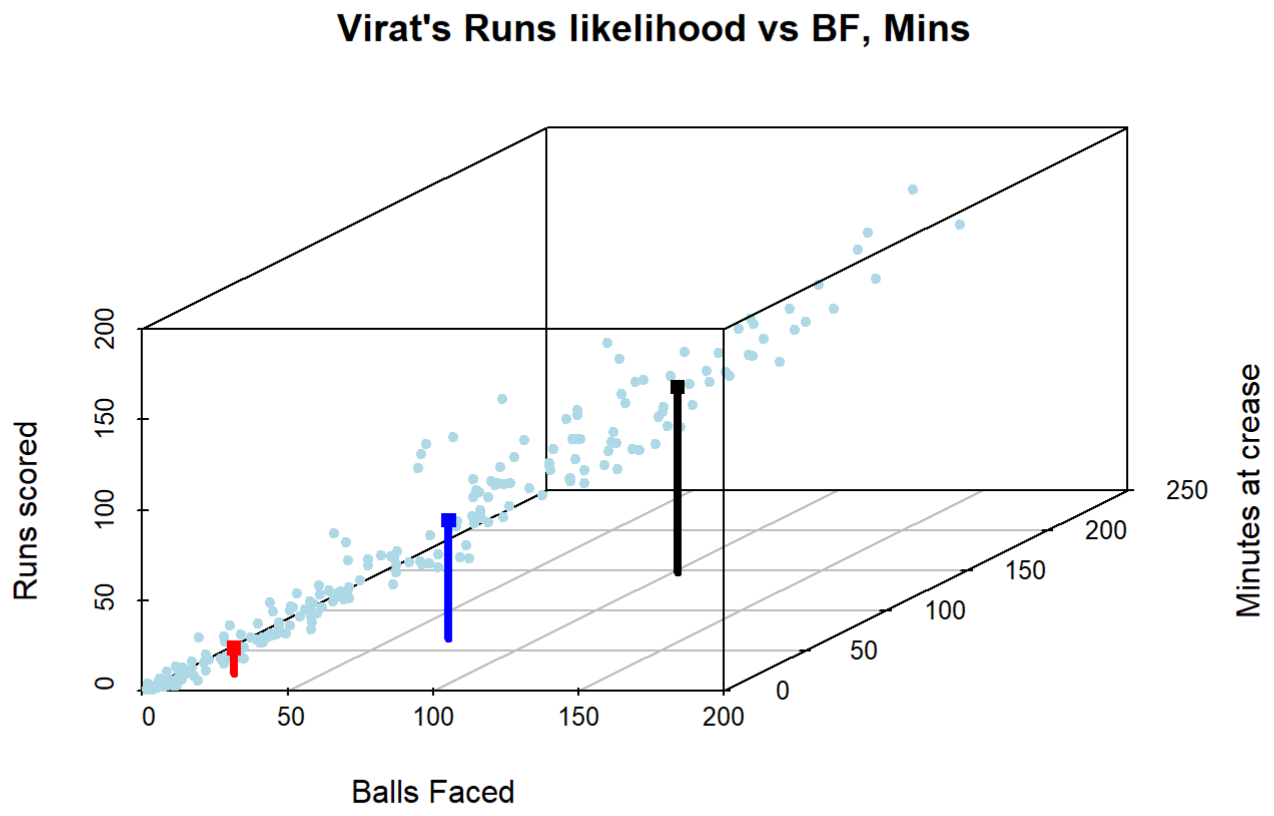
- The x-axis indicates the number of balls faced.
- The y-axis indicates the minutes spent at the crease.
- The z-axis indicates the runs scored by the player.

The significance of three clusters is to emphasize on the three phases of the batsman that is the beginning, the middle and the end of the innings. The first cluster formed is closer to the origin, the second one somewhere in the middle and the third cluster is far away from the origin. The percentage likelihood in the summary is calculated by dividing the sum of the data points in a particular cluster divided by the number of data points in all the three clusters.

```
batsmanRunsLikelihood("kohliOd","Kohli")
```

```
##         Runs        BF      Mins
## 1  65.55357  69.46429  64.64286
## 2  14.23009  19.78761  21.12389
## 3 102.31148 102.18033 147.50820
```

The above table gives the z,x and y values of the centers (column wise) of the three clusters(row wise) for Virat.



Data source-Courtesy:ESPN Cricinfo

```
## Summary of Virat's runs scoring likelihood
```

```
## There is a 24.35 % likelihood that Virat will make  66 Runs in  69 balls over 65  Min
utes
```

```
## There is a 49.13 % likelihood that Virat will make  14 Runs in  20 balls over  21  Mi
nutes
```

```
## There is a 26.52 % likelihood that Virat will make  102 Runs in  102 balls over 148
Minutes
```

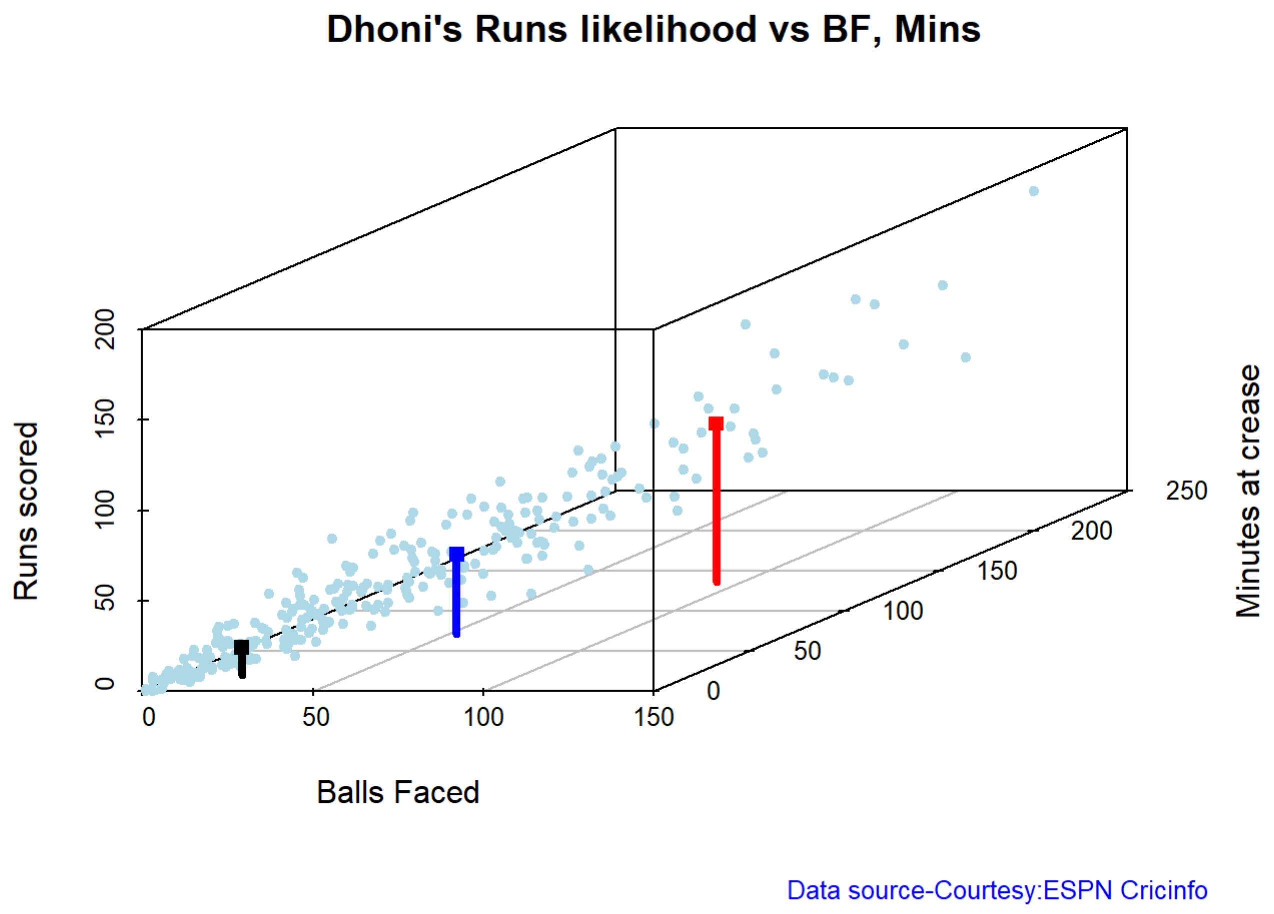
After the plot, the summary of the runs likelihood of Kohli has been displayed.

Similarly, we have used the `batsmanRunsLikelihood` function for predicting the runs likelihood in terms of percentage, based on the balls faced and the minutes spent at the crease of Dhoni along with a plot.

```
batsmanRunsLikelihood("dhoniOd","Dhoni")
```

```
##       Runs       BF      Mins
## 1 44.18519 52.75926  70.99074
## 2 87.48837 92.51163 136.41860
## 3 15.33562 17.99315  19.97260
```

The above table gives the z,x and y values of the centers (column-wise) of the three clusters(row-wise) for Dhoni.



```
## Summary of Dhoni's runs scoring likelihood
```

```
## There is a 36.36 % likelihood that Dhoni will make  44 Runs in  53 balls over 71  Min
utes
```

```
## There is a 14.48 % likelihood that Dhoni will make  87 Runs in  93 balls over  136  M
inutes
```

```
## There is a 49.16 % likelihood that Dhoni will make  15 Runs in  18 balls over 20  Min
utes
```

After the plot, the summary of the runs likelihood of Dhoni has been displayed.

Linear Regression for predicting the number of runs that can be scored

Next we have used the `batsmanRunsPredict` function available in *cricketr* package.This function is used for predicting the number of runs scored based on the balls faced and the minutes spent at the crease of Kohli. The parameters that need to be passed are:

- `file` which indicates the file name.Here, **kohliOd** and **dhoniOd** are the file names of Virat and Dhoni respectively.
- `name` which indicates the name of the player whose performance needs to be predicted.
- `newdataframe` which indicates the data for which you want to predict the runs that can be scored. Here, we are passing 20 values for balls faced from 10 to 200. Also, we are passing 20 values for minutes to be spent on the crease from 5 to 100. So using these two values we have predicted the runs that can be scored by Kohli and Dhoni.

The working of the function is explained below :

It makes use of simple linear regression model. This model is used for predicting an outcome based on given variables. Here, the outcome to be predicted is the number of runs. The variables that are used for prediction of this outcome are balls faced and minutes spent at the crease. Below outputs gives the values of the outcome as well as the variables.

```
BF <- seq(10,200, length=20)
Mins <- seq(5,100,length=20)
batsmanRunsPredict("kohliOd","Kohli",newdataframe=data.frame(BF,Mins))
```

```
##    Balls Faced Minutes       Runs
## 1           10       5   2.217927
## 2           20      10  10.808609
## 3           30      15  19.399292
## 4           40      20  27.989974
## 5           50      25  36.580657
## 6           60      30  45.171339
## 7           70      35  53.762022
## 8           80      40  62.352704
## 9           90      45  70.943387
## 10         100      50  79.534069
## 11         110      55  88.124752
## 12         120      60  96.715434
## 13         130      65 105.306117
## 14         140      70 113.896799
## 15         150      75 122.487482
## 16         160      80 131.078164
## 17         170      85 139.668847
## 18         180      90 148.259529
## 19         190      95 156.850212
## 20         200     100 165.440894
```

For example, if Kohli faces 10 balls and has spent 5 minutes on the crease then we will make around 3 runs. Similarly, we have used the `batsmanRunsPredict` function for predicting the number of runs scored based on the balls faced and the minutes spent at the crease of Dhoni.

```
batsmanRunsPredict("dhoniOd","Dhoni",newdataframe=data.frame(BF,Mins))
```

```
##    Balls Faced Minutes      Runs
## 1           10       5   5.70172
## 2           20      10  12.71439
## 3           30      15  19.72706
## 4           40      20  26.73973
## 5           50      25  33.75239
## 6           60      30  40.76506
## 7           70      35  47.77773
## 8           80      40  54.79040
## 9           90      45  61.80307
## 10         100      50  68.81574
## 11         110      55  75.82840
## 12         120      60  82.84107
## 13         130      65  89.85374
## 14         140      70  96.86641
## 15         150      75 103.87908
## 16         160      80 110.89175
## 17         170      85 117.90442
## 18         180      90 124.91708
## 19         190      95 131.92975
## 20         200     100 138.94242
```

For example, if Dhoni faces 10 balls and has spent 5 minutes on the crease then we will make around 6 runs.

So here we wrap up our cricket series. The objective of this series was to emphasize the role of analytics in one of India's most popular sports **Cricket**. Apart from this, we have covered how analytics can be performed using EDA and Machine Learning in R. We have restricted this series only to analyzing the performances of two batsmen. A lot more can be done besides this. You can also analyze the bowling performance or even predict the final result of the game using a large dataset. This series was just to give you all a basic idea of how analytics can be done using R. Hope you found this post simple and informative. Please feel free to comment or ask for more details in the comment section.

Thanks,

Tanvi (https://www.linkedin.com/in/tanvipareek)