ISEN 613: ENGINEERING AND DATA ANALYSIS PROJECT

Fédération Internationale de Football Association (FIFA)

Data Analysis

Name	**UIN**	**Percentage Contribution**
Ashiq Mohammed Abdul Razak	427009791	20%
Narayanan Vaidhyanathan	227007905	20%
Paul George	627003548	20%
Shikin Sadananda Shetty	727003968	20%
Srinivasan Valady Sivamani	527004431	20%

Table of Contents

1. Introduction

1.1. Importance of problem:

The data set considered, mentions the player details for Fédération Internationale de Football Association (FIFA). The data is used to analyze the overall performance of the soccer players. The statistical learnings are applied to identify the skills and performances of each player, classifying them according to their best skillsets and talent. When a new player is added to the data set, this will assist the management to identify the particular position he would fit to, depending on his skills. The team management can use this information to decide the overall rating of the players and their preferred positions. FIFA is one of the richest sport organization and the data analysis done will help the sport management to provide the clubs with details to select players. Similar analysis method can be used in any industries to identify the key performance indicators. (09) (10)

1.2. Objective:

1) To identify the predictors that add value to the data.
2) To predict the overall performance of individual player.
3) To classify the data into 3 classes which depicts the position that will best fit the player.

1.3. Scope of work:

Initial Analysis will consist of understanding the behavior of data by unsupervised learning approaches. While performing further analysis we will try out various statistical learning approaches learned in the course and suggest the best method based on the test errors. For the first objective we will be using new codes in R outside of the class scope to clean the data followed by applying cross validation and PCA. For the prediction of overall performance, we used various regression techniques. We will suggest the best process by computing the mean square error and cross validation for various methods. The third objective will be achieved by using the various classification techniques taught in class to classify the players into 3 classes based on their attributes. Using the various classification techniques, the confusion matrix for each method would be determined. The best method will be suggested for the model by using error rate obtained from the confusion matrix.

The threshold for each classification will be assumed by referring to football research papers. This data can be used to select the soccer teams by national team management as per their performances in the clubs. The Fans will also use this data to add value to their strategy adopted for betting sites. The data is mainly used for the prediction of performance of the players in the FIFA - 18 games.

2. Project Approach

2.1 Data Description:

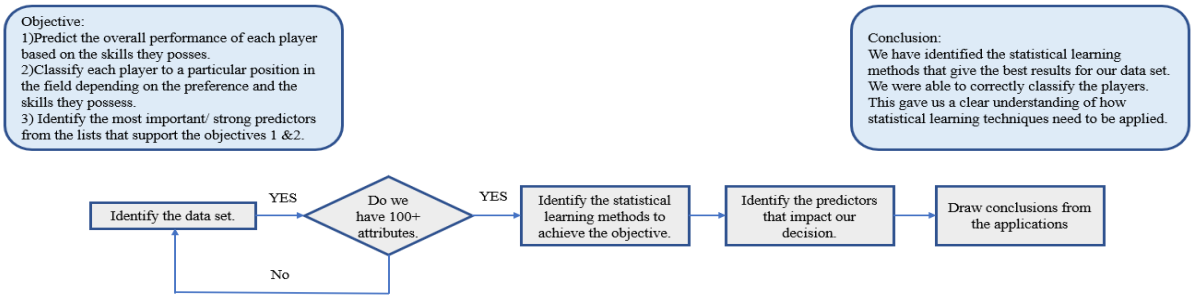
The data set is obtained from www.kaggle.com/ (09). The data was developed from sofifia.com and provides the details of almost all the players that is featured in the FIFA 2018 game. This data is a real-world depiction of the players. We are assuming that the data collected and populated is correct. There are more than 100+ attributes which includes the player image, Flag Images, Playing Position Data, Attributes derived from actual data of the latest EA's FIFA 18 game. Attributes also include all player style statistics like Dribbling, Aggression, GK Skills etc. Player personal data like Nationality, Photo, Club, Age, Wage, Salary etc.

In the domain of games research, there has been work exploring the correlations between player behavioral data and motivations for play. The data of players are important for computer games to adapt to the situation and be exceptionally good. (06) High dimensions of scales cover a temporal segment to reach years of real-time and a varying population of users. Combination of player details provides a way to discover detail patterns that are actionable for game developers. Interpretability and reliability of combination results are vital, as decisions based on them affect game design and thus ultimately revenue. (07) Our dataset is a result of previous research to develop new game sets. We will exploit this data to find new approaches to develop and improve the player statistics.

2.2 A brief about the game:

Developed in Zurich, early 1904. FIFA is an organization which governs the association of football, futsal and beach soccer. They have 211 national associations as its members. In the late 1990's FIFA in collaboration with Electronic Arts (EA sports) developed a video game which depicts real life soccer. In the game, players are given different positions based on their on-field potential in order to get the best outcome. Broadly the team is classified into Forwards, Midfielders, Defenders and Goal Keepers. The major role of forwards is to score goals, mid fielders are responsible to support both the defenders and forwards, defenders are in charge of guarding the goal post and helping the goal keeper to prevent the opponent from scoring. The position of each player is decided by the team manager depending on the skills each player possess. (08)

2.3 Project Flow:



2.4 The reason for our approach:

As we are new to this concept and the data set, we had to draw conclusions after performing the statistical learning methods. We performed all the widely used statistical learning techniques, calculated the test error and then identified the best method that can be used for this type of data set. Because the data set is not familiar to us, we had to perform all form of analysis know to us in order to draw conclusions. In order to apply all concepts, we identified two key issues faced in present world of sport management. One involves regression and the next involves classification issues. In the current scenario where sports are getting so much attention and importance, it is a humongous task for team managers to select the best team in order to adapt and beat the opposing team. Our statistical learnings and results can establish a strong platform for managers to make this decision.

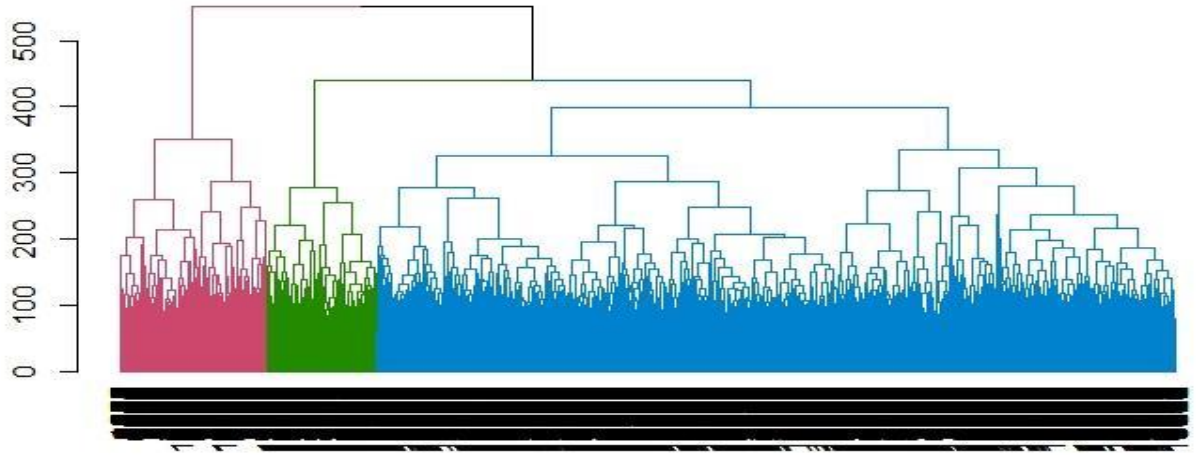
3. Implementation details

3.1 Data Preprocessing:

As a part of initial analysis, Entire FIFA data set was divided into Goalkeeper data and On field data. Missing data/NA values were removed from the data frame. Columns with similar logical values were also removed. The initial data consisted of 185 predictors from which we had to reduce.

3.2 Hierarchical Clustering:

Clustering was performed on On-field data set datasets to have an idea of subgroups of players having similar traits. Hierarchical clustering is a set of nested clusters that are organized as a tree. In hierarchical clustering, dissimilarity is measured by distance between pair of observation, such as Euclidean distance. Visual Representation of Hierarchical Clustering of FIFA Dataset.



3.3 Modelling:

We perform regression analysis on the goalkeeper dataset and the On-field dataset to predict overall performance parameter.

Various models have been implemented to minimize the test error.

After cleaning process, goalkeeper dataset was left with 47 predictors and on-filed dataset was left with 73 variables. The data was split into test and training data with 100 observations reserved for testing the model.

Following models were used to compare for Goalkeeper and On-field datasets.

No	Models	Goalkeeper	On-field
1	Linear	Yes	Yes
2	Backward Elimination	Yes	Yes
3	Lasso	Yes	Yes
4	Random Forests	Yes	Yes
5	Boosting	Yes	Yes
6	PCR	Yes	Yes

3.4. Linear

Regression analysis is used to predict the value of one or more responses from a set of predictors. It can also be used to estimate the linear association between the predictors and responses. Predictors can be continuous or categorical or a mixture of both. (04)

We first revisit the multiple linear regression model for one dependent variable and then move on to the case where more than one response is measured on each sample unit.

- Let $z_1, z_2, ..., zr$ be a set of r predictors believed to be related to a response variable Y.
- The linear regression model for the jth sample unit has the form

  $Y = \beta_0 + \beta_1 z_1 + \beta_2 z_2 + ... + \beta_r z_r + \varepsilon,$

  where ε is a random error and the β_r, r = 0, 1, ..., r is unknown (and fixed) regression coefficients. r is the number of predictors

- β_0 is the intercept and sometimes we write $\beta_0 z_0$, where $z_0 = 1$.

3.4.1 Goalkeeper

The mean square error was found out to be 0.0968 for goalkeeper dataset.
Summary of the fit is as follows.
Residual standard error: 0.3139 on 1879 degrees of freedom
Multiple R-squared: 0.9984, Adjusted R-squared: 0.9983

3.4.2 On-field

Linear regression was performed to fit the training data consisting of 73 predictors.
Mean square error observed was 3.026515.
Summary of the fit is as follows.
Residual standard error: 1.845 on 15815 degrees of freedom
Multiple R-squared: 0.9271, Adjusted R-squared: 0.9269

3.5 Backward elimination

In our discussion of regression to date we have assumed that all the explanatory variables included in the model are chosen in advance. However, in many situations the set of explanatory variables to be included is not predetermined and selecting them becomes part of the analysis.

There are two main approaches towards variable selection: the all possible regressions approach and automatic methods.

The all possible regressions approach considers all possible subsets of the pool of explanatory variables and finds the model that best fits the data according to some criteria (e.g. Adjusted R^2, AIC and BIC). These criteria assign scores to each model and allow us to choose the model with the best score.

The function regsubsets() in the library "leaps" can be used for regression subset selection. Thereafter, one can view the ranked models according to different scoring criteria by plotting the results of regsubsets().

3.5.1 Goalkeeper

Backward selection. Generates validation errors for 41 predictor models.

```
val.errorsgk

##  [1] 6.28447774 1.60801106 1.03228013 0.47609874 0.18042948 0.09527412
##  [7] 0.09497337 0.09655613 0.09805278 0.09772501 0.09693523 0.09610174
## [13] 0.09666488 0.09673815 0.09736894 0.09832025 0.09819136 0.09689167
## [19] 0.09625120 0.09628067 0.09582693 0.09555825 0.09600300 0.09659276
## [25] 0.09707590 0.09705780 0.09745677 0.09752376 0.09702829 0.09708261
## [31] 0.09681386 0.09706317 0.09706726 0.09703344 0.09704547 0.09697314
## [37] 0.09683112 0.09688152 0.09687332 0.09688316 0.09687714
```

Best Predictor models based on Adjusted R square, BIC , Cp(plots)

Val.errors	Adjusted R square	BIC	Cp
7	22	*19*	18

3.5.2 **On-field**

The backward selection methods generate validation errors for 57 predictor models

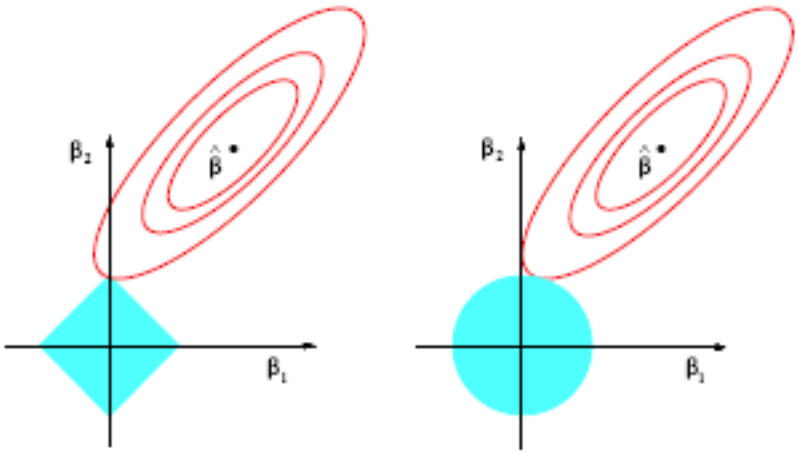
```
val.errors

##  [1] 28.018963  6.520218  4.409935  3.933665  3.577659  3.789524  3.843944
##  [8]  3.636177  3.473258  3.341393  3.260563  3.225369  3.131337  3.113720
## [15]  3.102726  3.150371  3.159087  3.159130  3.120233  3.071748  3.039719
## [22]  3.061525  3.044335  3.037699  2.989996  2.979571  2.965062  2.968600
## [29]  2.965396  2.961285  2.962883  2.964104  2.966744  2.981649  2.979404
## [36]  2.985658  2.997245  2.999669  2.998900  3.003176  3.006961  3.009275
## [43]  3.018435  3.023059  3.022625  3.023944  3.023037  3.024642  3.025533
## [50]  3.024920  3.027724  3.028856  3.028508  3.029265  3.029423  3.029527
## [57]  3.029515
```

Best Predictor models based on Adjusted R square, BIC , Cp

Val.errors	Adjusted R square	BIC	Cp
30	41	36	25

3.6 Lasso

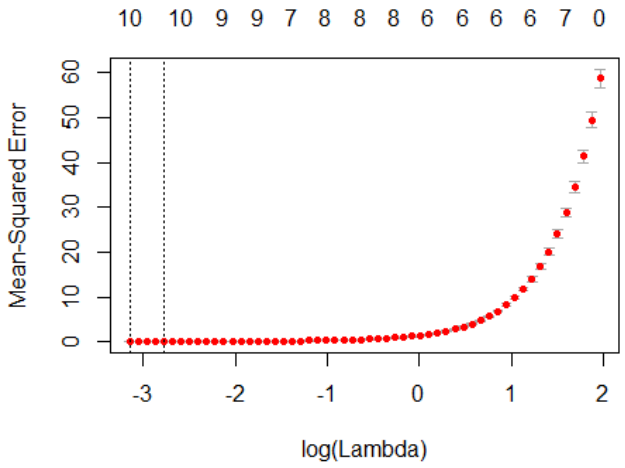


The lasso performs L1 shrinkage, so that there are "corners" in the constraint, which in two dimensions corresponds to a diamond. If the sum of squares "hits" one of these corners, then the coefficient corresponding to the axis is shrunk to zero. (03)

As p increases, the multidimensional diamond has an increasing number of corners, and so it is highly likely that some coefficients will be set equal to zero. Hence, the lasso performs shrinkage and (effectively) subset selection.
In contrast with subset selection, Lasso performs a soft thresholding: as the smoothing parameter is varied, the sample path of the estimates moves continuously to zero.

3.6.1 Goalkeeper

Library glmnet is used to run lasso analysis. We used cross validation to find the optimum



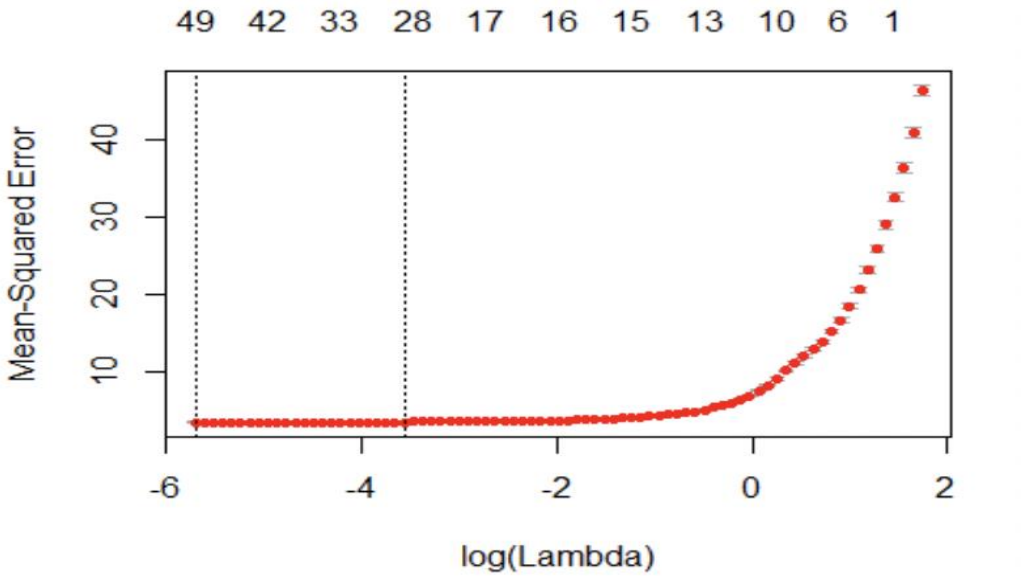
value of lambda.

The above plot determines the optimal lambda value with minimum value of mean-square error.

MSE(Goalkeeper) for lasso is 0.08956871. This MSE is comparatively lower than the error rate obtained from Linear regression and Backward elimination.

3.6.2 On-field

Cross-validation is used to find optimal value of lambda.



The above plot determines the optimal lambda value with minimum value of mean-square error.

MSE (On field Player) for lasso is 3.022436. This is slightly lower than the error rate from Linear Regression and Backward elimination.

3.7 Random Forest

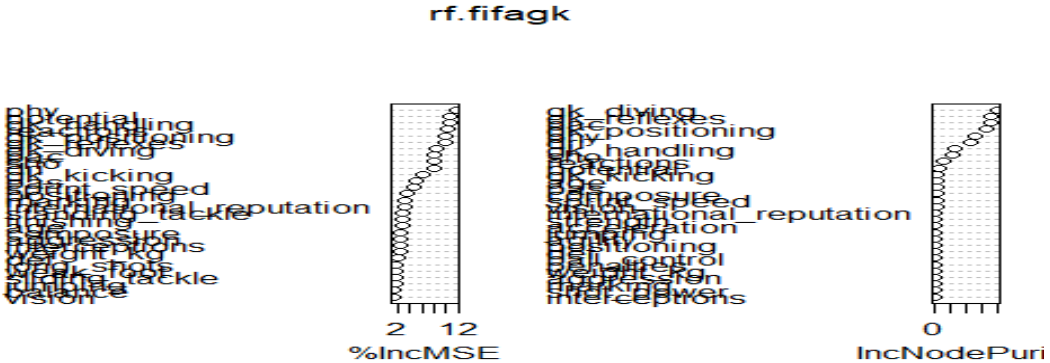
Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges as to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. (01)
Formally, a random forest is a predictor consisting of a collection of randomized base regression trees $\{rn(\mathbf{x},\Theta m,Dn),m \geq 1\}$, where $\Theta 1,\Theta 2,...$ are i.i.d. outputs of a randomizing variable Θ. These random trees are combined to form the aggregated regression estimate $\bar{rn}(\mathbf{X},Dn) = E\Theta\ [rn(\mathbf{X},\Theta,Dn)]$, where $E\Theta$ denotes expectation with respect to the random parameter, conditionally on $\mathbf{X}$ and the data set Dn. In the following, to lighten notation a little, we will omit the dependency of the estimates in the sample, and write for example $\bar{rn}(\mathbf{X})$ instead of $\bar{rn}(\mathbf{X},Dn)$. Note that, in practice, the above expectation is evaluated by Monte Carlo, that is, by generating M (usually large) random trees.

3.7.1 Goalkeeper

Random forest performed on p/3 predictors for regression analysis, where p=45. Thus, the model considers 15 predictors during each split. 100 trees were considered for averaging the response of predictors.

Test MSE was 0.3494. This shows that the linear models for Goalkeeper have better fit than Random forests. This shows that the true function is close to linear when predicting the overall for Goalkeeper.

Plot below shows important variables in terms of %increase in MSE and increase in Node Purity.

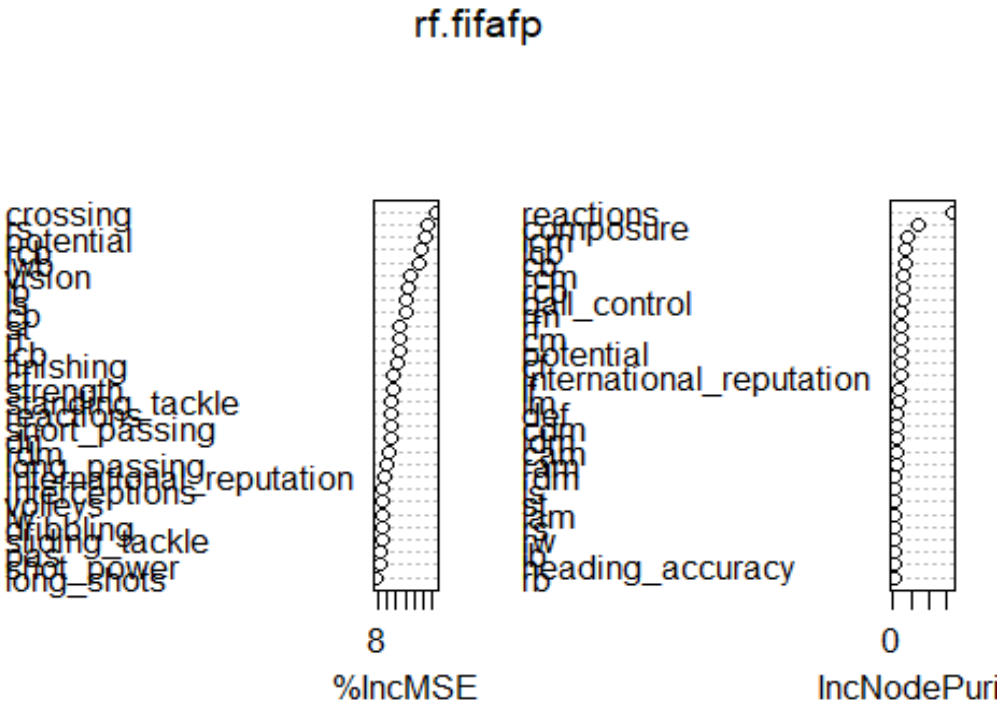


3.7.2 On-field

Random forest performed on p/3 predictors for regression analysis, where p=72. Thus, the model considers 24 predictors during each split. 100 trees were considered for averaging the response of predictors. Main advantage of random forest is that it decorrelates the tree models.

Test MSE turned out to be 0.4811761. Counterintuitively, while predicting the overall for on field players, the Random Forests model tend to perform better as compared to Linear Regression and LASSO.

Using VarImpPlot() function we can find relative importance of variables contributing to the response.
The below plot explains Increase in MSE and Node Purity with respect to different predictors.



3.8 Boosting

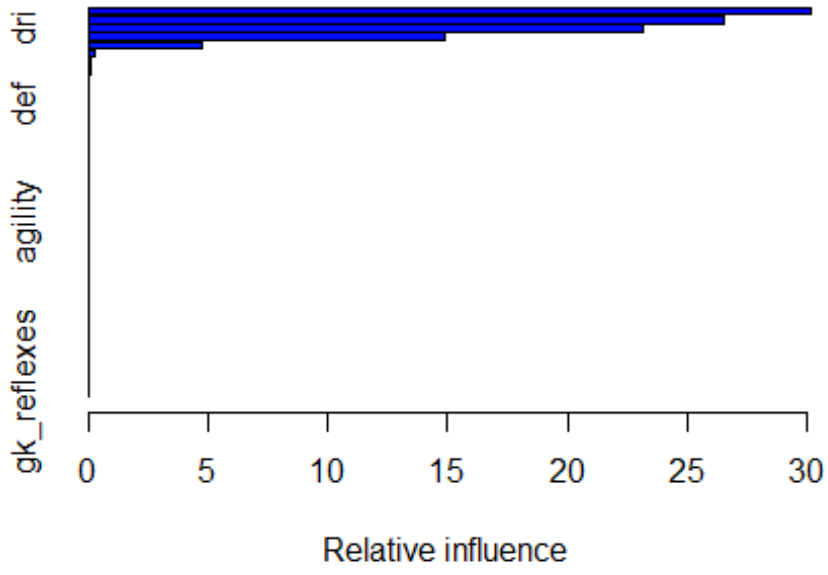
Boosting is another approach to improve the predictions resulting from a decision tree. Like bagging and random forests, it is a general approach that can be applied to many statistical learning methods for regression or classification. Recall that bagging involves creating multiple copies of the original training dataset using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model. Notably, each tree is built on a bootstrapped dataset, independent of the other trees.

Boosting works in a similar way, except that the trees are grown *sequentially*: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling; instead, each tree is fitted on a modified version of the original dataset.

- Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.
- Given the current model, you fit a decision tree to the residuals from the model. That is, you fit a tree using the current residuals, rather than the outcome Y, as the response.
- You then add this new decision tree into the fitted function in order to update the residuals. Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm. By fitting small trees to the residuals, you slowly improve $\hat{f}$ in areas where it does not perform well.
- The shrinkage parameter Lambda slows the process down even further, allowing more and different shaped trees to attack the residuals.

3.8.1 Goalkeeper

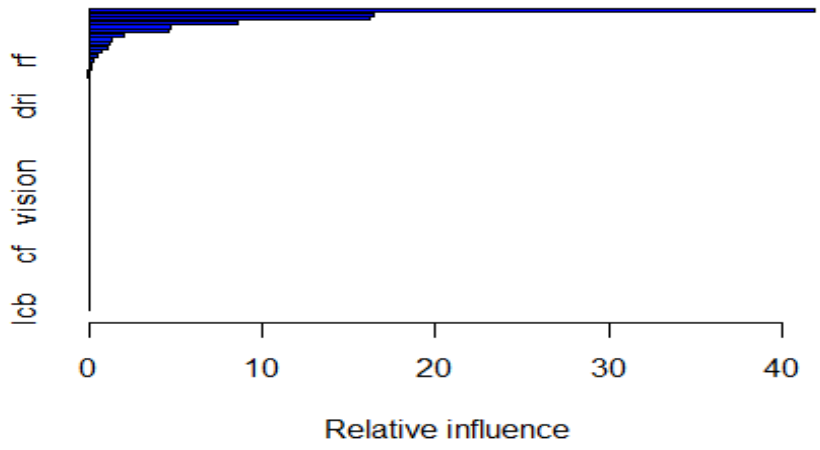
Model is fit using training data with number of trees to be 100 and interaction depth of 1.
gbm library in R package for fitting boosting model.
The plot for relative influence vs predictors is as shown



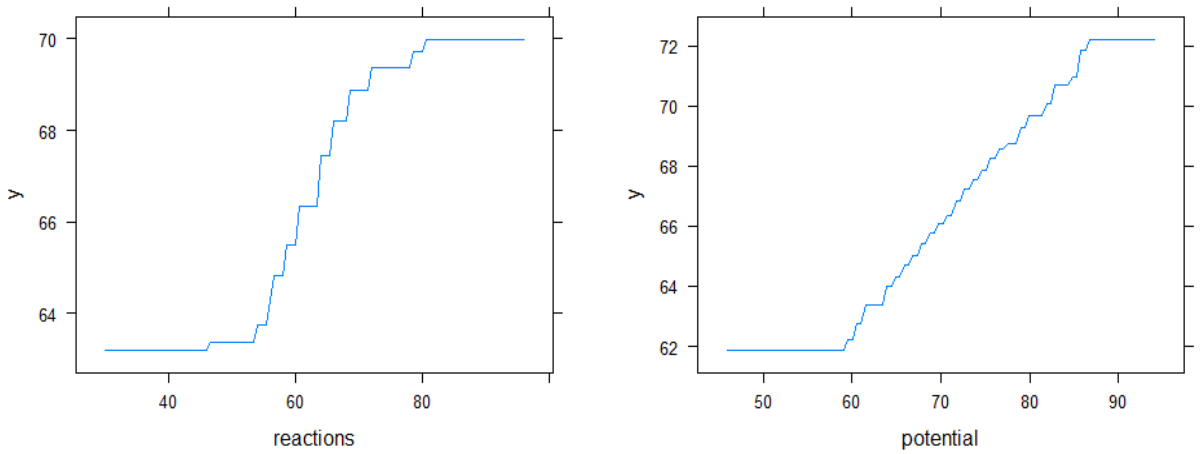
MSE using Boosting is 0.5700568. The MSE is higher for Boosting as compared to other models.

3.8.2 On-field

The model is fit with distribution to be gaussian, number of trees =100 and interaction depth of 1. Smaller the interaction depth, Model will learn slowly and have a better prediction accuracy. The graph of Relative influence vs predictors is as shown below



We try to plot the variation of y with respect to top 2 important predictors “reactions” and “potential”



The MSE using boosting was about 3.520272. The MSE for boosting is higher than that of Random Forests for On field Players.

3.9 PCR

Principal Components Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. (11)

3.9.1 Goalkeeper

```
TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
X           30.09    42.19    47.77    52.12    56.20    59.55    62.06
overall     82.54    96.40    96.72    97.03    98.54    98.56    98.64
          8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
X           64.33    66.45     68.36     70.19     71.97     73.67
overall     98.98    98.98     99.08     99.08     99.34     99.35
          14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
X            75.28     76.82     78.32     79.79     81.18     82.54
overall      99.36     99.37     99.38     99.41     99.55     99.65
          20 comps  21 comps  22 comps  23 comps  24 comps  25 comps
X            83.82     85.03     86.22     87.33     88.40     89.45
overall      99.65     99.65     99.65     99.67     99.67     99.68
          26 comps  27 comps  28 comps  29 comps  30 comps  31 comps
X            90.44     91.41     92.34     93.23     94.09     94.88
overall      99.69     99.69     99.69     99.70     99.71     99.71
          32 comps  33 comps  34 comps  35 comps  36 comps  37 comps
X            95.66     96.42     97.13     97.82     98.44     99.00
overall      99.71     99.71     99.71     99.72     99.74     99.77
          38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
X            99.47     99.81    100.00    100.00    100.00    100.00
overall      99.77     99.77     99.84     99.84     99.84     99.84
          44 comps  45 comps  46 comps
X           100.00    100.00    100.00
overall      99.84     99.84     99.84
> pcr_predgk <- predict(pcr_modelgk,newdata=test.gk,ncomp=21)
> mean((pcr_predgk - test.gk[,"overall"])^2)
[1] 0.2446676
```

3.9.2 On-field

```
TRAINING: % variance explained
          1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X           41.55    65.99    72.85    76.39    78.52    80.27    81.81    83.08
overall     49.46    68.82    74.80    77.55    78.89    78.89    83.70    83.70
          9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X           84.34     85.58     86.80     88.01     89.18     90.23     91.11
overall     83.70     83.87     83.88     83.95     84.74     85.50     85.52
          16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
X            91.91     92.64     93.27     93.88     94.47     94.97     95.42
overall      85.55     86.96     87.04     88.83     88.93     88.93     89.23
          23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
X            95.86     96.25     96.63     96.98     97.32     97.64     97.92
overall      89.58     89.88     90.07     90.14     90.20     90.22     90.22
          30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
X            98.18     98.42     98.65     98.87     99.08     99.25     99.42
overall      90.28     90.34     90.83     91.22     92.44     92.44     92.67
          37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
X            99.58     99.72     99.85     99.92     99.97     99.97     99.98
overall      92.68     92.68     92.68     92.68     92.70     92.70     92.70
          44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
X            99.98     99.98     99.99     99.99     99.99     99.99    100.00
overall      92.70     92.71     92.71     92.71     92.71     92.71     92.71
          51 comps  52 comps  53 comps  54 comps  55 comps  56 comps  57 comps
X           100.00    100.00    100.00    100.00    100.00    100.00    100.00
overall      92.71     92.71     92.71     92.71     92.71     92.71     92.71
          58 comps  59 comps  60 comps  61 comps  62 comps  63 comps  64 comps
X           100.00    100.00    100.00    100.00    100.00    100.00    100.00
overall      92.72     92.72     92.72     92.72     92.72     92.72     92.72
          65 comps  66 comps  67 comps  68 comps  69 comps  70 comps  71 comps
X           100.00    100.00    100.00    100.00    100.00    100.00    100.00
overall      92.72     92.72     92.72     92.72     92.72     92.73     92.73
          72 comps  73 comps
X           100.00    100.00
overall      92.73     92.73
> mean((pcr_predfp - test.fp[,"overall"])^2)
[1] 4.42562
```

3.10 Classification:

In classification, the team wanted to predict the class to which a player with a specific set of attributes would be assigned. From, the dataset, since the Goalkeeper was considered a single position, the team decided to eliminate the Goalkeeper since it has one class only leading to classification being impossible.

The team decided to aggregate the remaining on field positions were aggregated as a 'Position' column comprising Strikers, Midfielders and Defenders through exploratory analysis using R. Now, the objective in classification would be to treat Position as a response and remaining variables as the predictors and predict one of the three Position classes for each player.

We first split the data into training and test in order to train the program how the data is arranged and how the classification should take place. We removed the unwanted predictors as mentioned in the regression section. PCA was applied initially to identify which of the predictors gave the majority of the variances.

3.10.1 Random Forests

We then started off with the random forest since it gave very good results with respect to regression, we got the following result for 100 samples of test,

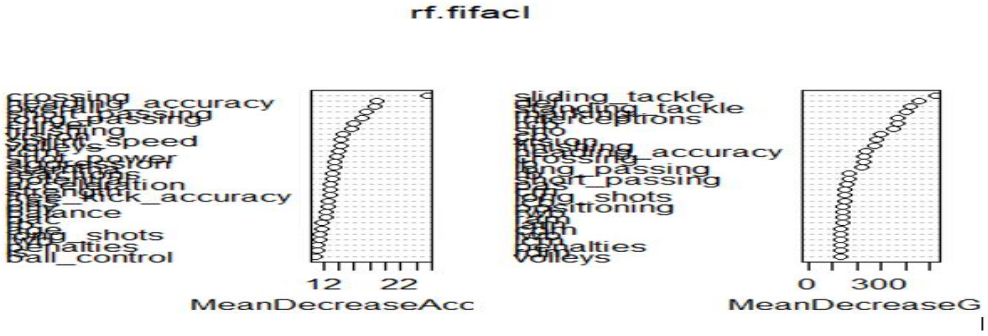
```
##
## pred.rf      Defender Midfielder Striker
##   Defender         29          3       0
##   Midfielder        3         29       6
##   Striker           0          3      27

##TEST ERROR RATE
test.error.rate.rf <- mean(pred.rf!=test$position)
test.error.rate.rf

## [1] 0.15
```

We were able to achieve 0.15 error rate for this particular classification. Here in random forests the program automatically decorrelates between the predictors. We use the mtry value as sqrt(p) for classification.

The importance of each variables is shown in the below graphs,



As we can see in the figure, the important predictors to classify the players into forward, midfielders, defenders are crossing, heading accuracy, short passing, sliding tackle etc. The left-hand side graph depicts the importance of variables with respect to accuracy and the right-hand side graph depicts the variable importance with respect to the node purity.

3.10.2 Linear Discriminant Analysis

One other approach we tried was the linear discriminant analysis for classification into 3 classes.

```
##
## lda.class    Defender Midfielder Striker
##   Defender         29          2       0
##   Midfielder        3         30       8
##   Striker           0          3      25

#TEST ERROR RATE
test.error.rate.lda <- mean(lda.class!=test$position)
test.error.rate.lda

## [1] 0.16
```

Above, we see the results for LDA classification in the form of a confusion matrix. The classification error for this method is 0.16, which is slightly higher than the random forest. The LDA works well with the linear data. The data we have is slightly non-linear hence the difference in the error rate.

3.10.3 K Nearest Neighbors

Next approach we went on with is the K – Nearest Neighbors. In KNN, the first task is to determine the value of K which leads to the lowest error rate. K values starting from 1 to 100 were taken and the error rates for each value of K were determined by running a loop in R.

The below results show the error rate associated with each value of K.

```
test.error.rate.knn

##   [1] 0.23 0.25 0.20 0.17 0.15 0.17 0.16 0.15 0.15 0.15 0.15 0.16 0.15
0.16
##  [15] 0.15 0.15 0.15 0.17 0.16 0.14 0.15 0.15 0.15 0.16 0.14 0.15 0.15
0.15
##  [29] 0.15 0.15 0.15 0.15 0.15 0.15 0.14 0.15 0.15 0.15 0.16 0.15 0.16
0.16
##  [43] 0.16 0.16 0.16 0.16 0.16 0.16 0.16 0.16 0.16 0.16 0.16 0.16 0.16
0.16
##  [57] 0.16 0.16 0.16 0.15 0.15 0.15 0.15 0.15 0.16 0.15 0.15 0.15 0.15
0.15
##  [71] 0.15 0.15 0.15 0.15 0.15 0.15 0.15 0.16 0.16 0.16 0.16 0.16 0.16
0.16
##  [85] 0.16 0.16 0.15 0.16 0.15 0.16 0.15 0.15 0.15 0.15 0.15 0.15 0.15
0.15
##  [99] 0.15 0.16

which.min(test.error.rate.knn)

## [1] 20
```

From the results, we found that the lowest error rate is obtained for the K value of 20.

Now we ran the classification again with the best value of K = 20 the results of this classification are plotted as a confusion matrix.

```
##LEAST ERROR RATE - 0.14 FOR K = 20
##REVISIT KNN FOR K = 20
knn.cl <- knn(fifa.num[train,-120],fifa.num[-train,-
120],fifa.num[train,]$position,k=20)
table(knn.cl,test$position)

##
## knn.cl       Defender Midfielder Striker
##   Defender         30          3       0
##   Midfielder        2         30       7
##   Striker           0          2      26

mean(knn.cl!=test$position)

## [1] 0.14
```

The above figure shows the error rate as 0.14 or 14 % of the data. So far, the best result is given KNN. This defines the nonlinear nature of the model and hence gives better results.

3.10.4 Boosting

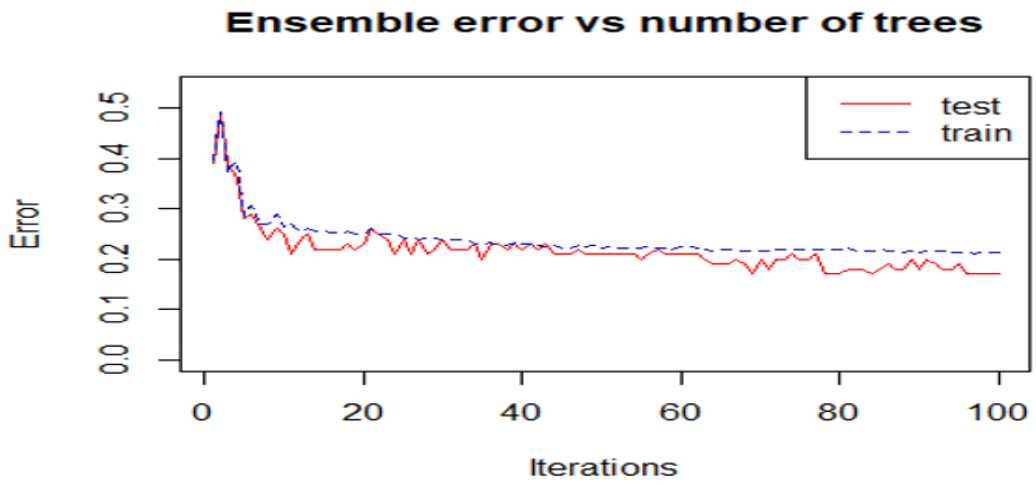
We also tried for boosting the classification of players into Strikers, midfielders, defenders. The result is as shown below:

```
##                Observed Class
## Predicted Class Defender Midfielder Striker
##      Defender         29          6       0
##      Midfielder        3         27       6
##      Striker           0          2      27

##TEST ERROR RATE
fifa.adaboost.pred$error

## [1] 0.17
```

The predicted test error is 17 % is on the higher side, but the graph below suggest the method gave an appropriate result.



In the figure the error are high initially and then reduces gradually as the number of iterations increases. As we have learnt the boosting is a slow learning process. Hence as the number of iterations increase the system learns from the previous predictions and make necessary changes in order to improve the predictions.

4. Comparison of Statistical Learning Methods

Regression:

No	Models	Goalkeeper (MSE)	On-field(MSE)
1	Linear	0.0968	3.0265
2	Backward Elimination	0.09497 Number of predictors = 7	2.9612 Number of predictors = 7
3	Lasso	0.08956	3.0224
4	Random Forests	0.3494	0.4811
5	Boosting	0.57	3.52
6	PCR	0.2446	4.42

Classification:

No	Models	Goalkeeper
1	Random Forests	0.15
2	LDA	0.16
3	KNN	0.14
4	Boosting	0.17

5. Executive summary

We were able to identify a method that gives optimal results for the prediction of overall performance of a player from the database. We were also able to identify the best classification method that can be adopted for multiple class segregation. Initial objectives for regression were to clean to the data and split the 'Overall' performance parameter to On Field Players and Goalkeepers. Similarly, for classification, aggregation of multiple redundant player positions into three vital positions was the key.

The main challenge that we faced was to identify the predictors that would be useful for our analysis. PCA, Random forests, Lasso etc. where few operations which helped us to analyze this. The cross-validation errors were used to rate our results in the case of regression and the confusion matrix was used to identify the error rate in case of classification. The methods chosen would prove to be a good fit to the real world data set and give accurate predictions.

5.1 Importance in Real World

In the real world the FIFA has a huge fan following for both the real world soccer as well as in video games. In the real world, the statistical learning methods employed in our research would enable team managers to predict with a fairly good level of accuracy, the overall performance of the players as well as their preferred positions by using their traits as the inputs. This would be very useful in areas such as scouting for new players, where calculated models can be employed rather than plain intuition. This approach would definitely prove to be a game winning tactic, where each manager can look for players with good overall performance and preferred positions that go well their tactics.

In the gaming world, this tool would be of good help to game developers, where during new player creation, the data set could be used to determine the overall performance and the preferred position. Since, gaming characters evolve over time in career modes, this statistical learning method wouloped over the years. We can also classify each player to each position. Till now the methods and selection was random and a lot of importance was given to influence and lucld also serve as a tool for the further machine learning.

5.2 Gap in Literature

We have found a way to link the performance of each player in a sport using the various skills that player had devek. But our statistical learning method will provide the user to select the best team by following actual data rather than just random guess.

5.3 What is the implication of the result in terms of the objectives you have set?

Our objective was to build a statistical model to predict the overall performance of players and classify the players based on their preferred positions. Through the various techniques, we were able to narrow down to few good methods that lead to fairly accurate results. However, since the player attributes would continuously evolve over time, the statistical model which yields least error might change. However, since there is no marked difference in the error rate, it is expected that the results over a period would mimic our existing predictions to the most extent.

6. Conclusion

There is no best method as such for any statistical learning method. The error rate will vary with the objective and the method we adopt for the analysis. In our data set random forests gave the best result to predict the overall performance of the players and goalkeepers. As for classification of players into forwards, mid fielders, defenders the error rate is minimum for KNN. Hence, it is a recommendation for the game developers, for machine learning to be performed to capture the evolution of players through time to arrive at the best predictions for a given instant.

Some future works could be explored

1) This method can be adopted to any sport.
2) We could further classify forward, mid fielders and defender in 27 sub classes for more clarity to further drill down.
3) Exploratory data analysis for best clubs or best countries which consistently produce high quality players can be classified.
4) This analysis would be helpful in cases where, people tend to bet on odds of a team winning or losing.

7. References

01. https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/.

02. R, Introduction to Statistical Learning with applications in.

03. http://statweb.stanford.edu/~tibs/lasso/lasso.pdf.

04. http://www.public.iastate.edu/~maitra/stat501/lectures/MultivariateRegression.pdf.

06. Predicting Skill Learning in a Large, Longitudinal MOBA Dataset – M.aung, et al.

07. Clustering of player behavior in computer games in the wild A drachen, R sifa, C Bauckhage.

08. https://www.easports.com/fifa.

09. https://www.Kaggle.com.

10. https://sofifa.com/.

11. https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Principal_Components_Regression.pdf.

Appendix

```
library(glmnet)
library(leaps)
library(randomForest)
library(gbm)
library(pls)
library(MASS)
library(class)
library(adabag)
library(caret)
```

#Regression

##ONFIELD OVERALL REGRESSION

```
#Load the CSV file
data <- read.csv(file.choose())
#Remove the unnecessary columns
fifa <- data[-c(1,3:6,8,9,12:19)]
##Split into Goal keeper and Field Player
fifa.gk <- fifa[which(fifa$prefers_gk=="True"),]
fifa.fp <- fifa[which(fifa$prefers_gk=="False"),]
##ONFIELD PLAYER
##Remove GK column
fifa.fp <- fifa.fp[-79]
##Remove Columns with only one level
fifa.fp <- Filter(function(x)(length(unique(x))>1), fifa.fp)
fifa.fp <-  na.omit(fifa.fp)
fifa.fp <- Filter(function(x)(length(unique(x))>1), fifa.fp)
##Remove name column
fifa.fp1 <- fifa.fp[-1]

##ONLY NUMERICAL PREDICTORS CONSIDERED FOR REGRESSION
fifa.fp2 <- fifa.fp1[-c(15:17,78:133)]
numpred=ncol(fifa.fp2)-1
fifa.fptemp=fifa.fp2[-c(4)]
fifa.fptemp
zscaletrain=preProcess(fifa.fptemp[,1:numpred])
scaledx=predict(zscaletrain,fifa.fptemp[,1:numpred])
corrpred=findCorrelation(cor(scaledx),cutoff=0.8)
uncor=scaledx[,-corrpred]
pcatrain=preProcess((uncor),method = "pca",thresh=0.8)
pcatrain
set.seed(3)
pr.out=prcomp(fifa.fptemp[1:numpred],scale=TRUE)
pr.out$rotation
names(pr.out)
pr.out$centre
pr.out$scale
pr.out$rotation
biplot(pr.out,scale=0)
pr.out$sdev
pr.var=pr.out$sdev^2
pr.var
pve=pr.var/sum(pr.var)
sum=0
```

```
pve


#cumulative sum of variance ratios
for (i in 1:numpred )
{
 if (i==1)
 {sum[1]=pve[i]}
 else
 {sum[i]=sum[i-1]+pve[i]}
}
sum[1:numpred]


#calculating pve indices for different % variances
varlim=0.8
j=0
for ( i in 2:numpred-1)
{
 if (sum[i]>=varlim && sum[i-1]<varlim)
 {j=i}
 else
 {j=j}
}
j


##SPLIT INTO TRAINING AND TEST DATA - ONLY NUMERICAL PREDICTORS
set.seed(3)
train.fp <- sample(1:nrow(fifa.fp2),nrow(fifa.fp2)-100)
test.fp <- fifa.fp2[-train.fp,]


##LINEAR MODEL
lm.fp <- lm(overall~.,data=fifa.fp2,subset=train.fp)
lm.predict <- predict(lm.fp,new=test.fp)
msefp <- mean((lm.predict-test.fp$overall)^2)
#The Mean Square Error is
msefp


##BACKWARD ELIMINATION WITH TRAINING DATA
regfit.bwd <- regsubsets(overall~.,data=fifa.fp2[train.fp,],nvmax=73,method="backward")
reg.summary <- summary(regfit.bwd)
reg.summary
#model matrix for test
test.mat <- model.matrix(overall~.,data=test.fp)
#Compute test mse
val.errors <- rep(NA,57)
for (i in 1:57){
 coefi <- coef(regfit.bwd,id=i)
 pred <- test.mat[,names(coefi)]%*%coefi
 val.errors[i] <- mean((test.fp$overall-pred)^2)
}
val.errors


#Find the best model
which.min(val.errors)
```

```
coef(regfit.bwd,30)

#Using adjR2
reg.summary$adjr2
which.max(reg.summary$adjr2)
#Using Cp
reg.summary$cp
which.min(reg.summary$cp)
#Using BIC
reg.summary$bic
which.min(reg.summary$bic)

##LASSO
#Fit the lasso with training data
x <- model.matrix(overall~.,fifa.fp2)[,-1]
y <- fifa.fp2$overall
lasso.mod <-
glmnet(x[train.fp,],y[train.fp],alpha=1)
#Use CV to find optimal lambda
set.seed(1)
cv.out <-
cv.glmnet(x[train.fp,],y[train.fp],alpha=1)
plot(cv.out)
bestlam <- cv.out$lambda.min
#Refit the lasso with the best lamda
lasso.mod <-
glmnet(x[train.fp,],y[train.fp],alpha=1,lam
bda=bestlam)
#Predict The Coefficient estimates
lasso.pred <-
predict(lasso.mod,s=bestlam,newx=x[-
train.fp,])
#MSE OF LASSO
mean((lasso.pred-y[-train.fp])^2)

##RANFDOM FORESTS
p <- round((ncol(fifa.fp2)-1)/3)
p
set.seed(3)
rf.fifafp <-
randomForest(overall~.,data=fifa.fp2,subs
et=train.fp,mtry=p,ntree=100,importance=
TRUE)
#Predict the test data
test.rf <- predict(rf.fifafp,newdata=test.fp)
#TEST MSE
mean((test.rf - test.fp[,"overall"])^2)
#Which Variables are important
varImpPlot(rf.fifafp)

#PCR

require(pls)
pcr_modelfp <-
pcr(overall~.,data=fifa.fp2,subset=train.fp,
scale=TRUE, validation="CV")
pcr_predfp <-
predict(pcr_modelfp,newdata=test.fp,nco
mp=21)
summary(pcr_modelfp)
mean((pcr_predfp - test.fp[,"overall"])^2)

#BOOSTING
set.seed(3)
boost.fifafp <-
gbm(overall~.,data=fifa.fp2[train.fp,],distri
bution="gaussian",

n.trees=100,interaction.depth=1)
summary(boost.fifafp)
```

```
par(nfrow=c(1,2))

plot(boost.fifafp,i="reactions")

plot(boost.fifafp,i="potential")

boost.fifafp.pred <-
predict(boost.fifafp,newdata=test.fp,n.trees
=100)

#MSE

mean((boost.fifafp.pred-test.fp$overall)^2)

#PCR

require(pls)
pcr_modelfp <-
pcr(overall~.,data=fifa.fp2,subset=train.fp,
scale=TRUE, validation="CV")
pcr_predfp <-
predict(pcr_modelfp,newdata=test.fp,nco
mp=21)
summary(pcr_modelfp)
mean((pcr_predfp - test.fp[,"overall"])^2)
```

#GOALKEEPER OVERALL REGRESSION

```
##GOALKEEPER PLAYER

head(fifa.gk)

colnames(fifa.gk)

fifa.gk <- fifa.gk[-79]

fifa.gk <-
Filter(function(x)(length(unique(x))>1),
fifa.gk)

head(fifa.gk)

fifa.gk <-  na.omit(fifa.gk)

head(fifa.gk)

fifa.gk <-
Filter(function(x)(length(unique(x))>1),
fifa.gk)

fifa.gk1 <- fifa.gk[,-1]

head(fifa.gk1)

ncol(fifa.gk1)

colnames(fifa.gk1)

head(fifa.gk1)

#Removing classification vectors

fifa.gk2 <- fifa.gk1[,-c(14,49:64)]

head(fifa.gk2)

numpred=ncol(fifa.gk2)-1

fifa.gktemp=fifa.gk2[-c(4)]

fifa.gktemp

zscaletrain=preProcess(fifa.gktemp[,1:num
pred])

scaledx=predict(zscaletrain,fifa.gktemp[,1:
numpred])

corrpred=findCorrelation(cor(scaledx),cut
off=0.8)

uncor=scaledx[,-corrpred]

pcatrain=preProcess((uncor),method =
"pca",thresh=0.8)

pcatrain

set.seed(3)

pr.out=prcomp(fifa.gktemp[1:numpred],sc
ale=TRUE)

pr.out$rotation

names(pr.out)

pr.out$centre

pr.out$scale

pr.out$rotation

biplot(pr.out,scale=0)

pr.out$sdev

pr.var=pr.out$sdev^2

pr.var

pve=pr.var/sum(pr.var)
```

```
sum=0
pve
#cumulative sum of variance ratios
for (i in 1:numpred )
{
 if (i==1)
 {sum[1]=pve[i]}
 else
 {sum[i]=sum[i-1]+pve[i]}
}
sum[1:numpred]

#calculating pve indices for different % variances
varlim=0.8
j=0
for ( i in 2:numpred-1)
{
 if (sum[i]>=varlim && sum[i-1]<varlim)
 {j=i}
 else
 {j=j}
}
j

##SPLIT INTO TRAINING AND TEST DATA - ONLY NUMERICAL PREDICTORS
set.seed(3)
train.gk <- sample(1:nrow(fifa.gk2),nrow(fifa.gk2)-100)
test.gk <- fifa.gk2[-train.gk,]

##LINEAR MODEL
lm.gk <- lm(overall~.,data=fifa.gk2,subset=train.gk)
summary(lm.gk)
lm.predictgk <- predict(lm.gk,new=test.gk)
msegk <- mean((lm.predictgk-test.gk$overall)^2)
#The Mean Square Error is
msegk
#3.48

##BACKWARD ELIMINATION WITH TRAINING DATA
ncol(fifa.gk2)
set.seed(3)
regfit.bwdgk <- regsubsets(overall~.,data=fifa.gk2[train.gk,],nvmax=54,method="backward")
reg.summarygk <- summary(regfit.bwdgk)
reg.summarygk
#model matrix for test
test.matgk <- model.matrix(overall~.,data=test.gk)
#Compute test mse
val.errorsgk <- rep(NA,1)
for (i in 1:41){
  coefigk <- coef(regfit.bwdgk,id=i)
  predgk <- test.matgk[,names(coefigk)]%*%coefigk
  val.errorsgk[i] <- mean((test.gk$overall-predgk)^2)
```

```
}
val.errorsgk
#3.269
#Find the best model
which.min(val.errorsgk)
coef(regfit.bwdgk,which.min(val.errorsgk)
)
#Using adjR2
reg.summarygk$adjr2
which.max(reg.summarygk$adjr2)
#35
#Using Cp
reg.summarygk$cp
which.min(reg.summarygk$cp)
#28
#Using BIC
reg.summarygk$bic
which.min(reg.summarygk$bic)
#14


#LASSO
#Fit the lasso with training data
x <- model.matrix(overall~.,fifa.gk2)[,-1]
x
y <- fifa.gk2$overall
lasso.mod <-
glmnet(x[train.gk,],y[train.gk],alpha=1)
#Use CV to find optimal lambda
set.seed(3)
cv.outgk <-
cv.glmnet(x[train.gk,],y[train.gk],alpha=1)
plot(cv.outgk)
bestlamgk <- cv.outgk$lambda.min
#Refit the lasso with the best lamda
lasso.modgk <-
glmnet(x[train.gk,],y[train.gk],alpha=1,la
mbda=bestlamgk)
#Predict The Coefficient estimates
lasso.predgk <-
predict(lasso.modgk,s=bestlamgk,newx=x[
-train.gk,])
#MSE OF LASSO
mean((lasso.predgk-y[-train.gk])^2)
#2.826
coef(lasso.modgk)


#RANFDOM FORESTS


p <- round((ncol(fifa.gk2)-1)/3)
p
set.seed(3)
rf.fifagk <-
randomForest(overall~.,data=fifa.gk2,subs
et=train.gk,mtry=p,ntree=100,
importance=TRUE)


#Predict the test data
test.rfgk <-
predict(rf.fifagk,newdata=test.gk)
#TEST MSE
mean((test.rfgk - test.gk,"overall"])^2)
#1.884
#Which Variables are important
rf.fifagk$importance
```

```
varImpPlot(rf.fifagk)


#PCR

require(pls)

pcr_modelgk <-
pcr(overall~.,data=fifa.gk2,subset=train.gk
,scale=TRUE, validation="CV")

summary(pcr_modelgk)

pcr_predgk <-
predict(pcr_modelgk,newdata=test.gk,nco
mp=21)

mean((pcr_predgk - test.gk[,"overall"])^2)

#0.244


#BOOSTING



set.seed(3)

boost.fifagk <-
gbm(overall~.,data=fifa.gk2[train.gk,],dist
ribution="gaussian",
n.trees=100,interaction.depth=1)

summary(boost.fifagk)

boost.fifagk.pred <-
predict(boost.fifagk,newdata=test.gk,n.tree
s=100)

#MSE

mean((boost.fifagk.pred-
test.gk$overall)^2)

#2.59


##CLASSIFICATION

##FP ATTRIBUTES CLASSIFICATION


##DATA SET WITH AGGREGATED
POSITIONS

fifa.agg <- fifa.fp1[,c(120:133)]

fifa.agg$position[fifa.agg$prefers_cf ==
"True"] <- "Striker"

fifa.agg$position[fifa.agg$prefers_cam ==
"True"] <- "Striker"

fifa.agg$position[fifa.agg$prefers_cb ==
"True"] <- "Defender"

fifa.agg$position[fifa.agg$prefers_cdm ==
"True"] <- "Midfielder"

fifa.agg$position[fifa.agg$prefers_cm ==
"True"] <- "Midfielder"

fifa.agg$position[fifa.agg$prefers_lb ==
"True"] <- "Defender"

fifa.agg$position[fifa.agg$prefers_lm ==
"True"] <- "Midfielder"

fifa.agg$position[fifa.agg$prefers_lw ==
"True"] <- "Striker"

fifa.agg$position[fifa.agg$prefers_lwb ==
"True"] <- "Defender"

fifa.agg$position[fifa.agg$prefers_rm ==
"True"] <- "Midfielder"

fifa.agg$position[fifa.agg$prefers_rw ==
"True"] <- "Striker"

fifa.agg$position[fifa.agg$prefers_rwb ==
"True"] <- "Defender"

fifa.agg$position[fifa.agg$prefers_st ==
"True"] <- "Striker"

fifa.agg$position[fifa.agg$prefers_rb ==
"True"] <- "Defender"


##CLASSIFICATION DATA SET

fifa.cl <- fifa.fp1[,-c(120:133)]

fifa.cl$position <- fifa.agg$position
```

```
fifa.cl$position <-
as.factor(fifa.cl$position)


##SPLIT INTO TRAINING AND TEST
DATA SET

set.seed(3)

train <-
sample(1:nrow(fifa.cl),nrow(fifa.cl)-100)

test <- fifa.cl[-train,]

fifa.cl <- na.omit(fifa.cl)


##RANDOM FORESTS

#No of predictors

p <- ncol(fifa.cl)-1

p <- round(sqrt(p))


set.seed(3)

rf.fifacl <- randomForest(position ~ .,
data= fifa.cl, subset = train, mtry=p,
ntree=100, importance=TRUE)

pred.rf <-
predict(rf.fifacl,newdata=fifa.cl[-
train,],type="class")

tab <- table(pred.rf,test$position)

##TABLE

tab

##TEST ERROR RATE

test.error.rate.rf <-
mean(pred.rf!=test$position)

test.error.rate.rf

##VARIABLES IMPORTANCE

varImpPlot(rf.fifacl)

##LDA

lda.fit <-
lda(position~.,data=fifa.cl,subset=train)

lda.pred <- predict(lda.fit,test)

lda.class <- lda.pred$class

#TABLE

table(lda.class,test$position)

#TEST ERROR RATE

test.error.rate.lda <-
mean(lda.class!=test$position)

test.error.rate.lda


##KNN

##CONVERT IN NUMERICAL
VALUES

fifa.num <- fifa.cl

for(i in 78:119)

{

  fifa.num[,i] <-
as.integer(fifa.num[,i]=="True")

}

lookup <- c("Right"=1,"Left"=0)

fifa.num$preferred_foot <-
lookup[fifa.cl$preferred_foot]

lookup1 <-
c("High"=3,"Medium"=2,"Low"=1)

fifa.num$work_rate_att <-
lookup1[fifa.cl$work_rate_att]

fifa.num$work_rate_def <-
lookup1[fifa.cl$work_rate_def]

fifa.num$position <- fifa.cl$position

##SPLIT THIS DATA

set.seed(3)
```

```
train <-
sample(1:nrow(fifa.num),nrow(fifa.num)-
100)

test <- fifa.num[-train,]

test.error.rate.knn <- c()

for(i in 1:100)

{

  set.seed(3)

  knn.cl <- knn(fifa.num[train,-
120],fifa.num[-train,-
120],fifa.num[train,]$position,k=i)

  tab <- table(knn.cl,test$position)

  #ERROR RATE

  test.error.rate.knn[i] <-
mean(knn.cl!=test$position)

}

test.error.rate.knn

which.min(test.error.rate.knn)

##LEAST ERROR RATE - 0.14 FOR K =
20

##REVISIT KNN FOR K = 20

knn.cl <- knn(fifa.num[train,-
120],fifa.num[-train,-
120],fifa.num[train,]$position,k=20)

table(knn.cl,test$position)

mean(knn.cl!=test$position)


##BOOSTING

##USING ADABOOST

set.seed(3)

fifa.adaboost <- boosting(position
~.,data=fifa.num[train, ], mfinal=100,
coeflearn="Zhu",
control=rpart.control(maxdepth=1))

fifa.adaboost.pred <-
predict.boosting(fifa.adaboost,
newdata=fifa.num[-train, ])

##CONFUSITON MATRIX TABLE

fifa.adaboost.pred$confusion

##TEST ERROR RATE

fifa.adaboost.pred$error

##Compare error evloution in training and
test set

errorevol(fifa.adaboost,newdata=fifa.num[
train, ])->evol.train

errorevol(fifa.adaboost,newdata=fifa.num[
-train, ])->evol.test

plot.errorevol(evol.test,evol.train)
```