FIFA Dataset Analysis

NAME	CONTRIBUTION
Aditya Wagholikar	20%
Biswajeet Mohapatra	20%
Subodh Ghuge	20%
Nayan Keni	20%
Neelesh Jayaraman	20%

Submitted to:
Dr. Na Zou,
Instructional Assistant Professor,
Dept. of Industrial and Systems Engineering
Texas A&M University
Submitted on:
December 7, 2018

Table of Contents

1. Introduction

1.1 Importance of the problem

Soccer is a field sport currently played by more than 200 million players across 200 countries and is the most popular sport globally. It is a sport played between two teams consisting of eleven players each on a rectangular grass field and the objective is to score as many goals as possible by sending the ball over the goal line and into the goal post at each end. Each game is played over 90 mins which are divided into two halves of 45 minutes each with the provision of extra time to compensate for any delays. Each team has a goalkeeper to guard the goal and is the only player allowed to touch the ball with hands inside the penalty box, all other players must not touch the ball with hands while ball is inside the playing field.

Association Football (soccer) is a $1.48 billion industry with stakeholders and fans across the globe. The increasing amount of talent and the money involved has attracted a lot of sports analytics firms to study the game and help the management in making crucial decisions about the team formations. Every major club around the globe is investing in analytics to understand more about the opponent as well their own strengths and weaknesses, to help them create a winning strategy for each game.

In our project we will be analyzing the dataset containing the strengths and weaknesses of each player and creating a model to help us predict the ideal position on field that we should play a specific player. This project will help us solve the problem of optimizing the playing position of each player. We can extend this project to create a winning strategy by predicting the opposition team and then suggest the best positions for each of our player to optimize our team formation.

1.2 Objective

Currently, managers and coaches spent hours deciding the formation and the best position for players on the soccer field. With this project, the manager and the think-tank would not have to spend hours identifying the same, instead they could focus their efforts on building strategies with the formation at hand. This project utilizes different data analytics techniques to identify the best position for a soccer player based on some criteria like his skill sets and other features. This is particularly useful when a coach or a manager needs to prepare the formation for his team before a match. It is also useful during in-game changes when you have the best position for a player and you could build your team around the star player.

This could also help us understand if a player prefers multiple positions. Using this information would allow the managers to make changes to the team based on the opponents' strengths and weaknesses. In a competitive team-sport like soccer, each vital bit of information about a player is worth goldmine.

Teams could leverage every piece of information available to create strategies and solutions to be victorious.

2. Project Approach

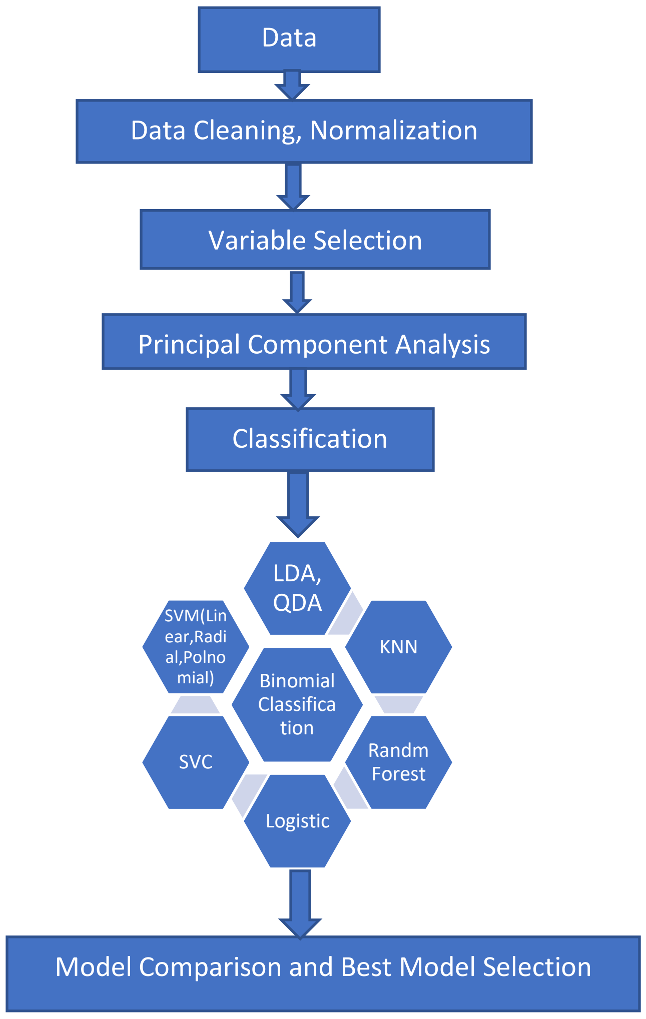
2.1 Data Description

The dataset used for this project is the FIFA 18 complete player dataset taken from Kaggle. There are approximately 18,000 observations/player in this dataset with each player having 185 fields which give information about the player like age, club, league, nationality, salary and other physical attributes.

Some of the attributes used to classify a player as an attacker or a defender include Overall, PAC, SHO, PAS, Dribbling, Crossing, heading accuracy, free kick accuracy etc.

The key challenge in processing the dataset is to find the appropriate attributes that would help to correctly classify the players either as attacker or defender, thereby ensuring optimal performance from the team.

2.2 Methodology


Data
Data Cleaning, Normalization
Variable Selection
Principal Component Analysis
Classification
LDA, QDA
SVM(Linear,Radial,Polnomial)
KNN
Binomial Classification
SVC
Randm Forest
Logistic
Model Comparison and Best Model Selection

In the beginning, data cleaning process was performed to remove the un-related variables as well as fields with zero or NULL values from the data set. The data set had 185 predictors initially, after cleaning the number of variables reduced to 105. Next, we conducted Principal Component Analysis, followed by correlation, thereby reducing the variables to 72 variables. Once we identified the predictors that contributed the most for prediction, we divided the reduced dataset into training and test data set. The train and test data split ratio were kept 1: 1. We used the training data to train the model and used the test data to validate the accuracy of models in prediction. We reduced the possible values of outcome variable i.e. Final position to binary (either 0(defending) or 1(attacking), thereby mapping the variables as below:

Position	Description	Boolean value
rs	Right Striker	1
rw	Right Winger	1
rf	Right Forward	1
ram	Right Attacking Midfield	1
rcm	Right Center Midfield	1
rm	Right Midfield	1
rdm	Right Center Midfield	0
Rcb	Right Center back	0
Rb	Right Back	0
Rwb	Right Winger Back	0
St	Striker	1
Lw	Left Winger	1
Cf	Center Forward	1
Cam	Center Attacking Midfield	1
Cm	Center Midfield	1
Lm	Left Midfield	1
Cdm	Center Defensive Midfield	0
Cb	Center Back	0
Lb	Left Back	0
Lwb	Left Winger Back	0
Ls	Left Striker	1
Lf	Left Forward	1
Lam	Left Attacking Midfield	1
Lcm	Left Center Midfield	1
Ldm	Left Defense Midfield	0
Lcb	Left Center Back	0
Gk	Goal Keeper	0

Table 1 - Mapping of football positions to Boolean Values

We used different classification models such as Linear Discriminant Analysis, Quadratic discriminant Analysis, Support Vector Machine (polynomial, linear and radial kernel), Support vector classifier, Logistic regression, random forest to train the data and then performed test on the test data to get the test error rate. At the end, we compared the models based on the test error rate and selected the best model.

2.3 Reason for our approach

The prime objective of our project was to get accurate prediction of the position (either attacking or defending) a player should be played. we realize that classification model would help us provide the appropriate results, though we have so many statistical models available to apply to this scenario. From the coursework we learnt that classification models LDA, QDA, Logistic, SVM, SVC, Random

Forest, KNN have equal probability of giving results, so we applied all these models to our dataset to find the best model based on the least test error rates obtained. We understand that multiple models can give similar results, but we had to do appropriate trade off and make the best decision.

3. Implementation Details

3.1 Data Cleaning/Normalization

After the initial scan of the dataset we found that few fields are not significant as well as few of the rows contain undesired values such as “NA” or “NULL” etc. We performed data cleaning to keep only 103 desired columns and discard the rest from the dataset as below:

```
colsToKeep <-
c("special","age","height_cm","weight_kg","overall","potential","pac","sho","pas","dri","def","phy",
"international_reputation","skill_moves","weak_foot","crossing","finishing","heading_accuracy","short
_passin",
"volleys","dribbling","curve","free_kick_accuracy","long_passing","ball_control","acceleration","sprint
_speed",
"agility","reactions","balance","shot_power","jumping","stamina","strength","long_shots","aggression
",
"interceptions","positioning","vision","penalties","composure","marking","standing_tackle","sliding_ta
ckle",
"gk_diving","gk_handling","gk_kicking","gk_positioning","gk_reflexes","rs","rw","rf","ram","rcm","rm"
,"rdm","rcb",
"rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk",
"prefers_rs","prefers_rw","prefers_rf","prefers_ram","prefers_rcm","prefers_rm","prefers_rdm","prefe
rs_rcb",
"prefers_rb","prefers_rwb","prefers_st","prefers_lw","prefers_cf","prefers_cam","prefers_cm","prefers
_lm",
"prefers_cdm","prefers_cb","prefers_lb","prefers_lwb","prefers_ls","prefers_lf","prefers_lam","prefers
_lcm",         "prefers_ldm","prefers_lcb","prefers_gk");
```

```
removePrefers <-
c("prefers_rs","prefers_rw","prefers_rf","prefers_ram","prefers_rcm","prefers_rm","prefers_rdm","pref
ers_rc",
"prefers_rb","prefers_rwb","prefers_st","prefers_lw","prefers_cf","prefers_cam","prefers_cm","prefers
_lm",
"prefers_cdm","prefers_cb","prefers_lb","prefers_lwb","prefers_ls","prefers_lf","prefers_lam","prefers
_lcm","prefers_ldm","prefers_lcb","prefers_gk")
```

Next, we created 2 new columns for preferred and final positions and set Boolean values for all possible positions as below:

#set of all possible positions

position =
c("rs","rw","rf","ram","rcm","rm","rdm","rcb","rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk");

booleanPosition = c(1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0);

3.2 Variable Selection:

At first, we performed correlation among variables on our dataset to know how many predictors are interrelated to make our analysis easy. Below are the results obtained from the function:
#plots to check collinearity

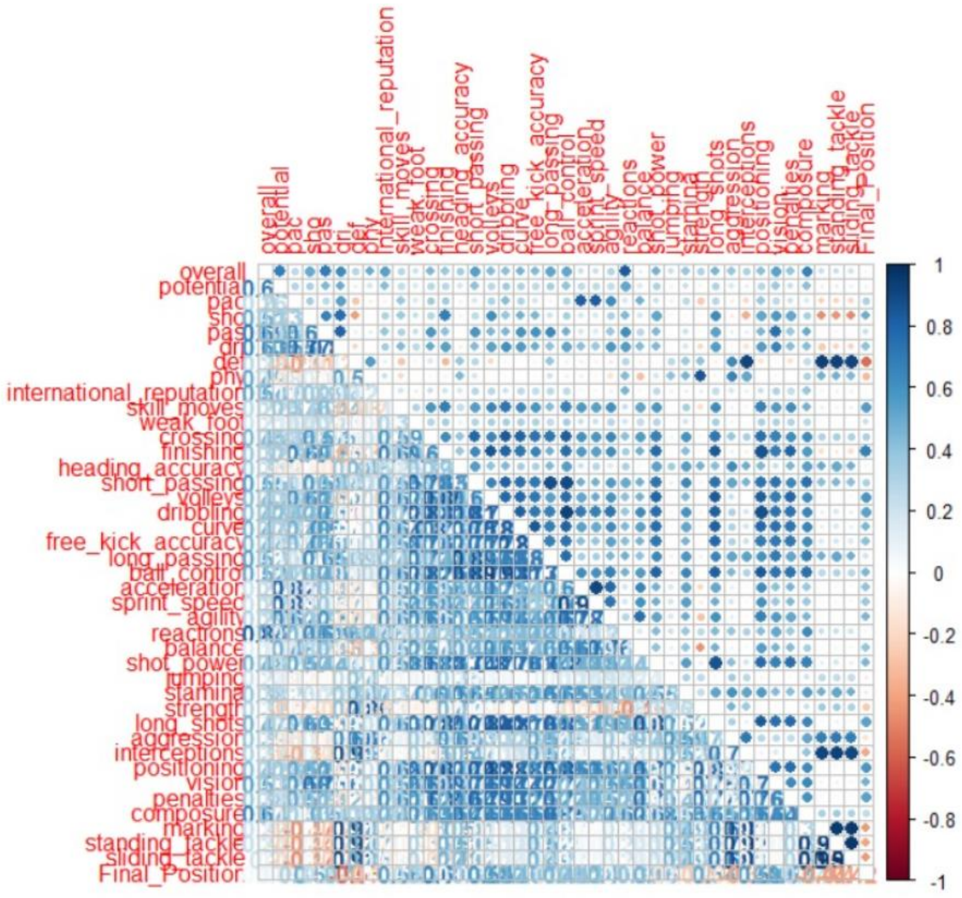


Figure 1 - Correlation plot

removeCharCols<- c("Pref_Position","special", "age", "height_cm","weight_kg", "gk_diving","gk_handling","gk_kicking","gk_positioning","gk_reflexes","rs","rw","rf","ram","rcm","rm"

,"rdm","rcb","rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk");str(withoutPrefers); correlation<-cor(withoutPrefers[,!colnames(withoutPrefers) %in% removeCharCols], method="pearson"); library(corrplot); corrplot.mixed(correlation,tl.pos = 'lt');

collinear<-c("gk_diving","gk_handling","gk_kicking","gk_positioning","gk_reflexes","rs","rw","rf","ram","rcm","rm","rdm","rcb", "rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk");

corrplot.mixed(cor(withoutPrefers[,collinear]),tl.pos = 'lt');

Next, we performed Principal Component Analysis to identify the crucial predictors that need to be considered for applying supervised classification models to predict the position player should be played. After applying PCA we found that approx. 85% variance is explained by 72 predictors in the dataset. Below is the result obtained from after running the PCA on our dataset.

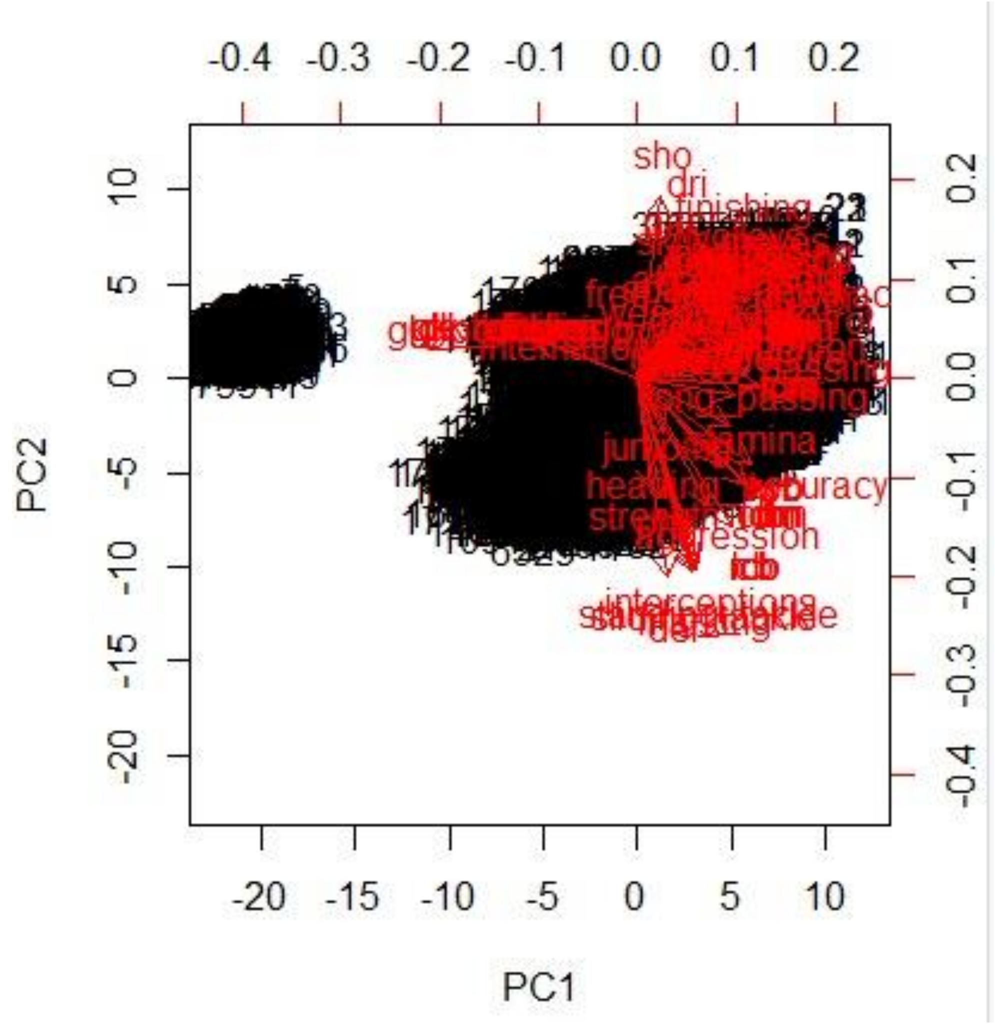


Figure 2- PCA Output

Code #

removePCAcols <- c("special","age","height_cm","weight_kg","Pref_Position","Final_Position");

PCAdata <- withoutPrefers[,!colnames(withoutPrefers) %in% removePCAcols]; apply(PCAdata, 2, mean); apply(PCAdata, 2, var)

3.3 Classification Models:

Next, we applied different classification models to our dataset and got below results:

3.3.1 Linear Discriminant Analysis:

Linear discriminant analysis is used to compute directions that will represent the axes to maximize the separation between classes and we got a mean prediction accuracy approx. 88%. Below are the plots for the same.

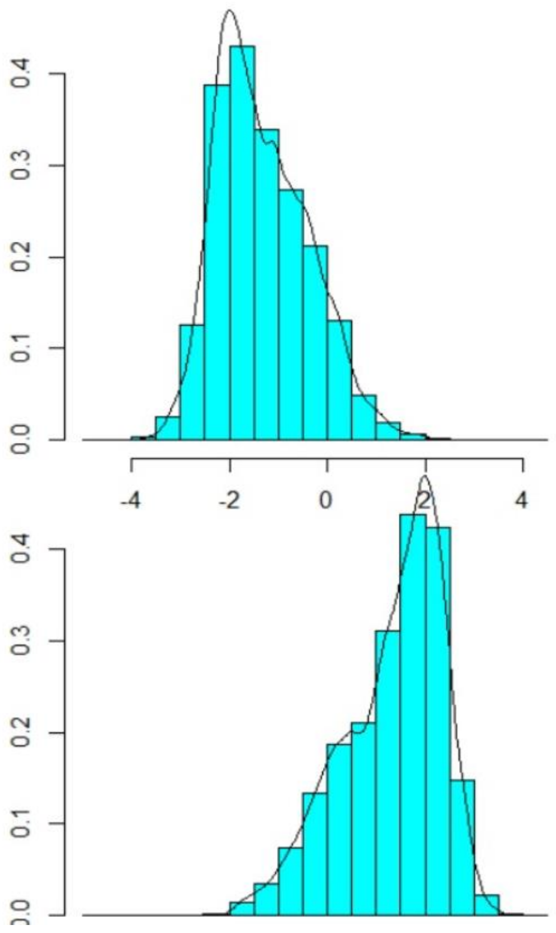


Figure 3- LDA plots

#apply LDA

library(MASS)

lda.fit<-lda(Final_Position ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position - overall

-gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb

- rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk, data=wp.train)

lda.pred = predict(lda.fit, wp.test)

table(lda.pred$class, wp.test$Final_Position)

Next, we plotted the AUC curve to access the performance and area was found to be 0.8788 as below:

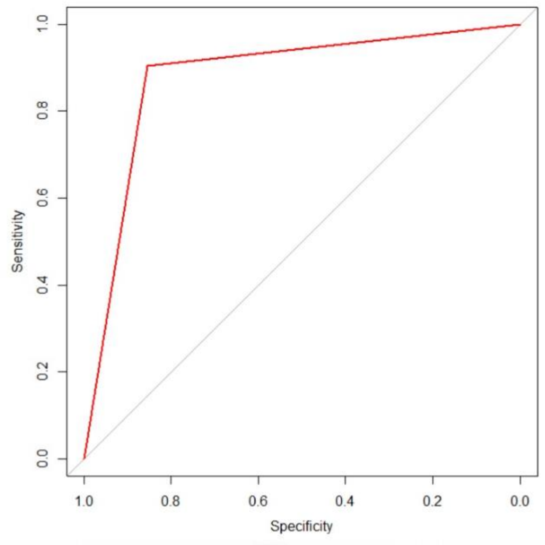
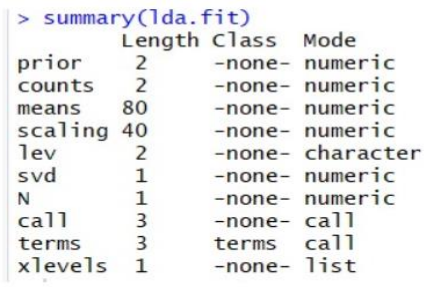


Figure 4 - AUC Curve

Below is the summary of results:



```
> summary(lda.fit)
        Length Class  Mode
prior    2     -none- numeric
counts   2     -none- numeric
means   80     -none- numeric
scaling 40     -none- numeric
lev      2     -none- character
svd      1     -none- numeric
N        1     -none- numeric
call     3     -none- call
terms    3     terms  call
xlevels  1     -none- list
```

Figure 5 - LDA Summary

3.3.2 Quadratic Discriminant Analysis:

Then, we performed quadratic discriminant analysis to increase the accuracy level on analysis and we didn't get better results with mean prediction accuracy as 0.74, thereby inferring that decision boundary as approximately linear.   Also, we plotted the AOC curve found the area as 0.8337.

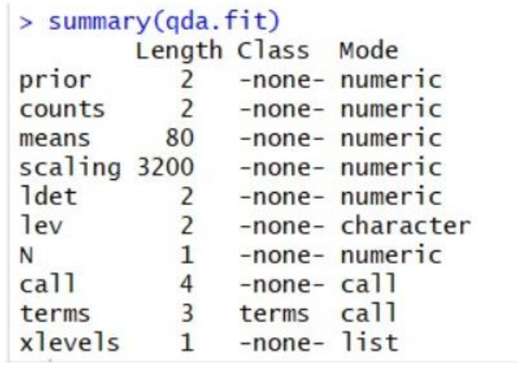
mean(qda.class ==wp.test$Final_Position)

[1] 0.7422751

> > rocnn<- roc(qda.class,wp.test$Final_Position)

> plot(rocnn,col="red")

Below is the summary of the results obtained:



```
> summary(qda.fit)
        Length Class  Mode
prior      2   -none- numeric
counts     2   -none- numeric
means     80   -none- numeric
scaling 3200   -none- numeric
ldet       2   -none- numeric
lev        2   -none- character
N          1   -none- numeric
call       4   -none- call
terms      3   terms  call
xlevels    1   -none- list
```

Figure 6 - QDA Summary

3.3.3 Logistic regression:

Next, we applied logistic regression to our dataset and obtained better results with mean prediction accuracy of 88.39 %. Logistic regression is one of the appropriate classification models that give accurate results when dependent variable is binary.

wp.logit<-glm(Final_Position ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position - overall -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb-rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk, data= wp.train, family="binomial"); wp.probs<-predict(wp.logit, wp.test, type="response")

Then we plotted ROC curve to compare the performance against the previous models and found AUC as 0.8839.

Test Error rate we got 1 – 0.8857 = 0.1143 ~ 11.43% which is good.

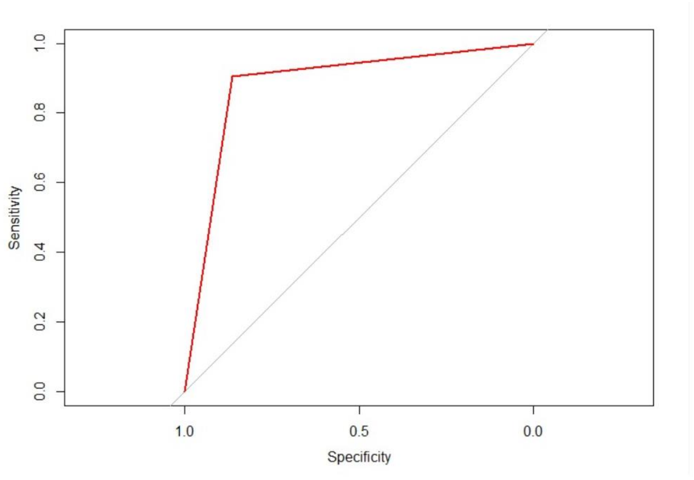


Figure 7- ROC Curve

We also plotted probability vs Data point graph as below and we can infer that many of the data points are close to either 0 or 1, thereby showing most of the points are correctly classified.

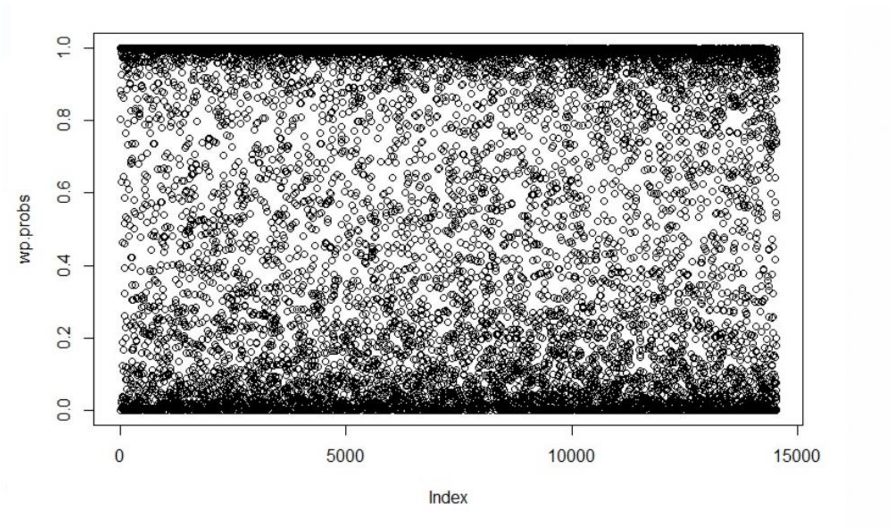


Figure 8 - probability vs Data point graph

3.3.4 K Nearest Neighbor:

As we have multiple predictors, we tried to check the situation whether the decision boundary can be non-linear. KNN, being a non-parametric approach, we don't do initial assumptions about the shape of decision boundary. After applying the KNN we found the prediction accuracy as 0.87. We chose k = 5 and k =10, but got better results with k =10.

knn.pred=knn(wp.train.predictors, wp.test.predictors, wp.train.response, k=10);

Below is the confusion matrix for KNN :

```
knn.pred    0    1
        0 5504   936
        1   948  7143                (AUC 0.8687)
```

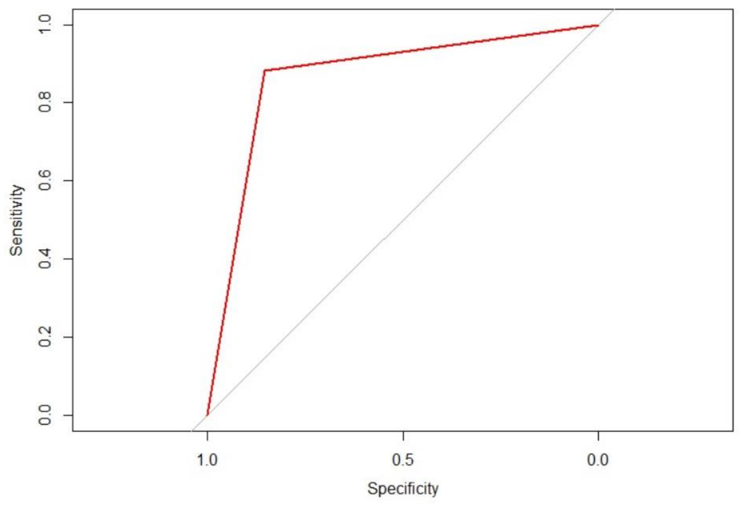
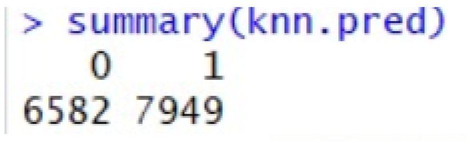


Figure 9 - AUC Curve

Below is the summary of the results obtained:



```
> summary(knn.pred)
   0    1
6582 7949
```

Figure 10 - Result Summary

i.e. 6582 data points belong to defending class (0) and 7949 data points belong to attacking class (1).

3.3.5 Support vector Classifier:

Then we applied support vector classifier to our dataset and found below results. we chose to select cost value as 0.01, 0.1 and 1 and found that cost=1 gave the optimal result with mean prediction accuracy 0.8826646.

library(e1071)

wp.svmfit <- svm(factors ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position - overall   -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -

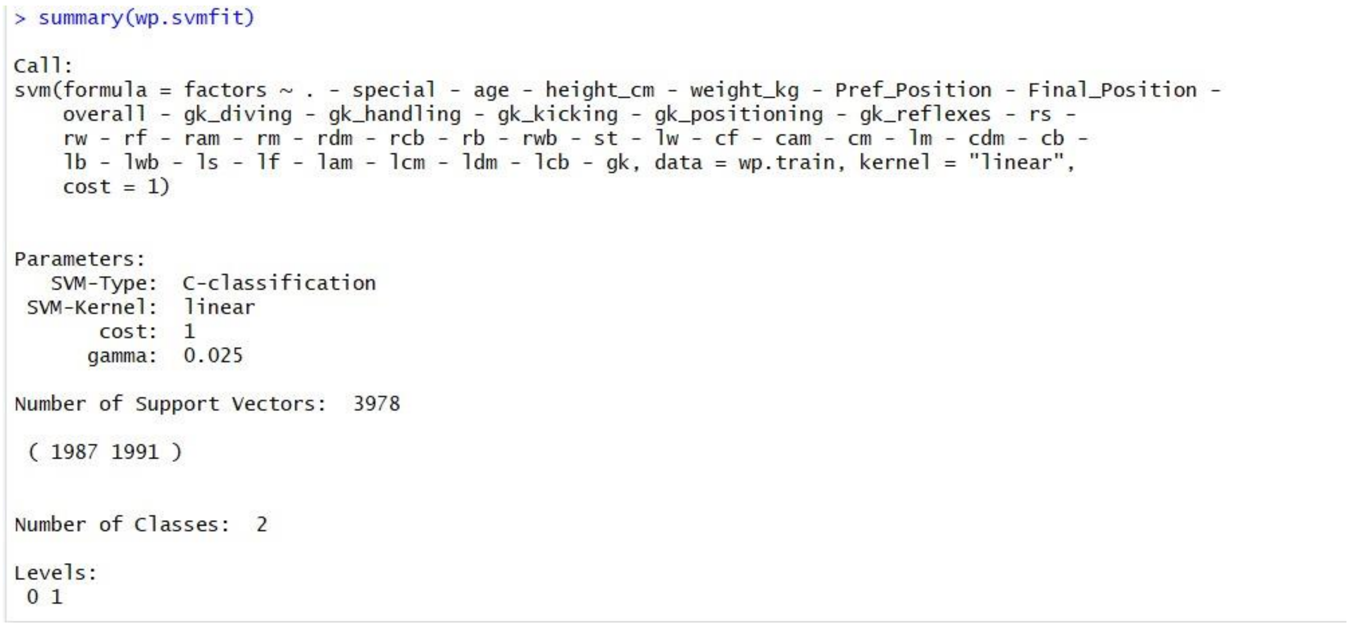
rdm -rcb- rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk , data= wp.train, kernel="linear", cost=1)

The confusion matrix obtained from the model:

```
predict    0      1
     0    5707   1021
     1    684    7119
```

Test Error rate: 11.8 %

Below is the summary of the results obtained :



```
> summary(wp.svmfit)

Call:
svm(formula = factors ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position -
    overall - gk_diving - gk_handling - gk_kicking - gk_positioning - gk_reflexes - rs -
    rw - rf - ram - rm - rdm - rcb - rb - rwb - st - lw - cf - cam - cm - lm - cdm - cb -
    lb - lwb - ls - lf - lam - lcm - ldm - lcb - gk, data = wp.train, kernel = "linear",
    cost = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
      gamma:  0.025

Number of Support Vectors:  3978

 ( 1987 1991 )


Number of Classes:  2

Levels:
 0 1
```

Figure 11 - Svm Summary

3.3.6 Support Vector Machine (Polynomial):

Next, we applied polynomial support vector machine to our dataset and got below results. Polynomial support vector machine gives better result in both linear or non-linear boundary with separable or non-separable cases.

We use below polynomial kernel function to calculate the response function which decides the class the data point belongs to.

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle$$

$$\downarrow$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

Figure 12 - SVM function

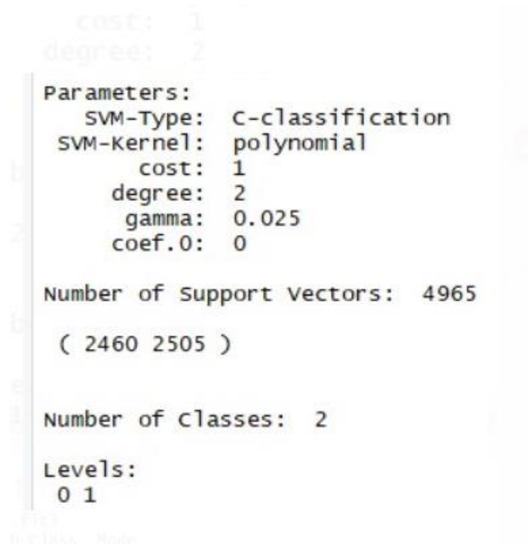
Polynomial kernel

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \quad \text{tuning parameter: } d$$

Figure 13 - Polynomial Kernel

wp.svmfit <- svm(factors ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position - overall -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk, data= wp.train, kernel="polynomial", degree=2 ,cost=1)

Below is the summary of the results obtained:



```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  2
      gamma:  0.025
     coef.0:  0

Number of Support Vectors:  4965

 ( 2460 2505 )


Number of Classes:  2

Levels:
 0 1
```

Figure 14 - SVM Summary

3.3.7 Support Vector Machine (Radial):

Next, we used radial kernel function to calculate response function and got mean prediction Accuracy rate as 88.5%.

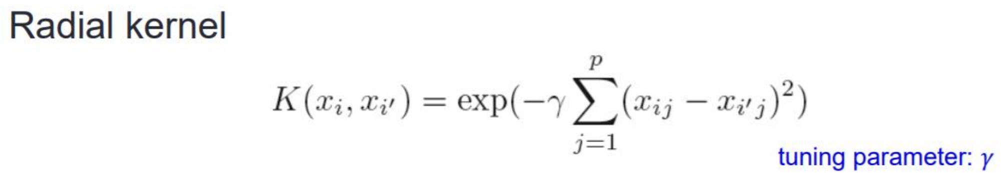
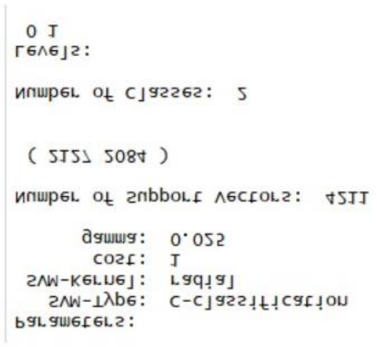

Radial kernel
$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$
tuning parameter: γ

Figure 15 - Radial Kernel

wp.svmfit <- svm(factors ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position - overall + -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb-rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk, + data= wp.train, kernel="radial", cost=1)

Below is the summary of results obtained



```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.025

Number of Support Vectors:  4211

 ( 2084 2127 )

Number of Classes:  2

Levels:
 0 1
```

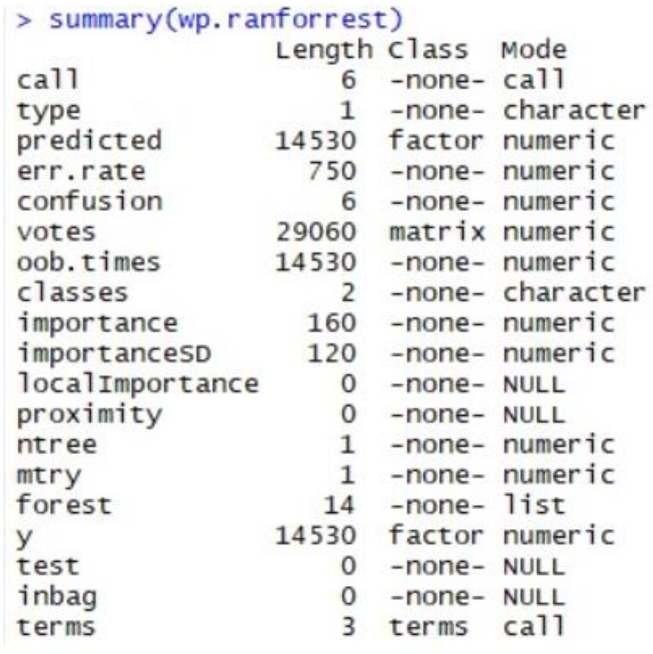
Figure 16 - Radial Kernel Summary

3.3.7 Random Forest:

Next, we applied random forest model to our dataset. Before applying the method, we had 39 training predictors and we chose m = sqrt(p) ~ 7, thereby resulting in 250 trees. The mean prediction accuracy rate obtained was 84.4%.

> wp.ranforrest= randomForest(factors ~. - special - age - height_cm - weight_kg - Pref_Position - Final_Position -overall + -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb+   - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk, + data= wp.train, mtry= 7, ntree= 250, importance=TRUE);

Below is the summary of the results obtained:



```
> summary(wp.ranforrest)
                Length Class  Mode
call                6  -none- call
type                1  -none- character
predicted       14530  factor numeric
err.rate          750  -none- numeric
confusion           6  -none- numeric
votes           29060  matrix numeric
oob.times       14530  -none- numeric
classes             2  -none- character
importance        160  -none- numeric
importanceSD      120  -none- numeric
localImportance     0  -none- NULL
proximity           0  -none- NULL
ntree               1  -none- numeric
mtry                1  -none- numeric
forest             14  -none- list
y               14530  factor numeric
test                0  -none- NULL
inbag               0  -none- NULL
terms               3  terms  call
```

Figure 17 - Random Forrest

4. Results Comparison:

Below is the table with error rate of classification models that we applied to our dataset.

Classification Method	Test Performance (Test Error Rate)
1) LDA	12 %
2) QDA	25.78 %
3) KNN	13%
4) SVC	11.8%
5) SVM(Polynomial)	11.86%
6) SVM(Radial)	11.5%
7) Random Forest	15.56%

Figure 18 - Result Comparison

Below is the table with confusion matrix with mean prediction accuracy for test data and Area Under Curve plot.

Method	Confusion Matrix	Mean Prediction accuracy for Test Data	AUC
LDA	$\begin{array}{ccc} & 0 & 1 \\ 0 & 5628 & 964 \\ 1 & 763 & 7176 \end{array}$	0.88	0.8788

QDA	0 1 0 2699 53 1 3692 8087	0.7422	0.8337
Logistic Regression	0 1 0 5695 903 1 757 7176	0.8857615	0.8839
KNN	0 1 0 5504 936 1 948 7143	0.8703462	0.8687
SVC	0 1 0 5707 1021 1 684 7119	0.8826646	0.8803
SVM (Polynomial)	0 1 0 5582 852 1 870 7227	0.8814947	0.8801
SVM (Radial)	0 1 0 5580 793 1 872 7286	0.8854174	0.8843
Random forest	0 1 0 5259 1068 1 1193 7011	0.8444016	0.8851

Figure 19 - Model Comparison

5. Executive Summary:

5.1 Importance of the problem

In soccer a player can play in multiple positions and it is important for any manager to understand the skills of each player and then use them to assign a particular position for the player for any match. A manager needs years of experience to observe the player and identify his strengths or weaknesses. To simplify this process of identifying the best position for a player we need an automated solution which analyzes historical data and then leverages that data to predict the optimal position.

5.2 Gap in literature

This project was started with the aim of providing a solution to the problem of identifying the best playing position for each soccer player. Even though there was a lot of literature available on identifying different playing positions, they were limited to one or two models at best. In order to obtain a holistic view of the problem we included multiple models which also included non-linear classification techniques which was an improvement over the existing linear and separable boundaries. Existing models only used supervised learning methodologies, we also tried unsupervised learning techniques like KNN to better understand how the classification works.

5.3 Implication of the result

We started this project to identify the best position of any player based on historical data. We used multiple models to predict the position of a player and evaluated the best model based on prediction accuracy and the test error rate.

It was important to identify the best model in terms of the accuracy as it would solve the headache for the managers when it comes to choosing the position for a player. We can use this model as a basis for further research in this area as well as expand it to include even more levels of positions or suggest the position for each player based on the expected formation of the opposition.

6. Conclusion

From the results of accuracy and area under the curve, it is observed that logistic regression has maximum mean test accuracy of 88.57%. It is closely followed by Support Vector Machine (Radial) with mean test accuracy of 88.54%. This is closely followed by Support Vector Classifier (88.26%) and Support Vector Polynomial (88.14%). Other models like LDA, KNN (for K=10), and Random Forest did a good job.

On the same lines, area under the curve measures is a key factor to determine the robustness w.r.t performance of a machine learning model. From our empirical findings in above table, it seems that Logistic Regression leads the pack with highest accuracy. However, area under the curve is more for SVM (Radial) with comparable accuracy. We have data with non-separable and non-linear class boundaries. So, in this case, SVM (Radial) can be come quite handy. In case where, clear separable boundary exists, even Logistic Regression can be used as the go to model to classify the players in terms of Defense and Attack position.

7. Future Scope

Currently, we classify the players into two positions- attack and defense. In the future we could extend this to add more positions to have a better understanding of a player's skillset. Furthermore, based on the position of the player, we could predict the transfer fee that should be assigned for a particular player. This would particularly help the football (soccer) clubs in making better investment choices.

If we have a multi-year data or multi-decadal data, this can be used to classify the player position more accurately in multiple positions. This can be used in Sports Analytics for player bidding (in different leagues), real-time dynamic positioning of players in FIFA game. In the world of VR-based multi-player gaming experience, these findings can come handy to marks player positions on the field based on user movements supported by a robust and scalable model

8. References:

1. James, G., Witten, D., Hastie, T., & Tibshirani, R. (n.d.). An Introduction to Statistical Learning with Applications in R. Retrieved from https://www-bcf.usc.edu/~gareth/ISL/ISLR Seventh Printing.pdf

2. Smola, A. J., & Scholkopf, B. (2003, September 30). A Tutorial on Support Vector Regression. Retrieved December 7, 2018, from https://alex.smola.org/papers/2003/SmoSch03b.pdf

3. Tahamtan, S. (n.d.). Player Wage Prediction. Retrieved November 15, 2018, from https://www.kaggle.com/stahamtan/player-wage-prediction

4. The Pennsylvania State University, (n.d.). 10.2 - Support Vector Classifier. Retrieved November 16, 2018, from https://onlinecourses.science.psu.edu/stat857/node/241/

5. DLao. (n.d.). FIFA 18 - Predict player's positions. Retrieved November 20, 2018, from http://www.citationmachine.net/apa/cite-a-website

9. Appendix:

```
install.packages("corrplot")
install.packages("pROC")
install.packages("MASS")
install.packages("glm2")
install.packages("class")
install.packages("e1071")
install.packages("randomForest")
# read Input
MyData <- read.csv(file="C:\\Users\\nayan\\Downloads\\fifa-18-more-complete-player-
dataset\\complete.csv", header=TRUE, sep=",")

#print the top 5 observations
head(MyData, 5)

#print all column names
colnames(MyData)

colsToKeep <-
c("special","age","height_cm","weight_kg","overall","potential","pac","sho","pas","dri","def","phy",

"international_reputation","skill_moves","weak_foot","crossing","finishing","heading_accuracy","shor
t_passing",

"volleys","dribbling","curve","free_kick_accuracy","long_passing","ball_control","acceleration","sprint
_speed",

"agility","reactions","balance","shot_power","jumping","stamina","strength","long_shots","aggression
",
"interceptions","positioning","vision","penalties","composure","marking","standing_tackle","sliding_ta
ckle",

"gk_diving","gk_handling","gk_kicking","gk_positioning","gk_reflexes","rs","rw","rf","ram","rcm","rm",
"rdm","rcb",

"rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk",

"prefers_rs","prefers_rw","prefers_rf","prefers_ram","prefers_rcm","prefers_rm","prefers_rdm","pref
ers_rcb",

"prefers_rb","prefers_rwb","prefers_st","prefers_lw","prefers_cf","prefers_cam","prefers_cm","prefer
s_lm",

"prefers_cdm","prefers_cb","prefers_lb","prefers_lwb","prefers_ls","prefers_lf","prefers_lam","prefer
s_lcm",
            "prefers_ldm","prefers_lcb","prefers_gk")

#columns to keep in data
MyData<-MyData[colsToKeep]

#create new column for preferred position
MyData$Pref_Position <- 'NA'
MyData$Final_Position <- 'NA'

colnames(MyData)

#New data set with cleaned data
cleanedData = data.frame();

#set of all possible positions
position =
c("rs","rw","rf","ram","rcm","rm","rdm","rcb","rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb
","lwb","ls",
          "lf","lam","lcm","ldm","lcb","gk")

booleanPosition = c(1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0)


#Creates a new dataset with specifying preferred position. In case of multiple preferred positions,
#duplicate rows of data is created with unique preferred positions for each row for each player

for( i in 1:length(MyData[,1])){
  for(j in 77:103){
    if(MyData[i,j] == 'True'){
      MyData[i,]$Pref_Position = position[j-76]
      MyData[i,]$Final_Position = booleanPosition[j-76]
      cleanedData = rbind(cleanedData, MyData[i,])
    }
  }
}


removePrefers <-
c("prefers_rs","prefers_rw","prefers_rf","prefers_ram","prefers_rcm","prefers_rm","prefers_rdm","pref
ers_rcb",

"prefers_rb","prefers_rwb","prefers_st","prefers_lw","prefers_cf","prefers_cam","prefers_cm","prefers
_lm",

"prefers_cdm","prefers_cb","prefers_lb","prefers_lwb","prefers_ls","prefers_lf","prefers_lam","prefers
_lcm",
               "prefers_ldm","prefers_lcb","prefers_gk")


withoutPrefers <- cleanedData[,!colnames(cleanedData) %in% removePrefers]
withoutPrefers[is.na(withoutPrefers)] <- 0
withoutPrefers$Final_Position <- as.numeric(withoutPrefers$Final_Position)

# Split training and test data
train<-sample(nrow(withoutPrefers),nrow(withoutPrefers)/2)
wp.train<- withoutPrefers[train,]
wp.test<-withoutPrefers[-train,]

#plots to check collinearity
removeCharCols<- c("Pref_Position","special", "age", "height_cm","weight_kg",
"gk_diving","gk_handling","gk_kicking","gk_positioning","gk_reflexes","rs","rw","rf","ram","rcm","rm"
,"rdm","rcb",
"rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk")
str(withoutPrefers)
correlation<-cor(withoutPrefers[,!colnames(withoutPrefers) %in% removeCharCols],
method="pearson")
library(corrplot)
corrplot.mixed(correlation,tl.pos = 'lt')

collinear<-
c("gk_diving","gk_handling","gk_kicking","gk_positioning","gk_reflexes","rs","rw","rf","ram","rcm","r
m","rdm","rcb",
"rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk")
corrplot.mixed(cor(withoutPrefers[,collinear]),tl.pos = 'lt')


#apply PCA
set.seed(1)
removePCAcols <- c("special","age","height_cm","weight_kg","Pref_Position","Final_Position")
PCAdata <- withoutPrefers[,!colnames(withoutPrefers) %in% removePCAcols]
apply(PCAdata, 2, mean)
apply(PCAdata, 2, var)
pr.out=prcomp(PCAdata, scale=TRUE)
names(pr.out)
pr.out$center
pr.out$scale
pr.out$rotation
dim(PCAdata)
dim(pr.out$x)
biplot(pr.out, scale=0)
pr.out$rotation=- pr.out$rotation
pr.out$x=-pr.out$x
biplot(pr.out, scale=0)



#apply LDA
library(MASS)
set.seed(1)
lda.fit<-lda(Final_Position ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position -
overall
        -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb
        - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk,
        data=wp.train)
summary(lda.fit)
lda.pred = predict(lda.fit, wp.test)
table(lda.pred$class, wp.test$Final_Position)
mean(lda.pred$class==wp.test$Final_Position)
plot(lda.fit, cex = 0.7, pch = 20,type="both")

#ROC Curve
library(pROC)
rocnn<- roc(lda.pred$class,wp.test$Final_Position)
plot(rocnn,col="red")
auc(rocnn)
```

```
#apply Logistic regression
set.seed(1)
library(glm2)
set.seed(1)
wp.logit<-glm(Final_Position ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position -
overall
            -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb
            - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk,
             data= wp.train, family="binomial")
summary(wp.logit)
wp.probs<-predict(wp.logit, wp.test, type="response")
wp.pred<-rep(0,nrow(wp.test))
wp.pred[wp.probs>0.5]=1
table(wp.pred, wp.test$Final_Position)
mean(wp.pred==wp.test$Final_Position)
plot(wp.probs)

#ROC Curve
rocnn<- roc(wp.pred,wp.test$Final_Position)
plot(rocnn,col="red")
auc(rocnn)

#apply KNN
unusedCols<- c("special","age","height_cm","weight_kg","Pref_Position","Final_Position", "overall",

"gk_diving","gk_handling","gk_kicking","gk_positioning","gk_reflexes","rs","rw","rf","ram","rcm","rm"
,"rdm","rcb",

"rb","rwb","st","lw","cf","cam","cm","lm","cdm","cb","lb","lwb","ls","lf","lam","lcm","ldm","lcb","gk")
wp.train.predictors <- wp.train[,!colnames(wp.train) %in% unusedCols]
wp.test.predictors<-  wp.test[,!colnames(wp.test) %in% unusedCols]
wp.train.response<- wp.train[,"Final_Position"]
set.seed(1)
library(class)
knn.pred=knn(wp.train.predictors, wp.test.predictors, wp.train.response, k=10)
summary(knn.pred)
table(knn.pred, wp.test$Final_Position)
mean(knn.pred==wp.test$Final_Position)

rocnn<- roc(knn.pred,wp.test$Final_Position)
plot(rocnn,col="red")
auc(rocnn)
```

```
#apply QDA
#QDA
set.seed(1)
library(MASS)
qda.fit<-qda(Final_Position ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position -
overall
          -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb
          - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk ,data=wp.train,
na.rm=TRUE)
summary(qda.fit)
qda.class = predict(qda.fit, wp.test, na.rm = TRUE)$class
table(qda.class, wp.test$Final_Position)
mean(qda.class ==wp.test$Final_Position)

rocnn<- roc(qda.class,wp.test$Final_Position)
plot(rocnn,col="red")
auc(rocnn)


#apply SVC
set.seed(1)
wp.train$factors<- as.factor(wp.train$Final_Position)
wp.test$factors<- as.factor(wp.test$Final_Position)
library(e1071)

wp.svmfit <- svm(factors ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position -
overall
            -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb
            - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk ,
            data= wp.train, kernel="linear", cost=1 )
summary(wp.svmfit)
train.predict<- predict(wp.svmfit, wp.train,  type="class")
table(predict=train.predict, truth=wp.train$factors)

test.predict<- predict(wp.svmfit, wp.test,  type="class")
table(predict=test.predict, truth=wp.test$factors)
mean(test.predict ==wp.test$factors)

library(pROC)
```

```
rocnn<- roc(as.numeric(test.predict),as.numeric(wp.test$factors))
plot(rocnn,col="red")
auc(rocnn)



#apply SVM radial
set.seed(1)
wp.train$factors<- as.factor(wp.train$Final_Position)
wp.test$factors<- as.factor(wp.test$Final_Position)
library(ISLR)
library(e1071)
wp.svmfit <- svm(factors ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position -
overall
        -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb
        - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk,
        data= wp.train, kernel="radial", cost=1 )
summary(wp.svmfit)
train.predict<- predict(wp.svmfit, wp.train,  type="class")
table(predict=train.predict, truth=wp.train$factors)

test.predict<- predict(wp.svmfit, wp.test,  type="class")
table(predict=test.predict, truth=wp.test$factors)
mean(test.predict ==wp.test$factors)

library(pROC)
rocnn<- roc(as.numeric(test.predict),as.numeric(wp.test$factors))
plot(rocnn,col="red")
auc(rocnn)



#apply SVM polynomial
set.seed(1)
wp.train$factors<- as.factor(wp.train$Final_Position)
wp.test$factors<- as.factor(wp.test$Final_Position)
library(ISLR)
library(e1071)
wp.svmfit <- svm(factors ~ . - special - age - height_cm - weight_kg - Pref_Position - Final_Position -
overall
        -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -rdm -rcb
        - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk,
        data= wp.train, kernel="polynomial", degree=2 ,cost=1 )
summary(wp.svmfit)
```

```
train.predict<- predict(wp.svmfit, wp.train,  type="class")
table(predict=train.predict, truth=wp.train$factors)

test.predict<- predict(wp.svmfit, wp.test,  type="class")
table(predict=test.predict, truth=wp.test$factors)
mean(test.predict ==wp.test$factors)

rocnn<- roc(as.numeric(test.predict),as.numeric(wp.test$factors))
plot(rocnn,col="red")
auc(rocnn)



#apply Random Forests
set.seed(1)
wp.train$factors<- as.factor(wp.train$Final_Position)
wp.test$factors<- as.factor(wp.test$Final_Position)
library(randomForest)
wp.ranforrest= randomForest(factors ~. - special - age - height_cm - weight_kg - Pref_Position -
Final_Position -overall
                    -gk_diving -gk_handling -gk_kicking -gk_positioning -gk_reflexes -rs -rw -rf -ram -rm -
rdm -rcb
                    - rb -rwb -st -lw -cf -cam -cm -lm- cdm -cb -lb -lwb -ls -lf -lam -lcm -ldm -lcb -gk,
                    data= wp.train, mtry= 7, ntree= 250, importance=TRUE)
summary(wp.ranforrest)
wp.rfpredict<- predict(wp.ranforrest, wp.test, type="class")
importance(wp.ranforrest)
table(predict=wp.rfpredict, truth=wp.test$factors)
mean(wp.rfpredict ==wp.test$factors)

rocnn<- roc(as.numeric(test.predict),as.numeric(wp.test$factors))
plot(rocnn,col="red")
auc(rocnn)
```