FIFA 2022 Soccer Position Predictor

Bhavana Chikkamuduvadi Renuka Gowda
Sumanth Herohalli Umashankar
Karthik Shingte

Abstract

The aim of this project is to analyze Soccer Players' data from 2022 FIFA dataset and gain insights from the data with respect to player positions. Initially, we shall explore the dataset to understand the structure of data by employing various analysis techniques. The bulk of this project will be in implementing KNN and LDA technique on subsets of the dataset based on the player positions and evaluating each model. Finally, we will summarize the results of each model.

Description of Dataset

This dataset was directly obtained from Kaggle and contains very detailed attributes for every player in the latest edition of the FIFA 22 database. The dataset contains many of the player's attributes, ranging from their age to their ball control and ranking in the game. Most of the skill set columns are on a ranking from 1 to 100. Since this dataset contains many columns of information pertaining to the ranking of the players for a particular skill, the following questions arise: Can we predict a player's general position (Forward, Goalkeeper etc.) based on the player's characteristics and ranking of skills?

The dataset contains 65 attributes and 19,239 observations. Attributes include details about the proficiency of each players' skill levels in different types of capabilities.

Data Exploration

To begin with we find a summary of the data. We find that there are 56 numeric variables and 9 categorical variables in the dataset. We also added the screenshot of more data related summaries to get an idea of the data we are using.

```
> length(select_if(soccer1, is.numeric))
[1] 56
> length(select_if(soccer1, is.character))
[1] 9
>
```

```
$ age                     : num [1:19239] 34 32 36 29 30 28 22 35 29 27 ...
$ height_cm               : num [1:19239] 170 185 187 175 181 188 182 193 187 188 ...
$ weight_kg               : num [1:19239] 72 81 83 68 70 87 73 93 85 89 ...
$ club_team_id            : num [1:19239] 73 21 11 73 10 240 73 21 241 18 ...
$ club_name               : chr [1:19239] "Paris Saint-Germain" "FC Bayern München" "Manchester United" "Paris Saint-Ge
main" ...
$ league_name             : chr [1:19239] "French Ligue 1" "German 1. Bundesliga" "English Premier League" "French Ligu
e 1" ...
$ club_position           : chr [1:19239] "RW" "ST" "ST" "LW" ...
$ club_jersey_number      : num [1:19239] 30 9 7 10 17 13 7 1 1 10 ...
$ nationality_name        : chr [1:19239] "Argentina" "Poland" "Portugal" "Brazil" ...
$ nation_position         : chr [1:19239] "RW" "RS" "ST" NA ...
$ preferred_foot          : chr [1:19239] "Left" "Right" "Right" "Right" ...
$ weak_foot               : num [1:19239] 4 4 4 5 5 3 4 4 4 5 ...
$ skill_moves             : num [1:19239] 4 4 5 5 4 1 5 1 1 3 ...
$ international_reputation: num [1:19239] 5 5 5 5 4 5 4 5 4 4 ...
$ release_clause_eur      : num [1:19239] 1.44e+08 1.97e+08 8.33e+07 2.39e+08 2.32e+08 ...
$ pace                    : num [1:19239] 85 78 87 91 76 NA 97 NA NA 70 ...
$ shooting                : num [1:19239] 92 92 94 83 86 NA 88 NA NA 91 ...
$ passing                 : num [1:19239] 91 79 80 86 93 NA 80 NA NA 83 ...
$ dribbling               : num [1:19239] 95 86 88 94 88 NA 92 NA NA 83 ...
$ defending               : num [1:19239] 34 44 34 37 64 NA 36 NA NA 47 ...
$ physic                  : num [1:19239] 65 82 75 63 78 NA 77 NA NA 83 ...
$ attacking_crossing      : num [1:19239] 85 71 87 85 94 13 78 15 18 80 ...
$ attacking_finishing     : num [1:19239] 95 95 95 83 82 11 93 13 14 94 ...
```

```
    shooting        passing        dribbling       defending        physic      attacking_crossing attacking_finishing
 Min.   :18.00   Min.   :25.00   Min.   :27.00   Min.   :14.0   Min.   :29.00   Min.   : 6.00      Min.   : 2.00
 1st Qu.:42.00   1st Qu.:51.00   1st Qu.:57.00   1st Qu.:37.0   1st Qu.:59.00   1st Qu.:38.00      1st Qu.:30.00
 Median :54.00   Median :58.00   Median :64.00   Median :56.0   Median :66.00   Median :54.00      Median :50.00
 Mean   :52.35   Mean   :57.31   Mean   :62.56   Mean   :51.7   Mean   :64.82   Mean   :49.58      Mean   :45.89
 3rd Qu.:63.00   3rd Qu.:64.00   3rd Qu.:69.00   3rd Qu.:64.0   3rd Qu.:72.00   3rd Qu.:63.00      3rd Qu.:62.00
 Max.   :94.00   Max.   :93.00   Max.   :95.00   Max.   :91.0   Max.   :90.00   Max.   :94.00      Max.   :95.00
 NA's   :2132    NA's   :2132    NA's   :2132    NA's   :2132   NA's   :2132
 attacking_heading_accuracy attacking_short_passing attacking_volleys skill_dribbling  skill_curve
 Min.   : 5.00              Min.   : 7.00           Min.   : 3.00     Min.   : 4.00   Min.   : 6.00
 1st Qu.:44.00              1st Qu.:54.00           1st Qu.:30.00     1st Qu.:50.00   1st Qu.:35.00
 Median :55.00              Median :62.00           Median :43.00     Median :61.00   Median :49.00
 Mean   :51.78              Mean   :58.87           Mean   :42.46     Mean   :55.66   Mean   :47.27
 3rd Qu.:64.00              3rd Qu.:68.00           3rd Qu.:56.00     3rd Qu.:68.00   3rd Qu.:61.00
 Max.   :93.00              Max.   :94.00           Max.   :90.00     Max.   :96.00   Max.   :94.00
```

Now let's look at the stats of the column player positions as we will be making prediction based on this category:

```
> table(soccer1$player_positions)

 CAM   CB  CDM   CF   CM   GK   LB   LM   LW  LWB   RB   RM   RW  RWB   ST
1151 3339 1665  142 2173 2132 1360 1016  435  171 1346 1028  495  178 2608
>
```

We can see that there are a lot of divisions within major positions, which we will group in 4 categories i.e., FWD, MID, DEF & GK.

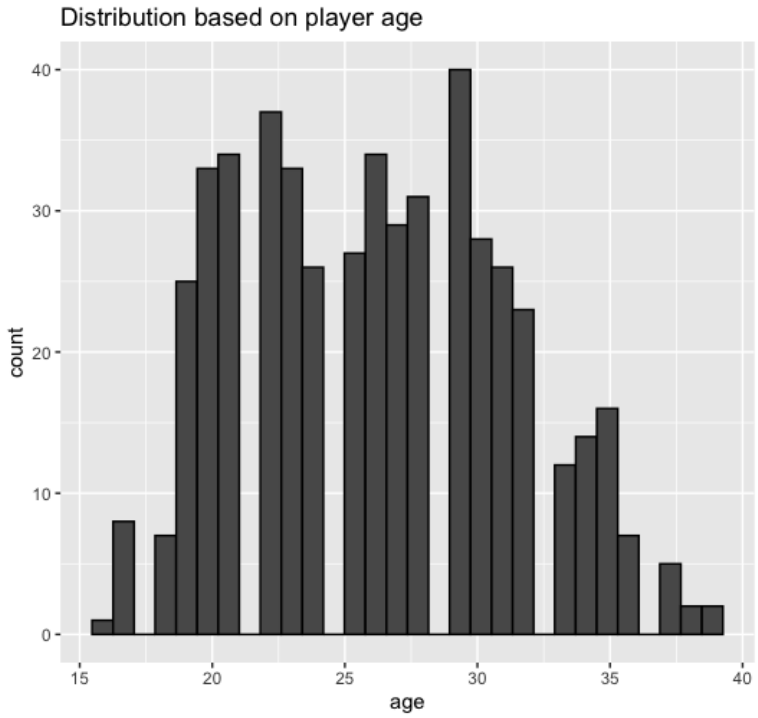
```
soccer.new<-soccer1%>%
  mutate(gen.position= ifelse(player_positions == "GK",
                              "GK",
                              ifelse(player_positions == "RW"|player_positions=="RF"|player_positions ==
"LF"|player_positions == "CF"|player_positions== "ST"|player_positions=="LS"|player_positions=="RS"|player_positions
=="LW",
                                     "FWD",
                                     ifelse(player_positions=="LM"|player_positions=="RCM"|player_positions
=="CM"|player_positions=="CAM"|player_positions=="CDM"|player_positions=="RAM"|player_positions
=="LAM"|player_positions=="RM"|player_positions=="LCM"|player_positions=="RDM"|player_positions=="LDM",
                                            "MID",
                                            ifelse( player_positions=="LWB"|player_positions=="LB"|player_positions
=="LCB"|player_positions=="CB"|player_positions=="RCB"|player_positions=="RB"|player_positions=="RWB",
                                                    "DEF",
                                                    "none"))
                              )))

table(soccer.new$gen.position)
```

```
> table(soccer.new$gen.position)

 DEF  FWD   GK  MID
6394 3680 2132 7033
>
```

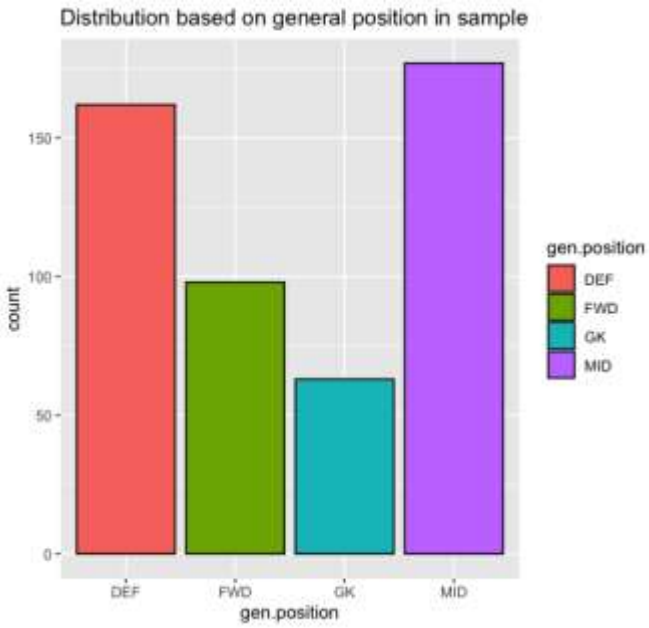
Additionally, we find that in the columns rating skills we find unimportant skills to carry value NA which we have normalized to 10. We have replaced NA value in columns 25 to 65.

For our analysis we will be taking 500 rows from the dataset, 100 from the top and rest 400 from the complete data randomly sampled.

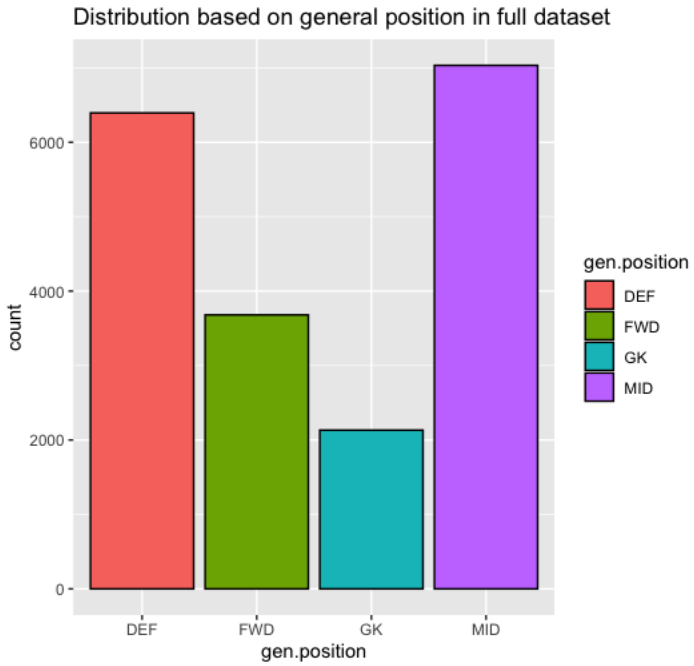
Plotting Age distribution of players in our 500-sample set.



Distribution based on general position in sample



Distribution based on general position in full dataset



We can see that our sampled set is similar to the original data w.r.t. the distribution of players in general position.

KNN Model

The KNN (K-nearest neighbor) model is a simple and effective classification and regression algorithm that is commonly used in machine learning. It is based on the idea of identifying the K nearest neighbors of an input data point, and then using those neighbors to make predictions or classifications.

For example, if you have a dataset of houses and their prices, you could use KNN to predict the price of a new house based on its features (such as square footage, number of bedrooms, etc.). To do this, the algorithm would identify the K houses in the dataset that are most similar to the new house, and then use those houses to predict the price of the new house.

One of the key advantages of KNN is that it is simple to implement and understand, and it can be applied to a wide range of problems. However, it can

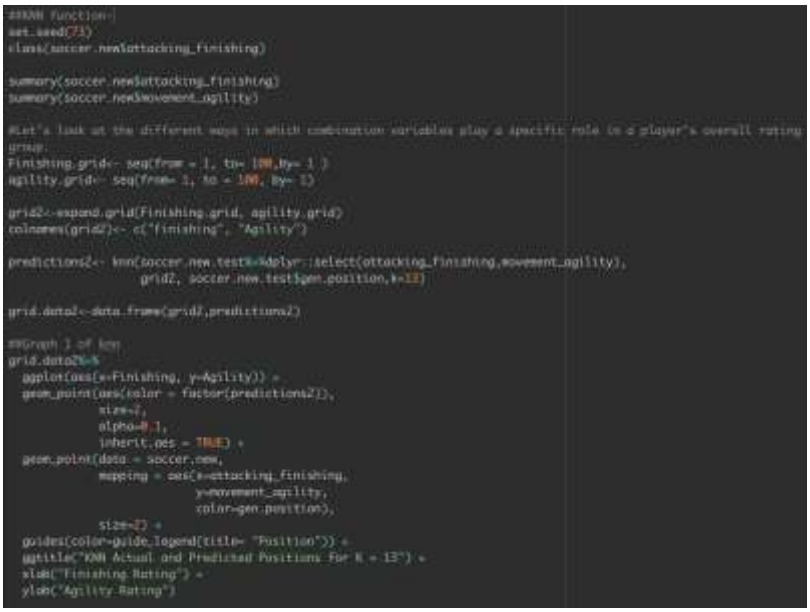
also be computationally intensive, particularly for large datasets, and it can be sensitive to the choice of K and other parameters.

Analysis

Now, we will split the sample into training and testing datasets with a split ration of 80:20. Then we will train a KNN model on the training dataset and make predictions using the testing dataset. Based on our knowledge of the domain we will use only attributes that we know are the most important to judge a player's position.

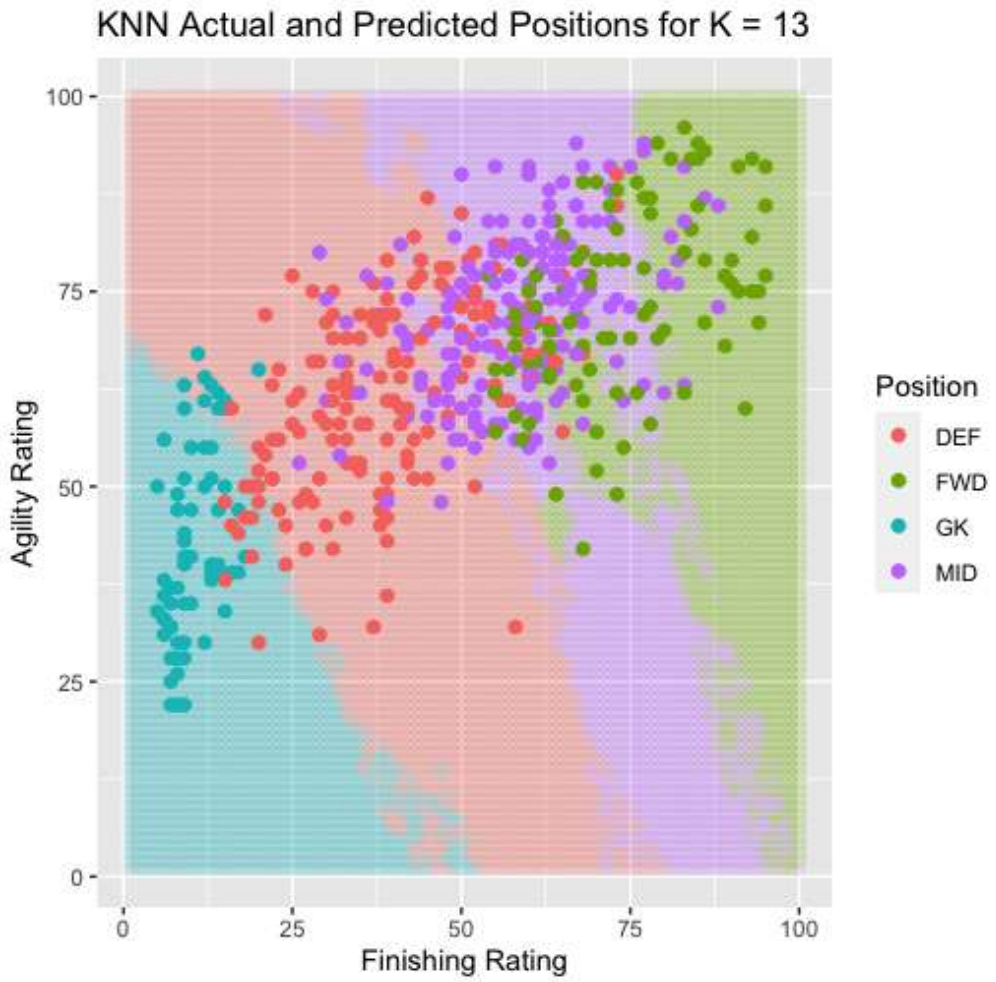
```
# split training and testing data
seed <- 88
set.seed(seed)
split = sample.split(soccer.new$gen.position, SplitRatio = 0.8)
split

soccer.new.train = subset(soccer.new, split == TRUE)
soccer.new.test = subset(soccer.new, split == FALSE)

## KNN with caret
trainControl <- trainControl(classProbs = TRUE)

knn.caret.pos<-train(factor(gen.position) ~ age+ overall+ attacking_finishing+ dribbling+
                       attacking_crossing+attacking_heading_accuracy+attacking_short_passing+
                       attacking_volleys+skill_curve+
                       skill_fk_accuracy+skill_long_passing+skill_ball_control+pace+movement_agility+
                       movement_reactions+movement_balance+power_shot_power+physic+
                       power_long_shots+ mentality_interceptions+
                       mentality_positioning+mentality_vision+ mentality_penalties+
                       mentality_composure+defending_marking_awareness+ defending_standing_tackle+
                       defending_sliding_tackle+ goalkeeping_handling,
                     data = soccer.new.train,
                     method="knn",
                     tuneLength = 12,
                     na.action=na.exclude)
```

The predictions are then built using the test data and two attributes – "Agility" & "Finishing"



```
###KNN Function#
set.seed(73)
class(soccer.new$attacking_finishing)

summary(soccer.new$attacking_finishing)
summary(soccer.new$movement_agility)

#Let's look at the different ways in which combination variables play a specific role in a player's overall rating
group.
Finishing.grid<- seq(from = 1, to= 100,by= 1 )
agility.grid<- seq(from= 1, to = 100, by= 1)

grid2<-expand.grid(Finishing.grid, agility.grid)
colnames(grid2)<- c("finishing", "Agility")

predictions2<- knn(soccer.new.test%>%dplyr::select(ottacking_finishing,movement_agility),
                   grid2, soccer.new.test$gen.position,k=13)

grid.data2<-data.frame(grid2,predictions2)

##Graph 1 of knn
grid.data2%>%
  ggplot(aes(x=Finishing, y=Agility)) +
  geom_point(aes(color = factor(predictions2)),
             size=2,
             alpha=0.1,
             inherit.aes = TRUE) +
  geom_point(data = soccer.new,
             mapping = aes(x=attacking_finishing,
                           y=movement_agility,
                           color=gen.position),
             size=2) +
  guides(color=guide_legend(title= "Position")) +
  ggtitle("KNN Actual and Predicted Positions for K = 13") +
  xlab("Finishing Rating") +
  ylab("Agility Rating")
```

Our KNN cluster distribution and confusion matrix results are as follows.



Confusion matrix results: The accuracy of the KNN model is 80.26%

```
> confusionMatrix(knn.caret.pos, mode = "everything")
Bootstrapped (25 reps) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction  DEF  FWD   GK  MID
       DEF 27.2  0.1  0.0  6.1
       FWD  0.0 15.3  0.0  4.7
       GK   0.0  0.0 12.8  0.0
       MID  5.2  3.6  0.0 24.9

 Accuracy (average) : 0.8026
```

LDA Model

The LDA (Linear Discriminant Analysis) model is a supervised machine learning algorithm that is commonly used for classification tasks. It is a linear model, meaning that it makes predictions based on a linear combination of the input features.

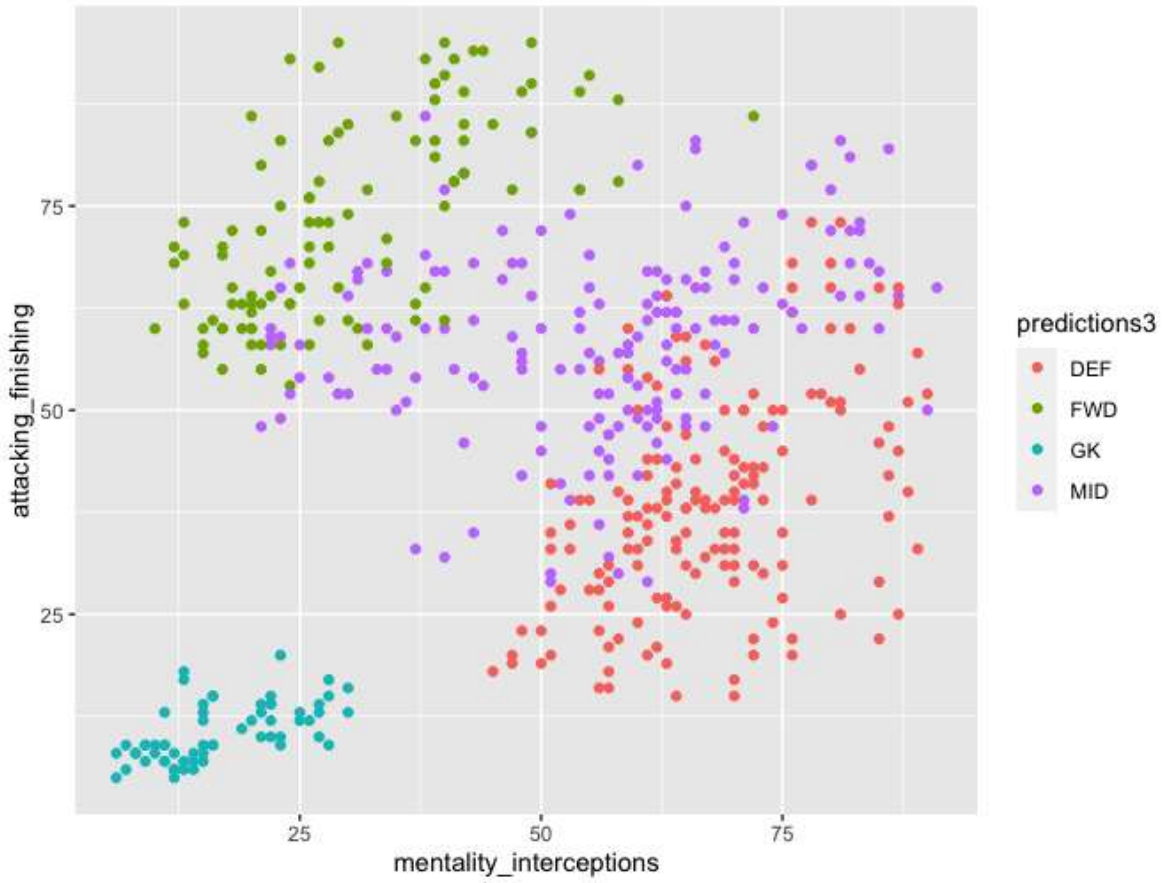
LDA is a generative model, which means that it models the distribution of the data and then uses that model to make predictions. It assumes that the data comes from a Gaussian distribution and that the classes are normally distributed with equal covariance matrices.

LDA works by finding the linear combination of the input features that best separates the classes. It does this by maximizing the ratio of the between-class variance to the within-class variance. This results in a hyperplane that can be used to make predictions about the class of new data points.

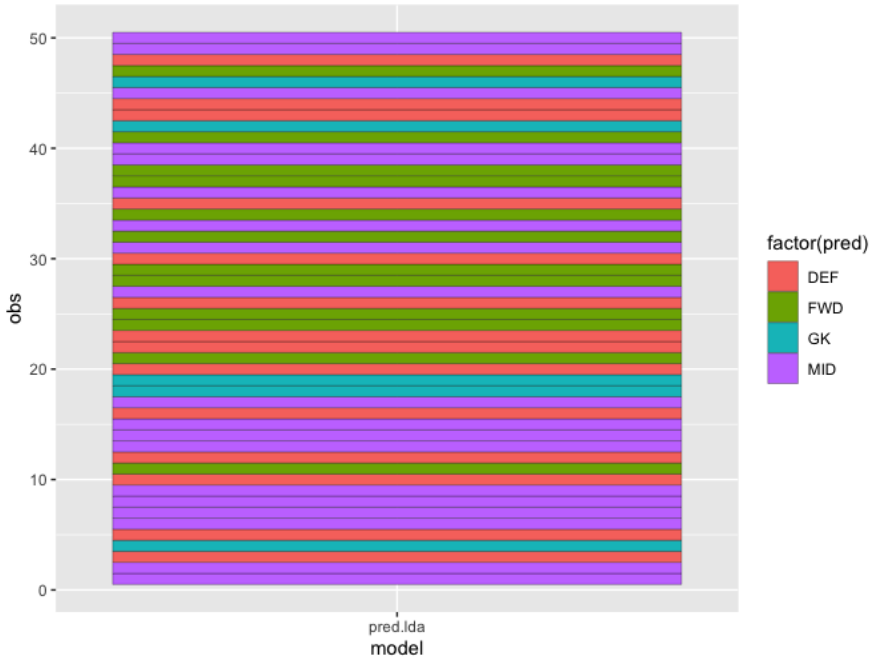
One of the key advantages of LDA is that it is computationally efficient and can be easily implemented. It is also a popular choice for dimensionality reduction, as it can reduce the number of input features while maintaining good classification performance. However, it can be sensitive to outliers and may not perform well if the assumptions about the data are not met.

We now train LDA model with a training data sample set. Here the attributes considered were “Finishing” vs “interceptions”. The results were found to be as follows:

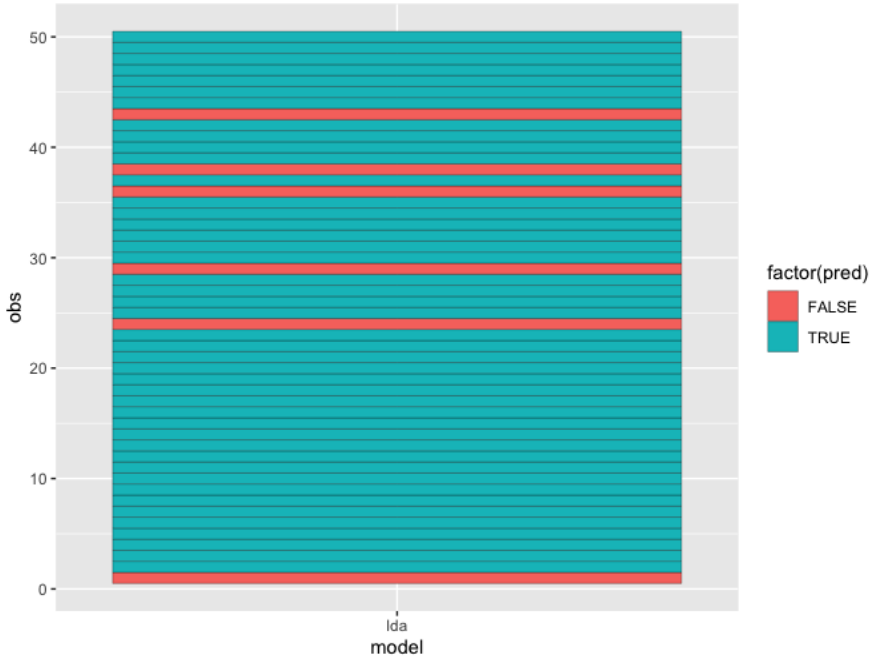
```
# LDA function
lda1.pos <- lda(gen.position~ age+ overall+ attacking_finishing+ dribbling+
                attacking_crossing+attacking_heading_accuracy+attacking_short_passing+
                attacking_volleys+skill_curve+
                skill_fk_accuracy+skill_long_passing+skill_ball_control+pace+movement_agility+
                movement_reactions+movement_balance+power_shot_power+physic+
                power_long_shots+ mentality_interceptions+
                mentality_positioning+mentality_vision+ mentality_penalties+
                mentality_composure+defending_marking_awareness+ defending_standing_tackle+
                defending_sliding_tackle+ goalkeeping_handling,
                data=soccer.new)


#Make some predictions and visualize their classification
predictions3 <- predict(lda1.pos)$class

new.default <- data.frame(soccer.new, predictions3)

new.default %>%
  ggplot(aes(x = mentality_interceptions,
             y = attacking_finishing)) +
  geom_point(aes(color = predictions3))
```



We will be splitting the data for training and testing in the ration of 80:20. Then we will run a prediction using our test data to represent our factors predictions.



Now that we have made our predictions let's check the accuracy of the model on this actual sample.



We have got a good prediction from our LDA model. Let's look at the confusion matrix and kappa statistics. We find the accuracy of the trained model to be 85.41%

```
Resampling results:

  Accuracy   Kappa
  0.8541033  0.793725

> confusionMatrix(lda.caret, mode = "everything")
Bootstrapped (25 reps) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction  DEF  FWD   GK  MID
      DEF 29.6  0.0  0.0  2.2
      FWD  0.0 15.1  0.0  4.6
      GK   0.0  0.0 11.3  0.0
      MID  3.2  4.5  0.0 29.4

 Accuracy (average) : 0.8541
```

Conclusion

For each of the model types we have compared the accuracy and predicted the player positions. LDA seems to have slightly higher accuracy than KNN for this data. But both models gave very good results.