Classification of FIFA Player Positions

Erik Elias Mikael Maresia
963724

January 2022

Abstract

The purpose of this paper is to understand what player attributes might suggest a football player to have a certain position on the field. This paper uses three tree-based learning methods to try to classify players based on player attributes. The result of this study is that boosting gives the most accurate results when classifying players.

1. Introduction

FIFA (Fédération Internationale de Football Association) is an international non-profit organisation that is the governing body of football. FIFA Football is a video game series developed by Electronic Arts. The video game simulates real football teams and players, by portraying realistic information about the teams and players.

Every player has a position on the field, which can be one of many. Player positions can be specific or generic. A generic position is one of four: goalkeeper, defender, midfielder or attacker. These generic positions can be further divided into more specific ones - a centre defender, right wing attacker, attacking midfielder etc.

Player objects are divided into multiple individual attributes, which make up a player. In this paper these attributes are synonymous with player skills. Examples of these attributes are acceleration, shot power, vision and player strength.

2. Data

The data analysed in this paper is player data from the FIFA Football video game of 2018. There are 75 variables in the dataset, but not all are of interest, including the player's age, nationality, name and the team it plays for.

Players have a variable called *Preferred.Positions*, which is a list of specific positions the player prefers to play in. The dataset has 26 unique specific player positions, so to simplify the classification problem, only generic positions are used. The specific positions must be mapped to a generic positions, since it does not exist in the dataset. Figure 1 shows that there are roughly the same amount of observations per position.

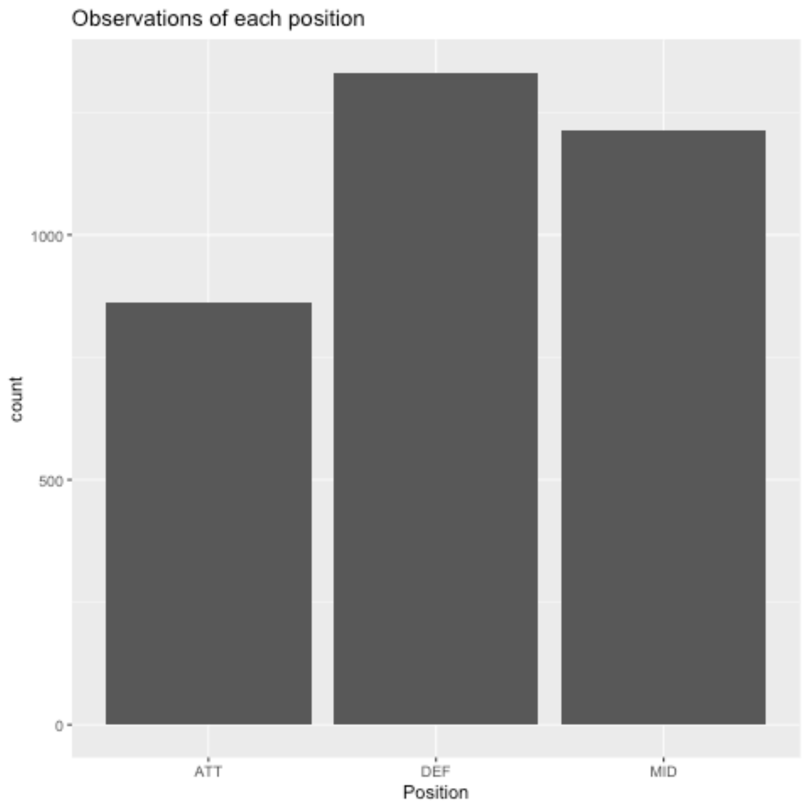


Figure 1

This paper is more interested in player attributes, which are classified as players in the field (i.e. not goalkeepers), so all goalkeepers are dropped from the dataset together with the direct attributes related to them.

Some positions are very similar to one another and a player playing in one of these *marginal* positions could (and sometimes does) play in a different position. An example of this would be an attacking midfielder. These marginal positions are disregarded.

Only measurable attributes are of interest, which might constitute a player's position. There are 29 of these attributes. These attributes are numerical variables that range from 0 to 100, with 0 being the worst and 100 the best. When thinking about football players, some skills should be able to reflect the position they play in. There are some skills that *make up* a defender, midfielder or attacker. This is somewhat evident when looking at the averages of players of position defence, midfield and attack (Figures 11-13).

Players have an attribute *Overall*, which is a type of weighted average of all the measurable attributes. To clean out noise from the data, only players who have an Overall of over 70 are taken into account. Since the dataset is ordered by the best overall players, the data is shuffled to have a random ordering.

3. Theory

This section briefly explains the core theory behind the statistical learning methods used in this analysis. This paper uses tree based learning methods for classification.

Statistical theory suggest that there is no one learning method, which is superior to all others. This is commonly known as the *no free lunch* theorem. A tree based learning method is used because EXPLAIN WHY. In the next sections the theory behind tree based learning methods is explained, together with some applications of them, namely random forests and boosting.

In the last section of this chapter, a technique called cross validation is explained.

3.1 Tree-Based Learning

Tree-based methods involves dividing the feature space into segments by different means. The rules for the segmentation can be shown as a tree diagram and thus is easy to interpret.

3.1.1 Decisions Trees

Decision trees are the most basic type of a tree-based learning method. It will segment the data into different regions of predictor space, which are known as *terminal nodes* or *leaves* of the tree. Although often oversimplifying the model, decision trees have one great advantage: they are very easy to interpret.

A tree is constructed via binary splitting. It starts with all observations belonging to the same region, splitting the predictor space into two new branches recursively. In the classification setting, there are two steps in building a decision tree:

1. Divide the predicted space in to J possible regions, $R_1, R_2, \dots, R_J$
2. For every observation that falls into a region R_j, the most common region is predicted.

Often the above will overfit the data, because the tree results too complex.

In order for the binary splitting to be done properly, there needs to be a criterion by which we the successfulness of the splits. The is known as the classification error rate. One of such error rates is called the *Gini index* and is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the kth class. It measures the total variance across the K classes. The Gini index is also referred to as the node purity. If a region predominantly contains observations from one class, the Gini index will have a small value.

3.1.2 Bagging and Random Forests

Decision trees are easy to interpret, but they suffer from high variance. This means that when modelled on some training data with a low error rate, the testing error rate will be high.

A way to reduce variance of a decision tree is through a technique called bootstrap aggregation or *bagging*. Bootstrapping is a resampling method to create many simulated samples of a larger dataset. Since averaging a set of observations reduces variance, bagging will build B separate prediction models $\hat{f}^{*b}(x)$ on the bth bootstrapped training set, and average the predictions. This is given by

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

In the classification setting, averaging will not work with qualitative variables - a majority vote will be taken instead.

While bagging usually improves accuracy, it looses interpretability over decision trees. This is because it is not possible to show the bagged tree as a diagram. One benefit of decision trees and their diagrams was to understand what variables were important. In the classification setting, bagging provides the importance of each variable in the order that they decrease the Gini index.

Random forests are an improvement to bagging. While bagging used all possible variables when a split in a tree is considered, random forests only use m out of p variables, where typically $m \approx \sqrt{p}$. In this way, the random forrest decorrelates the trees. It is quite easy to see, that if $m = p$, random forests are essentially the same bagging.

3.1.3 Boosting

The difference between bagging and boosting, is that while bagging *grew* trees simultaneously and then averaging the predictions, boosting grows the trees sequentially. In this way, boosting will learn slowly and not overfit the data.

Boosting has three tuning parameters:

1. The number of trees B
2. The shrinkage parameter λ, which is the rate at which the model learns.
3. The number of d splits in each tree, or the *interaction depth*.

3.2 Cross-Validation

To understand if a statistical model is performing well, it is useful to look at the test and training errors of the model. It is rare that there would exists a separate test dataset, so the available dataset has to be utilised for both the training of the model and the testing of it.

A simple way of doing this, would be to separate the data into two parts - a training set and a testing set. The problem with this approach is that there is no guarantee that the training set and the testing set are similar.

A popular cross-validation (CV) method is *k-fold CV*. It randomly divides the dataset into k groups of equal size. The first group is used as the testing set, while the rest $k - 1$ groups are used to train the model. This is repeated k times, with each time a different test set.

4. Analysis

This chapter will try to create a model, which will successfully classify FIFA players to a generic player position. Three decision-based learning methods are used - a decision tree model, a random forest and boosting. Each method and subsequent model will be analysed to see if it will fit the data adequately, or if a different method will do a better job at classifying.

4.1 Decision Tree

To begin with, a decision tree is used to model the data. It can be seen from Figure 2 that the model has partitioned the data into 8 segments. Heuristically the features used for the partitioning make sense, as these are skills that seem to fit the players in these positions. For example, the skill Sliding.tackle is often associated with more defensive players, while Finishing with attacking players. Inspecting the decision tree diagram is easy and

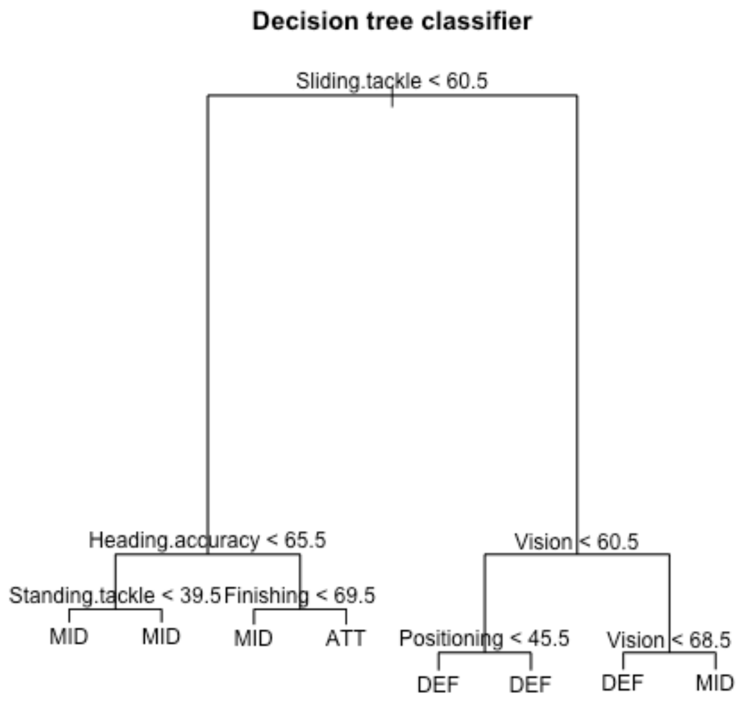


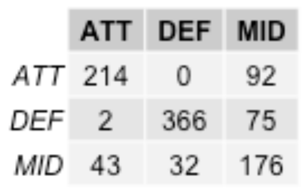
Figure 2

intuitive, but a more appropriate metric is needed to understand how the model has fitted - namely if it is under or overfitting the data.

As discussed in Chapter 3, there exists a way of measuring the error. The testing error rate can be calculated from a confusion matrix. A confusion matrix is a matrix, which shows how the model has classified the data. The confusion matrix for the model in Figure 2 can seen in Figure 3. The test error rate can be calculated by summing the true positives (diagonals) and dividing by the amount of test observations. In this case the test error results to be 0.244. In other words the model classifies data it had not seen before correctly with an accuracy of $75.6\,\%$.

The decision tree is pruned to figure out if the accuracy can be improved. Pruning a tree is the way.

The next model is going to be a random forest classifier. As mentioned in Chapter 3, a random forest classifier is an improved bagging classifier. The number of features to take into account every split, or m, has to be determined. This can be done by fitting the model many times with a



	ATT	DEF	MID
ATT	214	0	92
DEF	2	366	75
MID	43	32	176

Figure 3

different value of m and choosing the model which has the lowest error rate. The error rates can be seen from Figure 4, where the lowest error rate is given by $m = 8$.

As discussed in Chapter 3, using a random forest over a decision tree will loose interpretability, as it is no longer possible to draw a diagram of the tree, as in Figure 1. It is possible, however, to understand which variables are more

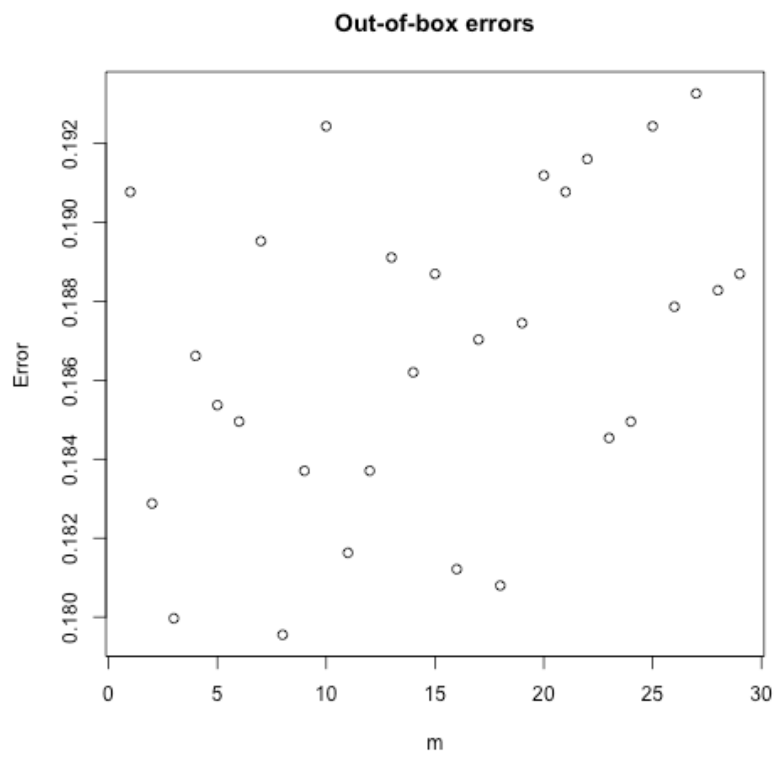


Figure 4

important than others. This can be seen in Figure 6, where there are two plots - MeanDecreaseAccuracy and MeanDecreaseGini. MeanDecreaseAccuracy represents how much the model looses in its accuracy if a variable is excluded from the model. MeanDecreaseGini is a measure of variable importance for the Gini Index, as discussed in Chapter 3. By just examining the first variables in the plots, it is easy to see that heuristically the importances make sense.

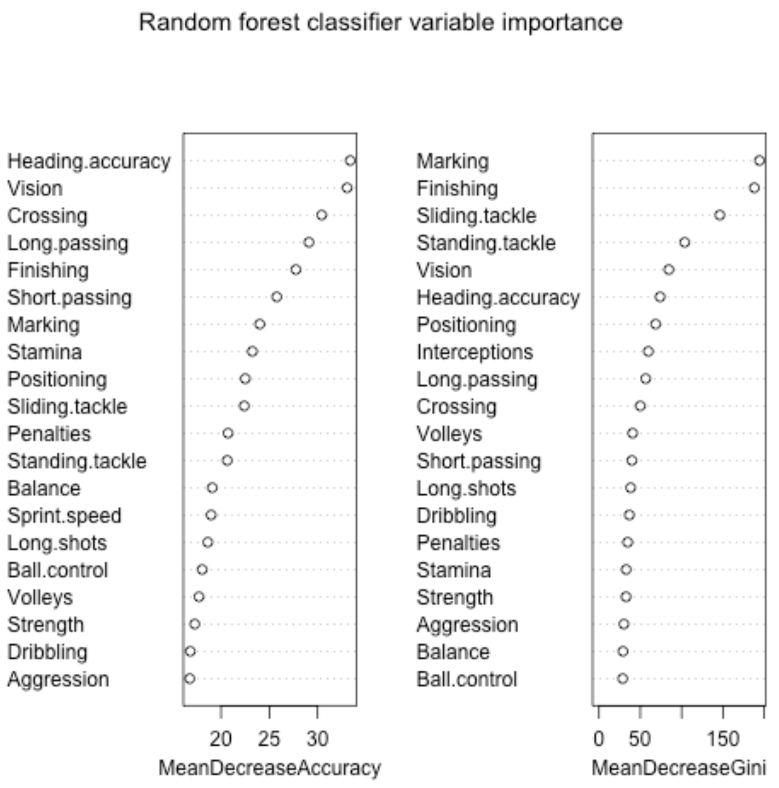
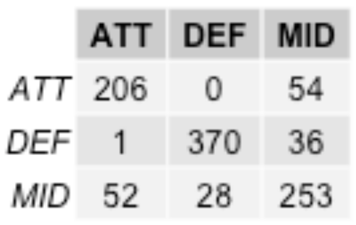


Figure 6

As was done in the case of examining the accuracy of the decision tree model, the accuracy of the random forest model will be assessed as well. The confusion matrix for the random forest model can be found in Figure 7. The model test error is 0.171, which translates to an accuracy of $82.9\,\%$. The random forest model is clearly better at classifying the player's position.



	ATT	DEF	MID
ATT	206	0	54
DEF	1	370	36
MID	52	28	253

Figure 7

Lastly, a boosting model is fit to the data. As discussed in Chapter 3, boosting grows the trees sequentially, so new parameters must be introduced - namely the number of trees B, the shrinkage parameter λ and the interaction depth d. These parameters will be decided by running a 5-fold cross-validation with the model. The three parameters are mapped to see how they perform. Three possible values are set for the shrinkage parameters: 0.10, 0.05 and 0.01. Three possible values are also set for the interaction depth: 1, 2 and 3. 15 different values for the number of trees is tested: 15 equispaced values ranging between 100 and 1500.

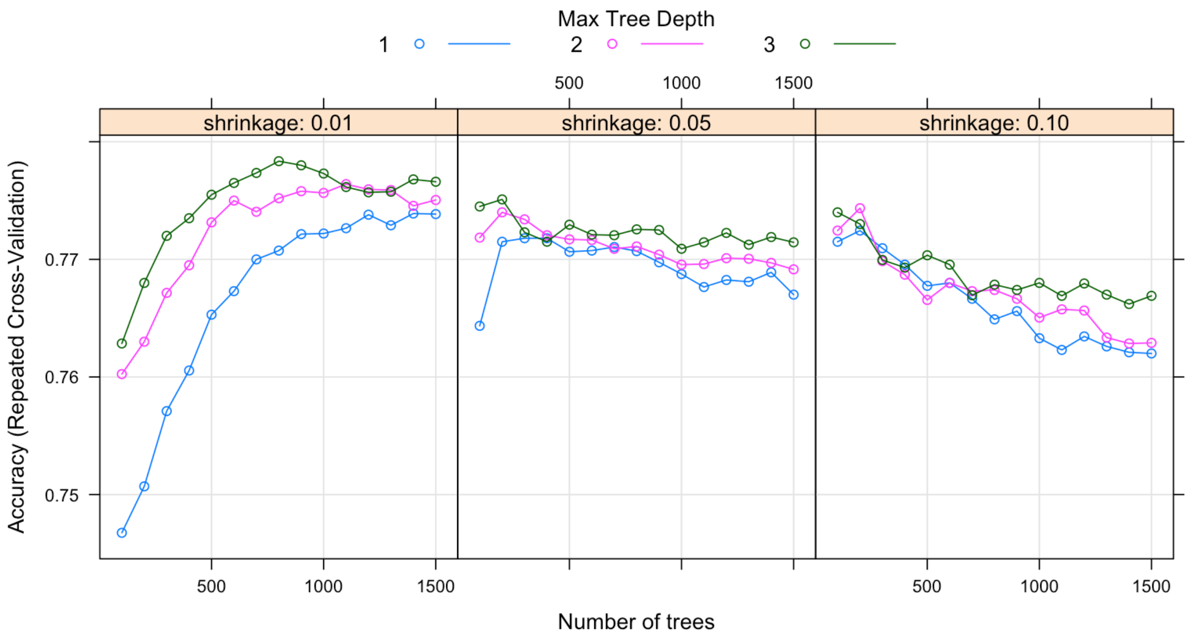
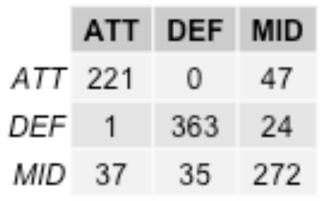


Figure 8

From Figure 8 it is easy to see that that the best accuracy was achieved when the shrinkage was 0.01, the number of trees was 800 and the max tree depth (interaction depth) is 3. These parameters are used for the boosting model.

To understand the model accuracy, it is once again necessary to look at the confusion matrix. From the confusion matrix seen in Figure 9, it is easy to calculate that the test error for the boosting model is 0.144. The model accuracy is therefore $85.6\,\%$, making it the most accurate model discussed in this paper.



	ATT	DEF	MID
ATT	221	0	47
DEF	1	363	24
MID	37	35	272

Figure 9

As was the case in the random forest model, a boosting model is not as interpretable as a decision tree model. Nonetheless it is possible to understand what variables have a greater effect in the classification than others. This is similar to what was seen in the random forest setting. Figure 10 shows the 15 most important variables in the boosting model. In it, it is seen that Marking is by far the most important variable, followed by Finishing and Sliding.tackle. Again this makes heuristically a lot of sense, since all these three variables are closely associated with a player position.

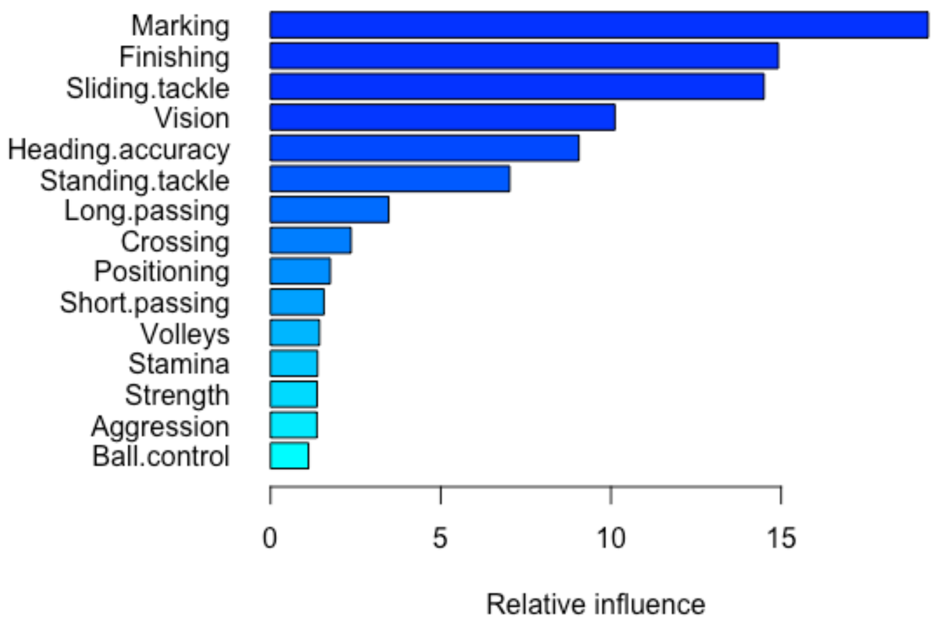


Figure 10

5. Conclusions

This paper used different tree-based learning techniques to try to understand variables in the FIFA football player dataset, which could help classify player positions.

The results of this paper are that player classification is possible with tree-based statistical learning methods. There are clearly certain attributes that suggest for a player to play in a certain position. The results show that while none of the models performed terribly, the random forest and boosting had better performance. This is explained by the high variance, or overfitting, which a decision tree does.

Both the random forest and boosting model gave great importance to the *Finishing* and *Marking* skill. From Figures 11 and 13 it can be seen that these are very clearly skills that differ between defenders and attackers and so it would make sense that they are decision boundaries for the classifiers.

6. References

- Gareth, James et al. An Introduction to Statistical Learning with Application in R, Springer, New York, USA, 2013, pp. 373–413.

Appendix

The R code that was used in this paper and the dataset can be found from: https://github.com/maresiaerik/statistical_learning_course

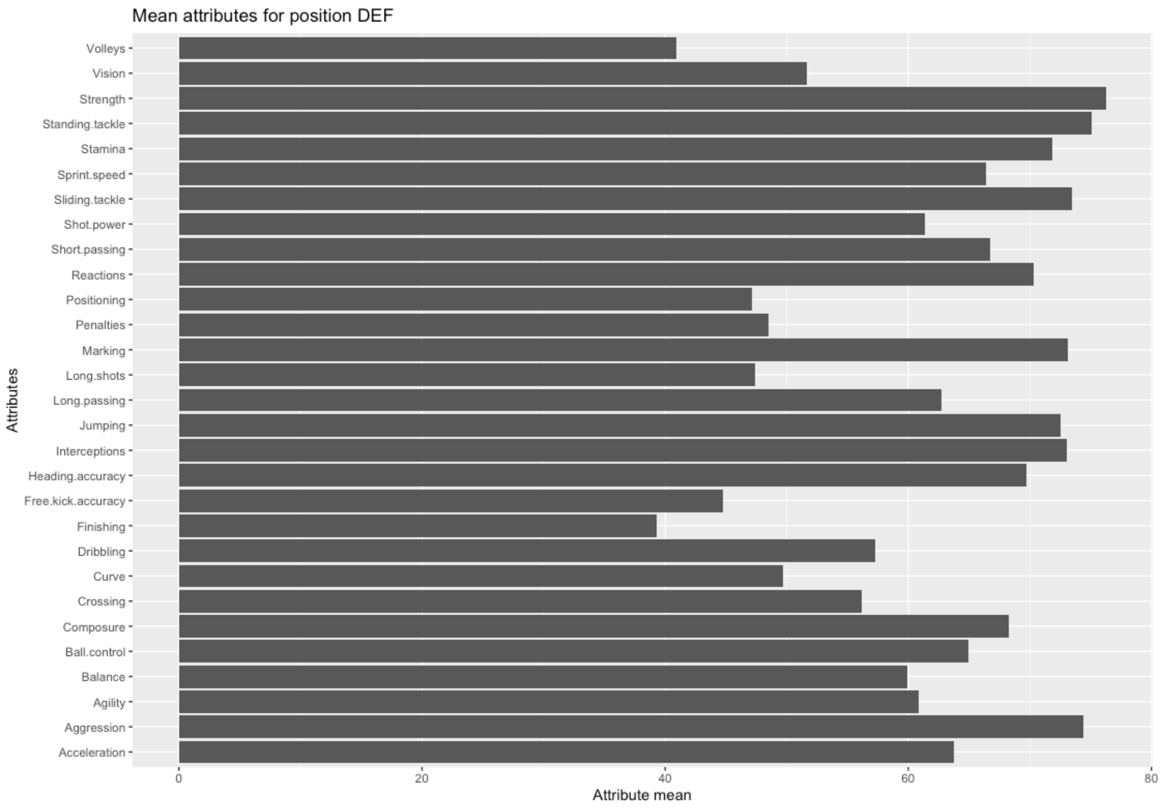


Figure 11

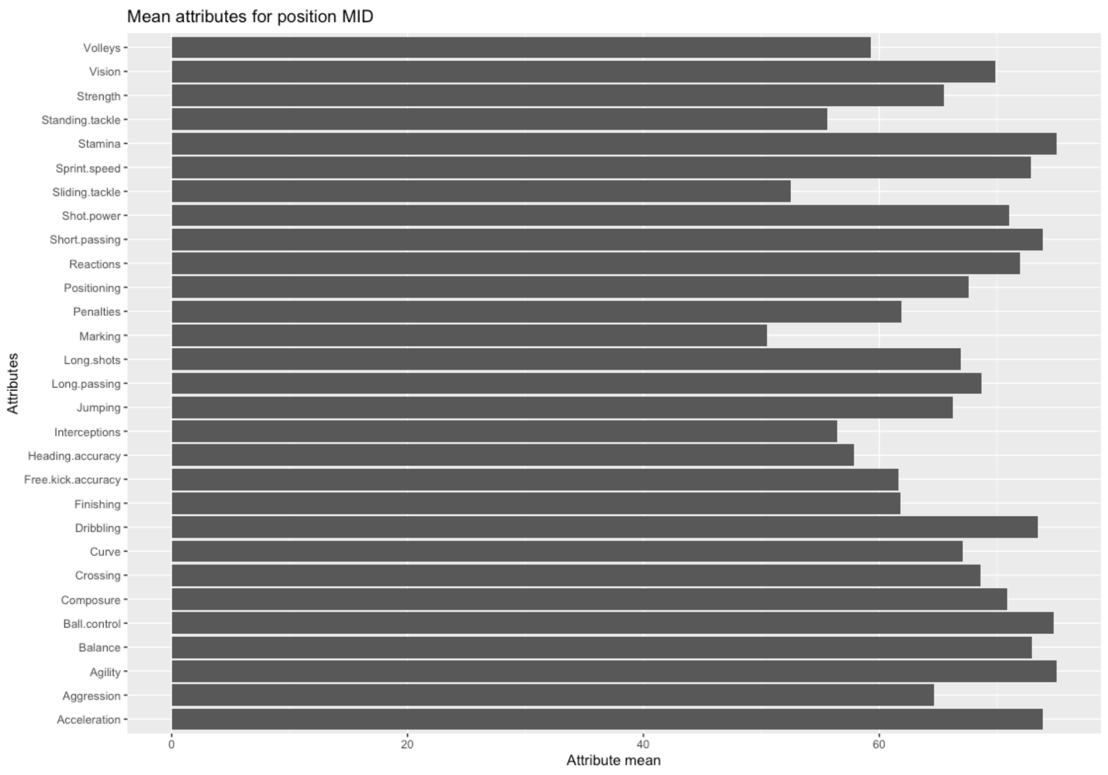


Figure 12

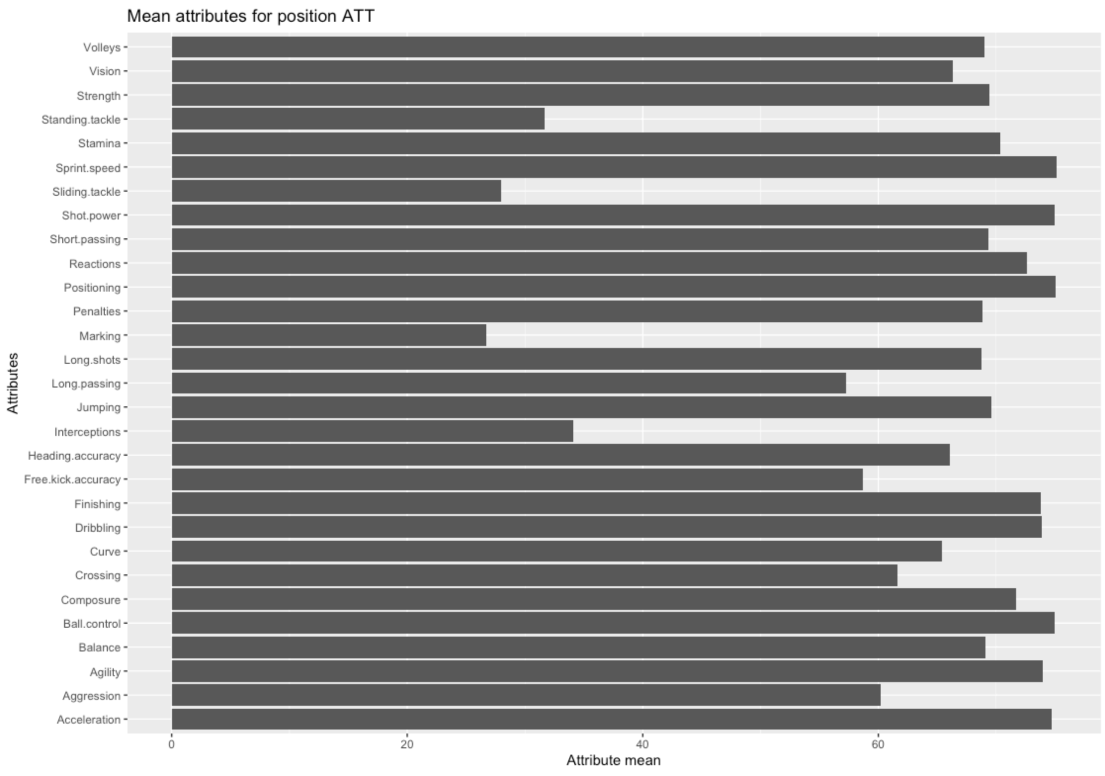

Mean attributes for position ATT
Volleys
Vision
Strength
Standing.tackle
Stamina
Sprint.speed
Sliding.tackle
Shot.power
Short.passing
Reactions
Positioning
Penalties
Marking
Long.shots
Long.passing
Jumping
Interceptions
Heading.accuracy
Free.kick.accuracy
Finishing
Dribbling
Curve
Crossing
Composure
Ball.control
Balance
Agility
Aggression
Acceleration
Attributes
0
20
40
60
Attribute mean

Figure 13