BA305: FIFA Position Recommendation

Using a decision tree model to predict a player's position based on their FIFA game attributes

Group 8
Akash Bhatnagar, Chung-Yeh Yang, Kimberly Low, Micah Lee

SUMMARY

Professional soccer players are the best within their sport. Each player has a specific combination of skills that make them unique. The FIFA video game rates a player's capabilities in areas of the sport on a 0-100 scale. Because there are so many different possible attributes a player can have, it can be difficult to determine the position that best suits the player. The model we have created allows us to determine the best position that a player should be played in: a forward/attacking position, a midfield position, or a defending position. By utilizing the attribute ratings, as well as physical characteristics (height & weight), we created prediction models for a player's best/most useful position. We excluded goalkeepers as we felt that this model is better suited for outfield players. Overall, we believe our model is effective and useful in predicting a player's preferred or best outfield position based on their specific skill set. This can be important when creating a player in FIFA - determining their skill set, and thereby needing to find the best and most effective position for the player. In FIFA career mode, or other game modes in which you should create a player, this model might be useful when a person has determined the player's attributes and therefore created a player's specific skill set.

DATA DESCRIPTION

This dataset *player_22 .csv* includes soccer players data for the career mode from the video game FIFA 22. There are 19,329 records and 110 different attributes. Each record corresponds to a specific professional soccer player; containing different attributes and information regarding said player. These attributes include position, height and weight, as well as other attributes relating to the skill, technical and athletic abilities of each player. The skill, technical and athletic attributes are created by FIFA, and are based on a player's performance during the season. This allowed us to obtain a data set with which most of the data is already normalized.

<u>DATA PRE-PROCESSING</u>

Selecting Attributes

Since there were too many attributes to construct a model, we selected the ones that were helpful for our prediction model. We first kept *player_positions* because our model aims to predict the players' positions. We also kept physical characteristics like height and weight and FIFA attribute scores like a players' attacking, skill, movement, power, and defending scores. The final dataset had 27 variables after the selections.

Combining and Sorting Attributes

Since 27 variables are still too much for a prediction model, we reduced them by combining variables that are highly correlated. Luckily, FIFA groups player attributes into five categories: attacking, skill, movement, power, and defending- each of which have their own subset of attributes. For example, attributes that rate a player's attacking include crossing, finishing, heading accuracy, short passing and volleys. We tested correlations to see if we could combine these attributes into one overarching category. After running a correlation table for each category, we found that most variables, under their respective aspect, are all positively correlated to each other. The correlations weren't all strong, but because they are positive, we didn't feel the need to drop any of them. Therefore, we combined the attributes under the same category by computing the average. We didn't need to standardize any attributes because they were already standardized by FIFA as most scores were on a scale of 0 to 100. The only exception was *skill_moves*, whose rating was on a scale of 1 to 5. We dropped this variable as normalizing it would increase the difference between each rating level too much - adding too much significance to the variable. After combining variables, we then obtained the following components: *attacking_average*, *skill_average, movement_average, power_average and defending_average.* We also ran a PCA test to see if we could reconfigure our categories, but found that we could not easily interpret the component outputs that were given to us. So, we used the average of attributes in a category as it was an easier and more efficient way of grouping.

Afterwards, we finalized our output variable *player_positions*. In soccer, a player can play different positions depending on the tactics of the team, so in the original dataset, some players have multiple positions in the *player_positions* column. We decided to choose the first position that appears in the column, which is the position that each player tends to play most. Since there are 17 unique positions in soccer (according to FIFA), we decided to simplify them into “forward”, “midfield”, “defender” and “goalkeeper”.

Removing Redundant Records

The next step for the data preprocessing was to remove records that were not helpful for the prediction model. Taking a glimpse at the dataset, we noticed that all goalkeepers had “NA” in the attribute “defending”. Other than that, goalkeepers obtained a rather lower score in other aspects compared to players in other positions. We decided to drop all goalkeepers’ records from the dataset. There are two reasons to remove goalkeepers. First, the model can’t run with all the “NA” values. More importantly, we don’t need our model to predict whether a player is a goalkeeper because their ratings are substantially different from others (and therefore it is easy to distinguish, without a model who is a goalkeeper and who is not) - this model is much more useful for outfield players.

OUR MODEL: THE DECISION TREE

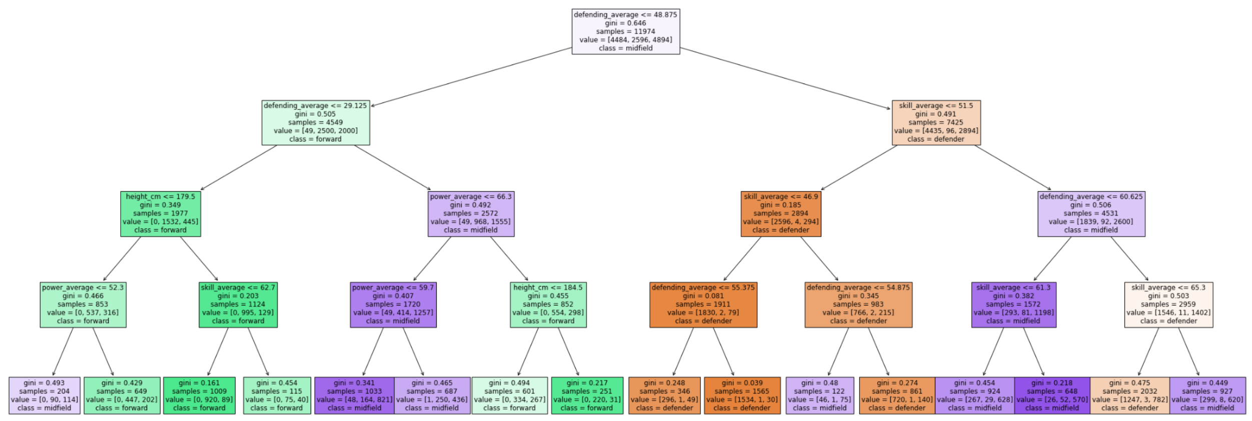
Before building our classification model, we had to settle on which algorithm to use. The four options that were available to us were: decision trees, k-nearest neighbors, logistic regression and neural networks. We ruled out logistic regressions as they can only make binary classifications and our model aims to classify data into three categories (midfielder, forward and defense). We considered using k-nearest neighbors and neural networks as both of these models would be able to deal with more than two classifications. However, given that these are black-box models, we would not be able to see the reasoning behind the predictions. Part of our reason for creating this model was to understand which mixture of statistics were more likely to be correlated to a certain position, so it was vital that we could interpret our model. Therefore, we settled for the decision tree.

Building the Decision Tree

The first step in building a decision tree is to split our data into a testing and training set. We decided to use a 70% training, 30% testing split as we had over 19,000+ observations total. This gave us the ability to maximize the number of observations used to train the model while still leaving enough for testing. It was also not necessary to stratify our data since all positions were represented in both data sets.

Next, we created a complete decision tree using the default specifications. This gave us a tree with too many leaves to visualize and a 100% accuracy on the training data (72% accuracy on test data), indicating overfitting. To create a more comprehensible tree, we ran GridSearch to find the optimal parameters for pruning the tree. The algorithm recommended using 20 for our *max_leaf_nodes*, 100 for *min_samples_leaf* and 40 for *min_samples_split*. Although the recommendation permitted our decision tree to go up to a max depth of 6 layers, we trimmed it at 4 so that it could be easier to interpret but still have enough depth for the rules to be meaningful. We used the Gini coefficient to measure accuracy for each node. The following is our optimal tree (figure 1).

Figure 1: Optimal Pruned Decision Tree


defending_average <= 48.875 gini = 0.646 samples = 11974 value = [4484, 2596, 4894] class = midfield
defending_average <= 29.125 gini = 0.505 samples = 4549 value = [49, 2500, 2000] class = forward
skill_average <= 51.5 gini = 0.491 samples = 7425 value = [4435, 96, 2894] class = defender
height_cm <= 179.5 gini = 0.349 samples = 1977 value = [0, 1532, 445] class = forward
power_average <= 66.3 gini = 0.492 samples = 2572 value = [49, 968, 1555] class = midfield
skill_average <= 46.9 gini = 0.185 samples = 2894 value = [2596, 4, 294] class = defender
defending_average <= 60.625 gini = 0.506 samples = 4531 value = [1839, 92, 2600] class = midfield
power_average <= 52.3 gini = 0.466 samples = 853 value = [0, 537, 316] class = forward
skill_average <= 62.7 gini = 0.203 samples = 1124 value = [0, 995, 129] class = forward
power_average <= 59.7 gini = 0.407 samples = 1720 value = [49, 414, 1257] class = midfield
height_cm <= 184.5 gini = 0.455 samples = 852 value = [0, 554, 298] class = forward
defending_average <= 55.375 gini = 0.081 samples = 1911 value = [1830, 2, 79] class = defender
defending_average <= 54.875 gini = 0.345 samples = 983 value = [766, 2, 215] class = defender
skill_average <= 61.3 gini = 0.382 samples = 1572 value = [293, 81, 1198] class = midfield
skill_average <= 65.3 gini = 0.503 samples = 2959 value = [1546, 11, 1402] class = defender
gini = 0.493 samples = 204 value = [0, 90, 114] class = midfield
gini = 0.429 samples = 649 value = [0, 447, 202] class = forward
gini = 0.161 samples = 1009 value = [0, 920, 89] class = forward
gini = 0.454 samples = 115 value = [0, 75, 40] class = forward
gini = 0.341 samples = 1033 value = [48, 164, 821] class = midfield
gini = 0.465 samples = 687 value = [1, 250, 436] class = midfield
gini = 0.494 samples = 601 value = [0, 334, 267] class = forward
gini = 0.217 samples = 251 value = [0, 220, 31] class = forward
gini = 0.248 samples = 346 value = [296, 1, 49] class = defender
gini = 0.039 samples = 1565 value = [1534, 1, 30] class = defender
gini = 0.48 samples = 122 value = [46, 1, 75] class = midfield
gini = 0.274 samples = 861 value = [720, 1, 140] class = defender
gini = 0.454 samples = 924 value = [267, 29, 628] class = midfield
gini = 0.218 samples = 648 value = [26, 52, 570] class = midfield
gini = 0.475 samples = 2032 value = [1247, 3, 782] class = defender
gini = 0.449 samples = 927 value = [299, 8, 620] class = midfield

Interpreting the Decision Tree

The tree indicates that defending scores, power scores, skill scores and height are the major determinants of a player's position. Breaking it down, we came up with a set of rules for each outcome. Although there are 16 end nodes, we combined some of them (leaf nodes with the same outcome that had branched out from the same previous node), so in total the tree has 11 rules as described in table 1 and shown in figure 2.

Figure 2: Simplified Optimal Decision Tree

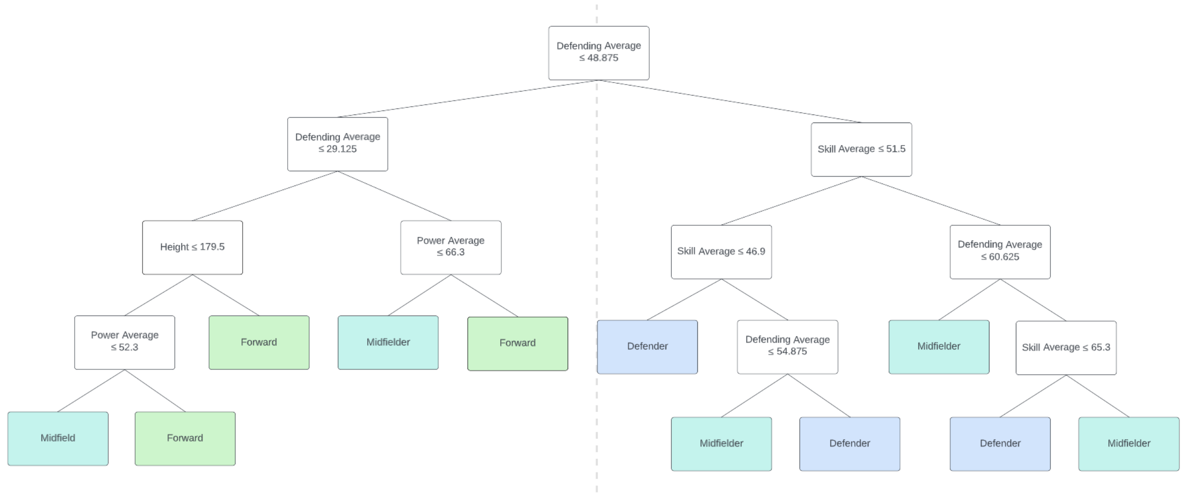


Table 1: Classification Rules Established by the Decision Tree

Position	Rules
Defender	A player is a defender if: 1. Their average defending score > 48.875, and average skill score ≤ 46.9. 2. Their average defending score > 54.875, and their average skill score is between 46.9 (exclusive) and 51.5 (inclusive). 3. Their average defending score > 60.625, and their average skill score is between 51.5 (exclusive) and 65.3 (inclusive).
Defenders follow the most straight forward rule- a high defending score likely leads to becoming a defender. However, there are variations in defending scores- with some defenders having higher scores than others. Surprisingly, skill scores seem to increase as defending scores increase. This could indicate that the overall competency of the player is improving, not just their ability to defend.	
Forward	A player is classified as a forward if:

	1. Their average defending score is ≤ 29.125, their height is ≤ 179.5cm. and their average power score > 52.3. 2. Their average defending score is ≤ 29.13, their height > 180cm. 3. If a player’s average defending score is between 29.13 (exclusive) and 48.88 (inclusive), and their average power score > 66.3.
Forwards generally have the lowest defending scores of the three groups, but medium to high power scores. Intuitively, this makes sense as forwards are typically noted for their shooting abilities and strength since it is considered an aggressive role.	
Midfield	A player is classified as a midfielder if: 1. Their average defending score is between 29.125 (exclusive) and 48.875 (inclusive), their average power score ≤ 66.3. 2. Their average defending score is between 48.88 (exclusive) and 54.88 (inclusive), and their average skill score is between 46.9 (exclusive) and 51.5 (inclusive). 3. Their average defending score is between 48.875 (exclusive) and 60.63 (inclusive). 4. Their average defending score > 60.625, and average skill score > 65.3.
The rules for midfielders are more difficult to comprehend as the midfield position tends to require a broader spectrum of skills. If forwards and defenders are positions on opposite ends of a spectrum, then midfielders are generally an in-between- with some being more offensively-oriented and others being more defensively-oriented. Therefore, they are expected to have attributes that fit both a forward and defender, though perhaps to a lesser scale. Our model shows that. The defending scores are lower than defenders but higher than forwards, and their skill scores can be higher than defenders.	

Evaluating our Model

In order to evaluate our model, we calculated both the total gini score and created a confusion matrix to break down our errors. Our gini spit out a total weighted value of 0.73 (similar to the accuracy created by the confusion matrix). Our training accuracy was 75.64% and testing accuracy was 73.47%. The following shows our confusion matrix (table 2).

Table 2: Decision Tree Confusion Matrix on Test Data

		Predicted		
		Defender	Forward	Midfielder
Actual	Defender	1556	0	354
	Forward	2	831	251
	Midfielder	478	277	1384

Midfielders have the lowest accuracy while defenders have the highest accuracy. A potential reason for this is that when we tested the correlations between the component attributes for each category, defending attributes had the highest correlations. So when averaged out into a single defending variable, there was less data loss. Defending scores play a large role in the decision tree, being the first criteria that the tree is split on. Additionally, as mentioned before, midfielders are usually regarded as in-between defenders and forwards, so it's harder to distinguish their preferred attributes from other roles.

Comparisons to Benchmarks

Knowing the accuracy of our decision tree, we now have to compare it to benchmarks to determine whether or not it is a useful model. We will evaluate it using the naïve rule. In the dataset, 41% of players are midfielders, 22% are forwards and 37% are defenders. Therefore the default position for classification is midfielders with a 41% accuracy rate. Looking at random classification, If we were to randomly assign a class to a player then our accuracy rate would be 33% as there are 3 possible classes. The accuracy of our model beats both of these. We judged the logic of the decision tree and concluded that it seems to be sound. So, we are satisfied with our model.

Limitations and Improvements

With that being said, there are still limitations as well as improvements that could be made. Here are some of the most important ones:

1. Finding a dataset with player attributes that are measurable

Although we can interpret roughly what each player attribute is measuring, EA (the FIFA game makers) do not provide an actual formula for how these stats are calculated. Therefore, we cannot be completely certain exactly what each of these attributes means. It also constrains our model to be used in FIFA game contexts only- or to players that are in FIFA. Our model cannot be applied to real-life soccer players without FIFA specific statistics.

2. Finding a better way to aggregate our input statistics

As mentioned before, averaging the attributes across categories is not the ideal way to aggregate our input variables. Most of the component attributes in the power category are not highly correlated. But we were forced to include this category in our models because if we dropped it, the model accuracy would be reduced too much. We also couldn't drop individual components within power because then there would be too few variables to average out. This input variable ended up playing a crucial role in our model as the power criteria appears multiple times. We had a similar problem where one component in the movement category (*movement_reactions*) did not correlate with the other variables highly. However, movement did not end up being a criteria used in the decision tree so this was not a problem.

3. Finding a way to account for multiple positions

Most professional soccer players are capable of playing multiple positions. This is reflected in the dataset as the position column was multivariable. However, for the purpose of this project, we only selected the primary position - the one they are most known for - of all players to simplify the data. Overall, this doesn't affect our model by much because we are looking at optimal playing positions and not all possible playing positions. However, just because a player usually takes on a certain position does not mean that position is actually their optimal position.

Another Model: Neural Networks

Another model we tested out was a neural network model. For this model, we split the data into x and y variables, then split the data set again for a training and a testing set. To form the model, we used a variation of the MLPRegressor function which was more suited to our classification intent: MLPClassifier. This function works about the same as MLPRegressor, only we feed a regular dataframe with classification variables into the model rather than a one with scaled numerical variables.

For the neural network we used one layer with 10 nodes, logistic activation, an lbfgs solver, and a max iteration limit of 1,000. These were all chosen through trial and error because they resulted in the highest prediction accuracy against the validation set, while also keeping the execution time within reason; the max iteration limit of 1,000 was required because the default value of 100 wasn't enough for convergence. Overall, this model had a score of 80.3%, or an error rate of 19.7%, against our test set. Although this score is fairly high, we chose not to go with this model as we were interested in how we came to those results, and neural networks most effectively used when only interested in results.

Another Model: K-Nearest Neighbors

We also tested out the model using K-nearest neighbors. We first turn the attributes *player_positions* into categorical variables to enable us to run the model. Next, we split the dataset into 70% training set and 30% testing set. We then ran the K-nearest neighbor model to see how many K returns the highest accuracy. The result shows that the accuracy of the model is 0.75 to 0.8, and it can be as high as 0.796 when K=27. Though this accuracy is slightly higher than the decision tree, we cannot interpret the reasoning behind how a player is positioned.

FINDINGS AND IMPLICATIONS

As our main decision tree model has shown us, the most important statistic when classifying players seems to be their aggregate defending stat. Not only is it used to separate the defenders from the other positions in our decision tree, but it is also used to distinguish between forwards and midfielders - forwards having a defending score less than 29.125 and midfielders being between 29.125 and 48.875. This makes sense because midfielders are often seen as supporting the forwards or defenders, so they should have a skill set that lends to both aspects of the game. Overall, there is a lot that goes into choosing player positions - especially outside of statistics and skill sets. Players could be lined up at certain positions because they have specific complimentary playstyles with their teammates. With this, we are happy with our model's 73.47% accuracy rate since we only observe given statistics for the players.

With our models, we envision that it could be a decent predictor of what position type a player should play in order to most effectively utilize their specific skill set. For example, we know that often in more amateur sports settings one athlete could take on multiple positions if they are seen as a generally skilled player. Using our model, players could be scored on a number of different variables similar to those we used when building the model, and be placed in a definitive position where the rankings are most similar.

More realistically, for users of the FIFA video game offered by EA Sports, this model would be useful with creating a player within the game, as you can affect different attributes and therefore create a unique player with a customized skill set. You can then utilize this model in order to predict the best position for which the player is best suited given their unique set of attributes. Alternatively, you could use this model to upgrade your favorite players' stats to fit a specific position in the Squad Builder mode. Overall, this model could affect the way the FIFA players view the game while creating their own players; while also being a potential tool for managers in allowing them to see their players' best and most effective positions based on their attributes in the FIFA game - given the player data that truly reflects the players' actual skill levels of their relevant attributes.