DATA 1030 Project Final Report

Kaishuo Zhang
Brown University DSI
https://github.com/Kingsley024/Data_1030_Project.git

Introduction

In this project, I used the dataset named "All Time Premier League Player Statistics". I am going to explain how I predict a player's best position using the statistics of their career in the English Premier League. The target value is the position of a given player, which is defined as forward, midfielder, and defender. Hence, it is a classification problem. This result could be used when deciding what position a player should be playing based on his performance on the pitch. For instance, if a defender is producing stats like a midfielder, it would be a good indicator that he will be a better fit in the midfield. This problem is important because when deciding a player's best position on the field, we can only look at his previous performance and many players found success after the transition to a new position at some point in their career. This data set was found on Kaggle, the author obtained the data from the official website of the premier league. There is a known issue that some of the players have incorrect goals per match stat and it will be discussed in the EDA section. Thanks to the best data collection technology of the premier league staff, the author was able to get 502 players (excluding Goalkeepers) with more than 50 different stats.

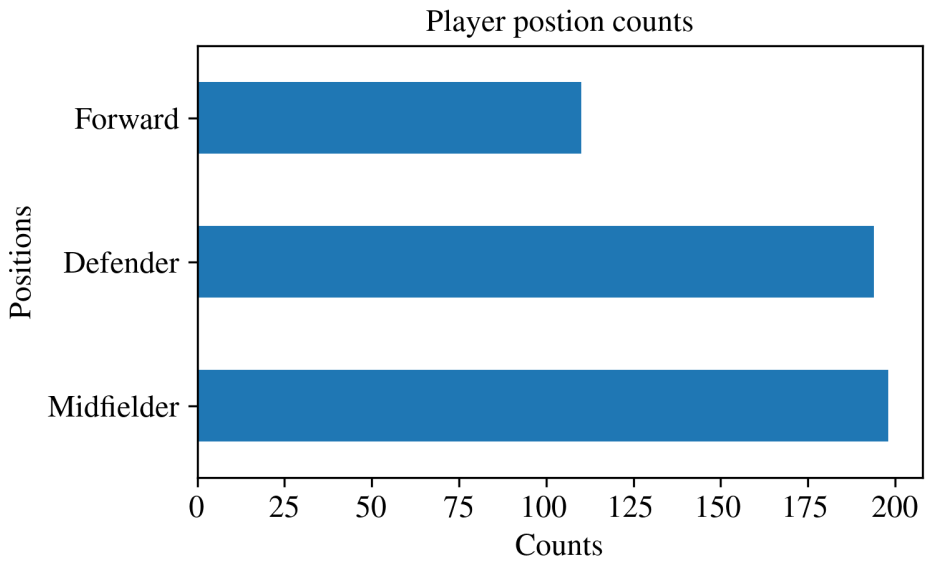
The dataset contains game stats associated with each player in different areas on the pitch. For example, shot attempts, goals, interceptions, and passes completed are all included. Some other interesting features are also recorded in this dataset, such as the player's nationality and jersey number. It is fascinating to see if these also have impacts on deciding what position the player is best suited for. In this data set, nationality, jersey number, and club are categorical features and all other features are continuous.

Even Though this dataset is found on Kaggle, there has been no similar research in this manner. Most people use this dataset to visualize the players' performance as a group in the league and also individual players' performance while trying to show how much better or worse a player is compared to his peers.

Exploratory Data Analysis

With the incorrect data mentioned in the last section, I found that all the impacted players had 0 goals in the league and for some reason, their appearances are used in that column. So I recalculated all the players' 'Goals per match' stat by dividing his 'Goals' by 'Appearances'.

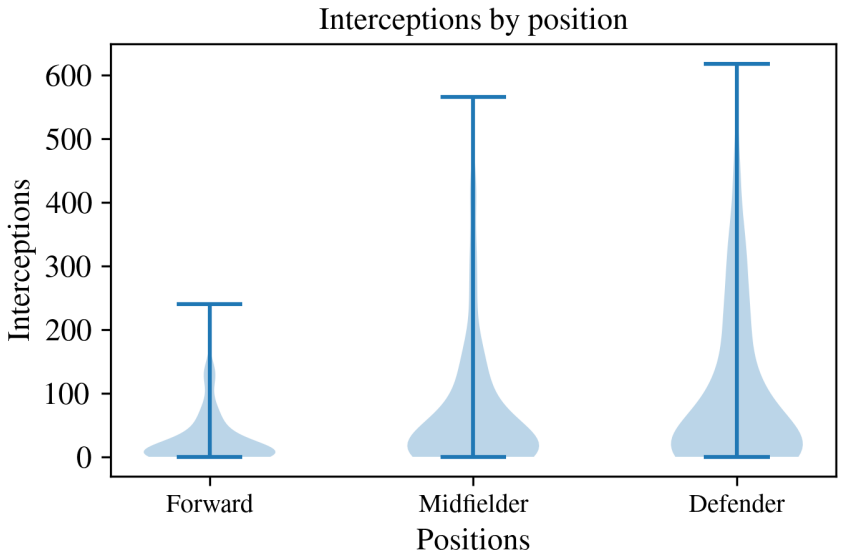
In terms of my target variable, there are three categories, 'Forward', 'Defender', and 'Midfielder'. As shown in the graph below, the dataset is not perfectly balanced as there are somewhat fewer Forward players.



This graph shows the distribution of players are in each position

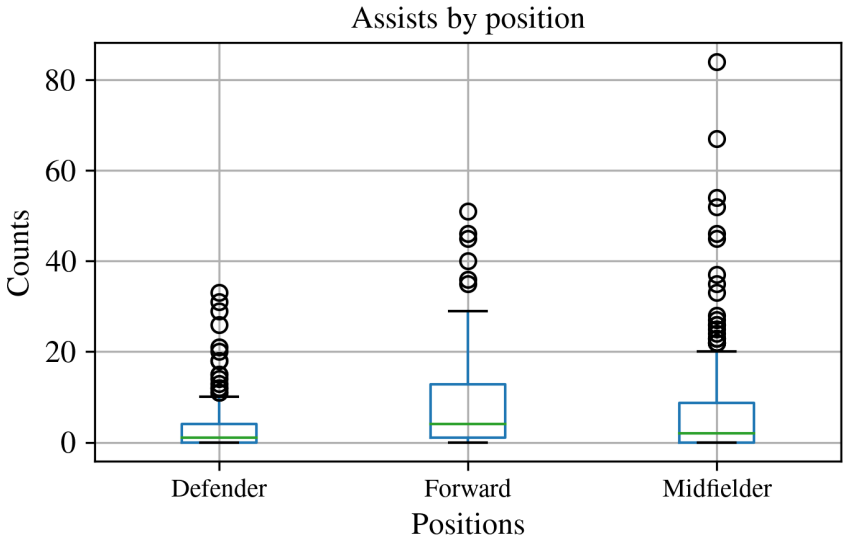
Some other interesting features are interceptions and assists because they are closely related to a player's position. I used the violin plot and the box plot below to show each feature's correlation to the target variable and the results are very interesting.

The expectation is that a player playing in a deeper position should have more interceptions than a player playing a more advanced role. As shown in the graph below, defenders have the most interceptions among all three positions while forwards have the fewest. This perfectly fits our expectations and it indicates that this feature could play a big role in the machine learning pipeline.



This graph shows the correlation between interceptions and positions

On the other hand, assists should be the opposite of interceptions where a player playing up the field should have more assists than a player playing in the backline. It would be easier for them to make an assist since they play closer to the opponent's goal. As you can see in the graph, defenders have the fewest assists of the three positions. While forwards and midfielders have approximately the same median, midfielders have a heavier tail because of outliers. This does make sense because most forwards love to shoot the ball instead of passing it and some midfielders have the pass-first mentality which helps some of them to have the most assists in this data set.



This graph shows the correlation between assists and positions

Methods

In terms of missing values, I imputed 0 for some of the features when it makes sense to do so. For example, many players have "penalty scored" as missing and that is most likely because they have never taken a penalty in their entire career. There are a couple of other features that have the same characteristics and I did the same for those. Other than that, there is one player who has his age and nationality missing and I imputed his age with the mean age of all players and imputed his nationality as None and treat it as a new category.

I chose to divide my data into a train set, a validation set, and a test set with the size 301, 100, and 101, respectively. In terms of the dividing method, I used K Fold even though my data set is not perfectly balanced. I made this decision because I did not see a significant improvement in test score when using stratified K fold. When encoding the features, I used OneHotEncoder on all of my categorical features, including "Jersey Number", "Club", and "Nationality". I used MaxMinEncoder on "Age" because it has a very clear bound. For all the other continuous features, I used StandardScaler. Finally, there will be 179 features to be fed into my model.

In the machine learning pipeline, firstly, I split out the test set before preprocessing and leave it on a side until calculating the test score. Secondly, I put the rest of the data into the pipeline which consists of a preprocessor and a machine learning algorithm. When training the model, I take advantage of the GridSearchCV to help me train the model with different hyper parameters in order to get the best combination of parameters. After the model is trained, I calculate the test score using the aforementioned test set and then save the models, test scores, test sets, and feature names after preprocessing.

I tried 4 different machine learning algorithms in this project:

For the Random Forest, the parameters I tuned are 'max_depth': [1, 3, 10, 30, 100,300,500], 'n_estimators': [1,3,10,30,100,300,500].

For the K neighbors Classifier, the parameters I tuned are 'n_neighbors': [1,3,10,30,50], 'leaf_size':[1,3,10,30,50], 'p':[1,2].

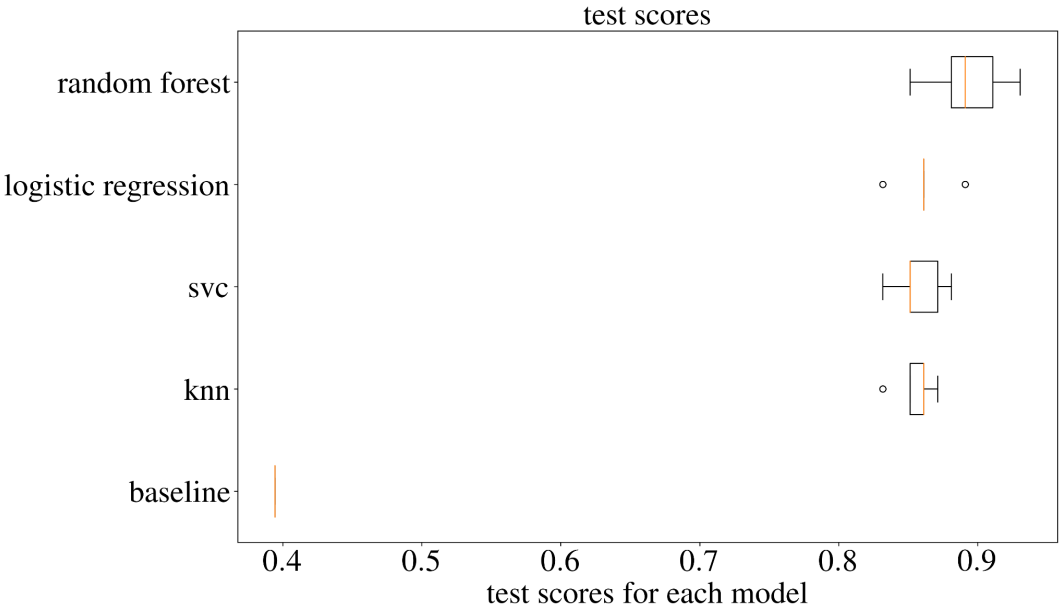
For the support vector machine, the parameters I tuned are 'gamma': [1,0.1,0.01,0.001], 'C': [0.1, 1, 10, 100, 1000], 'degree':[0, 1, 2, 3, 4, 5, 6],'kernel': ['rbf', 'poly', 'sigmoid'].

For the logistic regression, the parameters I tuned are 'C': [100, 10, 1.0, 0.1, 0.01], 'penalty':['l1','l2','elasticnet'].

In this project, I use accuracy as my evaluation metric because the dataset is not very imbalanced and I do care about how many players' positions are predicted correctly. In order to take uncertainties into account, for each model, I run my pipeline five times with five different random states and take the mean and standard deviation of the test scores of all five random states.

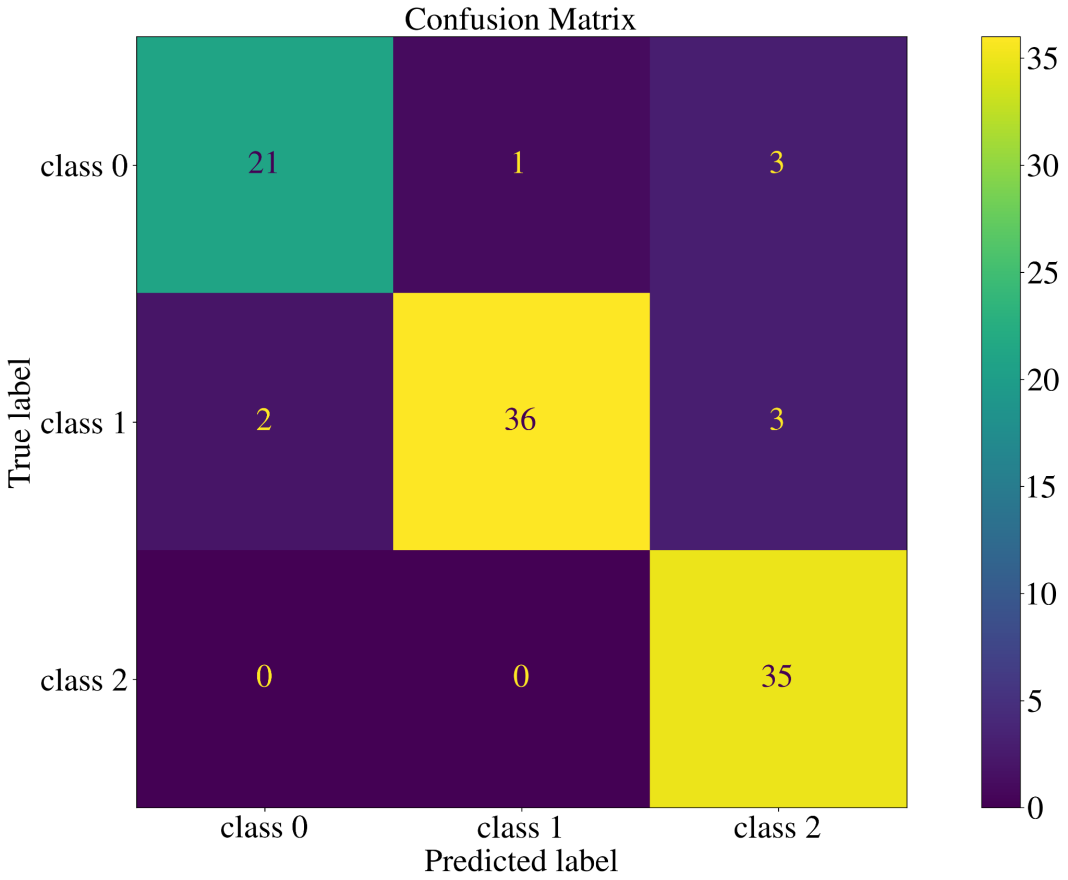
Results

The baseline score of the problem is 0.370 with a std of 0.026 and all of my models perform much better than baseline. As you can see in the figure below, three of my models have a test score of a little over 80%. Random forest is by far the best model among the four, which has an accuracy of 0.893 and a std of 0.027. It is about 20 std higher than my baseline score.



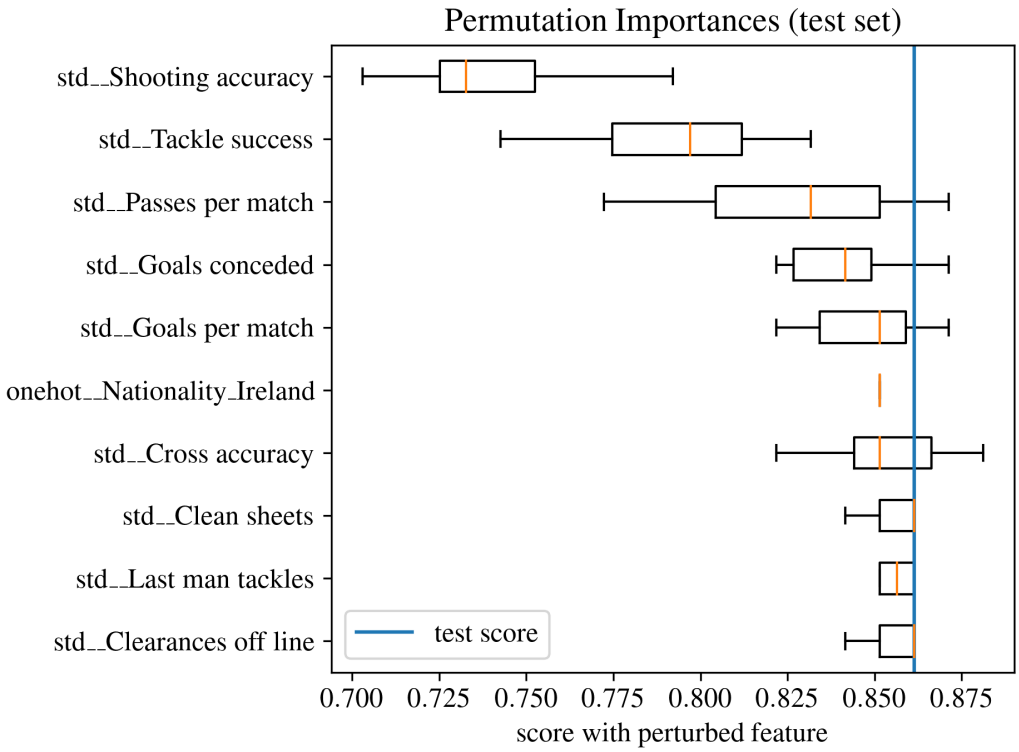
This graph shows all the test scores

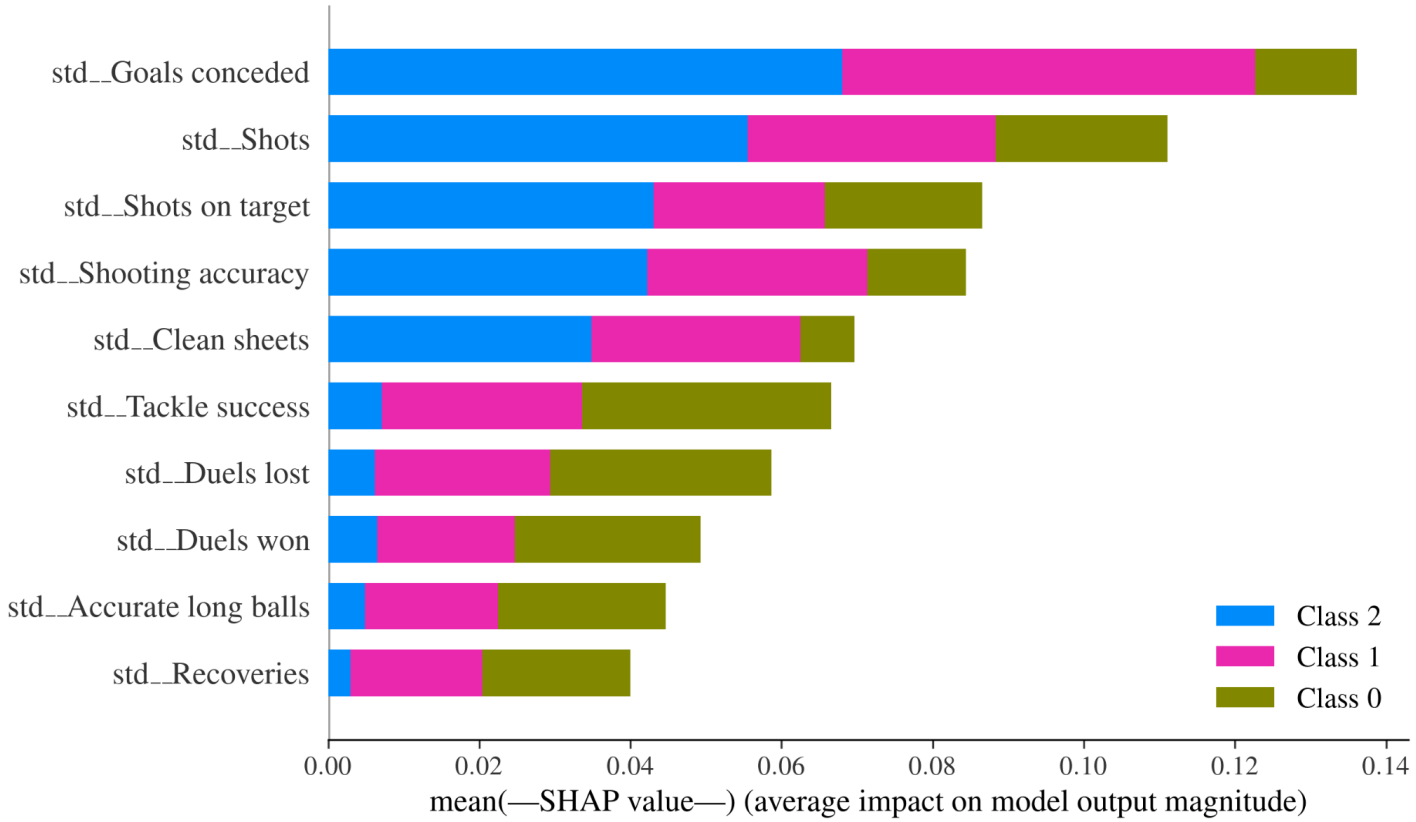
For the random forest (the most predictive model), the confusion matrix is shown below. The most notable thing here is that, for all the class 2 variables (defender), my model has a 100% accuracy. My model's performance on class 0 is the poorest and I believe it is because of the limited data I have on class 0 (forward).

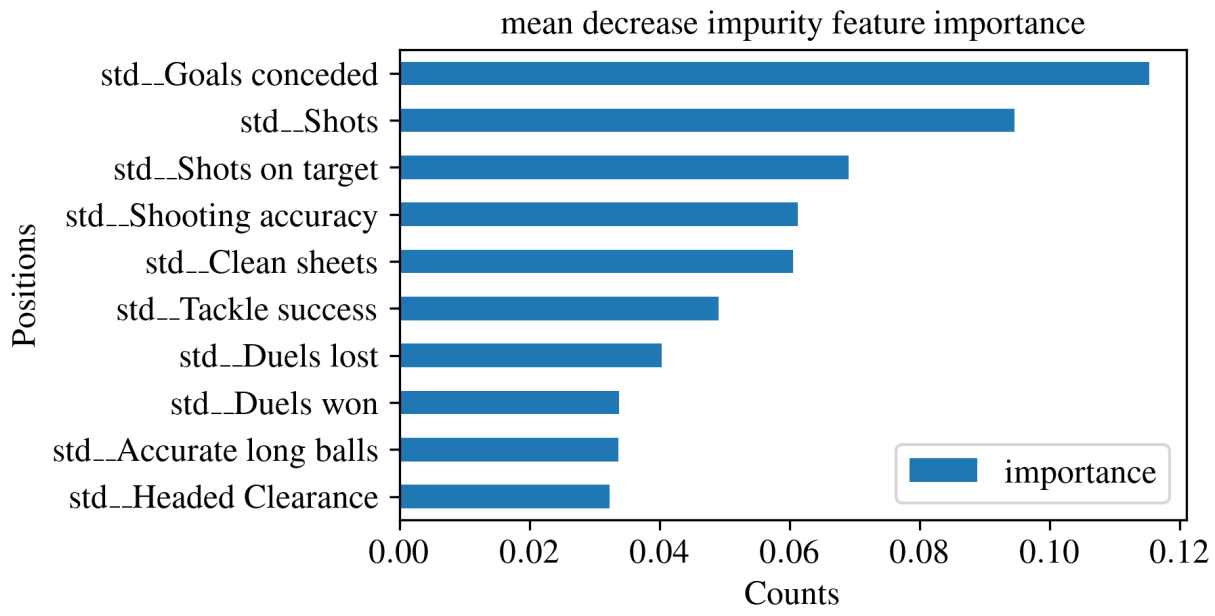


This is the confusion matrix

In terms of global feature importance, I calculated it in three different ways: permutation importance, SHAP global feature importance, and mean decrease impurity. The graphs are shown below. There are many overlaps among the top 10 most important features of the three methods. As expected, a player's position is determined by a position's responsibility on the field. For example, some common features we have on all three graphs are: tackle success, shooting accuracy, and Goals conceded. These are all stats that can only be produced when playing in a specific way. The model uses these features to distinguish players in different positions and that is why it was able to achieve such a high accuracy. Most of the important features are continuous and the one that caught my attention is onehot__Nationally_Ireland, it is funny how a player's position is affected by whether he is Irish or not. Another surprising finding is that 'Goals' did not appear in any of the top 10 most important features. As Goal per match, shooting accuracy and Shots are all very important, I did not see a reason why Goals would not be a very important feature. Overall, the model fits my expectations during EDA and is very interpretable.

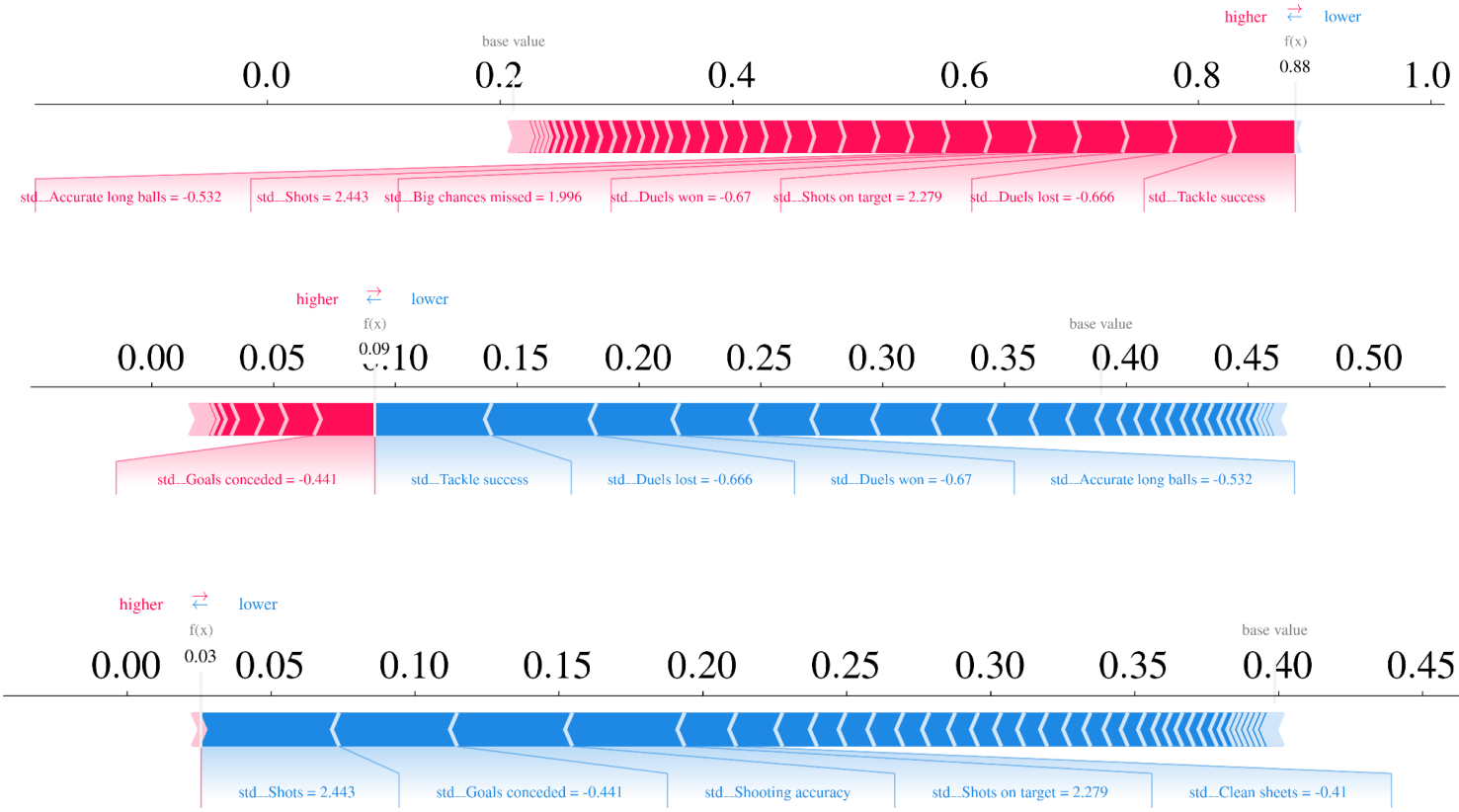

Permutation Importances (test set)
std__Shooting accuracy
std__Tackle success
std__Passes per match
std__Goals conceded
std__Goals per match
onehot__Nationality_Ireland
std__Cross accuracy
std__Clean sheets
std__Last man tackles
std__Clearances off line
test score
0.700
0.725
0.750
0.775
0.800
0.825
0.850
0.875
score with perturbed feature


std__Goals conceded
std__Shots
std__Shots on target
std__Shooting accuracy
std__Clean sheets
std__Tackle success
std__Duels lost
std__Duels won
std__Accurate long balls
std__Recoveries
Class 2
Class 1
Class 0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
mean(—SHAP value—) (average impact on model output magnitude)



This graph shows the mean decrease impurity feature importance

On the other hand, I also chose a player and calculated the local feature importance for that specific prediction. As you can see below, each graph represents the local feature importance on class 0, class1, and class 2, respectively.



This player is predicted to be class 0 (a forward) because he has very low tackle success %, duels lost and duels won. However, his shots on target and big chances missed are both very high. This indicates that he is not a good defender and plays very aggressively near the opponent's goal, therefore, his likelihood to be a forward is very high. These stats also pushes his likelihood to be a midfielder or a defender to very low points(0.09 and 0.03). It is a great example to show how the model predicts a player's position given his stats.

Outlook

In this project, I think there are a couple of things that I could consider other approaches if I had more time. First of all, I imputed 0 for many missing stats as I discussed before. Even though it makes perfect sense to do so since those stats are missing because the player has never done that in a game, I am curious to see what if I leave them as NaN and try some other models that can deal with missing values such as XGBoost. Secondly, the motivation of this project is to predict a player's best position if he is struggling on the pitch or he is from another league. Although my model works very well for players who are playing in the premier league, I did not find a way to test it on players from other leagues because different leagues have a different way of collecting data and most of them are not as complete as the premier league. This problem could be solved if I did what I mentioned in the previous point and trained a model that can deal with missing values. The last thing I would love to do is to play around with the models a little bit more and see if I can get better results off.

References

https://www.kaggle.com/datasets/rishikeshkanabar/premier-league-player-statistics-updated-daily