Predicting a Player's Position using the FIFA 19 Dataset: A Binary Prediction Problem using Various Machine Learning Classification Algorithms

Samaye Lohan
Final Project Report for Data 1030, Fall 2021 at Brown University
Supervised by: Dr. Andras Zsom
Github Repository: https://github.com/SamayeLohan/Data1030-Project

1 Introduction

The Federation Internationale de Football Association, or simply FIFA, is a non-profit organization that is the highest governing body responsible for managing and curating the sports football, futsal, and beach soccer. In an effort to increase the support for the game, EA Sports developed a football simulation video game for various gaming consoles (Playstation, Xbox, Nintendo Switch). FIFA 19 is one of the best selling video games of all time, selling over 260 million copies to date.

The target variable, *Position*, is categorical and we are trying to predict based on skill playing abilities. There are 27 distinct positions which can be grouped into 2 classes: *Offense* and *Defense*. We can define a classification problem in machine learning as the process of predicting a discrete class label output whereas in a regression problem, we are predicting a continuous quantity. With that in mind, we are dealing with a binary classification problem where we take certain attributes related to the target variable *Position* and use them to output our discrete class label, *Position*.

This project aims to predict a player's position through skills-based attributes like *sprint speed, field kick accuracy,* and *ball control* all while exploring various machine learning algorithms that optimize our classification strategies. This is an important task because like any sport, it must be fluid in terms of changing with the times especially when you have players that are able to play multiple positions with the same output. This can have a drastic effect on how the game is played especially when we are able to introduce *flex* players that can play multiple positions.

This original data set is from Kaggle and consists of 87 columns and 18, 207 rows but for our problem domain, I condensed it down to the features that were relevant to predicting a player's position, shown above, which brought our dataset down to 35 columns and 18, 147 rows. Although the data is very well documented on Kaggle, out of the 35 attributes, we have one categorical feature, *Preferred Foot*, and the remaining 33 features are continuous.

This project builds on top of the existing work done by Timotej Zatko and Tomas Hoffer who have utilized the same dataset but to predict a player's market value through physical and skills-based attributes. Their dataset consisted of 92 numerical attributes and 40 categorical attributes that were subject to various preprocessing techniques beyond the ones currently taught in the Data1030 course like *TransformerMixmin* and *BaseEstimator* [1]. They applied Random Forest, Decision Tree Classifier, Logistic Regression CV, and linear SVC as the primary machine learning models before concluding that Random Forest and Logistic Regression were the best performing ones [1]. We aim to extend their research by applying only skills-based attributes in determining a player's position all while selecting a subset of attributes that are directly relevant to a player in a specific spot on the field.

2 Exploratory Data Analysis

In this section of the report, we will highlight some of the visualizations created during the exploratory data analysis phase. As stated above, I have grouped the target variable, *Position*, into the following two categories:

- ➔ *Position 0*: offense = ["ST", "LW", "RW", "LF", "RF", "RS","LS", "CF", "CM","RCM","LCM", "CAM", "LAM", "RAM", "RM", "LM"]
- ➔ *Position 1*: defense =["CB", "RCB", "LCB", "LWB", "CDM","RDM","LDM", "RWB", "LB", "RB", "GK"]

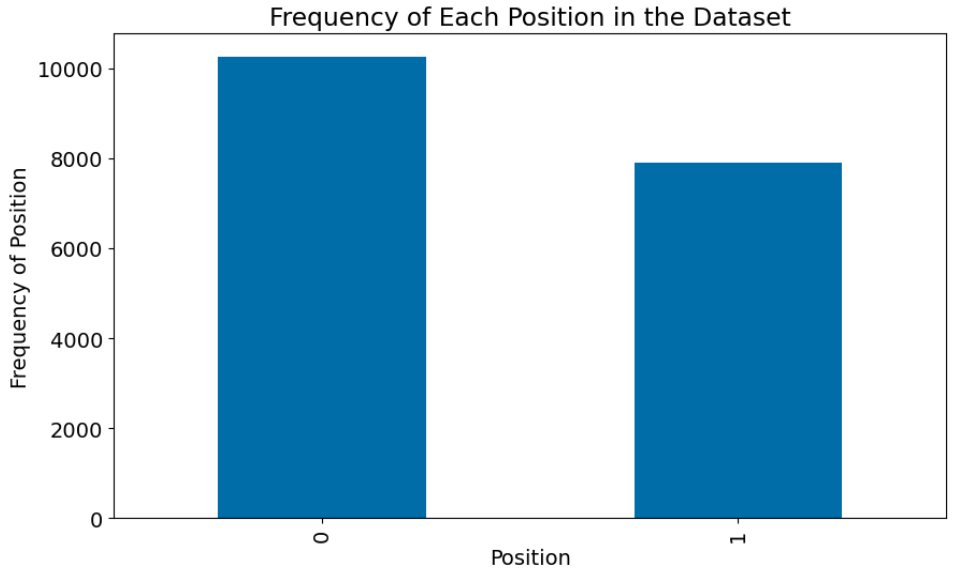

Figure 1: The bar graph, shown above, illustrates the frequency of the two position groups that are defined above. Generally, the offense tends to have the highest concentration of positions on the field at any given moment because these players act as a median to both the attacking and defending positions. This is validated by the bar plot as the offense position has the highest frequency followed by the defensive position.

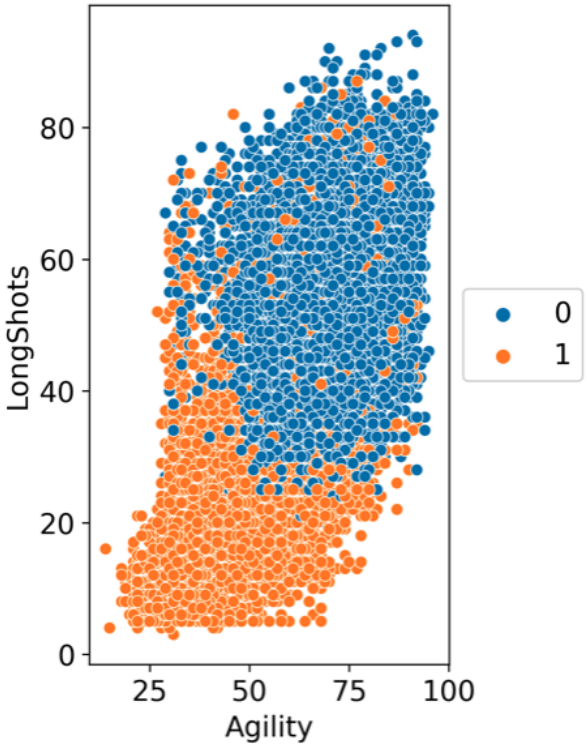

Figure 2: The scatter plot, shown above, shows the correlation between the *Agility* and *Long Shots* feature for the two respective positions. We can see that on average, offense (Position 0) tends to have the highest agility and long shot ability whereas defense tends to have the lowest. This makes sense because the offensive positions are required to cover more ground and facilitate a lot of the playmaking.

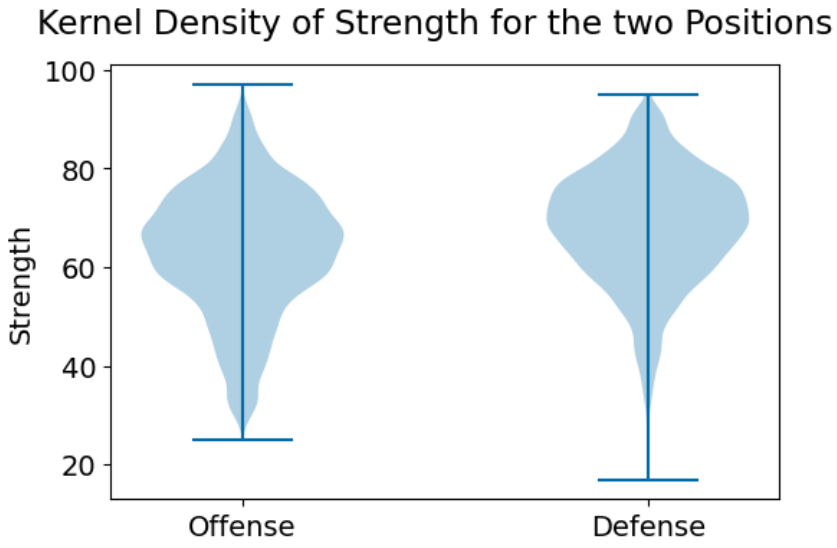

Figure 3: This figure illustrates the kernel density plots for the average strength of all the positions in FIFA19, segregated by the class variables. Observing the graph, it is evident that both positions represent a bimodal distribution where the defense class has a higher overall (mean and variance) strength ability as opposed to the offense class.

3 Methods

3.1 Data Preprocessing

As mentioned previously, there is one categorical feature in our dataset - *Preferred Foot* which takes on one of two values: *left* or *right*. Because we are dealing with unordered categorical data, I implemented a one-hot encoder for the feature. For context, a player in FIFA19 has a preferred choice of foot when playing the game which is crucial in determining a player's position because right-footed players are generally placed on the right side of the field.

For the remaining 33 numerical features, I implemented a MinMaxScaler because they are all bounded between 0 and 100. The EA Sports franchise ranks players on a scale of 0-100, meaning that no metric for any given player can exceed the value 100 or be less than 0. This sets a perfect bound and hence, we can apply the MinMaxScaler preprocessing technique.

3.2 Data Splitting & Machine Learning Pipeline

After conducting EDA, our refined FIFA19 dataset contains 35 columns, where 34 are the feature columns, and 18, 147 rows, which is 60 data points less than the original. The reason for this is because we dropped all the *NA* values in our target variable, *Position*, column. The FIFA19 dataset does not represent data over a given time period and cannot be represented as a function of time. In addition, our initial dataset consisted of raw information that was ungrouped but after the EDA component, I grouped the various positions into two classes to allow for a more accurate divergence into the specific categories. By definition, we do not have a direct group structure and hence, our dataset can be considered to be IID and as shown in Figure 1, our dataset is balanced because neither class falls below the 5% threshold - 10, 256 points belong to class 0 and 7891 points belong to class 1.

In terms of implementation, I defined an *MLPipe_KFold_AUC_ROC()* function which accepts 5 parameters: the feature matrix, target variable, preprocessor, machine learning algorithm being used, and the parameter grid. To ensure reproducibility, a for loop was initialized which encapsulated an 80/10/10 split using train_test_split and due to the large nature of the dataset, the secondary split was done using KFold so that all the models will use the same data splits. Next, we define the pipeline which takes in the preprocessor, from the data preprocessing step, and the machine learning algorithm as the parameters. I utilized sklearn's GridSearchCV to fit the different estimators on the training set and return the best parameters from the parameter grid. The primary scoring method used to evaluate the models was the ROC AUC score because it calculates it based on the predicted scores and thus, provides a better benchmark accuracy of the models as we do not want to overfit a single class. In addition, we are dealing with a binary classification problem and the objective of AUC is to measure the trade-off between TPR and FPR, so it evaluates the classifier as the threshold varies over all the different possibilities. The for loop is initialized to run five times and we multiply it by the random state to ensure differentiability during the splits. The four models that were experimented with were Random Forest, Logistic Regression w/ penalty, SVC, and Decision Tree.

```
Fitting 4 folds for each of 24 candidates, totalling 96 fits
Optimal Model Parameters:  {'ml_algo__criterion': 'entropy', 'ml_algo__max_features': 'log2', 'ml_algo__n_estimator
s': 1000}
ROC AUC Score:  0.9875951898486339
Fitting 4 folds for each of 24 candidates, totalling 96 fits
Optimal Model Parameters:  {'ml_algo__criterion': 'entropy', 'ml_algo__max_features': 'log2', 'ml_algo__n_estimator
s': 1000}
ROC AUC Score:  0.9870888943452735
Fitting 4 folds for each of 24 candidates, totalling 96 fits
Optimal Model Parameters:  {'ml_algo__criterion': 'entropy', 'ml_algo__max_features': 'log2', 'ml_algo__n_estimator
s': 1000}
ROC AUC Score:  0.9873362436575751
Fitting 4 folds for each of 24 candidates, totalling 96 fits
Optimal Model Parameters:  {'ml_algo__criterion': 'entropy', 'ml_algo__max_features': 'auto', 'ml_algo__n_estimator
s': 1000}
ROC AUC Score:  0.9877008190017524
Fitting 4 folds for each of 24 candidates, totalling 96 fits
Optimal Model Parameters:  {'ml_algo__criterion': 'entropy', 'ml_algo__max_features': 'auto', 'ml_algo__n_estimator
s': 500}
ROC AUC Score:  0.988005937829872
```

Figure 4: Random Forest Training for 5 Random States

Table 1: Hypertuned Parameters for Each Model

Model	Parameters
Random Forest	`param_grid = { 'ml_algo__n_estimators': [100, 200, 500, 1000], 'ml_algo__max_features': ['auto', 'sqrt', 'log2'], 'ml_algo__criterion': ['gini', 'entropy'] }`
Logistic Regression w/ Penalty	`param_grid = { 'ml_algo__solver': ['saga', 'newton-cg', 'lbfgs', 'liblinear'], 'ml_algo__max_iter': [1000], 'ml_algo__C': [1e6, 1e5, 1e4, 1e3, 1e2] }`
Support Vector Machines	`param_grid = { "ml_algo__kernel":['poly', 'rbf', 'sigmoid'], 'ml_algo__C':[10, 1, 0.1, 00.1], 'ml_algo__gamma': [1,0.1,0.01,0.001] }`
Decision Trees	`param_grid = { "ml_algo__criterion":['gini', 'entropy'], 'ml_algo__max_features':['sqrt', 'log2'], 'ml_algo__max_depth': [5, 10, 15] }`

The final output of the function consisted of the best hyperparameters chosen by GridSearchCV along with the ROC score for each random state, which was 5. Finally, a final function was defined for each model trained that picked the highest occurring values for each of the parameters defined in the param grid.

4 Results

4.1 Model Evaluation

Over the five random states, the baseline models returned an average ROC AUC score of 0.72 with a standard deviation of 0.06 whereas the trained models returned an average ROC AUC score of 0.93 with a standard deviation of 0.02. The baseline model's accuracy was roughly 42 standard deviations below the average of the trained models whereas the trained models achieved an accuracy that is roughly 11 standard deviations above baseline.

As mentioned in the methods section, the modified function we created for each of the four models outputted the classification report along with the final ROC AUC score for the hyperparameters we picked during the training phase. The final model scores can be seen in Figure 5, shown below, along with the ROC Curve for the best model, Random Forest, and the worst model, Decision Tree.

	Model	ROC AUC Score
1	Random Forest	0.965550
0	Support Vector Machines	0.953004
2	Logistic Regression	0.946450
3	Decision Tree	0.857801

Figure 5: ML Model Performances

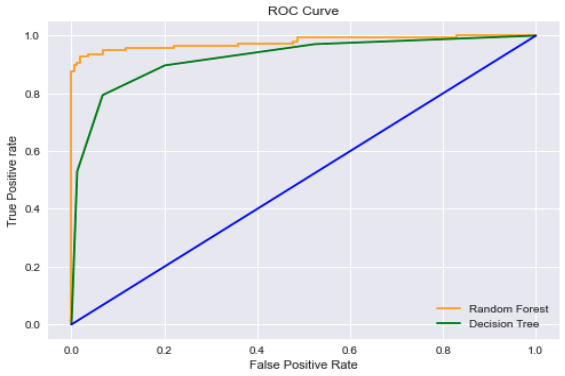

Figure 6: ROC Curve for RF and DT Models

4.2 Model Analysis

Global and Local Feature importance for the model was calculated using both the coefficients as well as the permutation test over 10 random shuffles and 34 random shuffles for each random state, respectively.

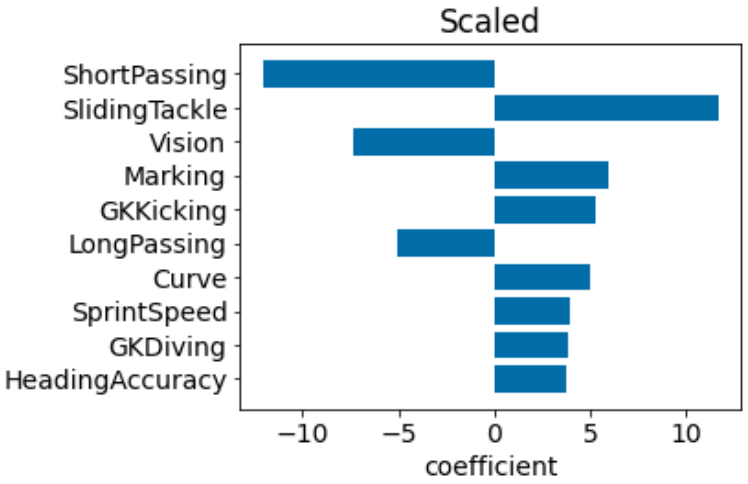

Figure 7: Top Average |Coefficient| Values

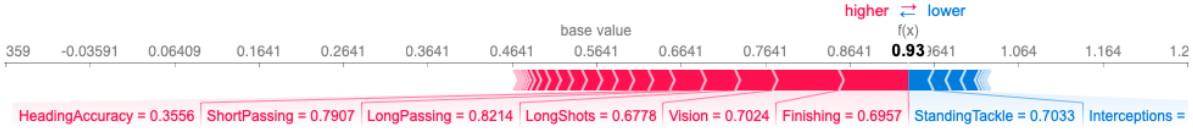

Figure 8: Local Feature Importance for Index = 1

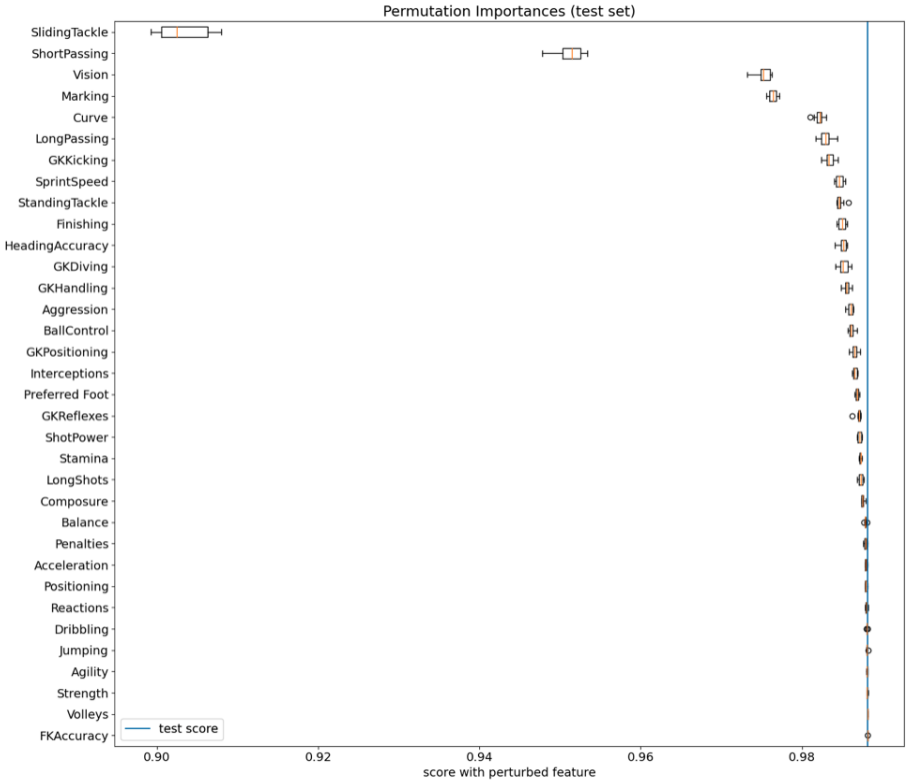

Figure 9: Top Average Permutation Importance Scores (Test Set)

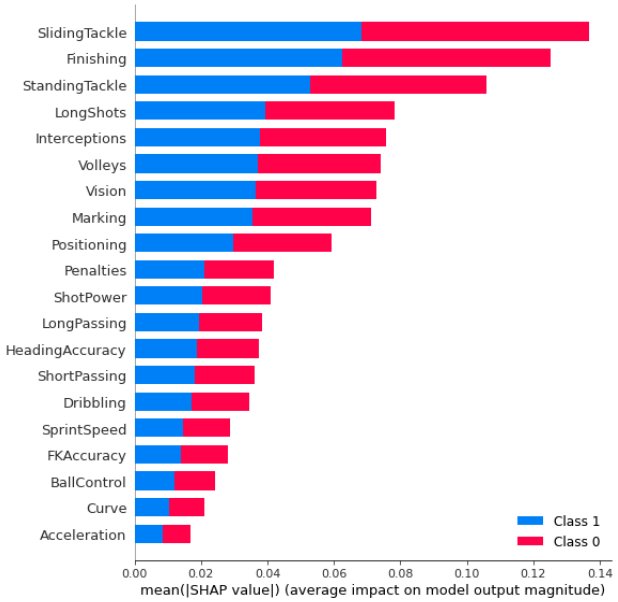

Figure 10: Summary of Mean SHAP Values

From the results, it can be seen that *Sliding Tackle*, *Finishing*, and *Vision* seem to be the most important features whereas *Volley*, *Curve*, and *GK Diving* seem to be the least important. The interpretations depicted above insinuate an accurate depiction of the two classes because features like *Slick Tackle* and *Interception* are crucial in the defensive positions (class 0). Similarly, features like *Finishing* and *Vision* are vital attributes in the offensive positions (class 1) because forwards and midfielders are the positions that predominantly score the goal and facilitate the plays during gameplay. It should also be noted that certain models were able to pinpoint on certain keywords such as *tackle* and *diving* which correspond to the defensive positions and thus, assign a high feature importance to it, which can be seen above.

5 Outlook

Given that the model is Random Forest, the results are very interpretable to the previous works done that used Logistic Regression and Support Vector Machines. To further improve the models, I would assign a higher feature importance weight to *Sprint Speed* and *Aggression* because they are statistically stronger parameters in determining which class they belong to. For example, *Sprint Speed* is a ubiquitous characteristic of the offensive position as it predominantly facilitates the playmaking whereas *Aggression* is a feature that is very prominent in the defensive positions because the objective is to obtain possession of the ball and clear it out of the 12-yard box as quickly as possible. In terms of model issues, the model marginally favours false negatives over false positives which ensures that the offensive positions are classified as class 0 simply because there exists more forward positions than defensive ones. We can improve this by considering an alternative scoring metric such as the f-score to ensure that the sensitivity of the defensive class is increased. In past works, when dealing with a binary classification problem with a balanced dataset, the ROC AUC is preferred and hence why the evaluation metric was chosen. Lastly, the offensive and defensive positions can be further split into “Forward”, “Midfielder”, “Defender”, and “Goalkeeper”, making it a multi-class problem and we can extend other features that provide a relation to the four classes. With new players coming into the FIFA league and the ratings constantly changing, prehistoric data that accurately reflects the prevalence of the two classes would significantly benefit this report.

6 References

[1] Timzatko, “timzatko/fifa-19-dataset-machine-learning: Player's value prediction and game position classification on FIFA 19 dataset.,” *GitHub*. [Online]. Available: https://github.com/timzatko/fifa-19-dataset-machine-learning. [Accessed: 12-Oct-2021].

[2] “Performing multi-class Classification on FIFA Dataset Using Keras,” *Analytics Vidhya*, 14-Jul-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/performing-multi-class-classification-on-fifa-dataset-using-keras/. [Accessed: 12-Oct-2021].

[3] P. D. Sadrach Pierre, “Exploratory Data Analysis of the FIFA 19 Dataset in Python,” *Medium*, 16-Sep-2020. [Online]. Available: https://towardsdatascience.com/exploratory-data-analysis-of-the-fifa-19-dataset-in-python-24eb27de9e59. [Accessed: 12-Oct-2021].

[4] Abhishek (ap1495), “Fifa 19 - Classification & Regression,” *Kaggle*, 29-Apr-2019. [Online]. Available: https://www.kaggle.com/ap1495/fifa-19-classification-regression. [Accessed: 12-Oct-2021].

[5] J. Brownlee, “Tune Hyperparameters for Classification Machine Learning Algorithms,” *Machine Learning Mastery*, 27-Aug-2020. [Online]. Available: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/. [Accessed: 07-Dec-2021].

[6] “AUC-ROC Curve in Machine Learning Clearly Explained,” *Analytics Vidhya*, 20-Jul-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/. [Accessed: 07-Dec-2021].