Expected Goal Analysis

Oriol Garrobé and Dávid Hrabovszki

28/08/2020

Abstract

The world is changing with the advance of technology and the vast amount of data available, and so does sport. Team sports such as football, are starting to introduce mathematical methods to analize performances, recognize trends and patterns and predict results. From this point, the purpose of this study is to develop a Machine Learning model that is capable to predict when a shot is going to end up being a goal.

To do so, many different mathematical approaches are trained on a dataset that tries to represent the exact moment of the shot as detailed as possible. Also, the dataset includes information about the players, their skills and the moment of the game. By doing so, a Logistic Regression algorithm is the one that yields the best performance when predicting goals, with an F1-score of 35.24%, improving the previous works on the field.

This study intends to provide new information and hidden insights to professionals - players, coaches or managers - in order to improve the game, looking for a better decision process when it comes to score a goal.

Introduction

Trying to play the best football possible in order to win all the titles is a quest that all the best teams in the world follow. Practicing methods have changed, and therefore football istelf. Because of the evolution of technology and the rapidly increasing amount of data - as done in other sports such as baseball - football is introducing datascience techniques to its toobox in order to improve the game.

Teams use data science and machine learning approaches to analyse which aspects of the games can be improved to become the best team. Artificial intelligence (AI) in sports is very relevant nowadays, and it changed the way coaches approach their practice sessions.

From this point of view, a large number of researchers have worked on sports analytics and in particular in football trying to build the state of the art models to find the hidden insights from the sport that the human eye cannot see by analysing data.

This project aims to study the probability of scoring of a player when shooting, so professionals can use this approach to look for better positions of shooting - those situations when a player has a big chance of scoring - and avoid those that apparently look good but they have a lower chance of goal.

As mentioned, there is a bast number of papers regarding football, but there is still room fore more. There are so many aspects of the game that can be analysed, and the game can be heavily improved using AI. From this regards, this project is based in a metric from StatsBomb called expected Goal (Larrousse 2019). The aim is to work over this metric and try to improve it by adding some other variables and try different machine learning models that could give a better result.

The data used is the datased provided by StatsBomb itself [reference this shit]. Apart from the StatsBomb dataset, aiming to go one step further, ratings from the players in the game are used, since it is believed that are players that in the same circumstances achieve different results. This means that not only the scenario is considered in this project but also the actors. For this purpose player ratings are included to the dataset coming from the FIFA video game.

A more sophisticated approach to analyse the shot situations in a football game would clearly help practitioners to design more technical interventions and strength and conditioning programmes for players. Accordingly, it is the purpose of this study to develop and validate a machine learning algorithm to identify the best shooting situations for players.

Related work

Methods

Dataset

The project is mainly based on a dataset from StatsBomb Academy (Academy, n.d.). This is a free dataset provided in order to new research projects in football analytics. The data from statsBomb includes very detailed and interesting features relevant for the project such as: location of the players on the pitch in any shot - including he position and actions of the Goalkeeper-, detailed information on defensive players applying pressure on the player in possession, or which foot the player on possession uses among others.

The data is provided as JSON files exported from the StatsBomb Data API, in the following structure:

- Competition and seasons stored in competitions.json.
- Matches for each competition and season, stored in matches. Each folder within is named for a competition ID, each file is named for a season ID within that competition.
- Events for each match, stored in events. The file is named for a match ID.
- Lineups for each match, stored in lineups. The file is named for a match ID.

In particular, the dataset only contains information about F.C. Barcelona games in the spanish national championship La Liga from 2005 up until 2019.

As stated earlier, player skills are also considered in this project. To have an unbiased rate of players, ratings from the most played around the world football video game FIFA from EA Sports (Arts, n.d.) are used. Also, it is considered which is the preferred foot of the player, information sourced from the same database. More in particular, this players information is gotten from *FIFAindex* (FIFAindex, n.d.).

Data Preprocessing

Data from StatsBombs comes in JSON format. One easy way to work with JSON data in R is through the **jsonlite** package that transforms JSON data - which is a combination of nested lists - to nested dataframes.

The StatsBomb dataset containing all the relevant information for the project is the **Events** dataset. Each data point in this data set is an event that occurred in a specific game, such as: pass, block, dribble or shots, among others.

The first step working with the dataset is to erase all the incomplete datapoints or those with wrong information that would afterwards make the models fail. From this point, and because this project is only focused on shots, the dataset is filtered by the variable **type** which contains the type of action of the event. This variable is filtered so only the events regarding shots remain. A **shots** dataset, way lighter than the one used before is created.

It is important to emphasize on the fact that this dataset contains an extremely useful information which is enclosed in the variable called **shot.freeze_frame**. This **shot.freeze_frame** states where all the players are positioned on the pitch at the moment of the event. It is considered that relevant since it allows to make a perfect picture of the scenario at the moment of the shot.

Table 1: Variables from the dataset defined.

Variable	Definition
id	Numeric. Number representing a unique player.
strong__foot	Boolean. States whether the player use the strong foot or not.
overall__rating	Numeric. Overall rating from FIFA ratings.
shot.power	Numeric. Shot power from FIFA ratings.
shot.finishing	Numeric. Shot finishing from FIFA ratings.
gk__rating	Numeric. Overall rating for goalkeepers from FIFA ratings.
gk.reflexes	Numeric. Goalkeeper reflexes from FIFA ratings.
gk.rushing	Numeric. Goalkeeper rushing from FIFA ratings.
gk.handling	Numeric. Goalkeeper handling from FIFA ratings.
gk.positioning	Numeric. Goalkeeper positioning from FIFA ratings.
shot.first__time	Boolean. States if the shot is the first of the game for the given player.
under__pressure	Boolean. States if the player shooting has opponents close enough to disturb the shot.
home	Boolean. States whether the player shooting is playing home or away.
dist	Numeric. distance to center of the goal from shot taker.
angle	Numeric. angle of the goal from the from shot taker (in degrees).
obstacles	Numeric. number of players (teammates & opponents NOT including GK) between goal and shot ta
pressure__prox	Numeric. Distance from closest opponent to shooter.
pressure__block	Boolean. Can the goalkeeper save the shot by being inside the triangle.
gk__obstacle	Boolean. States whether the goalkeeper is between the ball and the goal.
gk__pos	Numeric. goalkeeper's positioning, best if gk is standing on the line that halves the angle of the sho
gk__pos__adjusted	Numeric. same as gk__pos, but it is less strict with shots with a tight angle.
gk__dist__from__player	Numeric. distance between goalkeeper and shot taker.
gk__dist__from__goal	Numeric. distance between goalkeeper and the center of the goal.
goal	Binary. 1 the shot is goal, 0 the shot is not goal.

From this regards, and using the information in **shot.freeze__frame**, a number of new features can be computed and added to the dataset. More in particular, geometric features regarding the position of the striker, the defenders and the goalkeeper are computed. Such features will allow to know, for instance, which is the distance to target of the ball, whether the goalkeeper is properly positioned or if there are defenders close enough to disturb the striker.

Finally, every data point from the *shots* dataset, containing also the geometric features computed, is complemented with information of the players. For instance, the ratings of the striker and the goalkeeper in action are added or whether the player shooting is using his preferred foot or not.

Finally, some features from the **shots** dataset which are not relevant to study are removed, creating a dataset for analysis called **data** that will be the one used to train the models. The final features in the **data** dataset are:

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

Feature engineering

We created new features based on player locations for every shot in the dataset, similar to traditional xG metrics. The difference is that we calculated many more variables that the traditional approaches do not take into account. The statsbomb dataset contains the x and y coordinates of the shooter and other players that are relevant to the shot (both teammates and opponents including the goalkeeper). In the next part we explain what features we created and how.

Distance

Euclidean distance between the shooter and the center of the goal.

Angle

Angle of the goal from the shooter's point of view in degrees. Picturing a triangle, where one side is the goal line and the other two are the imaginary lines connecting each goalpost to the shooter, we need the angle opposite of the goal line. We used the following formula:

$$\alpha = \arccos\left(\frac{b^2 + c^2 - a^2}{2 \cdot b \cdot c}\right) \cdot \frac{180}{\pi}$$

where a is the goal line, and b and c are the imaginary lines between the shooter and the posts (Calculator, n.d.).

Obstacles

Number of players (teammates and opponents not including the opponent goalkeeper) between the goal and shooter – in other words, inside the triangle defined by the shooter and the goalposts. To calculate this, we first need to evaluate if a player is inside said triangle or not. If the area of this triangle is equal to the sum of the partial triangles defined by:

1. The shooter, one goalpost and the player being evaluated
2. The shooter, the other goalpost and the player being evaluated
3. The two goalposts and the player being evaluated

Then we conclude that the player being evaluated is inside the main triangle, therefore he is an obstacle. To calculate the area of a triangle based on the coordinates of its points, we used the following formula in R:

$$area = \left|\frac{a_1 \cdot (b_2 - c_2) + b_1 \cdot (c_2 - a_2) + c_1 \cdot (a_2 - b_2)}{2}\right|$$

where a, b and c are numeric vectors of length 2 representing the points of the triangle (Geeks, n.d.).

Pressure proximity

The Euclidean distance between the shooter and the opponent closest to him.

Pressure block

The closest opponent's physical ability to block the shot by being inside the triangle defined by the shooter and the goalposts. Boolean value.

Goalkeeper obstacle

The opponent goalkeeper's physical ability to save the shot by being inside the triangle defined by the shooter and the goalposts. Boolean value.

Goalkeeper positioning

Positioning of the opponent goalkeeper. The value is between 0 and 1, where a value of 1 means that the goalkeeper halves the angle of the shot, while a value of 0 means that the angle remains the same. This does not take into account if the goalkeeper is standing on the line or right in front of the shooter, only that he is positioned on the line that halves the angle of the shot. A larger kernel width means that the kernel distinguishes less between bad and good goalkeeper positioning, while a small value means that only split angles close to 0.5 can get a good mark for goalkeeper positioning, as it can be observed on Figure 1. We used a kernel width of 0.2 for this feature.

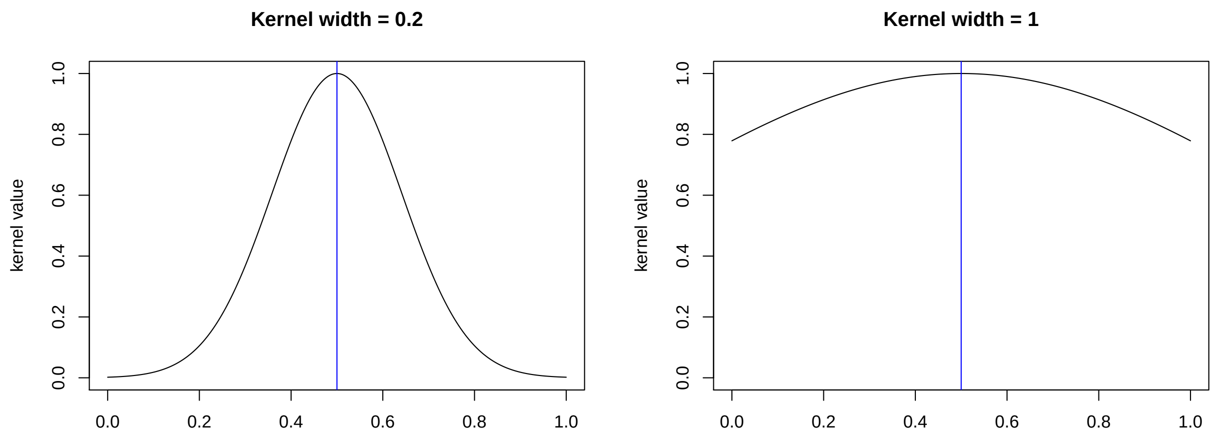

Figure 1: Gaussian kernels that give more weight to values around 0.5 (good split angle by the goalkeeper). Kernel width controls how strict the kernel is.

This feature was not included in the analysis, because it proved to be too strict when dealing with tighter shots.

Goalkeeper positioning adjusted

Very similar to Goalkeeper positioning, but this feature is adjusted with the shot angle, meaning that it takes into account the full shot angle when evaluating goalkeeper positioning. This is necessary, because the previous variable was too strict when it came to tight shots, and it gave a bad value for the goalkeeper, even though he was still positioned pretty well. For example, if the angle of the shot was 5 degrees, the goalkeeper should not be given a bad mark for his positioning if he splits the angle to 1 and 4 degrees. With such tight angles, it does not really matter where he stands, as long as he is obstructing the goal, of course.

We solved this problem by introducing another Gaussian kernel that gives a high output value for small input values (shot angles) and outputs a value close to 0.2 for larger input values (shot angles). The kernel width in this case was chosen to be 20. Then, this kernel value was used as the kernel width for the original Gaussian kernel described above in the previous feature. This way we achieved that the Goalkeeper positioning adjusted feature is more lenient towards tight shots than Goalkeeper positioning, thus more accurate in evaluating real life goalkeeper positioning.

Goalkeeper distance from player

Euclidean distance between shooter and goalkeeper.

Goalkeeper distance from goal

Euclidean distance between center of the goal and goalkeeper.

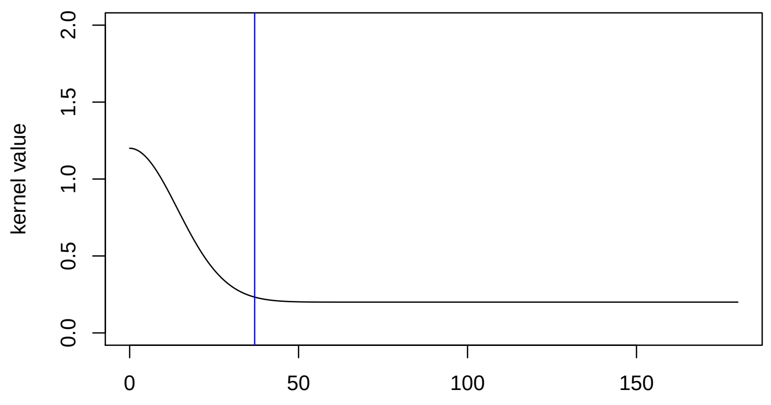

Figure 2: Adjusting Gaussian kernel that gives more weight to lower values (shot angles)

Machine Learning Models

Logistic Regression

From the moment that one tries to develop a classification algortihm, the first choice usually is the Logistic Regression. This study is no different.

The Logistic Regression model it is based on the logistic function (sigmoid) which is an S-shaped curve that takes any real number and maps it between 0 and 1, but never reaching this numbers. From this point, by setting a threshold, also between 0 and 1, it is possible to divide data points in two classes. For instance, if the threshold is set at 0.7, those data points with a probability higher than 0.7 will be classified as one class and those with a probability lower than 0.7 will be classified as the other. The regression coefficients (betas) of the Logistic Regression are estimated by training the data.

In particular, the threshold set for this study a threshold of 0.5. In order to run the Logistic Regression algorithm, the package **glm** (Schlegel 2019) from CRAN is used.

Random Forest and AdaBoost

Both Random Forest and AdaBoost models are very good choices when training a classification model. This models are built on the decision tree model.

The Random Forest model in particular generates a number of trees based on a bootstrapped subset of the sample and only considering a subset of variables at each step. The result is a wide variety of trees. From this point the data is run over all trees and that yield a decision, the decision given by more trees is the one that prevails. This is called Bagging. To run the Random Forest model, the package **randomForest** (Breiman and Wiener 2018) from CRAN is used.

AdaBoost on the other side also generates a number of trees, but only trees that have a root node and two leaves with no children, this trees are called stumps and are not great at making accurate classification, they are weak learners. From this point, once one stump is made the data is run through it, and a decision is made. The errors made by this stump influence when creating the next one and so on. Therefore, some stumps have a greater impact to the model. To run the AdaBoost model, the package **mboost** (Hofner 2020) from CRAN is used.

In order to get the best model possible and not to make it too computational expensive, in both models it is important to choose the optimal number of trees. To do so, both algorithms are run a number of times with different amount of trees. Based on the error rates, the best model in is chosen, and afterwords trained with the data to get the best results possible. In figure 3 it can be seen the error rates against the number of trees. The smallest error rate for training data in the Random Forest model appears when 40 trees are created, and for the AdaBoost model it appears when 30 trees are used. Therefore, the final models are set.

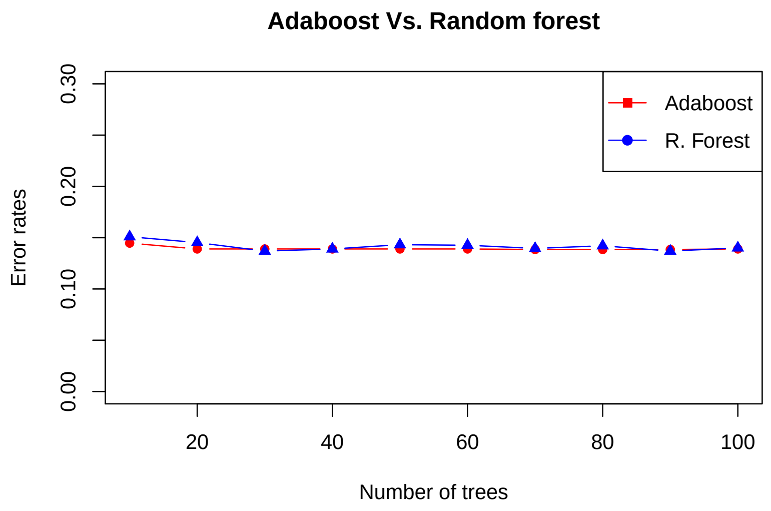

Figure 3: Random Forest and AdaBoost models error rates against the number of trees used.

KNN model

The KNN (K-Nearest Neighbours) model is a classification model that identifies the number k of closest labelled points to the one studying and estimates its class. The class chosen is the one with the bigger amount of neighbors with said class. This is a powerful algorithm but it is very important to choose the optimal number of neighbours, in other words, choose the value of k. This value is chosen with Cross Validation (CV), which is a model validation which partitions the data in complementary subsets, performs analysis on one subsets and validates on the others in order to give a more accurate model. The final number of neighbours used in the KNN algorithm is 9, which gives the best result for prediction. To run the KNN algorithm with CV, the package **kknn** from CRAN is used.

Results

- **Accuracy**. Stands for the ratio of correctly predicted observation over the total observations. If it is high it means that there are a lot of good predictions but it must be taken into account that for assymetric datasets or uneven classes - those where one class is larger than the other - it can be biased. In particular, in the dataset used there are way less shots that ended up being a goal than those that not. So, with a naive predictor all the datapoints could have been set to not goal/miss and the accuracy would have been still high. That is other metrics must be used in order to take this into account.
- **Recall**. Otherwise called sensitivity, stands for the correctly predicted positive observations to all observations in an actual class. This metric checks how many properly predicted points are predicted within a class, in this case **goal**. This means that it tries to show how many predicted goals are actually goals. Since in football there are not many goals, it is a very innacurate metric, as many times a situation that should clearly end up in goal in the end is not. This is why this one is not a very indicative metric for the purposes or the project. However, it is good to compute it in order to get some conclusions.

- **F1-score**. This metric is a weighted average of precision and recall, being precision the number of properly predicted observations in a class over all the observations fom that class. From this point, it takes both false positives and false negatives into account too, it is very useful when the goal is to predict uneven classes. In this case, as there are way more observations that are not goal than those that are goal, this metric is the one that mirrors best the quality of the algorithm.

Discusion

In this study, a Machine Learning algorithm to quantify the probability of a football player scoring is develooped, using the positioning of the players in the pitch and their skills. 4 classification approaches using 25 variables which derived from the particular scenario at the moment of the shot were examined. It has been proven that it is possible to classify whether a shot is going to be a goal or not with good accuracy, being the best performing method the logistic regression with an F-1 score of 35.24%, a recall of 23.72% and an accuracy of 85.89%. It also has improved the xG metric from statsBomb by adding player skills to the dataset and testing other Machine Learning approaches. This will provide professional of the sport such as players, coaches or managers some insights from the game that can be very useful to improve their performance.

Models based on decision trees such as Random Forest or AdaBoost provided mixed results that did not improve the previous works on the field. In particular, the optimal Random Forest algorithm with 40 trees yielded an F1-score of 32.24%, a recall of 22.05% and an accuracy of 60%. Looking at this results, it can be seen that eventhough the F1-Score is close to the one from xG, the overall accuracy is poorer, which means that many shots that were classified as miss where actually a goal. On the other hand, the optimal AdaBoost algorithm with 30 trees yielded an F1-score of 19.71%, a recall of 11.18% and an accuracy of 83.33%. All the results are worst than the xG metric, bringing no improvement to the field. This model failed as almost no goals were predicted, only 11.18% (recall) of the goals were predicted properly. From this point of view, these two algorithms do not bring improvements to the field or relevant information that can be used to improve the game.

Another model used to classfy goals is the K-NN algorithm. This algorithm yielded better results than models based on decision trees with an F1-score of 39.54%, a recall of 27.79% and an accuracy of 68.50%. It has the best F1-Score of all models used and also improves the xG metric. As stated before, the F1-score is the metric that mirrors best the quality of this algorithm, and this method could be the one chosen for the project. It also has the best recall among all the models. However, it has a relevant lower overall accuracy compared to xG or Logistic Regression. It yields a better F1-score and a better recall due the fact that predicts many more situations as goals than Random Forest, Adaboost or xG. However, many situations that were not goal were classified as goal, this is why this model has a lot of room to improve.

Finally, ### HERE WRITE ABOUT WHY WE CHOOSE LOGISTIC REGRESSION Notes: could have lowered threshold below 50% to get more predicted goals, but it doesn't make sense, and lowers accuracy (check if it lowers F1 score)

Player performances

In the previous sections we created models to predict goals, chose the best one, and now we are going to interpret the results on the player's level. We aim to find out how well each player performs from the aspect of converting shots to goals. We suspect that there are players that score more goals than they are expected to, and there are some that score fewer. It is important to note, that the database we used for analysis had 1044 goals, while we only predicted 320 goals with our best model for the whole dataset. Therefore, to make the amount of actual and predicted goals by each player comparable, we scaled up the amount of predicted goals for the sake of this performance analysis, so they also sum up to 1044.

The results can be observed in the table below, which is ordered by the amount of goals scored. It is no surprise that Messi scored 33.85 more goals (11% more) than he should have, based on his chances. This is in line with our belief that he is a very efficient striker. Even more efficient than Messi are Ivan Rakitic and Daniel Alves. The latter player is primarily a defender, but still managed to score 11 goals, even though none were predicted for him.

Table 2: Player performances (who scored more than 10 goals)

Name	Goals predicted (scaled up)	Goals scored	Diff. (amount)	Diff. (ratio)
Lionel Andrés Messi Cuccittini	300.15	334	33.85	0.11
Luis Alberto Suárez Díaz	133.76	106	-27.76	-0.21
Samuel Eto"o Fils	104.40	62	-42.40	-0.41
Pedro Eliezer Rodríguez Ledesma	55.46	48	-7.46	-0.13
Neymar da Silva Santos Junior	58.73	46	-12.73	-0.22
Thierry Henry	42.41	32	-10.41	-0.25
David Villa Sánchez	35.89	31	-4.89	-0.14
Xavier Hernández Creus	26.10	31	4.90	0.19
Alexis Alejandro Sánchez Sánchez	19.58	29	9.42	0.48
Andrés Iniesta Luján	22.84	25	2.16	0.09
Gerard Piqué Bernabéu	13.05	25	11.95	0.92
Ivan Rakitic	6.53	22	15.47	2.37
Francesc Fàbregas i Soler	13.05	20	6.95	0.53
Bojan Krkíc Pérez	9.79	19	9.21	0.94
Seydou Kéita	9.79	12	2.21	0.23
Zlatan Ibrahimovic	19.58	12	-7.58	-0.39
Daniel Alves da Silva	0.00	11	11.00	Inf
Ronaldo de Assis Moreira	6.53	11	4.47	0.69

The most underperforming players, when it comes to converting chances, were Samuel Eto'o and Zlatan Ibrahimovic. Luis Suárez also scored much fewer goals (27.76), than expected.

The next section will present some shots visually that might help explain these differences.

Visualising shots

Now we plot some of the shots to illustrate how such differences can occur between reality and prediction, and also to show some chances that had a high Scoring Probability, but were missed, or had a low probability, but still went in. We used the package **soccermatics** (Gallagher 2018) to plot the empty pitch, and then we wrote our own function that places the players and the shot itself on the pitch.

Figures 4 and 5 show two goals from Daniel Alves that he scored against the odds. These efforts contribute to him performing much better, than expected (see the table in the previous section).

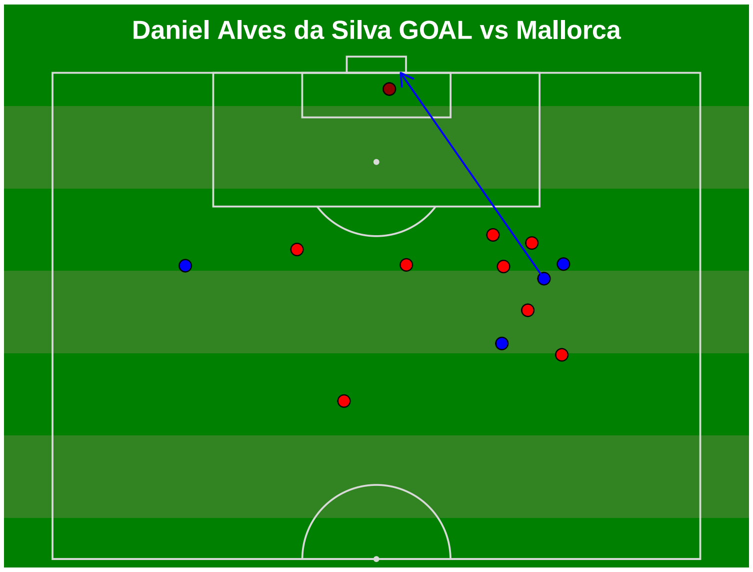

Figure 4: Daniel Alves goal (Scoring Probability = 0.029)

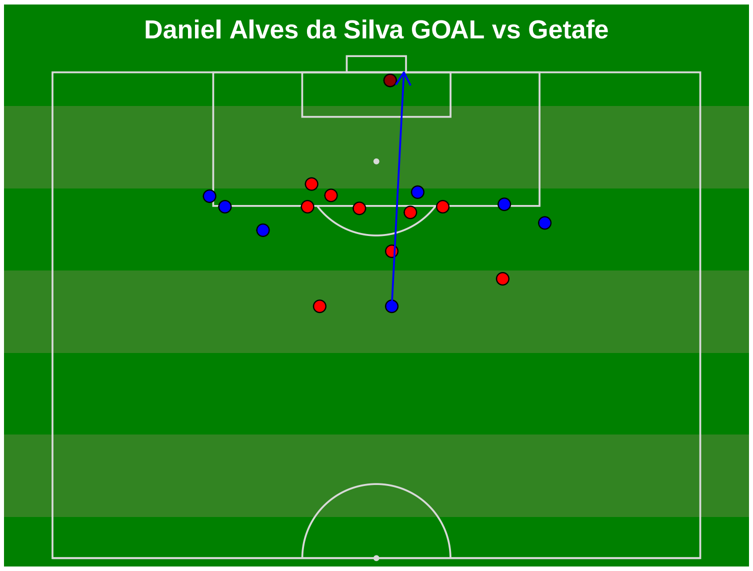

Figure 5: Daniel Alves goal (Scoring Probability = 0.038)

Other shots however, were not converted despite having a large Scoring Probability. Ludovic Giuly's header for example flew over the crossbar (Figure 6).

Samuel Eto'o failed to score from a relatively easy position (Figure 7) against Real Madrid.

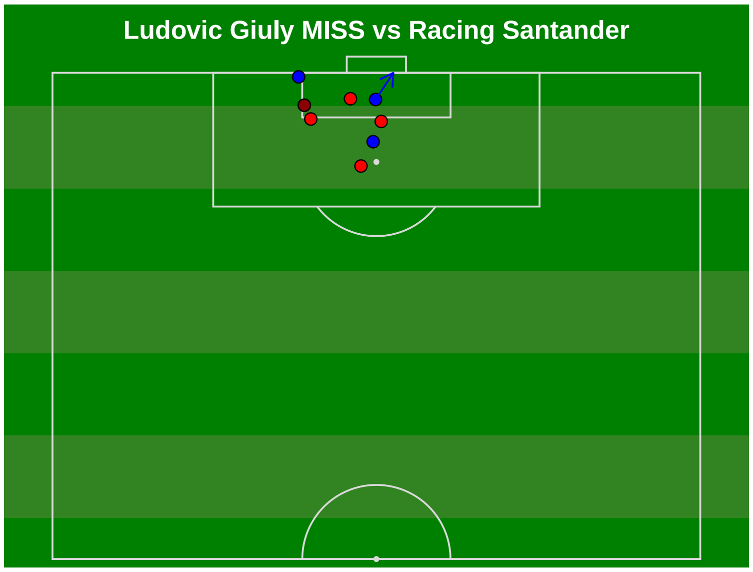

Figure 6: Ludovic Giuly miss (Scoring Probability = 0.886)

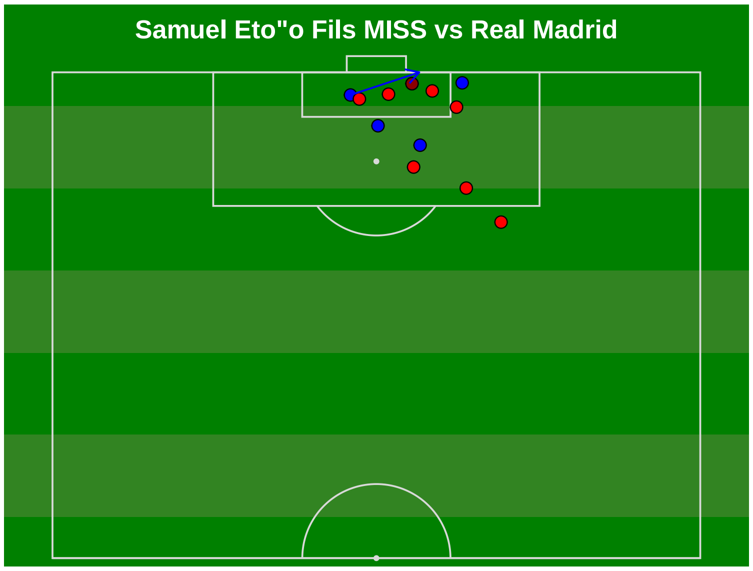

Figure 7: Samuel Eto’o miss (Scoring Probability = 0.823)

Finally a typical Lionel Messi goal against Levante, where he had less than 50% probability of scoring, but still managed to put the ball into the net.

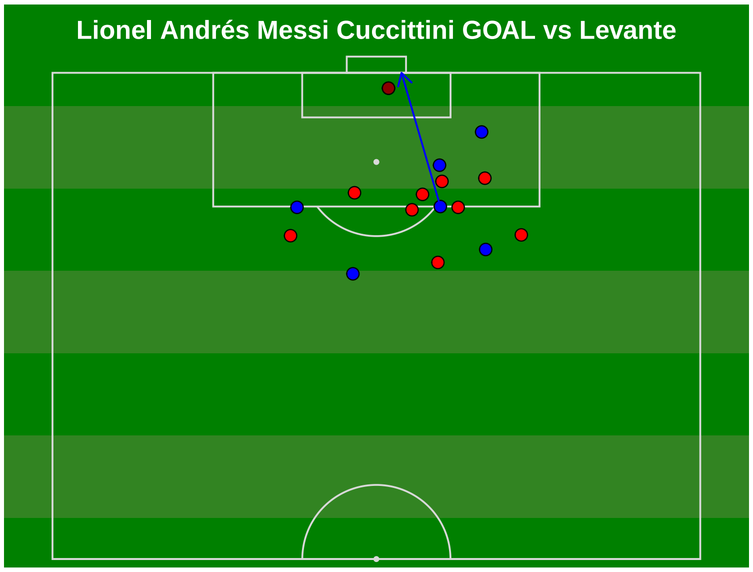

Figure 8: Lionel Messi goal (Scoring Probability = 0.120)

Future improvements

The limitations of this study must be acknowledged. The data sample consist only of **F.C. Barcelona** games in the national regular season **La Liga**. From this regards, Barcelona players - which are known to be very effective - playing against worse teams can lead to some biased results. It is also worth to be mentioned than as the dataset starts in 2005, many of the shots are done by **Leo Messi**, for many the best player in history and definetly one of the best strikers of all times. This can also lead to not very relevant results for the football comunity. This is why this project used player ratings, as an attempt to introduce this information to the models, that can be used therefore at all levels. It would be good therefore, to add more shot situations from many different competitions from different countries and different teams. This would give a more realistic dataset and also it would provide more data points that would definetly help train the algorithm, probably achieving better results.

Another possible improvement would be to create more variables. With the dataset provided the authors created a data set that comprised all those variables that could have an effect on the result of the shot. New variables were computed giving a proper dataset to work on. However, more variables could have been added or created. With more variables, a better picture of the situation of the shot is drawn. With this new dataset other Machine Learning approaches, more modern and complex could have been applied. For instance, would be good to train a Neural Net in order to predict goals. In order to do so, high computational resources are needed, that is why Neural Nets are not used in this studio, but could probably improve the results.

Another improvement to the study would be to apply the same methods to other types of events other than **shots**, such as success in **passes** or **tackles**. This would provide information not only to the strikers but to every player on the field, improving - as this is the final aim of Football Analysis - the game.

Conclusion

Football is a very demanding sport, and the high impact that it has in society makes teams strive for perfection. From this point, game demands are extensively analysed and new concepts coming from data science are introduced to coaches plans. This projects created a classifier that provides more information regarding the shooting positions of players. It intends to help players choose wiser when shooting, or in the other hand to advise them not to shoot when the chances are very low and a pass could create a better situation. Also it provides information, as in this project not only the situation but the players involved are studied, about which palyers should take which shots. It is commonly known that best players take more shots, but there are situations where a player which apparently is not that good is the best fit for that particular shot in that particular situation.

The algorithm, built on a simple logistic regression and only 25 relevant features classified the Scoring Probability with a combined precision and recall of 35.24% (F1-score), a recall of 23.72% and an accuracy of 85.89% improving the results of the xG study from statsBomb. This study tries to improve the football analysis comunity and football itself providing a tool to assess shooting scenarios in professional football.

References

Academy, StatsBomb. n.d. “StatsBomb Open Dataset.” https://statsbomb.com/academy/.

Arts, Electronic. n.d. “FIFA.” https://www.ea.com/en-gb/games/fifa/fifa-21.

Breiman, Cutler, Leo, and Matthew Wiener. 2018. “Breiman and Cutler’s Random Forests for Classification and Regression.” *CRAN*.

Calculator, Omni. n.d. “How to Find the Angle of a Triangle.” https://www.omnicalculator.com/math/triangle-angle#how-to-find-the-angle-of-a-triangle.

FIFAindex. n.d. “Player Stats Database.” https://www.fifaindex.com/players/top/.

Gallagher, Joe. 2018. “Visualise Spatial Data from Soccer Matches.” *CRAN*.

Geeks, Geeks for. n.d. “Check Whether a Given Point Lies Inside a Triangle or Not.” https://www.geeksforgeeks.org/check-whether-a-given-point-lies-inside-a-triangle-or-not/.

Hofner, Benjamin. 2020. “Model-Based Boosting.” *CRAN*.

Larrousse, Benjamin. 2019. “Improving Decision Making for Shots.” *StatsBomb*.

Schlegel, Benjamin. 2019. “Predicted Values and Discrete Changes for Glm.” *CRAN*.