UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

SPORTS ANALYTICS – POSTGRADUATE

# World Cup Post Shot Expected Goals (PSxG):

## Evaluation of performance and decision-making of the players

Universitat Politècnica de Catalunya

Miquel Camí Rodríguez
Pau Puertas Ametller

Academic course 2022/23

# Index

# 1. Introduction & Objectives

The objectives of this project are to evaluate the players' decision-making at the moment of the shot to goal and to appraise the performance of players and goalkeepers when making shots/saves.

To be able to assess these two objectives we created a Post-Shot Expected Goal (*PSxG*) metric that will return the probability to score a goal once we know the position of the player making the shot, the position of the goalkeeper and the destination of the shot.

We trained a data model with the events data of the World Cup (2018/2022) provided by StatsBomb, where the output is the *PSxG*, giving the position of the player and goalkeeper and the destination of the shot as an input.

To visualize this data, we created a heatmap colored by the probability to score a goal, representing a matrix (3 rows x 5 columns) inside the goal.

In addition, we created two additional charts combined with buttons and sliders to be able to interact with the heatmap. This interactivity allow us to change the player and goalkeeper position, updating the results dynamically.

# 2. Definition of Post-Shot Expected Goals (PSxG)

Post-Shot Expected Goals (*PSxG*) is a statistical metric used in soccer to evaluate the quality of a shot taken by a player, by taking into account the location of the shot and the position of the goalkeeper at the time of the shot. *PSxG* is an extension of the more general Expected Goals (*xG*) metric, which only takes into account pre-shot variables such as shot location and type.

To calculate the *PSxG* value of a shot, a statistical model is used to consider the location of the shot and the position of the goalkeeper at the time of the shot. The model assigns a probability to each shot, based on the likelihood of the shot resulting in a goal given the location and direction of the shot, as well as the position of the goalkeeper. The sum of these probabilities is the *PSxG* value.

*PSxG* can be used to evaluate the performance of individual players, as well as teams. It can be used to identify which players are creating the most high-quality scoring opportunities, and which areas of the pitch are the most effective for generating goal-scoring chances.

In addition, with this metric, we cannot only evaluate the performance of the players' shots, but also the goalkeepers' saves. This calculation can be made through inverting the probability of the shot.

Overall, Post-Shot Expected Goals is a valuable tool for soccer analysts and coaches, as it provides a more comprehensive measure of shot/saves quality that takes into account both pre-shot and post-shot variables.

# 3. Data Collection

## 3.1. Data Extraction

For our project we required shots event data, which essentially allow us to know the location of the player at the moment of kicking the ball, the direction of the ball (where the ball will go), and the position of the goalkeeper. If the events contain additional data, they can be used additionally to train the model.

After a search for this type of data, we found that StatsBomb provides this kind of data for many competitions, and that it maintains some databases freely accessible to all audiences.

The free data on this platform that fitted our expectations has been the events corresponding to the matches of the World Cup for men in 2018 and 2022. Being these two tournaments followed closely by both of us and having all the essential features to carry out our project on *PSxG*. We selected this data because we believe that is the best for this project.

If we merge the shots events data from 2018 and 2022, we obtain a total of 3,200 kicks with 39 different features. Furthermore, there is a goalkeeper-type event associated with each shot, which provides us with the goalkeeper's location at the moment of the shot, along with several other characteristics. We would like to have more shots data, but we are limited to choosing free data that does not carry any payments.

Related events such as passes (pre-shot pass), drives and crosses can also be extracted. In previous versions of our project we incorporated them into our dataset, but since our intention is to make a generalized model that allows us to calculate the goal probability without overfitting the data, we will not take this type of data into account.

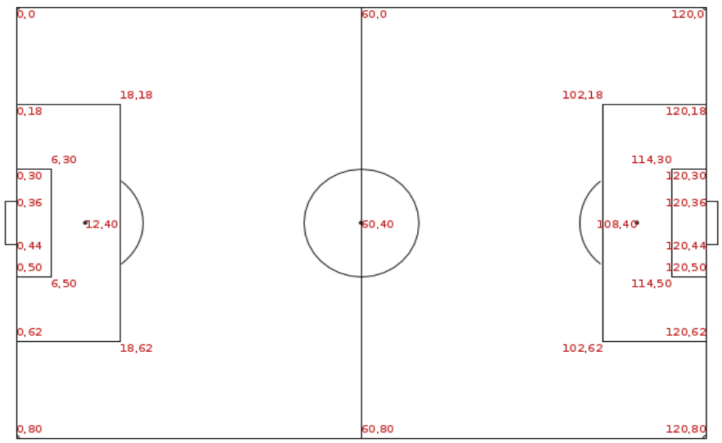
We attach in the bibliography a link[1] with a document provided by StatsBomb detailing all the features contained in the databases, as well as an explanation of them and the possible values they present:

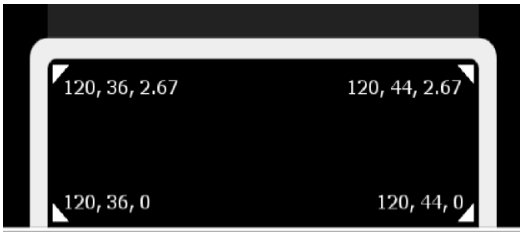
In addition, the data that we have taken into account for the following steps of the project is shown in the table below:

| **Feature** | **Description** | **Values** |
|---|---|---|
| *team* | Id / Name of the team of the event. | e.g. Argentina |
| *player* | Id / Name of the player of the event. | e.g. Lionel Messi |
| *location* | Array containing two integer values. These are the x and y coordinates of the event. | e.g. the center of the field is (60,40) |
| *related_events* | A comma separated list of the Ids of related events. For example, a shot might be related to the Goalkeeper event, and a Block Event.<br><br>The corresponding events will have the Id of the shot in their *related_events* column. | e.g. ("2b7d06c7-9bcb-4bbf-a6e5-08e54e1303ac",<br><br>"03b001b6-bf44-4691-ae47-e279f5a9b38c") |

| | | |
|---|---|---|
| *shot_statsbomb_xg* | The StatsBomb expected goals value calculated for the shot. | e.g. 0.15 |
| *shot_end_location* | Array of [x,y] or [x,y,z] with shot location in the goal where z is the height from the ground. | e.g., (120, 50) or (120, 32.5, 1.2) |
| *shot_outcome* | Id / Name for the attribute option specifying the outcome of the shot. | Blocked, Goal, Off T, Post, Saved, WayWard, Saved Off T, Saved To Post |
| *shot_body_part* | ID / Name of the body part has been used to shoot. | Head, Left foot, Other, Right Food |
| *shot_open_goal* | Added if the shot was taken with an open goal. | TRUE/FALSE |
| *shot_deflected* | Added if the shot was deflected. | TRUE/FALSE |
| *shot_redirect* | Added if the shot was redirected. | TRUE/FALSE |
| *shot_type* | Id / Name for the attribute option specifying the type of shot. | Corner, Free Kick, Open Play, Penalty, Kick Off |
| *gk* | Id / Name of the Goalkeeper of the event. | e.g. Keylor Navas |
| *goalkeeper_position* | Array containing two integer values. These are the x and y coordinates of the event. | e.g., [4.0, 40.0] when the gk is 4m in front of the line and in the center of the goal. |

We have also taken into account the dimensions of the field and the goal that corresponds to our data:



## 3.2. Data Cleaning

### 3.2.1. Goals

First, we have added a new Boolean column named *goal*, where 'true' will be stored when the column *shot_outcome* equals 'goal', and 'false' if it is different. We observed that from the 3.200 shots we had, 378 were goals and 2.822 were not. That means that 11.81% of the shots are goals. We considered this when making the model.

### 3.2.2. Isolated Game Situations

As we do not want isolated game situations, we eliminated the following shots from our database:

- Shot open goal
- Shot deflected
- Shot redirect
- Penalties
- Corners
- Free Kick
- Kick off

After this data cleaning there were 2.863 shots left.

### 3.2.3. Unrelated Features

We kept only the columns that had importance for our *PSxG* metric. That allowed us to create new features to help better train the model: *location, shot_end_location, shot_body_part,* and *goal*. The following image is a sample of five rows:

| | location | shot_end_location | shot_body_part | gk_location | goal |
|---|---|---|---|---|---|
| 0 | [98.0, 51.0] | [120.0, 36.2, 4.5] | Right Foot | [4.0, 41.0] | False |
| 1 | [88.0, 59.0] | [120.0, 46.7, 6.0] | Right Foot | [1.0, 40.0] | False |
| 2 | [107.0, 48.0] | [120.0, 34.4, 0.3] | Right Foot | [2.0, 38.0] | False |
| 3 | [82.0, 29.0] | [104.0, 39.0] | Left Foot | [2.0, 40.0] | False |
| 4 | [106.0, 25.0] | [120.0, 33.9, 0.2] | Head | [2.0, 43.0] | False |

### 3.2.4. Location Coordinates

What we did is separate by coordinates (x, y, [z]) each of the location features: *location, shot_end_location* and *gk_location*. Thereby, we could work with them more easily.

In the case of *shot_end_location*, which consists of the coordinates x, y and z, we deleted from our dataset all those shots that do not have the height of the shot (z) defined. Since it is very relevant information to take into account in our model, we are not including the shots without z informed to train it.

Finally, we removed the rows that had the same *shot_location_x* and *end_location_x*. The result of this was a total of 1,862 shots. The following image is a sample of five rows:

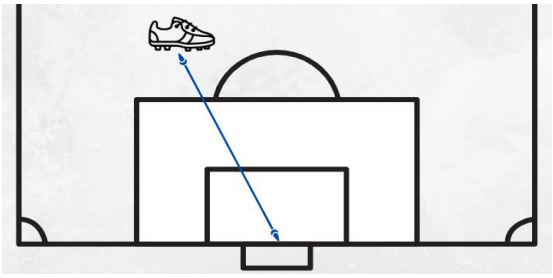
| | shot_body_part | goal | shot_location_x | shot_location_y | end_location_x | end_location_y | end_location_z | gk_location_x | gk_location_y |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Right Foot | False | 98.0 | 51.0 | 120.0 | 36.2 | 4.5 | 4.0 | 41.0 |
| 1 | Right Foot | False | 88.0 | 59.0 | 120.0 | 46.7 | 6.0 | 1.0 | 40.0 |
| 2 | Right Foot | False | 107.0 | 48.0 | 120.0 | 34.4 | 0.3 | 2.0 | 38.0 |
| 4 | Head | False | 106.0 | 25.0 | 120.0 | 33.9 | 0.2 | 2.0 | 43.0 |
| 6 | Left Foot | False | 97.0 | 20.0 | 120.0 | 45.6 | 0.2 | 4.0 | 45.0 |

## 3.3. Feature Engine

Afterwards, we created some features that allowed our model to learn correctly according to the shots parameters in a more general way.

### 3.3.1. Distance to Goal

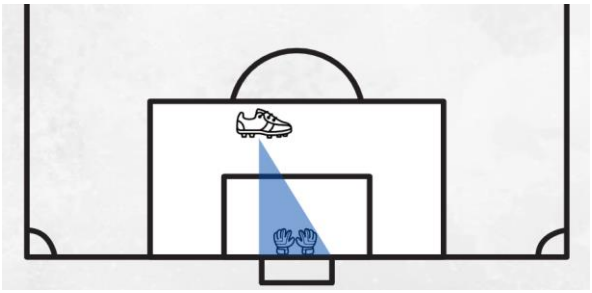
The first feature that we created was the distance to the goal at the time of the shot. To do so, we calculated the distance between the *shot_location* point and the center of the goal (120.0, 40.0).



### 3.3.2. Angle to Goal

Another Feature is the angle available to the player when shooting. It is calculated, taking into account the *shot_location* and the points corresponding to the two posts of the goal (120, 36) and (120.0, 44.0).

The closer and more centered the player is to the goal, the greater the angle will be. Conversely, the farther away or closer to the corners the player is located, the smaller the angle will be.

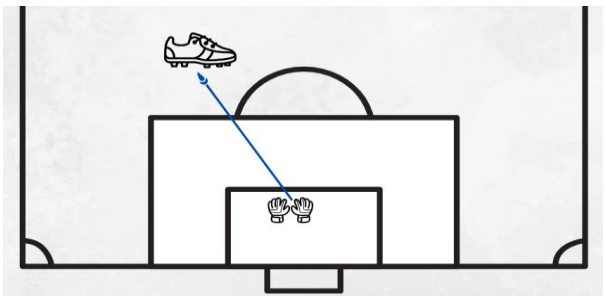


Example values to understand the angle to goal:

| **Player Location** | **Angle To Goal** |
|---|---|
| (90, 40) | 0.3 |
| (110, 40) | 0.8 |
| (100, 30) | 0.3 |
| (110, 70) | 0.1 |

### 3.3.3. Distance to Goalkeeper

We are also interested in registering the distance between the player and the goalkeeper. We will be able to calculate this feature by the distance between the *shot_location* and the *gk_location*.



### 3.3.4. Project End Location to the Goal Line

One of the problems with the *end_location* is that in many cases the *end_location_x* is not 120, because the destination of the shot is stopped before the goal line. This can happen if the goalkeeper or a defender blocks the kick before it reaches the line.

We needed to solve this problem, otherwise all the shots that do not reach x=120 would be evaluated by our model as No Goal.

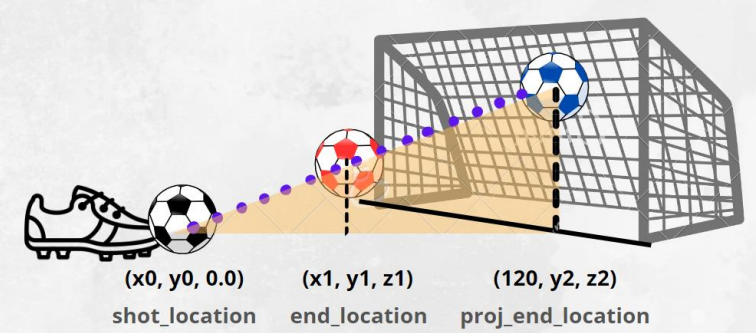
What we did for the shots that have *end_location_x* less than 120, is project the *end_location_y* and the *end_location_z*, trying to predict in which direction the shot was directed from the goal over the height of the goal line.

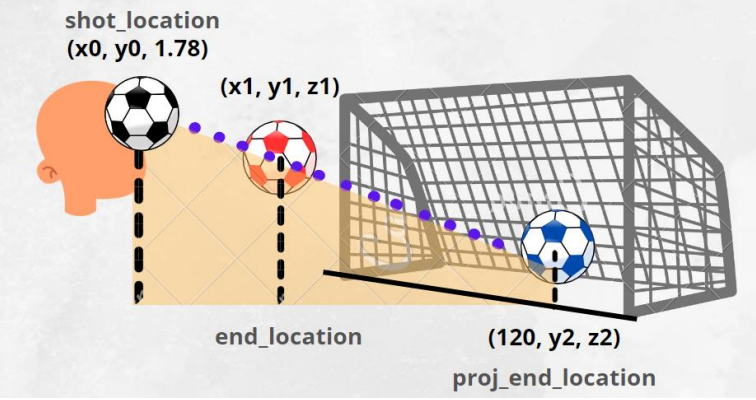
Since we did not have the exact information of the height from which the shot comes out, we assumed that:

- When the shot is made with the foot, the height from where the shot comes out will be 0m (*shot_location_x* = 0.0).
- When the shot is made with the head, the height from where the shot comes out will be 1.78m, which is approximately the average height of the players.

From here, knowing the origin of the shot and the point that we have registered as *end_location*, we projected a straight line up to x=120.0, to know in which coordinate the shot would be located above the goal line.

For all these cases, since we did not have more information on the effect and trajectory of the shot, the projection will be in all cases a straight line.



The function we have made also calls another function called *correct_rare_values*, to correct exceptional cases where the projected position may have unrealistic values:

Our *y_projected* can take values only between 18 and 62, which is the size of the area. If the shot was expected to go farther away, we register the value as 18 or 62.

Our *z_projected* cannot be less than 0.2, which we consider that is the minimum value possible when it is a ground ball.

Our *z_projected* can be a maximum of 8, as it is the maximum height we have specified.

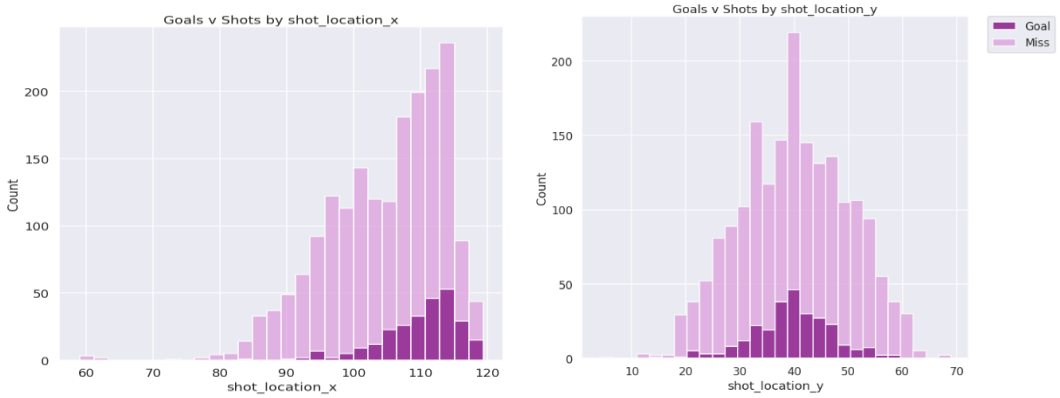
If the *end_location* has x=120, our function returns this position as the projected one.

## 4. Data Exploration

After deleting the columns that we have used to create our features, and we no longer need, we are left with the following dataset:

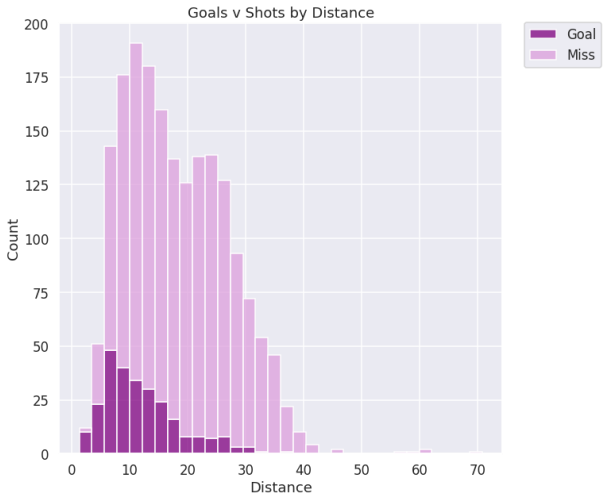
| | goal | shot_location_x | shot_location_y | gk_location_y | distance_to_goal | angle_to_goal | distance_to_gk | end_position_ext_y | end_position_ext_z |
|---|---|---|---|---|---|---|---|---|---|
| 0 | False | 98.0 | 51.0 | 41.0 | 24.6 | 0.3 | 20.6 | 36.2 | 4.5 |
| 1 | False | 88.0 | 59.0 | 40.0 | 37.2 | 0.2 | 36.4 | 46.7 | 6.0 |
| 2 | False | 107.0 | 48.0 | 38.0 | 15.3 | 0.4 | 14.9 | 34.4 | 0.3 |
| 4 | False | 106.0 | 25.0 | 43.0 | 20.5 | 0.3 | 21.6 | 33.9 | 0.2 |
| 6 | False | 97.0 | 20.0 | 45.0 | 30.5 | 0.2 | 31.4 | 45.6 | 0.2 |

Now, we will see the distribution of the different variables and check when the optimal moments and positions are to score goals:



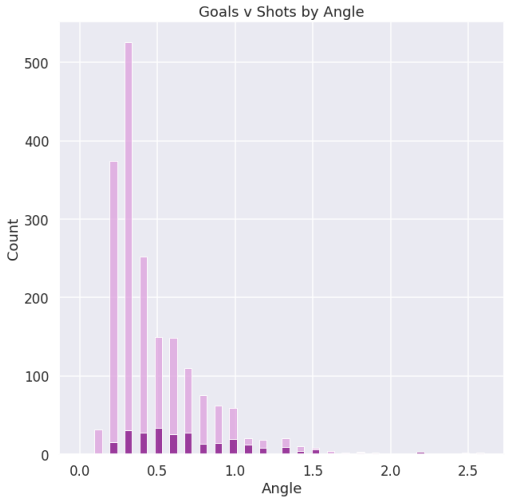

Knowing that x=120 is the goal line, we appreciate how most goals are found between 100<x<120. The area goes up to 102, so there are few goals outside of it, nevertheless, we can see that even when 90<x<100, the players keep trying to shoot regularly, despite not scoring that many goals.

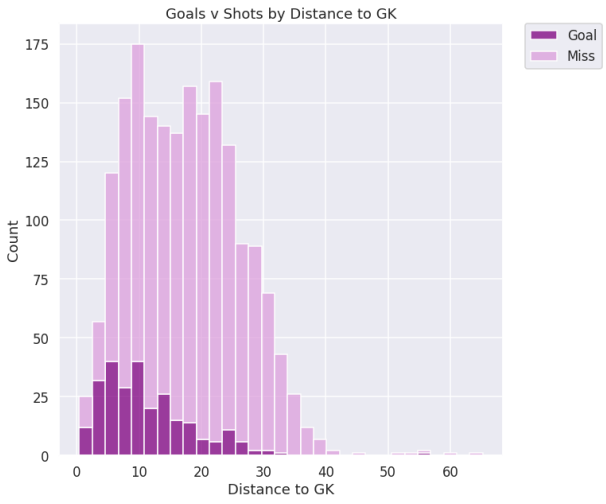
Regarding y, knowing that 40 is half the field on the vertical axis, we see how most goals come from central positions. The more you move away from the center the less likely it is to score. There are no huge differences between shooting from one side or the other.


When we talk about the location of the goalkeeper, we see how the goalkeeper is most of the time located near the center of the goal y=40. The mean is 40.27 and the median 40.1. We can observe that when the goalkeeper is moved to the sides, there is a similar number of goals as when he is in the center, but with much fewer shots. Therefore, this indicates, except in specific situations, it is always better for him to be located near the center goal posts to save more goals.

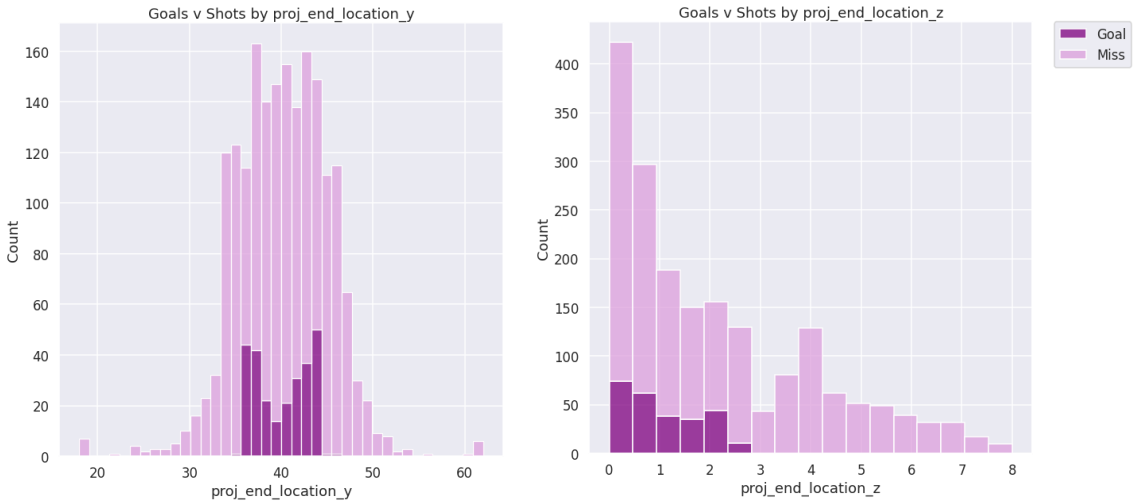
Turning to the players' distance to the goal, when they are very close, they score almost all the goals, and when they move away, the ratio of goals per shot decreases. From 15 meters onwards (outside the area), they rarely score, although there are a big number of shots. Afterwards, we will discuss if it is worth trying to score from far distances.




Upon analyzing the angle in relation to the goal, we observe that there are similar numbers of goals for different angles, but significantly fewer shots when the angle is larger. We can infer that it is more probable to score a goal from shots at an angle of 0.5 compared to smaller angles. However, the challenge arises when one is far away or cornered, as it is easier to take shots at a smaller angle than when centered and close to the goal.

In conclusion, the closer the player is to the goalkeeper, the more probable it is to score a goal, since it is most likely to be closer to the goal. Although, it seems that being too close to the goalkeeper can lead to a miss.




In the chart of the goal sides (y), it is interesting to observe how the number of goals rises when the shot is farther from the center (when the shots are made inside the goal). With this, we deduce that shooting to the sides is more effective than to the center, as long as we do not shoot outside the goal, as we observe a large number of shots outside of it. As a reminder, the goal is located in the meters 36 to 44.

Talking about the height of the shot (z), we see how most of the shots are close to the ground. Anyway, if we look at the correlation between goals/shots, we see that the best one is 1.5 meters above the ground, as it has the most number of goals with the less number of shots.

# 5. Data Modeling

## 5.1. Data Preprocessing

In this section, we will talk about the construction of our model. First, we removed the *shot_location_x* and *gk_location_x* features, since we already have the distance to the goal and the goalkeeper to measure them.

What we did next was divide our sample into X (with all the features) and Y (with the target goal variable). Afterwards, we separated our data between training and testing.

We decided to use 75% of our data for the train and 25% of the data for the test. Then, we did the feature scaling to our X_Train and X_Test. Feature scaling is important when training a model because it ensures that all features are on a similar scale, which helps the model learn patterns and relationships in the data more effectively. Without scaling, features with larger magnitudes can dominate the learning process and bias the model towards those features.

The concern about our data is that most of the shots are misses (1.214 – No goal, 182 – Goal). If we had trained our model directly with these data, in the testing, nearly all the shots would be classified as no goals.

Therefore, to take care of the class imbalance problem, we used oversample to the minority class, in this case Goal=True. What we have done is SMOTE (Synthetic Minority Over-sampling Technique). This is a data augmentation technique commonly used to address imbalanced datasets in machine learning. It works by creating synthetic minority class samples by

interpolating between existing minority class samples, with the aim of balancing the class distribution. Specifically, SMOTE randomly selects a minority class sample, identifies its k nearest neighbors, and creates new samples by linearly interpolating between the original sample and its neighbors in the feature space. This process is repeated until the desired level of class balance is achieved.

Once we did it, we obtained 1.214 shots that are goals, and 1.214 that are no goal.

We would like to emphasize that it is generally recommended to apply feature scaling before applying SMOTE to the dataset. The reason for this is that feature scaling can affect the distance metric used in the SMOTE algorithm, particularly when using distance-based k-nearest neighbors techniques to identify the nearest neighbors. If the features are not scaled, certain features with larger scales may dominate the distance metric, leading to biased and less effective sampling.

## 5.2. Evaluation Metrics

Since the purpose of the model is to predict the probability to score after a shot is made, considering both goals and misses, we need to evaluate our model's performance using appropriate metrics. The evaluation metrics used in this project are Recall, ROC, Accuracy, and F1-score.

- **Recall** is a measure of how well our model is able to identify all positive examples (goals) in the dataset. Specifically, it is the ratio of the true positive predictions (goals correctly identified) to the total number of actual positive examples (all goals in the dataset). A high recall means that our model correctly identifies most of the positive examples in the dataset.
- **ROC** (Receiver Operating Characteristic) is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds. The area under the ROC curve (AUC-ROC) is a measure of how well our model is able to distinguish between positive and negative examples. A higher AUC-ROC means that our model is better able to discriminate between positive and negative examples.
- **Accuracy** is the proportion of correctly classified examples (both positives and negatives) to the total number of examples. It is a measure of overall performance of the model.
- **F1-score** is the harmonic mean of precision and recall and is a measure of the balance between precision (the proportion of predicted positives that are actually positive) and recall (the proportion of actual positives that are correctly predicted). A high F1-score means that our model has a good balance between precision and recall.

These evaluation metrics are important because they provide us with different perspectives on how well our model is performing.

# 6. Models

The models we are using in this project are:

- **Decision Tree**: A decision tree is a tree-like model that recursively partitions the data into subsets based on the value of one feature at a time. At each node of the tree, the model selects the feature that maximally reduces the impurity (e.g., entropy or Gini index) of the subsets. The tree continues to grow until some stopping criteria (e.g., maximum depth or minimum samples per leaf) is met. A decision tree model can be interpreted visually and can be useful in understanding the importance of different features in the data.
- **Random Forest**: A random forest is an ensemble model that combines multiple decision trees. It works by building multiple decision trees on different subsets of the data (selected randomly with replacement), and then averaging their predictions to make a final prediction. The goal of this approach is to reduce overfitting and improve the model's generalization performance. In a random forest, each decision tree is built independently, and the final prediction is based on the mode of the predictions of all trees.
- **Gradient Boost**: Gradient Boosting is an iterative model that combines multiple weak prediction models (e.g., decision trees with small depth) into a strong prediction model. The model starts with an initial prediction (e.g., the mean of the target variable), and then iteratively adds new prediction models that correct the errors made by the previous models. The new models are fit to the residual errors of the previous model. The final prediction is the sum of the initial prediction and the predictions of all the models.
- **XG Boost**: XG Boost (Extreme Gradient Boosting) is an optimized implementation of Gradient Boosting that aims to improve the model's performance and computational efficiency. It includes additional regularization terms to reduce overfitting and employs a parallelized learning system that can handle large-scale datasets. XG Boost also supports custom loss functions, which can be useful in situations where the standard loss functions are not appropriate for the problem at hand.

These four models are frequently used in machine learning and can effectively predict binary outcomes such as whether a shot is a goal or not. Each model has its own advantages and limitations, and the selection of a model depends on the specific characteristics of the data and the problem at hand. By employing multiple models and assessing their performance using the appropriate evaluation metrics we mentioned earlier, we can obtain a better understanding of which models are most efficient in predicting goals and make informed decisions about how to enhance our predictions.

We have developed a function that enables us to evaluate the models with multiple potential parameters, returning the most effective ones for the given model.

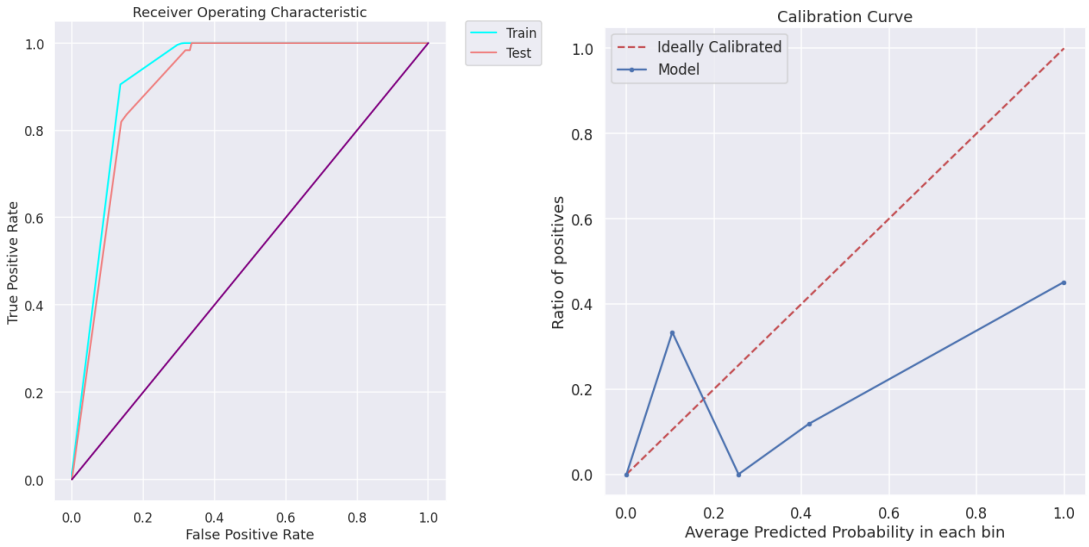
We detail below for each model with the most efficient parameters, what are its results. To solve the problem of overfitting, we adjusted some parameters in order to add some variety into the models.

### 6.1.1. Decision Tree

Model Report Test:

| | False | True | accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|
| precision | 0.971671 | 0.451327 | 0.845494 | 0.711499 | 0.903558 |
| recall | 0.846914 | 0.836066 | 0.845494 | 0.841490 | 0.845494 |
| f1-score | 0.905013 | 0.586207 | 0.845494 | 0.745610 | 0.863281 |
| support | 405.000000 | 61.000000 | 0.845494 | 466.000000 | 466.000000 |

Test - Area Under the Curve: 0.9

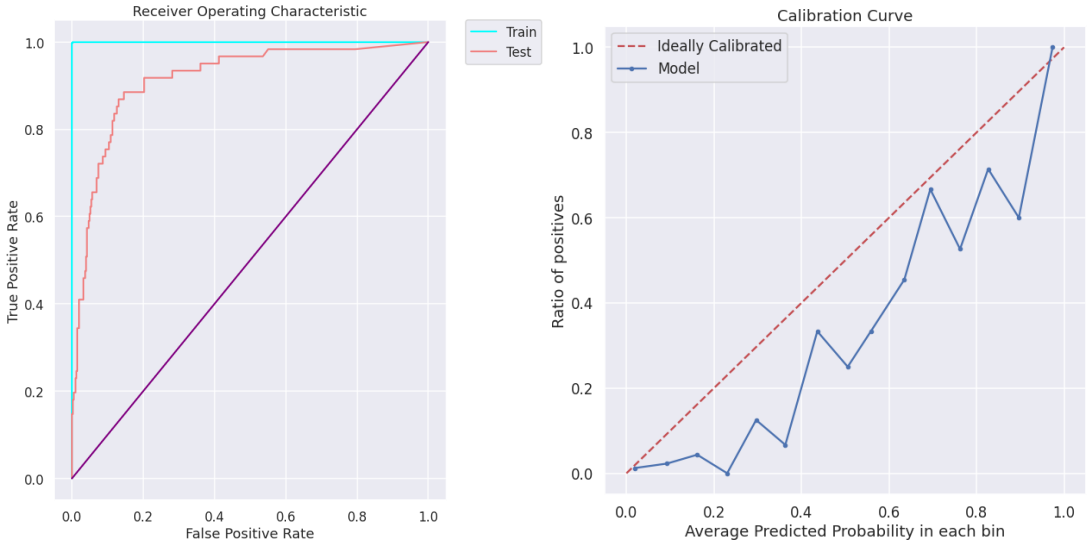



### 6.1.2. Random Forest

Model Report Test:

| | False | True | accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|
| precision | 0.956186 | 0.564103 | 0.890558 | 0.760144 | 0.904861 |
| recall | 0.916049 | 0.721311 | 0.890558 | 0.818680 | 0.890558 |
| f1-score | 0.935687 | 0.633094 | 0.890558 | 0.784390 | 0.896077 |
| support | 405.000000 | 61.000000 | 0.890558 | 466.000000 | 466.000000 |

Test - Area Under the Curve: 0.92

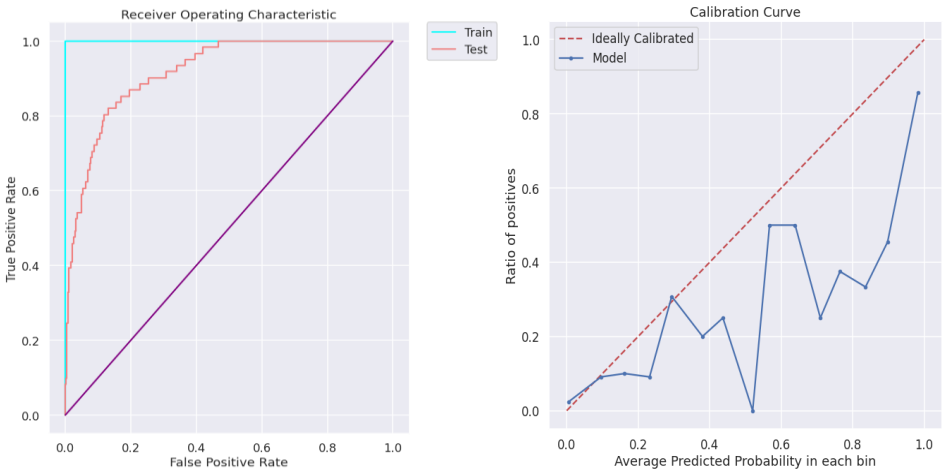



### 6.1.3. Gradient Boost

Model Report Test:

| | False | True | accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|
| precision | 0.953608 | 0.551282 | 0.886266 | 0.752445 | 0.900943 |
| recall | 0.913580 | 0.704918 | 0.886266 | 0.809249 | 0.886266 |
| f1-score | 0.933165 | 0.618705 | 0.886266 | 0.775935 | 0.892002 |
| support | 405.000000 | 61.000000 | 0.886266 | 466.000000 | 466.000000 |

Test - Area Under the Curve: 0.92

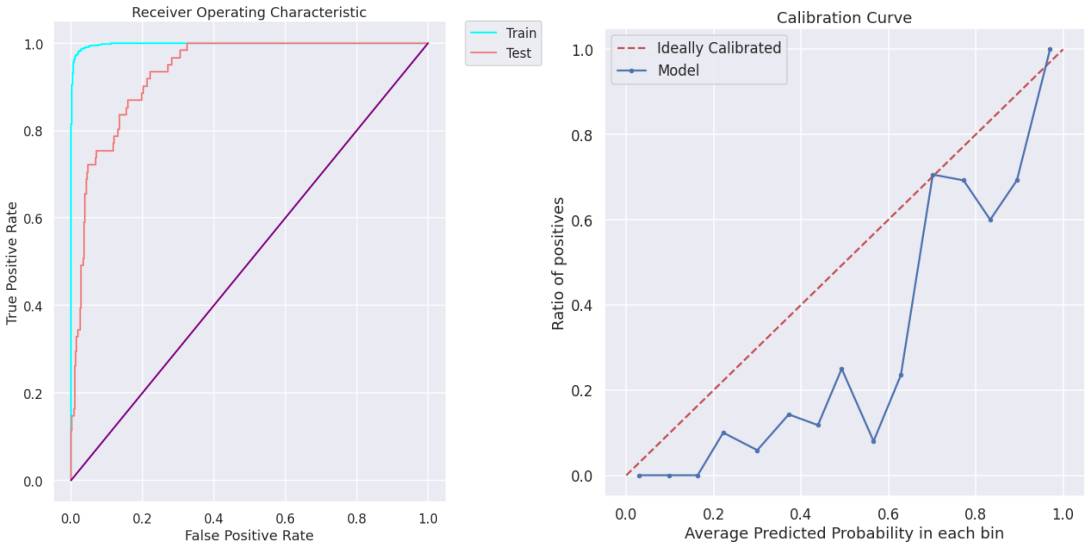


### 6.1.4. XG Boost

Model Report Test:

| | False | True | accuracy | macro avg | weighted avg |
|---|---|---|---|---|---|
| precision | 0.957179 | 0.637681 | 0.909871 | 0.797430 | 0.915356 |
| recall | 0.938272 | 0.721311 | 0.909871 | 0.829792 | 0.909871 |
| f1-score | 0.947631 | 0.676923 | 0.909871 | 0.812277 | 0.912195 |
| support | 405.000000 | 61.000000 | 0.909871 | 466.000000 | 466.000000 |

Test - Area Under the Curve: 0.93




When we use the XGBoost (and the other ones) classifier to predict whether shots are goals or not, we want to know the probability of a shot being a goal. However, when we use the xgb_best.predict_proba() method, we might get predicted values that are outside the range of [0, 1]. This is because we are using the 'binary:logitraw' objective in our XGBoost model, which returns the raw predicted values of the linear function before the logistic transformation.

To obtain probabilities between 0 and 1, we need to apply the logistic transformation to the raw predicted values manually. We can do this by using the sigmoid function, which maps any real number to a value between 0 and 1. Alternatively, we can change the objective of our XGBoost model to 'binary:logistic', which automatically applies the logistic transformation and returns probabilities between 0 and 1. We get worse results when changing to it, so we will use 'binary:logitraw' with the sigmoid.

## 6.2. Model Comparison

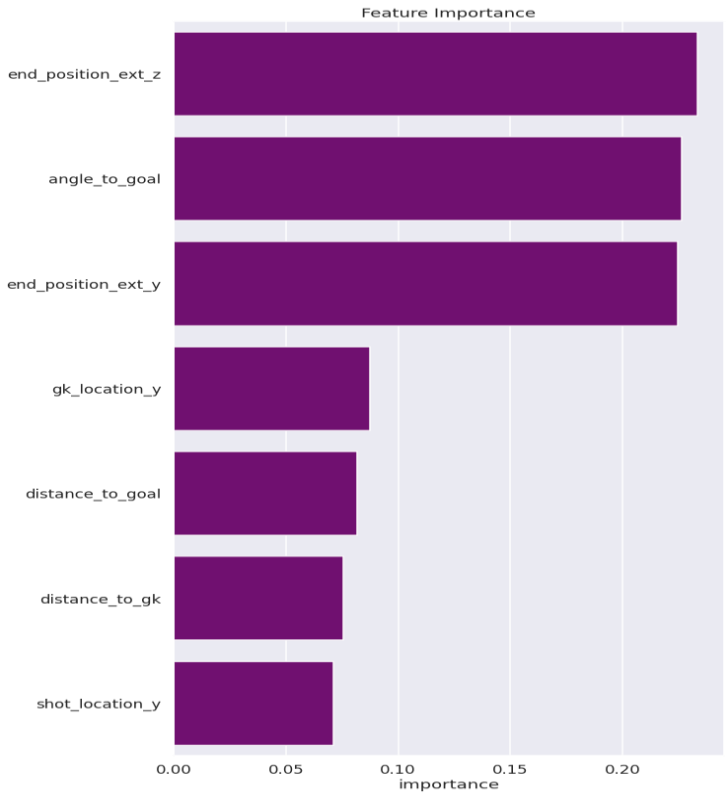
We made a table to compare Recall, ROC, Accuracy and f1-Score for each model:

| | Model | Recall | ROC | Accuracy | f1-Score |
|---|---|---|---|---|---|
| 3 | XG Boost | 0.72 | 0.93 | 0.91 | 0.91 |
| 1 | Random Forest | 0.72 | 0.92 | 0.89 | 0.90 |
| 2 | Gradient Boost | 0.70 | 0.92 | 0.89 | 0.89 |
| 0 | Decision Tree | 0.84 | 0.90 | 0.85 | 0.86 |

We selected XG Boost as it is the one with better f1-score, better accuracy and better ROC.

## 6.3. Feature Selection

We wanted to know the importance of each of the features for our xgb model, which is the one we have chosen. This is the feature importance:



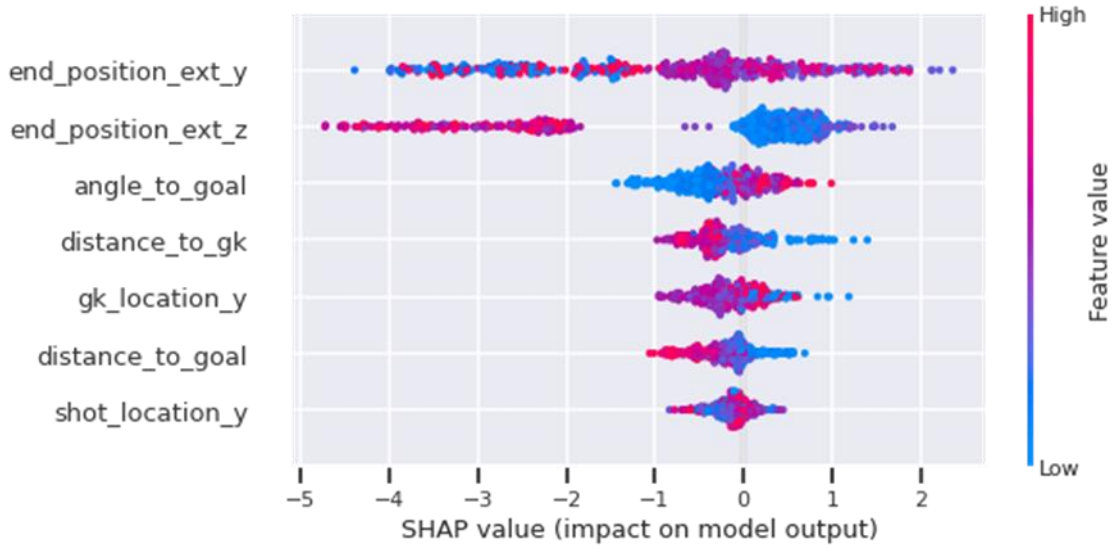
We can clearly see how the two *end_position* (z,y) variables and the *angle_to_goal* variable are by far the most important for the model.

In addition, we see that all of the features contribute more than 0.07 of importance, so we cannot get them away.

A possible change would be to keep only the three most important features, which would make a much more general model. The problem with this, is that it would only take into account the destination/direction and the origin of the ball (which would be calculated by the angle), ignoring the goalkeeper position and the exact place of the field where the shots are being made. That would not be realistic, as with a similar angle but different position on the pitch (eg. in the corner vs. the middle of the field), we would get the same probability.

Moreover, we have created a SHAP (SHapley Additive exPlanations) explainer object for our XGBoost classifier.

The SHAP values measure the contribution of each feature to the predicted outcome (in our case, whether a shot is a goal or not) for each individual sample in our dataset. These values are computed by analyzing the effect of each feature on the output of the XGBoost model for all possible combinations of features, taking into account the interactions between the features:

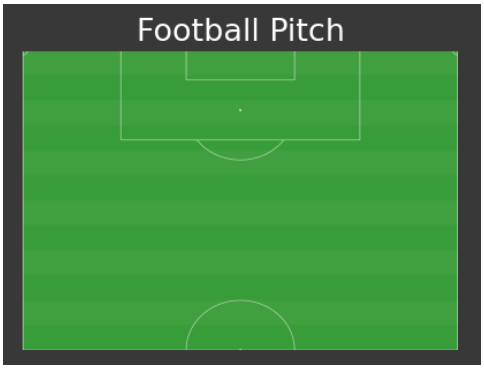



What we consider important is:

- High values of *end_position_ext_z* makes the probability decrease a lot. This is because it is taking the shots where the players shot above the goal posts. We see that there is also a cluster of red points when the SHAP values are high, indicating that shots with big z (without being more than the 2.67 of the goal post), the bigger the probability is.
- We clearly see that as bigger is the angle, bigger is the probability of scoring.
- In the distances features we see that as less distance there is, more probability of goal.

# 7. Data visualization

### 7.1.1. Football pitch

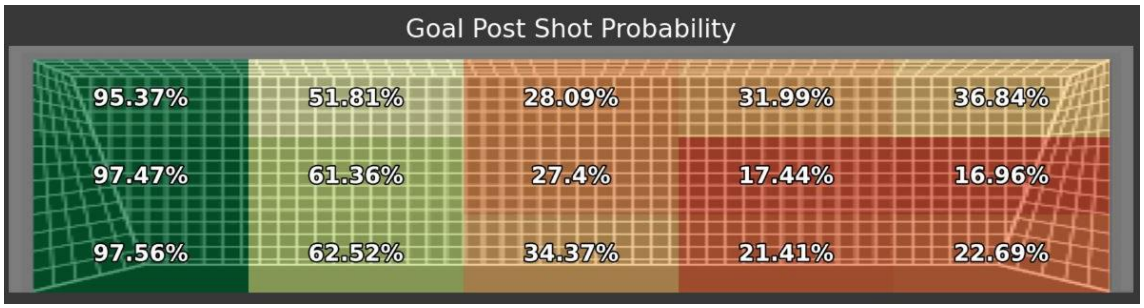



It shows a half of the football pitch. This visualization is interactive, that means that you can select a coordinate (x,y) by clicking any location in the football pitch. There is any coordinate selected until you click in the football field. A red circle will appear in the coordinates that you clicked in. The x coordinate can have values between 0 and 80, while the y coordinate can take values from 0 to 55. The maximum y coordinate can be 55 because we fitted the image to the plot. In addition, when a coordinate is selected, we are not taking in count only the exact x and y, but also all the coordinates in all directions with a meter of margin. With this, we calculated all the probabilities for each coordinate and made the average for each shot in order to smooth the *PSxG*.

### 7.1.2. Goalkeeper position


This visualization shows the position (between the left and right goal posts) of the goalkeeper. The goalkeeper can move through the x-axis by moving the slider 'Goalkeeper position'. The position of the goalkeeper will be updated according to this slider. In the beginning, the position of the goalkeeper will be the center. Once the goalkeeper is moved either to the left or to the right, you can center him by clicking the 'Set Goalkeeper To Center' button.

### 7.1.3. Goal Post Shot Probability




In this visualization, we can see a heatmap inside a goal. This heatmap represents each zone of the goal, colored by the probability of scoring a goal after a shot is made. This way, we can see if a shot/save was good or badly executed. On one hand, we can prove if a player scored/failed a goal that was very difficult/easy to score, and on the other hand, we can see if a goalkeeper made a difficult save or he conceded an easy goal to save. With this, we can compare if a player/goalkeeper is performing above or below the mean. In the first execution, all the probabilities will be 0%, as any coordinate in the field is selected. This visualization will be updated once the 'Update Goal Heatmap' is clicked. In addition, this visualization provides us the ability to evaluate the shot decision-making of the players, taking in count if the location in the goal where the shot was made, was more/less probable to score. In addition, when we calculate the *PSxG* for each zone, we begin with a matrix of 25 rows by 38 columns. After that, we made the average probability for a matrix of 3 rows by 5 columns. This is made to smooth the probabilities.

## 7.2. Visualization interactivity

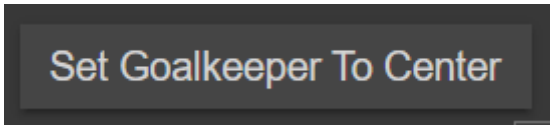
### 7.2.1. Meters in front of the goal (Goalkeeper)


This selector is a text box that we can use to set the number of meters that the goalkeeper is in front of the goal line. The selector can be changed either by introducing a number between 0 and 5 or by clicking the arrows (up/down) next to the number in the text box. The numbers can be increased/decreased by 0.5. The average position of the goalkeepers in the goal is 2.5, that is why this number appears in the first place.

### 7.2.2. Goalkeeper position

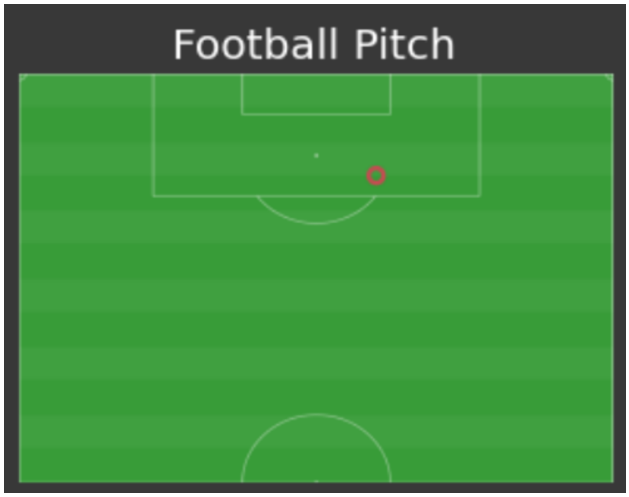

For this interaction, we decided to put a slider to move the goalkeeper through the x-axis (left and right). This slider provides the ability to put the goalkeeper in the position of the goal when a shot is made. The goalkeeper will move accordingly to the slider.

### 7.2.3. Set goalkeeper to center



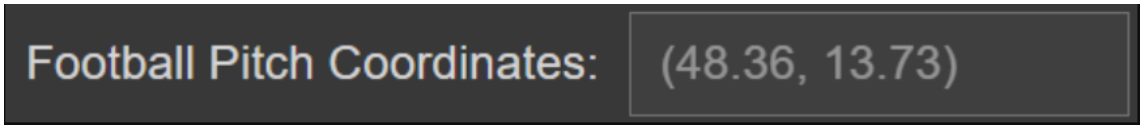

This button has the ability to set the goalkeeper to the center position, as once is moved through the slider, if we want to evaluate another shot, it is difficult to put it back again to the initial position with the slider.

### 7.2.4. Select a coordinate in the football pitch



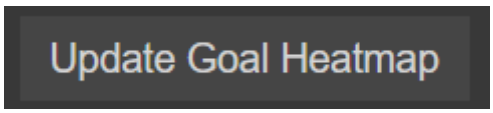

As we mentioned previously, we can select a coordinate by clicking in the football pitch. Once it is clicked, a red circle will appear in the coordinate and the text box with the football pitch coordinates will be updated indicating the selected coordinate.

#### 7.2.5. Football pitch coordinates




This text box will display the coordinate selected in the football pitch.

#### 7.2.6. Update goal heatmap




With this button, we can update the goal heatmap with the selected coordinates and goalkeeper position. In case that we do not select any coordinate in the football pitch, we display a message saying that before updating the heatmap, selecting a coordinate is needed.

# 8. Examples & Conclusions

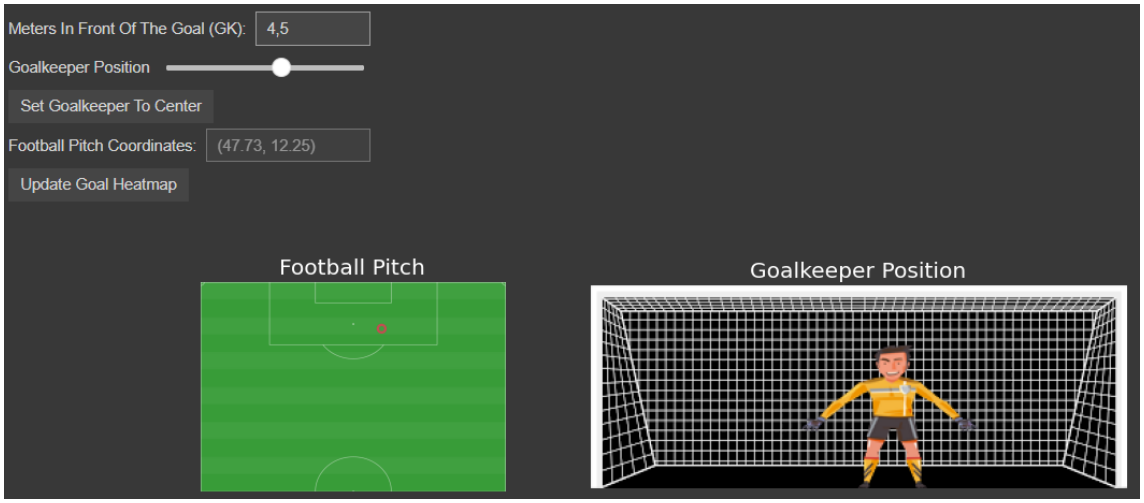
## 8.1. Visualization examples

What we can do with our visualization engine is to evaluate the performance of the goalkeeper, the player and their decision-making.

#### 8.1.1. Goalkeeper performance

If we talk about the World Cup, the performance of “Dibu” Martínez in the final against France is the more relevant execution. Specifically the spectacular save in the extra time in which he makes against Kolo Muani, saving Argentina and allowing his national team to win the World Cup.


We can select in our visualization tool the parameters that adjust to this situation:




We obtain the following Goal Post Shot Probability matrix:


The location of the French player's shot was at the bottom right, with which there was a 61% score by the player according to our model. The goalkeeper therefore made a very valuable save for his team. We can conclude from this probability that almost 1 of every 2 shots would be goal in this situation.

### 8.1.2. Decision Making

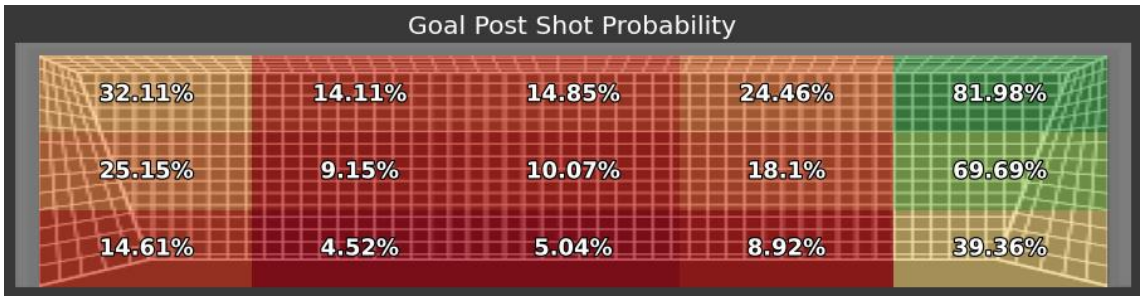
In the same example, we can evaluate if Kolo Muani's decision-making was the most accurate in this situation.

He decided to shoot down to the right, which gives him a higher or similar probability than shooting to any central position of the goal (any of the three columns in the middle).

What we can observe is that if he had been able to cross the ball to the left side of the goal, he would have increased the chances of scoring by more than 30%, making it practically impossible for the Argentine goalkeeper to stop the ball.

An example where an impeccable decision is clearly shown is the following:


Where Enzo Fernandez vs Mexico, he decides to shoot towards the right corner of the goal. Once we place the parameters of the shot, we obtain this matrix:



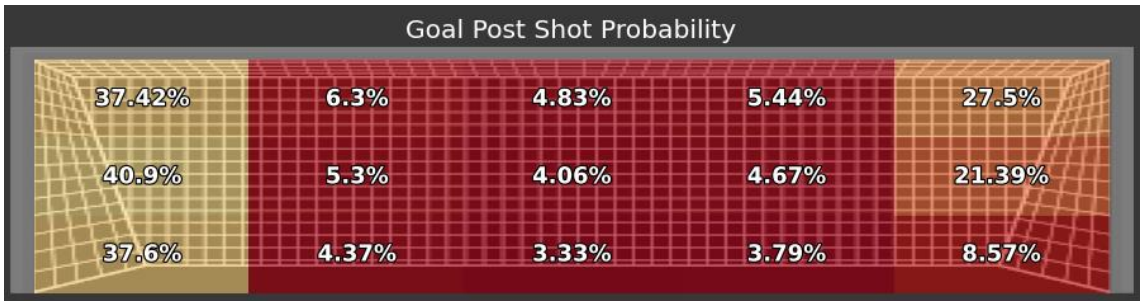

We clearly see how he decides the best possible destination, making a real great goal.

### 8.1.3. Player performance

To evaluate the performance of the players, we can select plays in which it was unlikely to score, if the player has scored it indicates that he has made a really good kick, marking differences with respect to the average number of players.

In the next shot by the Korean player Seung-Ho Paik against Brasil, taken from approximately the position (44,28), a real great goal was scored.

The probability matrix was as follows, being almost unlikely to score a goal from this position:




The player shoots to the right at medium height, having a 21% score. Despite this, the player makes an unstoppable shot.

## 8.2. Conclusions

To make our conclusions, we made a table with the goalkeepers, players and teams. The shots that we took in account were the used in the test, as if we used the training, we can take the risk of introducing bias into our analysis, as the model has already seen this data and may be overfitting to it.

In the table of the goalkeepers we can found their respective goals received Post expected goals received, the difference between the goals received and the Post expected goals received, the expected goals (Not post shot) from StatsBomb, and the shots received.

| gk | goal | PSxG | PSxG_difference | shot_statsbomb_xg | Shots_received |
|---|---|---|---|---|---|
| Kasper Schmeichel | 2 | 5.76 | -3.76 | 1.64 | 17 |
| Dominik Livaković | 2 | 5.26 | -3.26 | 2.76 | 19 |
| Manuel Neuer | 2 | 3.37 | -1.37 | 1.47 | 11 |
| Vladimir Stojković | 2 | 3.37 | -1.37 | 1.13 | 11 |
| Jordan Pickford | 2 | 3.22 | -1.22 | 2.36 | 17 |
| Lawrence Ati-Zigi | 4 | 3.19 | 0.81 | 2.18 | 11 |
| Shūichi Gonda | 2 | 3.15 | -1.15 | 1.08 | 9 |
| Vanja Milinković Savić | 2 | 2.98 | -0.98 | 1.96 | 8 |
| Yann Sommer | 0 | 2.96 | -2.96 | 0.68 | 13 |
| Edouard Mendy | 2 | 2.94 | -0.94 | 1.03 | 9 |

The conclusion to this grid is that Kasper Schmeichel is the goalkeeper that had the most probability to receive more goals, but he conceded less than the mean of his Post Shot expected goals. On the contrary, we see that Lawrence Ati-Zigi had more goals than the Post Shot expected goals.

We made another table to compare the PSxG of the players between their actual goals:

| player | goal | PSxG | PSxG_difference | shot_statsbomb_xg | Shots_done |
|---|---|---|---|---|---|
| Kylian Mbappé Lottin | 4 | 2.61 | 1.39 | 1.36 | 9 |
| Giorgian Daniel De Arrascaeta Benedetti | 2 | 2.40 | -0.40 | 1.14 | 3 |
| Mathew Leckie | 1 | 1.94 | -0.94 | 0.31 | 4 |
| Ahmed Musa | 2 | 1.91 | 0.09 | 0.72 | 3 |
| Neymar da Silva Santos Junior | 1 | 1.91 | -0.91 | 1.33 | 9 |
| José Paolo Guerrero Gonzáles | 0 | 1.76 | -1.76 | 0.28 | 3 |
| Romelu Lukaku Menama | 1 | 1.71 | -0.71 | 1.57 | 7 |
| Memphis Depay | 1 | 1.55 | -0.55 | 0.61 | 4 |
| Ivan Perišić | 1 | 1.54 | -0.54 | 0.41 | 8 |
| Vinicius José Paixão de Oliveira Júnior | 0 | 1.51 | -1.51 | 0.32 | 4 |

This table is not 100% representative as we have only few shots for every player and we can not assume that this are the correct ratios.

The last table made for the teams is the following:

| team | goal | PSxG | PSxG_difference | shot_statsbomb_xg | Shots_done |
|---|---|---|---|---|---|
| Brazil | 4 | 11.02 | -7.02 | 4.48 | 32 |
| Croatia | 5 | 7.92 | -2.92 | 3.24 | 38 |
| Belgium | 4 | 6.50 | -2.50 | 2.43 | 23 |
| Germany | 4 | 6.26 | -2.26 | 2.25 | 18 |
| France | 6 | 4.92 | 1.08 | 3.69 | 31 |
| South Korea | 5 | 4.71 | 0.29 | 1.28 | 13 |
| Mexico | 2 | 4.36 | -2.36 | 1.35 | 20 |
| Spain | 3 | 4.23 | -1.23 | 1.60 | 16 |
| Uruguay | 3 | 4.05 | -1.05 | 2.38 | 17 |
| Morocco | 2 | 3.80 | -1.80 | 1.11 | 20 |

Finally, the conclusion to this table is that Brazil made more opportunities to score a goal than the rest of the teams, while they did not score even the mean of their *PSxG*. Furthermore, this team will in long term, if they keep generating this opportunities the have a greater chance to score more goals. In comparison, we have France, which scores more goals than the *PSxG*.

To sum up, we can say that after all the training of our model we obtained a smooth changing probabilities, close to the reality, which we can use to evaluate the actions taken by the players compared to the mean results of all the players.

# 9. Future improvements

As we had some time and resources limitations, we know some future improvements that can be made to improve the model:

- **Add more data to the model:** As we mentioned in the document, we know that we have not that much shots to build a generalized model for each case. This can be improved by adding more shot events into the training model, including for example data from national leagues or European tournaments. In some cases, this kind of data is not free of charge, so this can be a limitation in the future if we expect to include into our model only data with no cost.
- **Add the number of players inside the angle of the shot:** Another enhanced feature to add to our model to be more realistic would be to add the number of players inside the angle of the shot. This would be either a feature in our model and an interactive selector to change de goal matrix. With this, we would cover more cases and evaluate more accurate the performance of a player.
- **Project the goal probability outside the goals posts:** With our current model, we are saying that if a player shoot to a post had a 0% chance of scoring, while this is not the reality. In some cases, a closer shot to the posts will be more probable to score a goal, taking into consideration the fact that the ball can be deflected by any other player or even the post and end inside the goal. This would refine the accuracy of our model and add more cases relative to the real world. In addition, this would be useful to add veracity to the expected goal probability of the players, taking in account that if a player shoots close to the posts, would be more probable to score a goal.
- **Add the shot velocity/speed:** In our current dataset, we do not have this feature available. It would be nice to have it as a selector and as a feature to train our model. This way we would be able to get closer to the truth, with the ability to compare the same shots with different speeds.
- **Add the ball trajectory after a shot:** If we had the ball trajectory to train our model, we could infer this as a feature to our model as well as representing it in a visualization/selector, having the capacity to select a shot between a ground floor shot and a curved aerial shot, for example.

# 10. Bibliography:

[1] https://github.com/statsbomb/open-data/blob/master/doc/Open%20Data%20Events%20v4.0.0.pdf

Caley, M. (2015). A new way to evaluate goalkeepers using post-shot expected goals. Retrieved from https://cartilagefreecaptain.sbnation.com/2015/12/11/9885968/a-new-way-to-evaluate-goalkeepers-using-post-shot-expected-goals

Groom, R. (2018). The math behind expected goals (xG) in soccer. Retrieved from https://www.thoughtco.com/expected-goals-xg-in-soccer-4120059

Singh, K. (2018). What are expected goals (xG)? The stats behind the soccer metric. Retrieved from https://www.si.com/soccer/2018/06/21/what-are-expected-goals-xg-stats-behind-soccer-metric

Scikit-learn documentation: https://scikit-learn.org/stable/index.html

Towards Data Science: https://towardsdatascience.com/understanding-data-science-evaluation-metrics-bfa9f95db967

Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/10/how-to-choose-evaluation-metrics-for-classification-model/

Medium: https://towardsdatascience.com/how-to-evaluate-a-classification-model-15abfb18b27b