1. Problem Setting:

Football (soccer) is a sport where scoring goals is the ultimate objective, and understanding the factors influencing the likelihood of a successful goal is crucial. Expected Goals (xG) is a statistical metric that quantifies the probability of a shot resulting in a goal. This project aims to leverage machine learning techniques to predict the xG of football shots based on various features.

2. Problem Definition:

The primary objective is to build a predictive model that can estimate the xG of a given shot. The model will take into account features such as shot distance, angle to the goal, and potentially other factors derived from applying feature engineering on match events. We aim to predict the probability of a shot converting into a goal.

3. Data Sources:

The project utilizes football event data obtained from the Wyscout platform, specifically the events datasets for the top five European football leagues (England, Spain, Italy, Germany, France). The data is publicly available and has been cited from the research paper by Luca Pappalardo et.al. https://doi.org/10.6084/m9.figshare.c.4415000.v5

4. Data Description:

The datasets have 12 columns and approx. of 600k rows that include information on various match events, with a focus on shots (eventId=10) i.e. 40,461rows. The relevant features for building the xG model include:

- Shot positions (start and end coordinates)
- Tags indicating the outcome of the shot (e.g., goal, miss)
- Other relevant features derived from event data

A sample of variables includes shot distance, shot angle, and the outcome of the shot (goal or miss). The datasets provide a comprehensive set of information on which the required features can be selected and calculated using feature engineering for building and evaluating the xG model.

The head of the required data frame is shown below.

In [36]:
```
finalShotData.head()
```

Out[36]:

	eventId	subEventName	tags	playerId	positions	matchId	eventName	teamId	matchPeriod	eventSec	subEventId	id
0	10	Shot	[{'id': 101}, {'id': 402}, {'id': 201}, {'id':...	25413	[{'y': 41, 'x': 88}, {'y': 0, 'x': 0}]	2499719	Shot	1609	1H	94.595788	100	177959212
1	10	Shot	[{'id': 401}, {'id': 201}, {'id': 1211}, {'id'...	26150	[{'y': 52, 'x': 85}, {'y': 100, 'x': 100}]	2499719	Shot	1631	1H	179.854785	100	177959247
2	10	Shot	[{'id': 101}, {'id': 403}, {'id': 201}, {'id':...	14763	[{'y': 52, 'x': 96}, {'y': 100, 'x': 100}]	2499719	Shot	1631	1H	254.745027	100	177959280
3	10	Shot	[{'id': 401}, {'id': 201}, {'id': 1215}, {'id'...	7868	[{'y': 33, 'x': 81}, {'y': 0, 'x': 0}]	2499719	Shot	1609	1H	425.824035	100	177959289
4	10	Shot	[{'id': 402}, {'id': 201}, {'id': 1205}, {'id'...	7868	[{'y': 30, 'x': 75}, {'y': 0, 'x': 0}]	2499719	Shot	1609	1H	815.462015	100	177959429

The shape of the required data frame is show below.

```
finalShotData.shape
```

```
(40461, 12)
```

The info of the required data frame is shown below.

```
finalShotData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 40461 entries, 46 to 632802
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   eventId       40461 non-null  int64
 1   subEventName  40461 non-null  object
 2   tags          40461 non-null  object
 3   playerId      40461 non-null  int64
 4   positions     40461 non-null  object
 5   matchId       40461 non-null  int64
 6   eventName     40461 non-null  object
 7   teamId        40461 non-null  int64
 8   matchPeriod   40461 non-null  object
 9   eventSec      40461 non-null  float64
 10  subEventId    40461 non-null  object
 11  id            40461 non-null  int64
dtypes: float64(1), int64(5), object(6)
memory usage: 4.0+ MB
```

4. Data Mining Tasks:

The first steps we take in performing data mining tasks is to extract the exact X-Y coordinates from our data frame as the coordinates are stored as a list of dictionaries. By performing feature extraction, we can now use x and y coordinates to find position of the shots taken.

positions
[{'y': 41, 'x': 88}, {'y': 0, 'x': 0}]
[{'y': 52, 'x': 85}, {'y': 100, 'x': 100}]
[{'y': 52, 'x': 96}, {'y': 100, 'x': 100}]
[{'y': 33, 'x': 81}, {'y': 0, 'x': 0}]
[{'y': 30, 'x': 75}, {'y': 0, 'x': 0}]

will be converted to

startY	startX	endY	endX
41	88	0	0
52	85	100	100
52	96	100	100
33	81	0	0
30	75	0	0

The next step of feature extraction and data reduction is to use tags mentioned in our data frame and find required tags based on our project requirements. The descriptions of the tags are mentioned in the research paper cited above. For example, 101 is represented as a Goal, 301 is Assist, 401 is Left Leg Used, 602 is anticipation. Link to Tags

For this projection we need to know if the shot taken has resulted in a goal or not. The variable will be considered as our Target Variable while designing the model. To extract this feature, we create a table that extracts all the tags of each event and extract tag 101 and label it to Goal or No Goal.

	0	1	2	3	4	5
0	{'id': 101}	{'id': 402}	{'id': 201}	{'id': 1205}	{'id': 1801}	None
1	{'id': 401}	{'id': 201}	{'id': 1211}	{'id': 1802}	None	None
2	{'id': 101}	{'id': 403}	{'id': 201}	{'id': 1207}	{'id': 1801}	None
3	{'id': 401}	{'id': 201}	{'id': 1215}	{'id': 1802}	None	None
4	{'id': 402}	{'id': 201}	{'id': 1205}	{'id': 1801}	None	None
...	...	...	...	...	...	...
40456	{'id': 401}	{'id': 201}	{'id': 1205}	{'id': 1801}	None	None
40457	{'id': 402}	{'id': 201}	{'id': 1214}	{'id': 1802}	None	None
40458	{'id': 401}	{'id': 2101}	{'id': 201}	{'id': 1802}	None	None
40459	{'id': 401}	{'id': 201}	{'id': 1201}	{'id': 1801}	None	None
40460	{'id': 101}	{'id': 401}	{'id': 201}	{'id': 1208}	{'id': 1801}	None

This data is converted to

startY	startX	endY	endX	goal
41	88	0	0	1
52	85	100	100	0
52	96	100	100	1
33	81	0	0	0
30	75	0	0	0

To perform the next data mining procedure, we will use tags 401, 402, and 403 to determine the body part that was used to perform the shot i.e. right foot, left foot, or using header or body. This will later help us determine if a player used his strong or week foot. This will create an important categorical feature that will contribute to our model building.

```
   eventfoot
0      right
1       left
2     header
3       left
4      right
```

Now to determine if the foot used was strong foot or week foot, we will be importing a second dataset provided in the same research paper that provides the details of the players.
Link to Player Dataset We will be merging this dataset to our current data using left join on "playerid". This will give us additional information about the player who was involved in the event. We will then compare the preferred foot of the player to the foot used in the event to determine if it was strong foot or week foot.

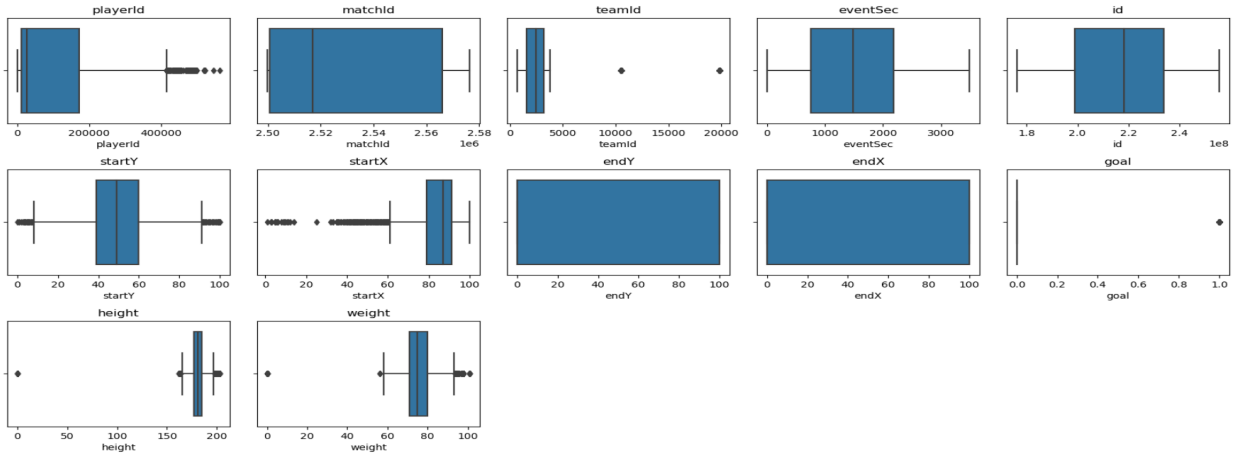
startX	...	endX	goal	firstName	lastName	foot	height	weight	role.name	eventfoot	shot_foot
88	...	0	1	Alexandre	Lacazette	right	175.0	73.0	Forward	right	strong foot
85	...	100	0	Riyad	Mahrez	left	179.0	62.0	Midfielder	left	strong foot
96	...	100	1	Shinji	Okazaki	right	174.0	70.0	Forward	header	header
81	...	0	0	Alex	Oxlade-Chamberlain	right	175.0	70.0	Midfielder	left	weak foot
75	...	0	0	Alex	Oxlade-Chamberlain	right	175.0	70.0	Midfielder	right	strong foot

Using this new dataset will also provide additional information that can be used to build our model. We will now go through our data frame and drop the columns/features that are not required or not have impact towards the model or are duplicates in the dataset, this will be one of the dimension reductions. The columns that are being dropped are 'eventfoot','eventId','eventName', 'positions','startYX','endYX'.
The next step is to check for any missing values that are present in our merged dataset.

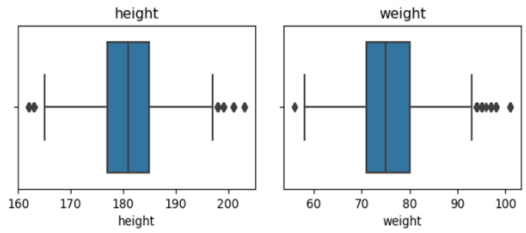
```
goal          0
firstName     3
lastName      3
foot          3
height        3
weight        3
role.name     3
shot_foot     0
```

Since the number of null values are very low, we can drop those rows.

We will now find any outliers in our dataset that can be done by constructing boxplots for the numerical values.



It's very evident from the boxplots that height and weight have few extreme outliers that may impact our model negatively. From the visual representation we can see that the height and weight are zero, which is not possible in a case of a humans. So, we are going to remove these zero values by removing the rows that consist of those values.



This is the new boxplots post removal of 0 values.

We can perform a chi-squared test to analyze the relationship between the player's foot used to shoot suck as strong foot or week foot (shot_foot) and the outcome of whether a goal was scored (goal). This will help us determine if there is a significant association between the player's preferred foot and their likelihood of scoring a goal.

```
Chi-squared statistic: 46.202173189667654
P-value: 9.275248755192316e-11
```

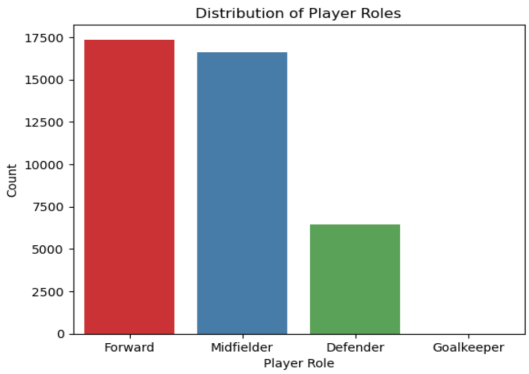
Based on the output, The chi squared value represents the result of the chi-squared test. It indicates the strength of the association between the two variables. In this case, the chi-squared statistic is approximately 46.20 and the p-value is approximately 9.28e-11 (or approximately 0), which is extremely low. Therefore, we can conclude that there is a significant association between a player's preferred foot and their likelihood of scoring a goal.

5. Data Exploration:

The statistical description of our final dataset is shown below.

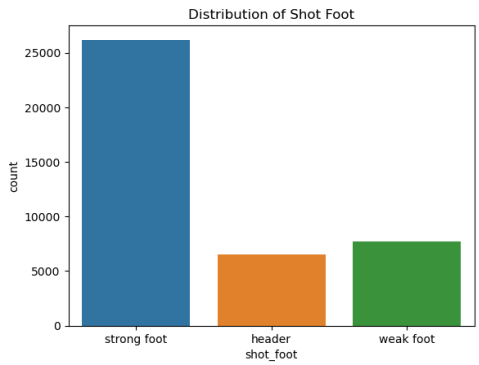
	playerId	matchId	teamId	eventSec	id	startY	startX	endY	endX	goal	height	weight
count	40458.000000	4.045800e+04	40458.000000	40458.000000	4.045800e+04	40458.000000	40458.000000	40458.000000	40458.000000	40458.000000	40458.000000	40458.000000
mean	96666.318750	2.532575e+06	2607.857902	1472.327127	2.173399e+08	49.232537	84.828266	44.522715	44.522715	0.105566	181.057813	75.423427
std	122838.088397	3.319150e+04	2209.194665	818.662880	2.145643e+07	13.761569	8.096794	49.699702	49.699702	0.307285	6.792741	6.950713
min	36.000000	2.499719e+06	674.000000	1.238426	1.761217e+08	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	11294.000000	2.500764e+06	1612.000000	762.299077	1.989030e+08	39.000000	79.000000	0.000000	0.000000	0.000000	177.000000	71.000000
50%	25715.000000	2.516887e+06	2455.000000	1480.584749	2.181691e+08	49.000000	87.000000	0.000000	0.000000	0.000000	181.000000	75.000000
75%	173214.000000	2.565869e+06	3197.000000	2181.203862	2.337738e+08	60.000000	91.000000	100.000000	100.000000	0.000000	185.000000	80.000000
max	564512.000000	2.576338e+06	19830.000000	3490.826794	2.557092e+08	100.000000	100.000000	100.000000	100.000000	1.000000	203.000000	101.000000

Count Plot (Distribution Of player roles)



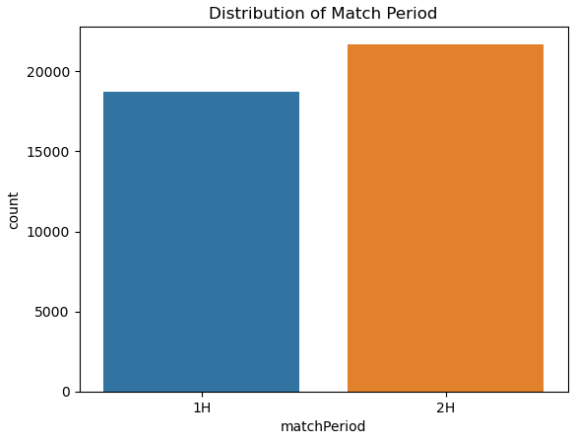
The count plot displays player role distribution with Forward and Midfielder roles being most common, followed by Defenders, and then Goalkeepers, as expected. This visualization aids in understanding dataset composition, reflecting typical team formations and potential insights into player involvement in events like shots and goals.

Count Plot (Distribution of Shot Foot)



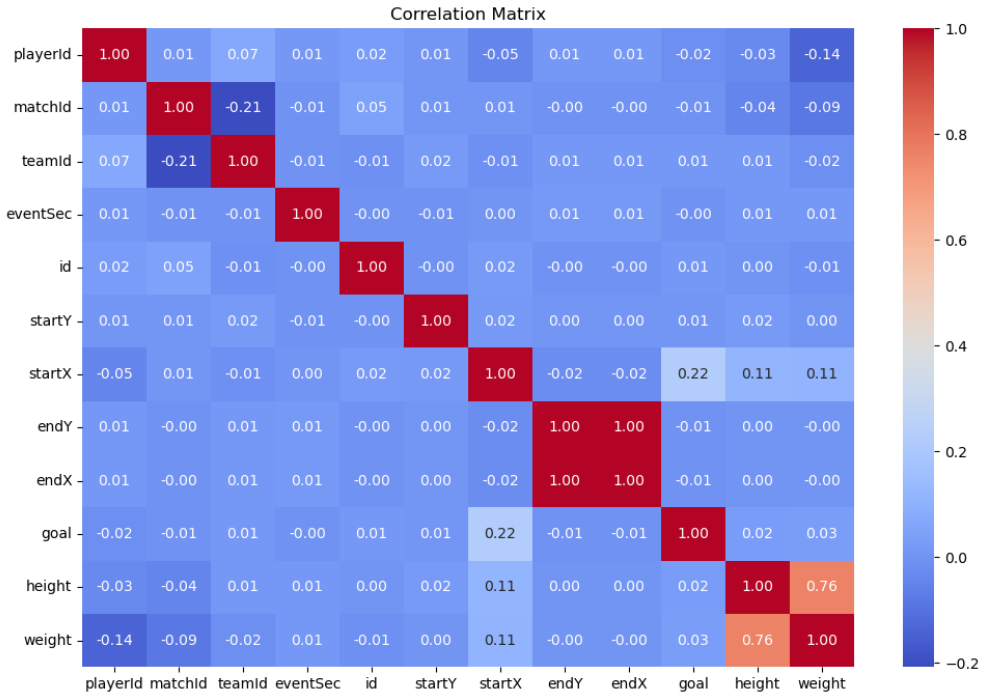
The count plot illustrates shot distribution by foot type in football data. Strong foot shots are most common, followed by headers and weak foot shots, indicating player preferences. This insight is valuable for xG model predictions, as strong foot shots may have higher goal probabilities.

Count Plot (Match Period)



The count plot compares event occurrences in the first half (blue) and second half (orange) of football matches, revealing increased activity in the latter. This insight aids in strategizing around players' stamina and game dynamics as matches progress.

Correlation Matrix



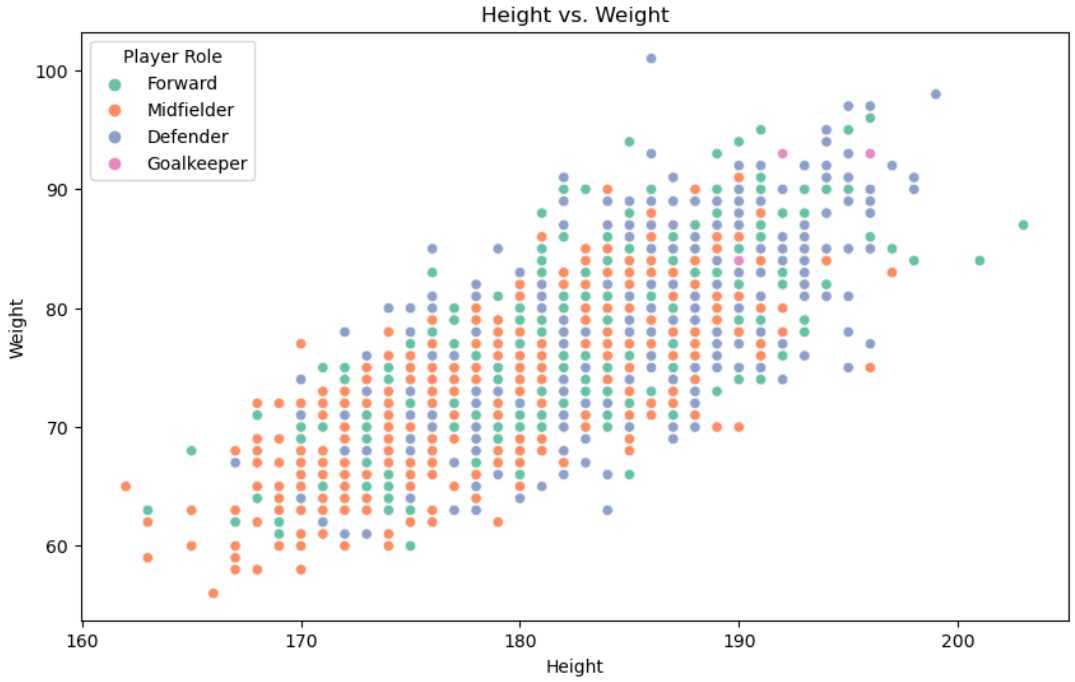
The correlation matrix highlights relationships between variables in a dataset. Key findings include:

Low correlations for playerId, matchId, teamId, and id, indicating they are independent.

Positive correlations between startX and endX (0.22), and goal and endY (0.22).

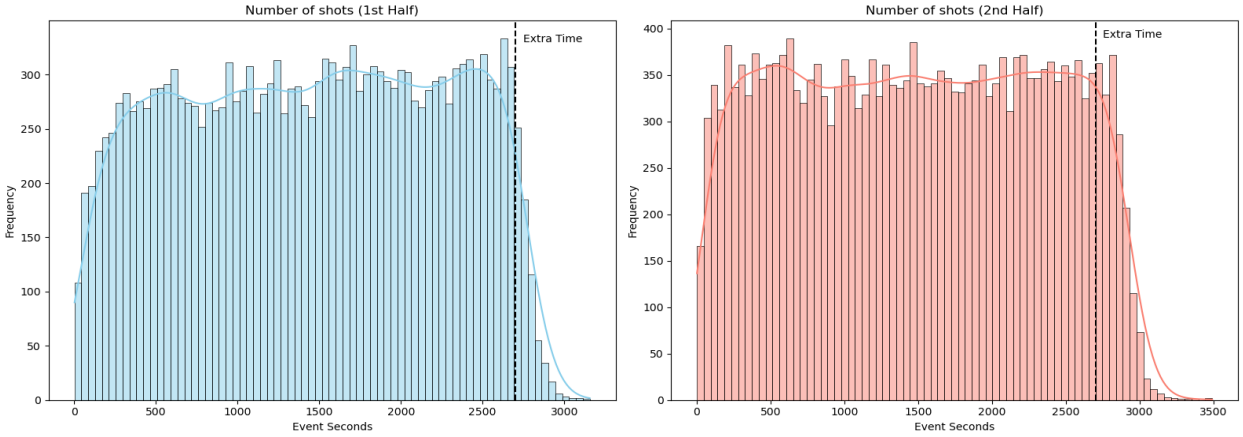
A strong positive correlation between height and weight (0.76), suggesting taller players tend to be heavier. These insights aid in feature selection for xG models, helping identify variables that influence shot outcomes.

Scatter Plot (Height vs Weight)



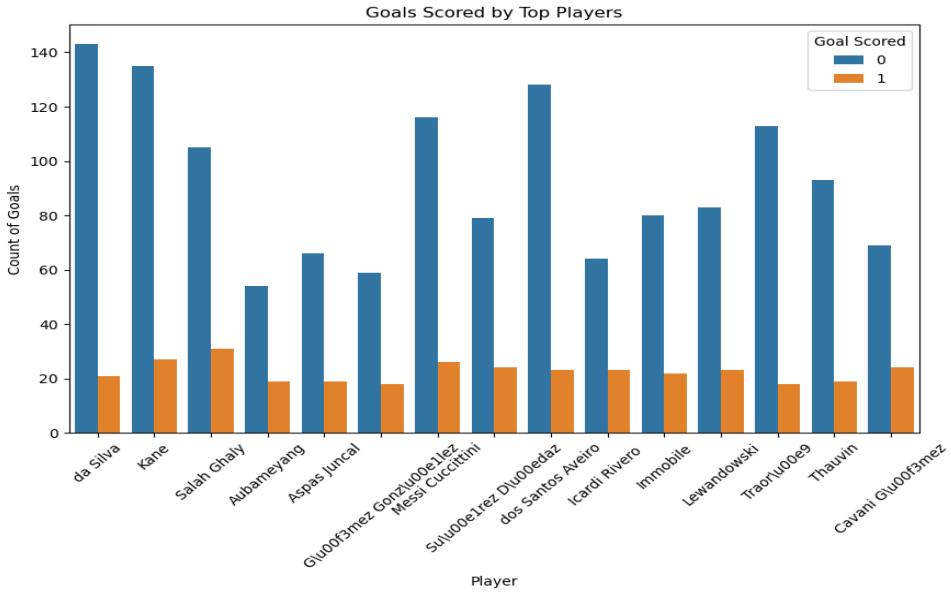
The scatter plot shows the relationship between height and weight for players in different positions: Forward, Midfielder, Defender, and Goalkeeper. It reveals a positive correlation between height and weight and highlights potential physical profiles associated with each position, such as taller and heavier players for defenders and goalkeepers. This insight aids in understanding positional characteristics and their impact on player performance.

Histogram (Shot frequency over time)



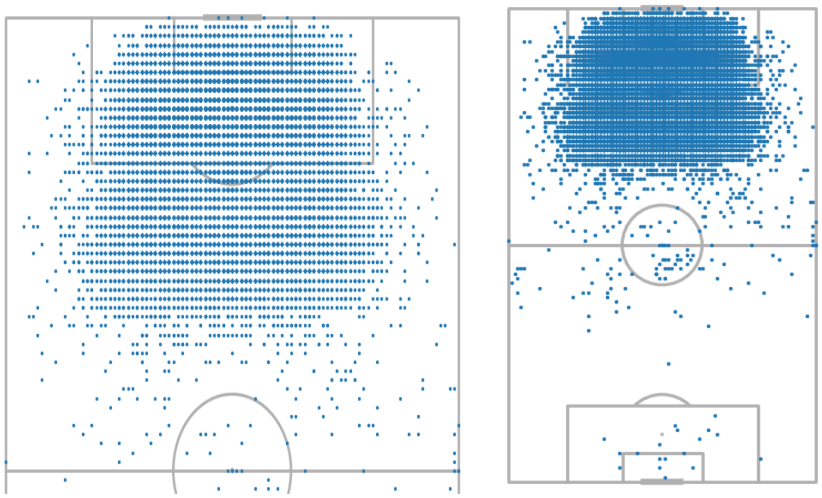
The histograms display shot frequency over time in football matches' first and second halves. They show peaks at the beginning and end of each half, indicating heightened attacking play, while tapering off in the middle as teams possibly adopt a more cautious approach. The sparse events during "Extra Time" align with its shorter duration. These insights aid in strategizing defensive and attacking tactics throughout a match.

Bar Graph (Goals scored by top players)



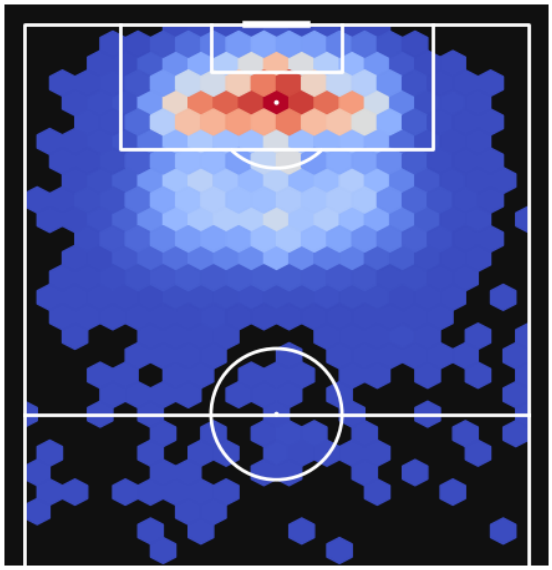
The bar chart compares goals scored and shots without goals for top players, aiding in assessing goal-scoring efficiency and shooting activity. It helps identify players with higher conversion rates and those who take more attempts but with lower success rates, informing performance analysis and tactical strategies.

Pitch Plot (Shot Distribution)



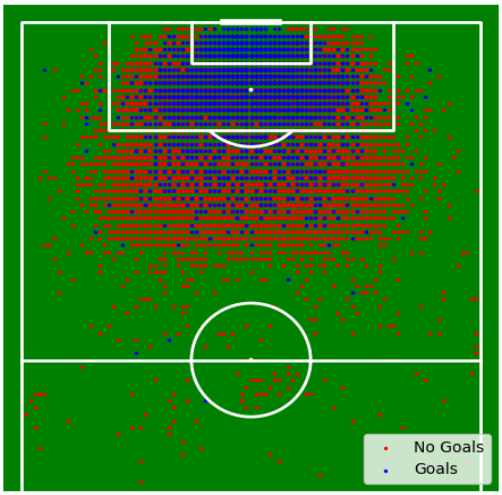
Each dot on the plot corresponds to the location of a shot, with the density of the dots giving an indication of which areas shots are most frequently taken from. A higher concentration of dots around the goal area would suggest that more shots are taken close to the goal, which is consistent with football tactics. Such visualizations are crucial for Expected Goals (xG) models, as they provide insights into shooting patterns and preferred shooting zones, which can help in understanding the probability of a shot resulting in a goal from different locations on the pitch.

Heatmap (Shot Distribution)



The is a heat map overlaid on a soccer field, representing the locations from where shots are taken. The intensity of the colors likely corresponds to the frequency of shots from those areas — warmer colors (red to yellow) indicate higher shot frequency, while cooler colors (blue) indicate fewer shots. This type of visualization is useful for analyzing spatial patterns in shot-taking, such as which areas of the field yield more attempts on goal. For example, a concentration of warmer colors around the penalty area suggests a high volume of shots taken from that region, which is expected as it's closer to the goal. Such insights can inform teams about defensive strengths or weaknesses and offensive strategies

Shot Map (Goal or NO Goal)



From this visualization, one can infer where most shots are taken, and which areas have a higher success rate. For instance, if the blue dots are concentrated around the goal area, it would indicate that shots taken close to the goal are more likely to result in goals, which is a common finding in soccer analytics. This type of data is invaluable for teams to develop tactical strategies and for coaching staff to understand patterns in gameplay.

5. Data Mining Models

Selecting the right machine learning model is critical for building an effective predictive model for estimating expected goals (xG) in football shots. In this analysis, we'll evaluate several data mining models and methods, considering their applicability, strengths, and limitations in relation to our dataset. By understanding how each model aligns with our objectives, we aim to choose the most suitable approach for predicting goal outcomes in football shots.

1. Logistic Regression

- Applicability: Well-suited for binary classification tasks like goal or no goal prediction. It can handle both numerical and categorical features present in your dataset, such as player attributes and match details.
- Pros: Simple and interpretable, which can be advantageous for understanding the factors influencing goal outcomes. It's also computationally efficient and doesn't require extensive tuning.
- Cons: Assumes a linear relationship between the features and the log odds of the target variable. It may not capture complex interactions between features without extensive feature engineering.

2. Decision Trees

- Applicability: Capable of handling both numerical and categorical data, making it suitable for our mixed dataset. Decision trees can capture non-linear relationships between features and the target variable.
- Pros: Easy to interpret and understand, making it valuable for extracting insights into the factors affecting goal outcomes. Requires minimal data preprocessing compared to some other models.
- Cons: Prone to overfitting, especially with complex datasets or deep trees. It may not generalize well to unseen data without careful pruning or regularization.

3. Random Forest

- Applicability: Particularly well-suited due to its ability to handle mixed data types and its robustness to overfitting compared to single decision trees.
- Pros: Can model complex relationships between features and the target variable, providing high predictive accuracy. Random forests are less prone to overfitting than individual decision trees and require less parameter tuning.
- Cons: Less interpretable than decision trees, as it's a collection of multiple trees. Training can be computationally intensive for large datasets, but it's generally manageable.

4. Gradient Boosting Machines (GBM)

- Applicability: Effective for structured datasets, where capturing complex relationships between features and the target variable is crucial.

- Pros: Offers high predictive accuracy by iteratively improving on the weaknesses of decision trees.
- Cons: Requires careful hyperparameter tuning to prevent overfitting. Training can be computationally expensive, especially with large datasets, but it often pays off in improved performance.

5. Support Vector Machines (SVM)
- Applicability: Can be used for binary classification tasks like goal prediction after appropriate preprocessing of our dataset.
- Pros: Effective in high-dimensional spaces and versatile with different kernel functions, which can capture complex relationships between features and the target variable.
- Cons: Requires significant data preprocessing, such as scaling features, and may not perform well with large datasets due to its computational complexity. Choosing the right kernel and tuning hyperparameters can also be challenging.

6. Neural Networks
- Applicability: Suitable for capturing complex and non-linear relationships present in our dataset, potentially improving prediction accuracy.
- Pros: Highly flexible and capable of modeling very intricate patterns in the data. Neural networks can automatically learn relevant features from the data, reducing the need for extensive feature engineering.
- Cons: Requires a large amount of data to train effectively and prevent overfitting. Neural networks are computationally intensive and may require significant resources for training, especially with deep architectures.

7. k-Nearest Neighbors (k-NN)
- Applicability: Simple to apply and can make predictions based on similarities between data points in the feature space.
- Pros: Easy to understand and implement, making it useful for quick prototyping. It doesn't require model training and can adapt well to changes in the data over time.

- Cons: Computationally expensive as the dataset grows, especially during prediction time. Performance heavily depends on the choice of distance metric and the value of k, which may require experimentation.

Selection Criteria

- Data Complexity and Size: Considering the complexity of relationships and the size of our dataset, models like Random Forest, GBM, or Neural Networks might offer better predictive performance.
- Interpretability Needs: If interpretability is crucial for our analysis, simpler models like Logistic Regression or Decision Trees might be preferred.
- Computation Resources: For limited computational resources, Logistic Regression or Decision Trees are more suitable. However, if resources allow more complex models like Random Forest or Neural Networks could yield better results.