1. Introduction

In this project, we developed a machine learning model to predict Expected Goals (xG) in football. The xG metric quantifies the likelihood of a goal being scored from a particular shot based on various factors such as shot location, angle, and body part. By leveraging historical data and advanced machine learning techniques, the model aims to provide more accurate predictions of goal-scoring opportunities, offering valuable insights for teams, analysts, and fans to better understand match dynamics and player performance. We used data of the 2022 World Cup provided by Hudle StatsBomb and retrieved thanks to the *mplsoccer* library.

"In this report, it is football, not soccer."

2. Method

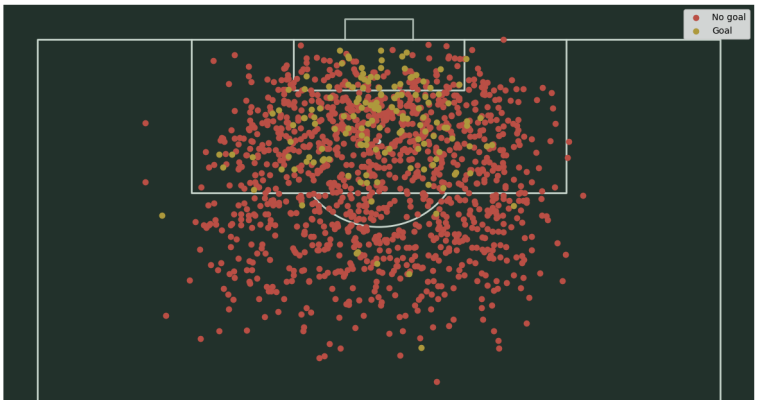
2.1 Data Collection

Hudl Statsbomb is a leading company in the sports data industry, particularly in football. In recent years, the company has released data from major tournaments. For this project, we are focusing on the 2022 FIFA World Cup in Qatar (WC), the last tournament with 32 teams and 64 matches, as the 2026 World Cup will feature 48 teams. While the size of the dataset is substantial, there is potential for more data from other competitions. However, it's important to recognize that international football differs from club football, so any additional data should be from the same context to ensure consistency. The dataset includes nearly 4,000 events per match, with approximately 20 events are shots. In total, there are almost 1500 shots in the tournament ($n=1,453$).

We collected the data from StatsBomb using the *mplsoccer* library. This allowed us to access all matches from the World Cup, retrieve their match IDs, and extract the corresponding events for each match.

```
# iterate through all matches to get the events data
df_matches = {}
for i, id in enumerate(df_match['match_id']):
  df_matches[id] = {}
  df_matches[id]['event'], df_matches[id]['related'], df_matches[id]['freeze'], df_matches[id]['tactic'] = parser.event(id)
```

Using matplotlib and mplsoccer we can see all the shots in the tournament



2.2 Preparing Data

To prepare the data, we created a new DataFrame containing only the features required for training the model. The selected features, which we believe have the most significant impact, include:

- Location (x, y)
- Outcome (e.g., saved, goal, etc.)
- Type (e.g., open play, corner, etc.)
- Body part (e.g., head, right foot, etc.)
- Under pressure (true or false)
- Technique (e.g., volley, normal, etc.)
- StatsBomb xG (a value provided by StatsBomb's model)

By including the xG values from the more advanced StatsBomb model, we can compare our results directly against their predictions.

Next, we filtered the data to include only shots taken during the first four periods: the 1st half, 2nd half, 1st extra time, and 2nd extra time. This step is crucial because StatsBomb data also includes shots from penalty shootouts, which could distort the model's understanding of typical match scenarios.

```
new_features = ['x', 'y', 'outcome_name', 'sub_type_name', 'body_part_name', 'under_pressure', 'technique_name', 'shot_statsbomb_xg']
df_shot = pd.DataFrame(columns=new_features)

for id in df_match['match_id']:
  mask_shot = (df_matches[id]['event'].type_name == 'Shot') & (df_matches[id]['event'].period <= 4)
  shots_temp = df_matches[id]['event'].loc[mask_shot, new_features]
  df_shot = pd.concat([df_shot, shots_temp]).reset_index(drop=True)
```

In football, the angle and distance from which a shot is taken are critical factors in determining the likelihood of scoring a goal. A wider shooting angle increases the chances of scoring, as the goalkeeper has more area to cover. Similarly, shots taken closer to the goal are generally more likely to result in goals due to the goalkeeper's reduced reaction time.

The angle is calculated by measuring the space between the shooter's position and the two goalposts, using mathematical operations to determine the size of the angle. This value is then converted to degrees for easier interpretation, ensuring the result is always positive. Similarly, the distance is calculated by measuring the straight-line distance from the shooter's position to the closest point on the goal line, accounting for whether the shot is above, below, or aligned with the goal area.

```
def calculate_angle(x, y):
  # 44 and 36 is the location of each goal post
  g0 = [120, 44]
  p = [x, y]
  g1 = [120, 36]

  v0 = np.array(g0) - np.array(p)
  v1 = np.array(g1) - np.array(p)

  angle = np.math.atan2(np.linalg.det([v0,v1]),np.dot(v0,v1))
  return(abs(np.degrees(angle)))
```

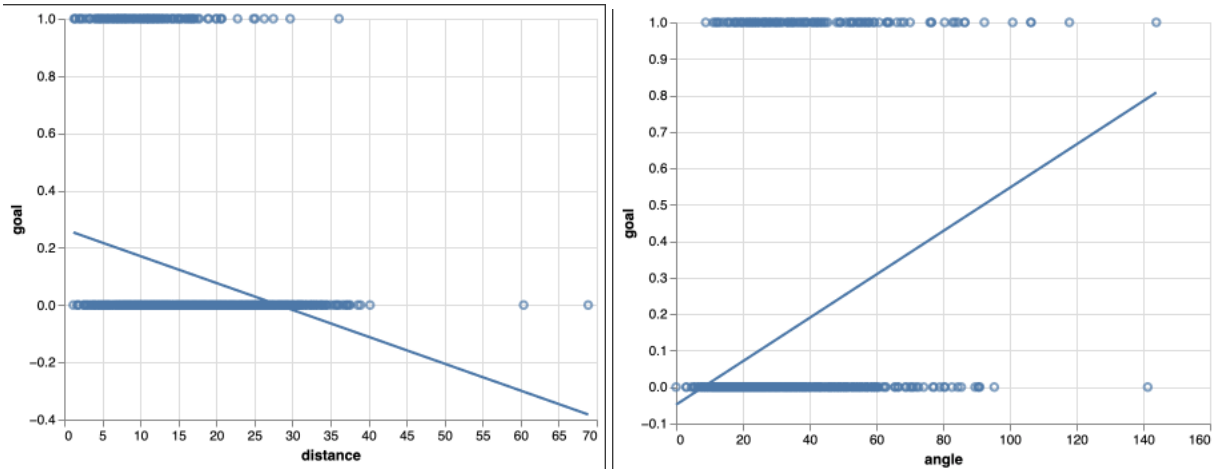
```
def calculate_distance(x, y):
  x_dist = 120-x
  y_dist = 0
  if (y<36):
    y_dist = 36-y
  elif (y>44):
    y_dist = y-44
  return math.sqrt(x_dist**2 + y_dist**2)
```

Then, we added two new columns to the dataset. One column indicates whether the shot was a header: 0 if it wasn't a header and 1 if it was. Similarly, another column classifies the outcome of the shot: 0 for shots that did not result in goals and 1 for those that did.

We transform categorical data into numerical form to prepare the dataset for model training. The technique name and type name columns are converted into dummy variables, creating separate columns for each unique value in those categories, with binary values indicating the presence of each technique or sub-type. The under pressure column is first filled with 0 for missing values and then converted to integers (0 or 1) to represent whether a shot was taken under pressure, making it easier for the model to process.

2.3 Data Analysis

In the analysis, we used Altair to visualize the relationship between shot angle, distance, and the likelihood of scoring a goal. First, we plotted a scatter plot of shot angle versus goal outcome, then added a regression line to model the relationship. Similarly, we created a scatter plot for shot distance versus goal outcome and calculated the correlation between these features and the goal outcome. The correlation coefficients for both shot angle and distance were computed to assess their impact on scoring likelihood. The correlation for distance was -0.25 and 0.31 for the angle, suggesting that the angle of a shot has a stronger influence on the likelihood of scoring than the distance from the goal.



The first chart shows the relationship between shot distance and the likelihood of scoring. The regression line slopes downward, indicating that the probability of scoring decreases as the distance increases. Most shots, especially from longer distances, result in no goal, though a few rare outliers show goals from farther away. This suggests that closer-range shots are more effective.

The second chart illustrates the relationship between shot angle and goal probability. The positive slope of the regression line shows that the likelihood of scoring increases with the angle. However, most data points are clustered at 0 (no goal), suggesting that other factors also play a key role in scoring. The few goals from difficult angles highlight that while it's possible to score from tough positions, it's still uncommon.

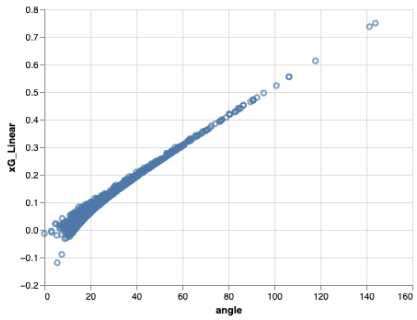
2.4 Modelling

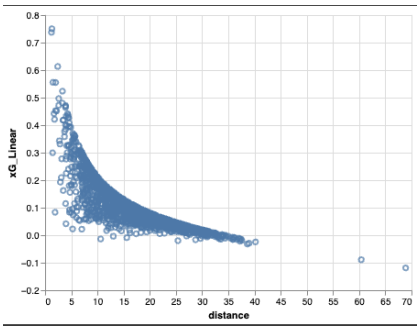
The first step is to select the model we want to use. In this case, we will train both Linear and Logistic regression using the sklearn library to determine which one performs best. For simplicity, we will use distance and angle as our features (X). We will train both models and evaluate their performance by calculating the R^2 score to see how each model responds to the data.

```
[19] from sklearn.linear_model import LinearRegression
     from sklearn.linear_model import LogisticRegression

     model_names = ['Linear', 'Logistic']
     models = {}
     models['Linear'] = {}
     models['Linear']['model'] = LinearRegression()
     models['Logistic'] = {}
     models['Logistic']['model'] = LogisticRegression()

[20] X = df_shot[['angle', 'distance']]
     y = df_shot['goal']

[21] from sklearn import metrics
     for mod in model_names:
       models[mod]['model'].fit(X, y)
       if mod == 'Logistic':
         models[mod]['y_pred'] = models[mod]['model'].predict_proba(X)[:, 1]
       else:
         models[mod]['y_pred'] = models[mod]['model'].predict(X)

       models[mod]['r2_score'] = metrics.r2_score(y, models[mod]['y_pred'])
       print("R2 of model {}: {}".format(mod, models[mod]['r2_score']))
```

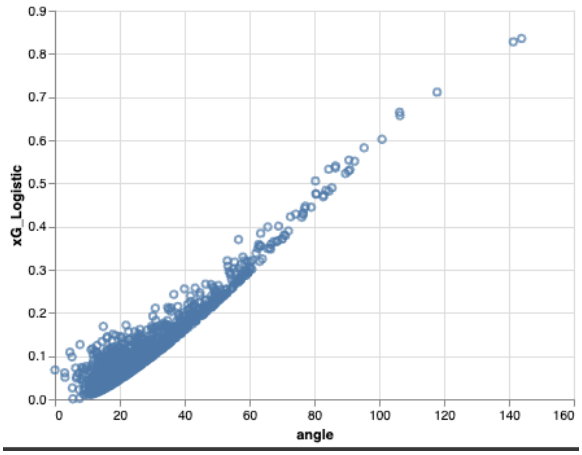
The R^2 score of the Linear model is 0.099, and the R^2 score of the Logistic model is 0.096. These low R^2 values suggest that neither model explains much of the variance in the data, meaning their predictions don't closely match the actual outcomes. In comparison, the Statsbomb xG model has an R^2 score of 0.20, indicating that we have room to improve.

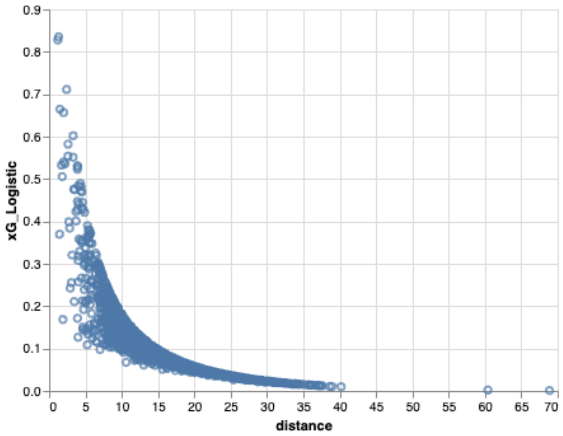
To choose the best model, we need to dig deeper into the analysis. First, the Linear regression model produces at least 45 negative xG values, which is not possible because expected goals (xG) cannot be negative regardless of the shot's distance or angle. In the first image, we can see the relationship between shot angle and the xG predicted by the Linear model. The second image shows the angle again, helping to understand the model's prediction pattern and how the shot angle correlates with the expected goal value. The negative values in the Linear regression model are a red flag, indicating that this model is not appropriate for predicting xG, as xG values should always fall between 0 and 1. Since the model produces impossible values, we cannot rely on it for this type of prediction, and thus it is not suitable for our analysis.





For the Logistic Regression model, we don't encounter negative values, which is a positive aspect since xG values should lie between 0 and 1. Next, we analyze the relationship between angle and distance to assess if there is a correlation, which will help us understand how these features contribute to the model's prediction of xG.





Now that we choose the model we are using, is time to selected a set of features from the dataset to use as input variables for the model. The features chosen include factors such as whether the shot was taken under pressure, the shot angle, distance from the goal, the player's preferred side, whether the shot was a header, and different types of techniques and shot scenarios like open play or penalties. We created a new subset of the data containing only these selected features, which will be used to train the model. The first few rows of this subset were displayed to ensure the data was properly prepared for the next steps in the analysis.

```
X_cols = ['under_pressure', 'angle', 'distance',
       'preferable_side', 'header', 'technique_name_Backheel',
       'technique_name_Diving Header', 'technique_name_Half Volley',
       'technique_name_Lob', 'technique_name_Normal',
       'technique_name_Overhead Kick', 'technique_name_Volley',
       'sub_type_name_Corner', 'sub_type_name_Free Kick',
       'sub_type_name_Open Play', 'sub_type_name_Penalty']
X = df_shot[X_cols]
```

In this analysis, the target variable we are trying to predict is whether the shot resulted in a goal or not. This variable, labeled as 'goal', takes a value of 1 if the shot was a goal and 0 if it was not. It represents the outcome of the shot, which we aim to predict based on the features we selected earlier.

The code trains a logistic regression model on the features (X) to predict the likelihood of a shot resulting in a goal (y). The predict_proba(X) method returns the predicted probabilities for both classes (goal or no goal), and by selecting [:, 1], we extract the probability of the shot being a goal. With these new features, the R^2 score improves to 0.19, showing a significant enhancement in the model's ability to explain the variance in the data.

```
adv_model = LogisticRegression()
adv_model.fit(X, y)
y_pred = adv_model.predict_proba(X)[:, 1]
metrics.r2_score(y, y_pred)
```

3. Results

3.1 Evaluaitng in All Matches

To evaluate all matches and calculate expected goals (xG) for each shot, we followed a structured approach. The first step involved preparing the data. We created a ‘df_summary’ DataFrame, which contains essential match details such as ‘match_id’, ‘home_team_name’, and ‘away_team_name’. This DataFrame serves as a repository for the results of our evaluation for both the home and away teams.

Next, we looped through each match using a loop over the ‘df_match’ DataFrame. For every match, we retrieved the event data that includes all shot attempts. This was achieved by filtering for "Shot" events in the ‘df_evaluate’ DataFrame for each match ID. This gave us a dataset focused solely on shots taken during the match.

To ensure we were only considering relevant data, we applied an additional filter to focus only on shots taken during open play. This was done by excluding shots from periods greater than 4. By doing so, we ensured that our analysis was based on regular gameplay only, aligning with the standard xG calculations.

With the data filtered, we then proceeded to calculate the xG for each shot using the ‘calculate_xg_adv’ function. This function takes various shot features, such as the angle, distance, body part used, technique, and shot type, and passes them into the logistic regression model. The model then predicts the likelihood of the shot resulting in a goal, providing an xG value for each attempt.

After calculating the xG for each shot, we separated the shots by the home and away teams. For each team, we calculated three key statistics: the number of goals scored (by counting shots with the outcome "Goal"), the total xG (by summing the individual xG values for all shots taken by the team), and the StatsBomb xG (by summing the StatsBomb-provided xG values for comparison).

Once the calculations were completed, we stored the results in the ‘df_summary’ DataFrame. The results were saved under columns for the home and away teams, including ‘home_goal, home_xg’, ‘home_xg_sb’, ‘away_goal, away_xg’, and ‘away_xg_sb’.

Finally, after the loop finished processing all matches, the ‘df_summary’ DataFrame contained a summary of the statistics for each match. These statistics included the goals scored by each team and the xG values, both from our model and StatsBomb, for both home and away teams.

This approach allows us to assess the performance of the xG model for each shot across all matches, providing insights into the model's accuracy and highlighting any discrepancies between expected and actual goals scored.

In these image we can see 10 matches of the WC and our results compared with the score and the Statsbomb model.

	match_id	home_team_name	away_team_name	home_goal	home_xg	home_xg_sb	away_goal	away_xg	away_xg_sb
0	3857256	Serbia	Switzerland	2	1.493781	1.189004	3	2.483093	3.103515
1	3869151	Argentina	Australia	2	1.324700	1.481579	0	0.586725	0.426118
2	3857257	Australia	Denmark	1	0.404915	0.469723	0	1.326828	0.737155
3	3857258	Brazil	Serbia	2	1.960283	2.123890	0	0.209117	0.163327
4	3857288	Tunisia	Australia	0	1.316651	1.052170	1	0.490532	0.359038
5	3857267	Ecuador	Senegal	1	1.102636	1.001663	2	1.751185	1.707465
6	3869321	Netherlands	Argentina	2	0.693612	0.569538	2	1.622771	1.939197
7	3857287	Uruguay	South Korea	0	0.634220	0.417294	0	0.609650	0.492993
8	3869486	Morocco	Portugal	1	0.848070	0.972023	0	0.847005	0.744121
9	3869685	Argentina	France	3	2.775122	2.758306	3	1.928040	2.272618

3.2 Results Analysis

The results of the analysis demonstrate the performance of the custom logistic regression model for predicting expected goals (xG) and its comparison with StatsBomb's xG values. The model produced realistic xG predictions, as it did not generate negative values, which would be illogical in the context of football. By analyzing the total goals scored and comparing the total xG values for both teams (home and away) across all matches, it became clear that while the model showed promise, there were occasional discrepancies between the predicted xG and actual goals.

When comparing the model's predicted xG to the StatsBomb xG values, we observed that StatsBomb's predictions generally aligned more closely with the actual goals scored. This indicates that StatsBomb’s model may have captured more complex nuances of shot outcomes that our simpler logistic regression model might have missed, such as defensive pressure or player positioning. The custom model, while fairly accurate, showed minor over- or under-predictions, which might be due to the limitations of the features used in the regression.

In terms of model accuracy, using metrics like total xG per team and the number of goals scored, our model performed reasonably well but showed a slight gap when compared to StatsBomb's more refined approach. These differences suggest areas for improvement, such as including additional features (e.g., shot velocity, match context, player positioning) and tuning the model further.

4. Conclusion

Overall, the custom logistic regression model provided a good starting point for predicting expected goals (xG) based on shot features, but it is clear that there is significant room for enhancement. The comparison with StatsBomb's xG highlights the potential of the model while also identifying areas where it could be improved. In particular, incorporating more granular features related to the shot (e.g., defensive pressure, player positioning, shot technique) could improve the model's predictive power.

While the custom model has potential, StatsBomb's xG, which is based on a much larger dataset and more sophisticated algorithms, performed better in terms of predicting actual goal outcomes. The custom model's ability to avoid negative xG values is an advantage, but further development is needed to match the accuracy and reliability of StatsBomb.

In conclusion, although the model is a promising tool for evaluating shot quality and providing insights into team and player performance, it still has limitations that need to be addressed. Future improvements to the model, including more complex feature sets and advanced machine learning techniques, would enhance its ability to predict goals and offer more accurate insights for football analysis.

Bibliography

Simulating a football season. (2023, October 31). IB Maths Resources From Intermathematics. https://ibmathsresources.com/2019/06/19/simulating-a-football-season/

User Guide — pandas 2.2.3 documentation. (n.d.-b). https://pandas.pydata.org/docs/user_guide/index.html

GeeksforGeeks. (2024, June 20). *Logistic regression in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/understanding-logistic-regression/

Jidge, A. (2022, March 30). The Complete Guide to Linear Regression Analysis - towards Data Science. *Medium*. https://towardsdatascience.com/the-complete-guide-to-linear-regression-analysis-38a421a89dc2

Hudl Statsbomb, Data Champions. (2024, October 2). *What is xG? How is it calculated? | Hudl Statsbomb | Data Champions*. https://statsbomb.com/soccer-metrics/expected-goals-xg-explained/

Quick start — mplsoccer 1.4.0 documentation. (n.d.). https://mplsoccer.readthedocs.io/en/latest/index.html