Investigation into the relationship between the shot location and probability of scoring in football

Topic: Math AI Internal Assessment

Page count: 20

Session: May 2023

Contents page:

Introduction

With an anticipated 5 billion spectators of the World Cup final this year, football is the most popular sport in the world (2022). Having played football since I was four years old and watched it long before that, it has always been my biggest interest. As I grew up, I became fixated on the debate over who was the best player in the world and used numerous statistics to support my claims. Expected goals (xG) can be seen as the statistical measure of the "quality" of a chance or shot. It uses data in order to assess the probability that on a normal day of football with an average player relative to the dataset a shot results in a goal from a specific location. The xG takes numerous things into consideration when calculating the probability of scoring a shot such as shot location, pressure from other players on the shooter (how near opponents are to the shot), shot type (whether the shot was a header, volley or ground shot) and type of assist. Using this information, an expected goal value is calculated for each shot in a game. xG can be used by teams to inform players of which situations to take a shot from, and is therefore very important in how football is played in the modern day. In order to further my personal knowledge on the field, this exploration will seek to model the probability of scoring a shot in football from a specific location (xG).
The data used in this exploration is from Wyscout and is specifically used due to the size of sample as well as the extent of the data. Wyscout is the biggest public collection of football logs that has spatiotemporal events. It contains all events (shots, passes, goals and misses) and the location at which they occurred, for every single game in the Premier League season 2018/19 season. The aim of this project is to model the relationship between location in which a shot is taken and the probability of the shot resulting in a goal and to compare the model to final results from g ames. Various mathematical methods will be used in order to model the probability and visualise the data that is attained. Logistic regression, used in order to model the probability of scoring based on location.

Context/background & data collection

The sample of data was collected by Wyscout, but publicly released by Luca Pappalardo and Emanuele Massucco on Figshare. Each event that occurred was coupled with a type of event (assessing the shot type or pass type) as well as the position on the pitch measured in X and Y coordinates. Where the X axis was measured as the vertical distance of the position from the baseline and the Y measured as the distance from the centre of the goalline, for the team that was in possession of the ball when the event occurred.

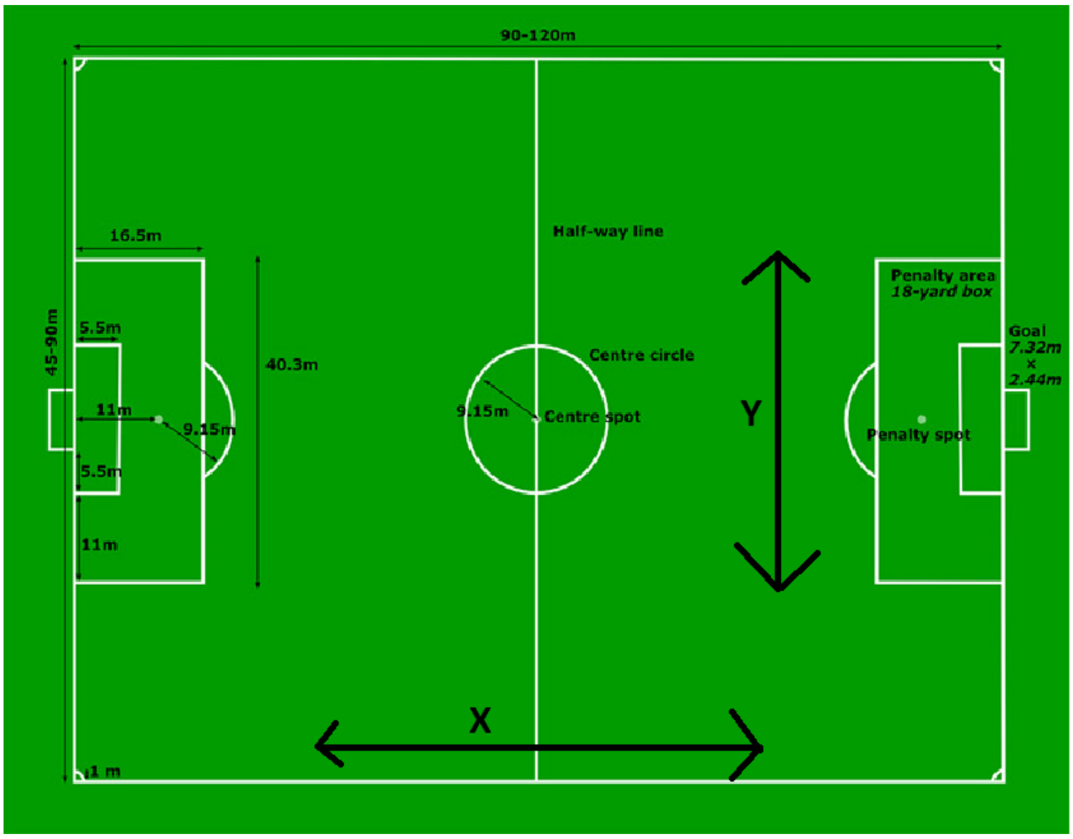

Figure 1: Representation of the way wyscout assesses data with reference to field

An important factor to consider when looking at the statistics is if this makes sense from a football perspective. Such as, an assumption that can be made simply by understanding football would be that generally when the distance to the goal is smaller then the probability of the goal being scored is higher. However, the relationship between distance and probability of scoring could be misrepresented by as much as a "lucky goal" in a low probability of scoring position where not many people happen to take shots.

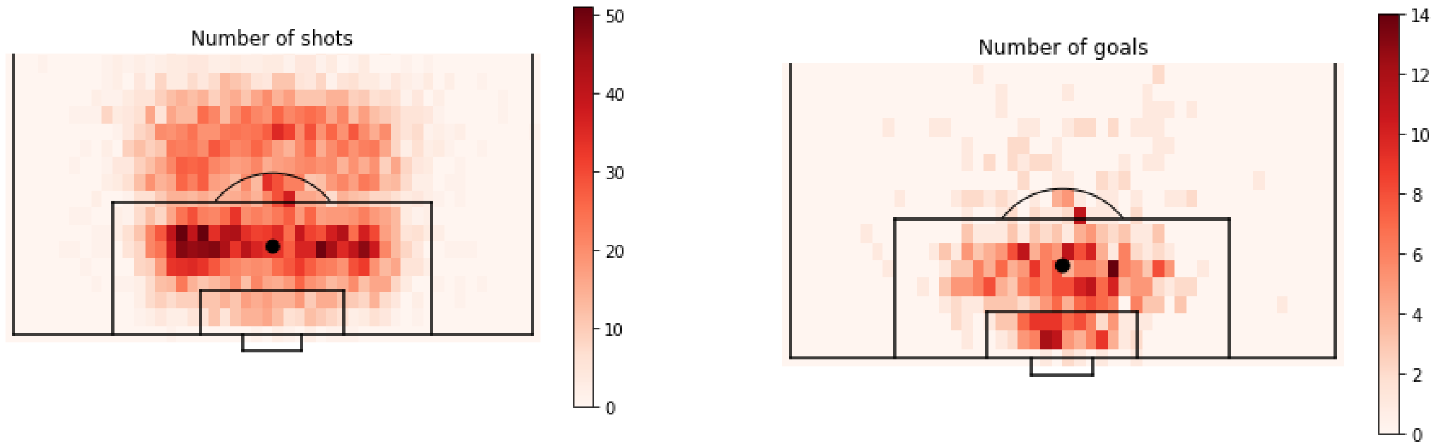

Figure 2: Comparison between where the most shots are taken from and goals are scored

Here it can be seen the number of shots that were taken from various locations in the wyscout dataset. Using these figures, one can plot the proportion of shots from each location that result in a goal by dividing the number of goals scored by the number of shots taken from the location:

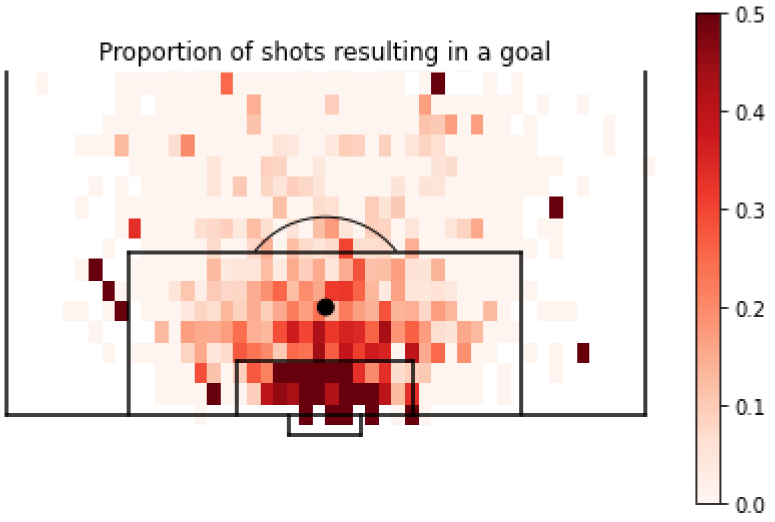

Figure 3: Proportion of shots that result in a goal (goals/shots)

As shown by figure 3, generally more shots are converted closer to the goal, however, there are some outliers that exist due to a "fluke" or a lucky shot. These are constaints of having used only one season worth of premier league football. As with a bigger dataset flukes like this won't be as common.

<u>Distance and angle:</u>

From understanding football, it can be assumed that generally, the likelihood of converting a shot into a goal is higher the closer you are to the goal. Meaning that the distance is a key factor when modelling the probability of scoring based on location. But what it fails to consider is the angle to the goal at which the shot is taken from, in other words "how much of the goal you can see". If the model were to rely entirely on the distance to the goal it would misrepresent the likelihood of scoring a goal on the baseline. As shown in the leftmost event in figure one, although the distance to the goal is at a shorter range to that of the event to the right of it, it has a statistically lower chance of being converted because the ball can "see" less of the goal.

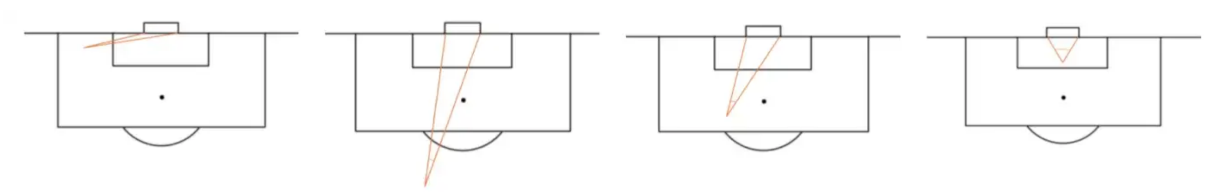

Figure 4: Depiction of angle from different shooting locations[1]

The angle at which a shot is taken shows the relationship between location and shot conversion better than the pure distance from the goal. The angle of a shot will increase with a decrease in distance to the goal. As seen in figure 7 the distance between the goal and shot in event 1 is higher than that in event 2, and correspondingly the angle Θ is smaller. Therefore, the angle of Θ better represents the probability of shot-conversion because it considers both the distance as well as how much of the goal the ball can "see".

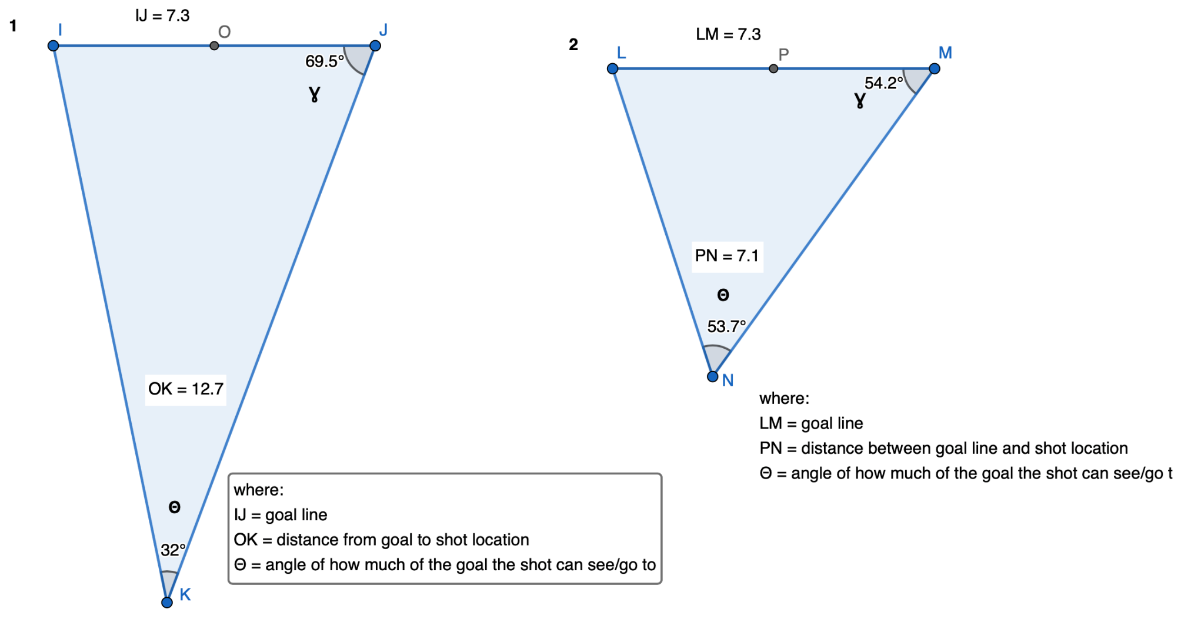
1
I
IJ = 7.3
O
J
69.5°
ɣ
OK = 12.7
Θ
32°
K
where:
IJ = goal line
OK = distance from goal to shot location
Θ = angle of how much of the goal the shot can see/go to
2
L
LM = 7.3
P
M
54.2°
ɣ
PN = 7.1
Θ
53.7°
N
where:
LM = goal line
PN = distance between goal line and shot location
Θ = angle of how much of the goal the shot can see/go t

Figure 5: How angle affects the distance a goal is scored at.

Logistic regression

Logistic regression uses a type of statistical model (also known as *logit model*) that is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

[1] Dragulet, Ian. "An Introduction of Expected Goals." *Medium*, Towards Data Science, 1 Jan 2021, https://towardsdatascience.com/a-guide-to-expected-goals-63925ee71064. Accessed 3 January 2023.

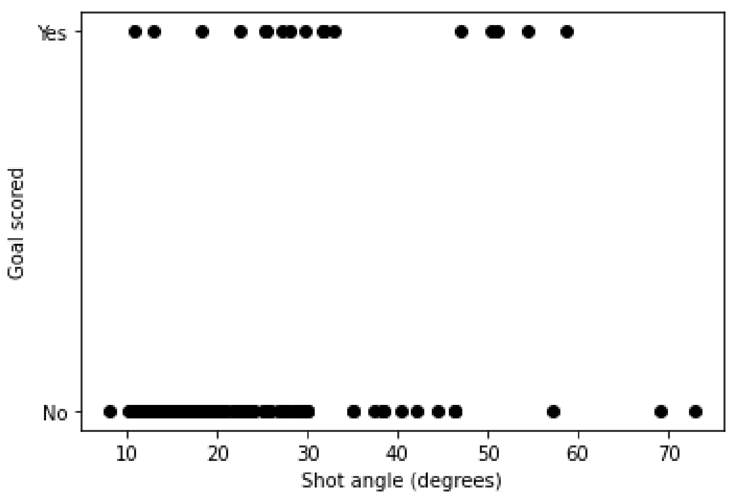

Figure 6: Dataset of first 60 shots plotted on the binary outcome and angle of the shot

Considering the dataset seen in figure 1, the variable on the y axis is on nominal scale and has two categories, either the shot was scored (yes) or the shot was missed (no). In binary logistic regression, there is a single binary dependent variable, coded by an indicator variable (indication if event occurred or not, in this case 1 and 0) and the independent variables might be binary variables or continuous variables (any real value), in the case of this investigation it will be a continuous variable. Probability is defined as the ratio of occurred favourable outcomes to that of the total amount of outcomes (or trials).

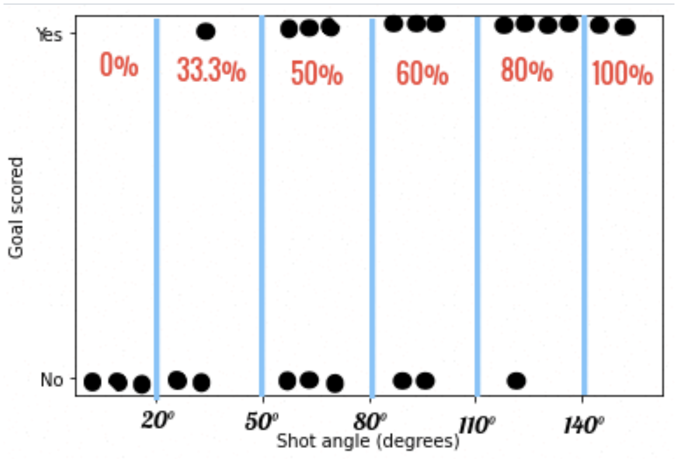

Figure 7: Probability of scoring based on shot angle (made up data)

In splitting a data set such as the one as seen above in figure 2, the probability of scoring based on the data for that specific sector of the graph can be determined. Where in the first sector between 0° and 30° there are only missed shots meaning based on the data there is a 0% chance of scoring the goal. Whereas in the second sector between 30° and 60° there is a goal scored out of a total of 3 attempts, subsequently the probability of scoring is 33.3%, and so on for the remaining sectors. These probabilities can be used to model a function that represents the probability of an event occurring based on an independent variable as seen below in figure 3. For example, based on figure 3, if a shot is taken with a shot angle of 80°, the probability of scoring is close to 50%.

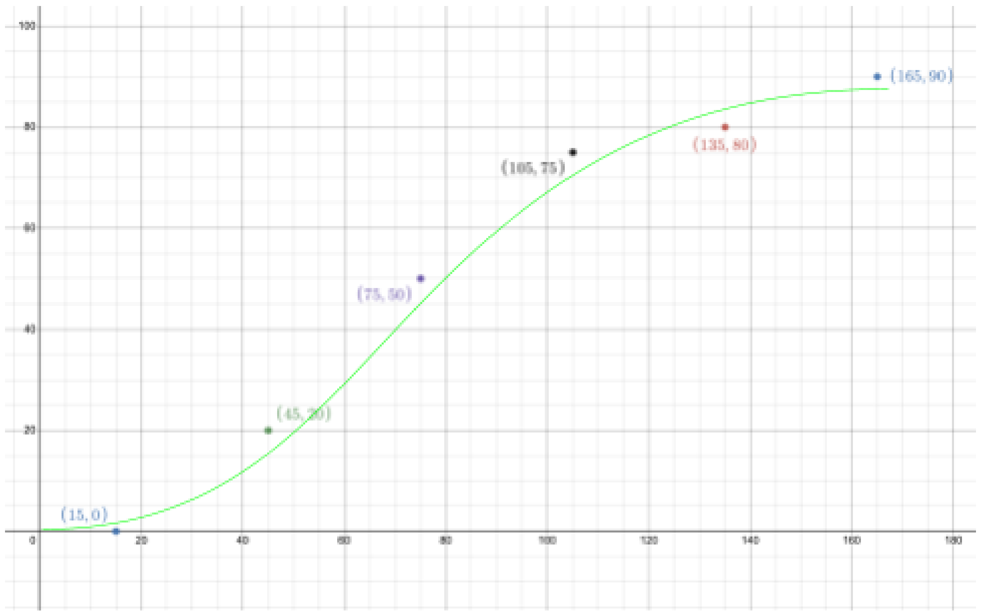

Figure 8: Graph representing a logistic curve fitting to the data points (logistic curve drawn by hand and is used as a visualisation).

Logistic regression uses previous observations from a data set to predict a binary outcome, such as yes or no. By examining the correlation between one or more already present independent variables (shot location), a logistic regression model forecasts a dependent data variable(probability of converting the shot into a goal). Logistic regression is preferable to linear regression when modelling probability because it is designed to model binary outcomes, while linear regression is designed to model continuous outcomes. Logistic regression uses a logistic function (sigmoid function) to model the probability of a binary outcome, which is better suited to this type of data. Additionally, logistic regression can be extended to model multi-class outcomes, while linear regression cannot. The formula that models the probability of a shot resulting in a goal is denoted by the function:

$$p(y) = \frac{1}{1 + e^{(b_0 + b_1 \cdot x)}}$$

$$\begin{gathered} where: \\ p(y) = the\ probability\ that\ the\ output\ is\ equal\ to\ 1\ (or\ a\ goal)\ given\ the\ input\ features\ x, \\ e\ = the\ base\ of\ the\ natural\ logarithm\ (approximately\ 2.718) \\ b\ =\ the\ log - odds\ or\ logit, the\ model\ parameters. \\ x =\ the\ angle\ that\ the\ shot\ location\ makes\ to\ the\ goalposts \end{gathered}$$

Log-odds

The parameters (b_0, b_1) and the maximum likelihood estimation will be calculated by the python library module statsmodel.families.Binomial()).fit(), for computational ease. Log-odds, also known as logit, is the natural logarithm of the odds of an event occurring (the ratio of the probability of an event occurring to the probability of it not occurring). Log-likelihood, on the other hand, is a measure of the fit of a statistical model to a set of data. It is the logarithm of the likelihood function, which is a function that describes the probability of observing a particular set of data given the parameters of a model. Log-likelihood is used in model selection and parameter estimation to compare the fit of different models or to find the best-fitting parameters for a given model. The difference between the two is, log-odds are used to model the probability of a binary outcome, while log likelihood is used to evaluate the fit of a model to a set of data.

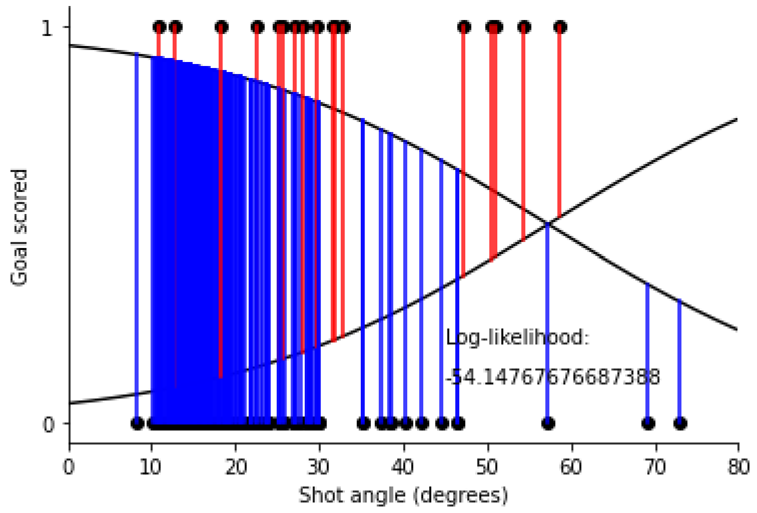

Figure 9: Log odds & Log likelihood based on the dataset that it was given (in this case only a training set of the first 200 shots)

. For the purpose of this example imagine the first 10 shots of the training set are taken and could be represented by the following:

Table 1:

Shot (#)	Status (goal or no goal)	Angle (x) ○	Probability	$odds = \frac{p}{1-p}$	$ln\left(\frac{P}{1-P}\right)$

1	No	20	0.025	0.2564102564	-3.663561646
2	No	70	0.4	0.6666666667	-0.4054651081
3	Yes	120	0.78	3.5454545454	1.265666373
4	Yes	110	0.73	2.703704	0.9946225751
5	No	100	0.67	2.03030303	0.708150579
6	No	50	0.2	0.25	-1.386294361
7	No	90	0.59	1.43902439	0.3639653772
8	Yes	80	0.5	1	0
9	Yes	140	0.85	5.6666666667	1.734601055
10	No	75	0.45	0.8181818182	-0.2006706955

As an example, the function in figure 8 can be used to predict the probability of scoring depending on shot angle. This probability is then applied to calculating the odds of the shot. These odds can be thought of as in the case of shot number 6, the probability of scoring is 20% meaning that the odds is equal to 0.2/(1-0.2). The odds of the function is 0.25, meaning that the likelihood of the event occurring is 0.25 times more likely than not occurring. This can also be thought as for every 4 times the event does not occur (shot is missed) there will be 1 event that occurs (shot is scored).

The natural log of these odds produces the log-odds, which can lie anywhere within positive and negative infinity, whereas, odds is constrained to positive infinity and probability to between 0 and 1. It can also be noted that if the odds of a shot is above 1 it will result in a positive log-odds value, whereas, if it is below 0 it is negative, also as seen in the case of shot number 8, is the odds is equal to one it results in a log-odds of 0. If these log-odds are plotted with relation to the angle of the shot it is possible to see the relationship between how the log-odds change with a change in angle.

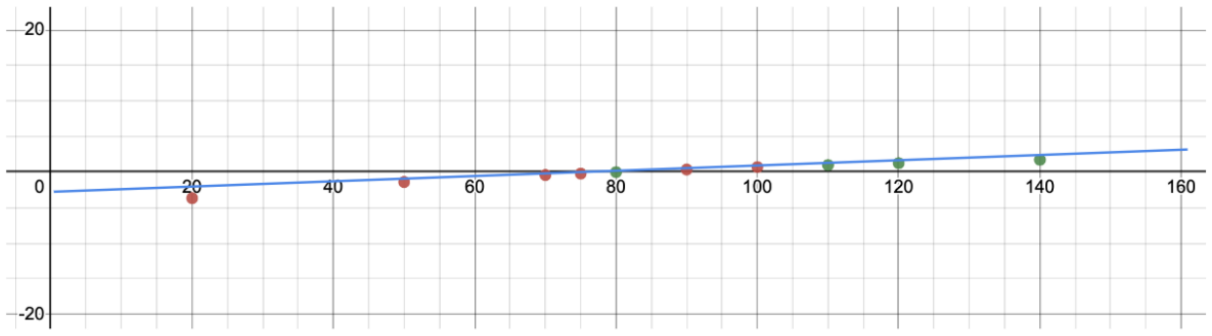

Figure 10: Representation of log-odds as shown in table 1 (line of best fit is not mathematically calculated and is meant to indicate the trend).

As shown in figure 10, log odds seem to follow a straight line, this line represents the relationship between the increase in shot angle and the log-odds of the shot. The coefficient b_0 is the intercept of the line when it reaches the y-axis and the b_1 coefficient is equal to the slope of the line. The goal is to use this data in order to create a logistic regression model that will, based on the angle of the shot, forecast the likelihood of converting the shot into a goal. But first the log-odds of scoring a goal, as a function of the shot angle needs to be determined in order to establish the coefficients of the function. The logistic regression equation can be used to produce this:

$$\begin{gathered} log - odds \ = \ b_0 + b_1 \cdot x \\ where: \\ x \text{ is the shot angle.} \\ b_0 \text{ and } b_1 \text{ is the coefficients.} \end{gathered}$$

Mathematics

Angle calculation:

To calculate the angle Θ as seen in figure 11 to assign to each shot in the dataset, it can be assumed that the goalline (AB) is 7.32 metres as a constant throughout every shot that is taken in the dataset. Where D or the position that the shot occurs is established by the x,y coordinates.

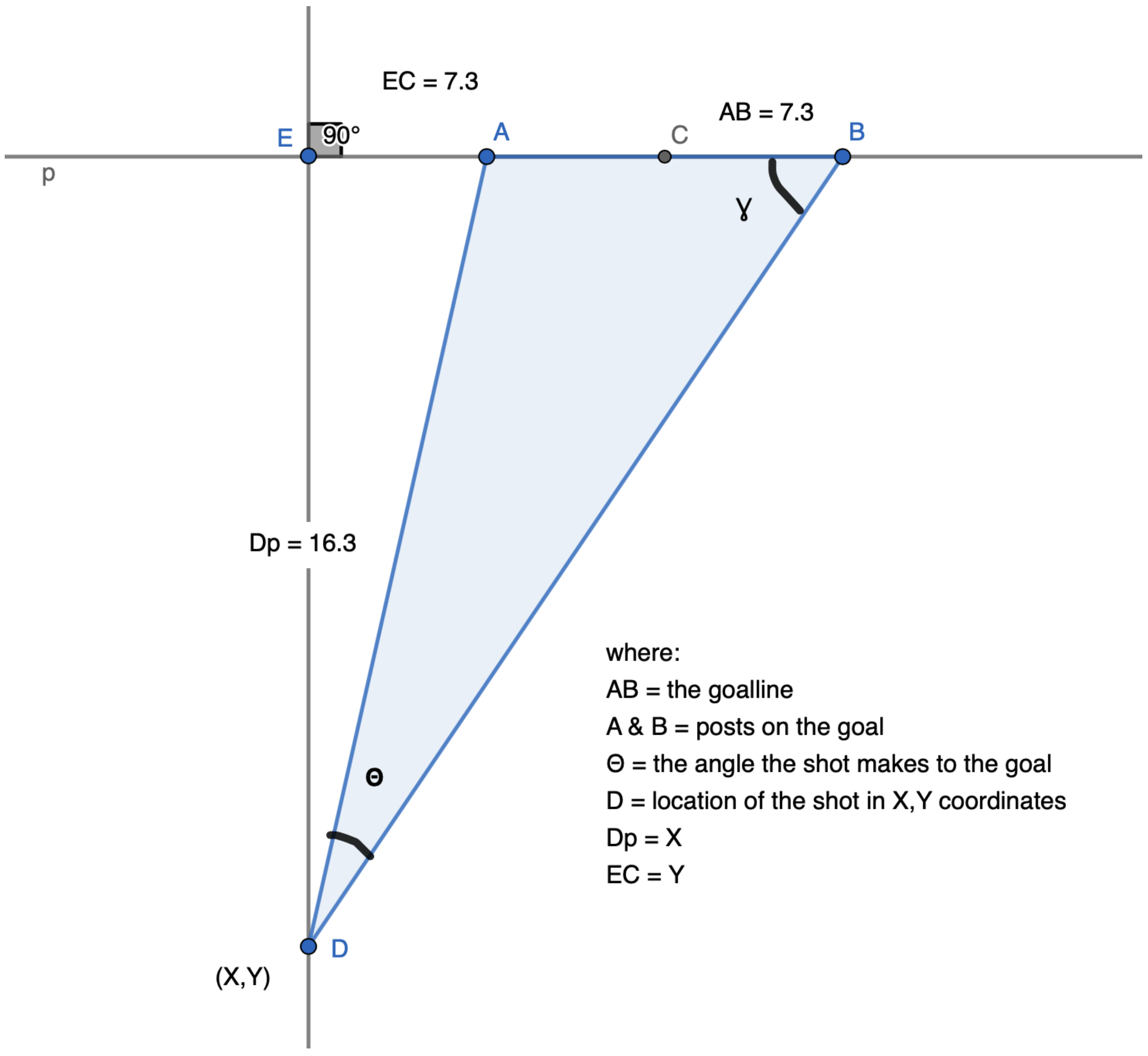
EC = 7.3
AB = 7.3
E
90°
A
C
B
p
ɣ
Dp = 16.3
where:
AB = the goalline
A & B = posts on the goal
Θ = the angle the shot makes to the goal
D = location of the shot in X,Y coordinates
Dp = X
EC = Y
Θ
D
(X,Y)

Figure 11: Diagram of a specific made up shot event

To find Θ, a series of other values must be calculated first. It can be shown that if lies within the goalposts then a different method must be used.

Using pythagoras' theorem it can be shown that:

$$AD^2 = x^2 + \left(y - \frac{AB}{2}\right)^2$$

$$BD^2 = x^2 + \left(y + \frac{AB}{2}\right)^2$$

Considering x is the length of the pitch and y is measured from the midpoint of the goalline because BD and AD lie on the goal posts, the additional distance that is measured in the data is equal to half of AB. Where:

$$sin(\gamma) = \frac{x}{BD}$$

It can then be determined using sine rule that:

$$\frac{AB}{sin(\Theta)} = \frac{AD}{sin(\gamma)} \Longrightarrow \frac{AB}{cos(\Theta)\cdot tan(\Theta)} = \frac{AD\cdot BD}{x}$$

$$\frac{AD\cdot BD}{x} = \frac{AB}{cos(\Theta)\cdot tan(\Theta)} \Longrightarrow cos(\Theta) = \frac{AB\cdot x}{AD\cdot BD\cdot tan(\Theta)}$$

$$\textit{Note that } sin(\Theta) = cos(\Theta)\cdot tan(\Theta)$$

Using this, when applying cosine rule:

$$\textit{where } \Theta \textit{ is the angle opposite from } b: \; b^2 = a^2 + c^2 - 2\cdot a\cdot c\cdot cos(\Theta)$$

$$where: AD^2 = x^2 + \left(y - \frac{AB}{2}\right)^2, BD^2 = x^2 + \left(y + \frac{AB}{2}\right)^2 \; and \; cos(\Theta) = \frac{AB\cdot x}{AD\cdot BD\cdot tan(\Theta)}$$

$$AB^2 = 2x^2 + 2y^2 + \frac{(AB^2)}{2} - 2\cdot AD\cdot BD\cdot\left(\frac{AB\cdot x}{AD\cdot BD\cdot tan(\Theta)}\right) \Longrightarrow$$

$$AB^2 = 2x^2 + 2y^2 + \frac{(AB^2)}{2} - 2\cdot\left(\frac{AB\cdot x}{tan(\Theta)}\right) \Longrightarrow$$

$$2x^2 + 2y^2 - \frac{(AB^2)}{2} - 2\cdot\left(\frac{AB\cdot x}{tan(\Theta)}\right) = 0$$

$$\Longrightarrow \frac{2\cdot AB\cdot x}{tan(\Theta)} = \left(2x^2 + 2y^2 - \frac{(AB^2)}{2}\right)$$

$$\Longrightarrow tan(\Theta) = \frac{2\cdot AB\cdot x}{2x^2 + 2y^2 - \left(\frac{AB^2}{2}\right)} \Longrightarrow tan(\Theta) = \frac{AB\cdot x}{x^2 + y^2 - \left(\frac{AB}{2}\right)^2}$$

If applied with distance of the goalline (AB):

$$tan(\Theta) = \frac{7.32\cdot x}{x^2 + y^2 - \left(\frac{7.32}{2}\right)^2} \Longrightarrow tan^{-1}\left(\frac{7.32\cdot x}{x^2 + y^2 - \left(\frac{7.32}{2}\right)^2}\right) = \Theta$$

These calculations were applied to the python code used to visualise the findings and can be found in the appendix.

<u>Parameters calculation:</u>

Derivation of the formula seen in context:

$$P(success) = \frac{e^{b_0+b_1 \cdot x}}{1 + e^{b_0+b_1 \cdot x}}$$

$$P = probability\ of\ success$$

$$\frac{e^{b_0+b_1 \cdot x}}{1 + e^{b_0+b_1 \cdot x}} = P$$

$$e^{b_0+b_1 \cdot x} = P\left(1 + e^{b_0+b_1 \cdot x}\right)$$

$$e^{b_0+b_1 \cdot x} = P + \left(P \cdot e^{b_0+b_1 \cdot x}\right)$$

$$e^{b_0+b_1 \cdot x} - \left(P \cdot e^{b_0+b_1 \cdot x}\right)$$

$$e^{b_0+b_1 \cdot x}(1 - P) = P$$

$$e^{b_0+b_1 \cdot x} = \frac{P}{1 - P}$$

Therefore as seen in the definition for log-odds:

$$ln\left(\frac{P}{1 - P}\right) = b_0 + b_1 \cdot x$$

$$ln(x) = log_e(x)$$

$$log_e\left(\frac{P}{1 - P}\right) = b_0 + b_1 \cdot x$$

$$\left(\frac{P}{1 - P}\right) = e^{b_0+b_1 \cdot x}$$

$$e^{ln\left(\frac{P}{1 - P}\right)} = \left(\frac{P}{1 - P}\right)$$

Findings:

Based on the model parameters determined by the statistical software used the model parameters for the logistic function can be estimated as $b_0 = 3.7097$ and $b_1 = -3.4553$, and therefore:

$$G(y) = \frac{1}{1 + e^{(b_0+b_1 \cdot x)}} \implies G(y) = \frac{1}{1 + e^{(3.7097-3.4553 \cdot x)}}$$

The model of this can be seen below in figure 12.

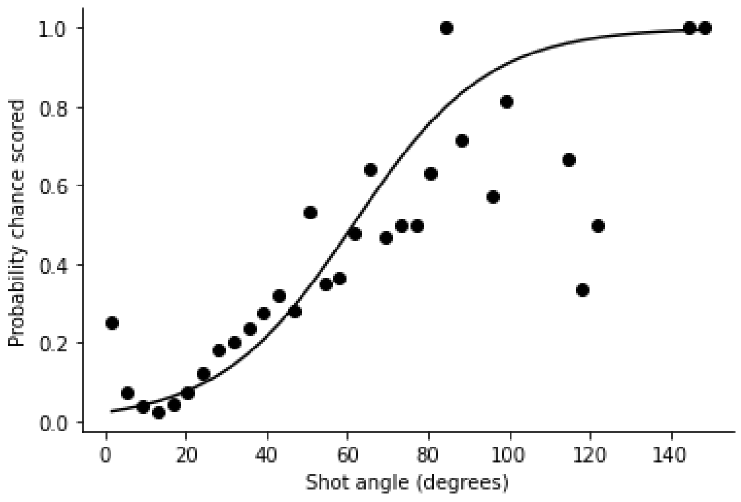

Figure 12: Model of the functions and data spread of "bins" of data points (modelled by code in appendix)

Real-world applications:

There are several real-world applications of using this model in order to predict shots, one of which include using it in order to assess whether a team had a "lucky" win (meaning there shots had a low probability of going resulting in a goal but did. This could lead to a better understanding of what team or players simply are better or worse in a match and if they are unlucky or not. In order to do this data from understat[2] will be used for a log on the shot locations from recent games played in the premier league from the 2022-2023 season. The probability of all of the shots combined should produce a value close to the final result of the game (unless one or both teams have been very fortunate or misfortune).

The first example of this can be the recent match between Manchester United FC and Leicester City FC in the 2022/2023 season.

Table 2: (calculations generated by using the python code seen in the appendix, not ordered chronologically)

Table 2:

Manchester United			Leicester City		
X position	Y position	Probability	X position	Y position	Probability

[2] "Manchester United xG stats for the 2022/2023 season." *Understat*, 19 February 2023, https://understat.com/team/Manchester_United/2022. Accessed 27 February 2023.

12	3	0.1456493688602311	7	8	0.1122474596538975
6	7	0.140805018551319	7	1	0.39520810203238194
5	2	0.5806252466621815	9	9	0.09382154848917702
10	5	0.15395831841625127	11	9	0.0898485624314304
9	7	0.12644992299450247	30	3	0.05320850731400037
8	1	0.3147422613015317	29	4	0.0542737897466166
17	17	0.049259983814358276	16	7	0.08393428666187609
17	15	0.05376611977525217	27	11	0.05175391546548881
13	17	0.048285020647841646	26	11	0.05282867021205697
16	1	0.10328963682043084	25	10	0.05520248224979184
5	7	0.13794606415605692	17	8	0.07613310451385513
3	7	0.10631657810578574	12	8	0.09581130209115185
14	5	0.10666121986472292	8	13	0.05682594769022607

12	5	0.12631143397021077	15	14	0.05729731000718066
11	0	0.1839246209492319	16	17	0.04924641930808062
9	0	0.2610442583285438	12	6	0.11576180187799574
7	0	0.4060279776161539	13	4	0.12344234263470845
32	3	0.05071767460668329	15	2	0.11084488677256062
8	10	0.08103802223544228			
5	0	0.6585813103499796			
11	6	0.1252145024180351			
13	11	0.07143841724206271			
9	1	0.25618319011178176			
6	1	0.5034730186575358			
8	4	0.2283444290352661			
15	12	0.06447105540257952			

Sum of total shot probability: 5.08452671	Sum of total shot probability: 1.673417
Actual goals scored: 3	Actual goals scored: 0

The probability of each shot was calculated by the equation:

$$G(y) = \frac{1}{1 + e^{(3.7097 - 3.4553 \cdot x)}}$$

The probability of each shot was then added up for each team respectively and denoted as the sum of shot probability which can be understood as the expected number of goals or xG.

$$Expected\ Goals\ (xG) = P(shot\#1) + P(shot\#2) + ... + P(shot\#i)$$

$$where\ i\ is\ the\ total\ number\ of\ shots\ for\ each\ team$$

Therefore:

$$\sum_{n=1}^{i} P(n) = xG$$

In the case of this football match between Manchester United and Leicester City, according to my findings it seems that Manchester United definitely should have won the game producing an xG of 5.08452671 but in fact only converting 3 goals out of the expected 5.08452671. Leicester City also did not convert their chances as expected where they had an xG of 1.673417 goals and failed to convert any of their shots. This indicates that the Manchester United forwards (attacking players) were more proficient at creating chances and played better. It is also expected that the Leicester City defence may not have had a very good game due to the amount of chances that Manchester United had. In order to try to validate this, these assumptions can be compared to the player ratings provided by various football pundits, commentators and reviewers.

According to the average score of the match ratings for players it can be seen that the Manchester United goalkeeper David De Gea was given a score of 8/10 for his performance indicating that he had a very good game and might have been responsible for stopping some of the shots from Leicester City that were “expected to score”. The Manchester United forward Marcus Rashford, responsible for 2 out of 3 of the goals scored was given a score of 8.875/10 for his performance, indicating that played a very good game, likely creating lots of good scoring opportunities. Whereas, the Leicester defence was given an average score of

4.125/10 for their performance, which indicates that they may have made it too easy for the Manchester United players to create chances.

Evaluation:

Limitations of investigation:

1. This investigation focuses on how the shot location has an effect on the probability of scoring a goal, however, in football that is played in the real world, numerous other factors. These kinds of factors have a significant impact on the probability of the shot being scored. These factors include:
    a. Whether the shot was taken by a weak foot or dominant foot.
    b. The shot type (whether it is a volley, still shot, bouncing, etc).
    c. The amount of surrounding players (players that may be blocking the "route" to the goal or even just close by which leads to a rushed shot).
    d. Weather conditions
    e. The game state (current score of the match, relates to the mentality of the player).
    f. Pattern of play (fast break, free kick, corner, open play, etc.).
2. The data used in this investigation is expressed as a percentage from the WyScout data, then transformed into the distance in metres, followed by being rounded to the nearest integer. This therefore, may skew the results of the investigations slightly as it slightly alters the resulting angle of the shot.
3. The data used in this investigation is also only based on one league, therefore, may be inaccurate in representing the relationship between the shot location and probability of scoring, as different leagues have different paces (intensity), different play styles and different quality of players. Therefore, the findings of this investigation are only applicable to the Premier League.
4. Old data: The data provided by WyScout is from the 2018-19 Premier League football season. This could lead to skewed results in the findings as the relationship between shot location and scoring probability has changed over the years since.

Assumptions made in this investigation:

1. All players have the same probability of scoring (club, position, age, "skill" and features do not affect the probability of scoring).
2. That the shot angle is the only factor that affects the probability of scoring a goal.

3. Every shot that was taken in the dataset, are independent from each other.
4. Every league "is the same" and has the same probability of scoring, thereby, all players are the same and have the same quality. Meaning that all the goalkeepers have the same skill and same likelihood of saving the shot.

Extensions to make in this investigation:

1. Consider each player: This investigation could produce a more accurate representation of the probability of scoring from a specific shot location if it considers the player themselves, either making a specific function that models the probability of scoring for that specific player, or considering the player's position on the field and modelling separate functions for the probability of a player scoring for each position.
2. If the data that is used in this investigation was larger, and considered shots from different leagues it would produce a better general function to model the probability of scoring from different shooting locations.
3. Consider the amount of players surrounding the shooting player, and or block the "route" to the goal. This could be through considering the position of each player that is inside of the angle of the shot and the distance they have to the shooter.
4. Consider the goalkeeper that the player is shooting against, this could be done by referencing each probability of scoring with the probability of saving the shot (which takes in the goalkeeper's perspective).

Conclusion:

Based on the findings of this investigation, the function that modelled the probability of scoring a goal in football based on the specific shot location was acquired. The function assesses the position that the shot was taken and calculates the shot angle, which is an overall better factor to consider in comparison to the distance from the goal because it considers how much of the goal the ball can "see" as well as accounting for the distance. The function can be used to determine how well a team plays and how many goals the team were expected to score in a match. This can be compared to the actual result of the game in order to determine how wasteful a team is and how lucky they are (how much they convert their shots). This information can also be used in order to aid and inform players on where to take shots from and can be used by opposing teams to analyse where a specific team has the highest probability of scoring from.

Bibliography:

Addagatla, Arun. “Maximum Likelihood Estimation in Logistic Regression | by Arun Addagatla | Medium.” *Arun Addagatla*, 26 April 2021, https://arunaddagatla.medium.com/maximum-likelihood-estimation-in-logistic-regression-f86ff1627b67. Accessed 6 January 2023.

AV contents team. “Logistic Regression R | Introduction to Logistic Regression.” *Analytics Vidhya*, 1 November 2015, https://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/. Accessed 21 January 2023.

Boyd-Graber, Jordan. “Logistic Regression.” *YouTube*, JordanBoydGrabe, 25 September 2020, https://www.youtube.com/watch?v=9BcxOiwE4Ds&ab_channel=JordanBoyd-Graber. Accessed 20 January 2023.

Brownlee, Jason. “A Gentle Introduction to Logistic Regression With Maximum Likelihood Estimation - MachineLearningMastery.com.” *Machine Learning Mastery*, 28 October 2019, https://machinelearningmastery.com/logistic-regression-with-maximum-likelihood-estimation/. Accessed 5 February 2023.

Danny. “Torvaney | xG.” *Match Expected Goals Simulator*, https://torvaney.github.io/projects/xG.html. Accessed 10 January 2023.

Davis, Lynne. “More mathematical notation you need for A-level.” *B28 Maths Tutor*, 12 January 2023, https://b28mathstutor.co.uk/more-mathematical-notation-for-a-level/. Accessed 9 February 2023.

Dragulet, Ian. “An Introduction of Expected Goals.” *Medium*, Towards Data Science, 1 Jan 2021, https://towardsdatascience.com/a-guide-to-expected-goals-63925ee71064.

Accessed 3 January 2023.

Kluyver, Thomas. “User Guide — pandas 1.5.3 documentation.” *Pandas*, NumFOCUS, Inc., 18 January 2023, https://pandas.pydata.org/docs/user_guide/index.html. Accessed 28 January 2023.

“Manchester United xG stats for the 2022/2023 season.” *Understat*, 19 February 2023, https://understat.com/team/Manchester_United/2022. Accessed 27 February 2023.

Marin, Mike. “5.6 Logistic Regression: Estimating Probability of Outcome Using Model Equation.” *YouTube*, marinstatlectures, 12 January 2021, https://www.youtube.com/watch?v=TY91omiZzlk. Accessed 20 January 2023.

The Matplotlib development team. “Matplotlib.” *Matplotlib — Visualization with Python*, 23 November 2022, https://matplotlib.org/. Accessed 2 February 2023.

Neal, Radford M. “1.1. Linear Models — scikit-learn 1.2.2 documentation.” *Scikit-learn*, https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression. Accessed 17 January 2023.

Pappalardo, Luca. “Soccer match event dataset.” *Figshare*, 1 August 2020, https://figshare.com/collections/Soccer_match_event_dataset/4415000. Accessed 3 April 2023.

Sumpter, David. “Football Analytics.” *David Sumpter*, https://www.david-sumpter.com/football-analytics/. Accessed 12 January 2023.

Sumpter, David. “How to Build An Expected Goals Model 1: Data and Model.” *YouTube*, Friends of tracking, 3 May 2020, https://www.youtube.com/watch?v=bpjLyFyLlXs. Accessed 10 January 2023.

Tilevik, Andreas. “Logistic regression : the basics - simply explained.” *YouTube*, TileStats, 21 February 2021, https://www.youtube.com/watch?v=yhogDBEa0uQ&ab_channel=TileStats. Accessed

20 January 2023.

Appendix:

Python code:

Python code for the calculation of the parameters and the visualisation of the curve with the input data:

```
#code comes from the friendsoftracking youtube video done by friendsoftracking the link to
#the source can be found in the references
import matplotlib.pyplot as plt
from matplotlib.patches import Arc

def createPitch(length,width, unity,linecolor): # in meters
    # Code by @JPJ_dejong

    """
    creates a plot in which the 'length' is the length of the pitch (goal to goal).
    And 'width' is the width of the pitch (sideline to sideline).
    Fill in the unity in meters or in yards.

    """
    #Set unity
    if unity == "meters":
        # Set boundaries
        if length >= 120.5 or width >= 75.5:
            return(str("Field dimensions are too big for meters as unity, didn't you mean yards as
unity?\
                        Otherwise the maximum length is 120 meters and the maximum width is 75
meters. Please try again"))
        #Run program if unity and boundaries are accepted
        else:
            #Create figure
            fig=plt.figure()
            #fig.set_size_inches(7, 5)
            ax=fig.add_subplot(1,1,1)

            #Pitch Outline & Centre Line
            plt.plot([0,0],[0,width], color=linecolor)
            plt.plot([0,length],[width,width], color=linecolor)
            plt.plot([length,length],[width,0], color=linecolor)
            plt.plot([length,0],[0,0], color=linecolor)
            plt.plot([length/2,length/2],[0,width], color=linecolor)

            #Left Penalty Area
            plt.plot([16.5 ,16.5],[(width/2 +16.5),(width/2-16.5)],color=linecolor)
            plt.plot([0,16.5],[(width/2 +16.5),(width/2 +16.5)],color=linecolor)
            plt.plot([16.5,0],[(width/2 -16.5),(width/2 -16.5)],color=linecolor)

            #Right Penalty Area
            plt.plot([(length-16.5),length],[(width/2 +16.5),(width/2 +16.5)],color=linecolor)
            plt.plot([(length-16.5), (length-16.5)],[(width/2
+16.5),(width/2-16.5)],color=linecolor)
            plt.plot([(length-16.5),length],[(width/2 -16.5),(width/2 -16.5)],color=linecolor)

            #Left 5-meters Box
            plt.plot([0,5.5],[(width/2+7.32/2+5.5),(width/2+7.32/2+5.5)],color=linecolor)
            plt.plot([5.5,5.5],[(width/2+7.32/2+5.5),(width/2-7.32/2-5.5)],color=linecolor)
            plt.plot([5.5,0.5],[(width/2-7.32/2-5.5),(width/2-7.32/2-5.5)],color=linecolor)

            #Right 5 -eters Box

plt.plot([length,length-5.5],[(width/2+7.32/2+5.5),(width/2+7.32/2+5.5)],color=linecolor)

plt.plot([length-5.5,length-5.5],[(width/2+7.32/2+5.5),width/2-7.32/2-5.5],color=linecolor)
            plt.plot([length-5.5,length],[width/2-7.32/2-5.5,width/2-7.32/2-5.5],color=linecolor)

            #Prepare Circles
```

```
            centreCircle = plt.Circle((length/2,width/2),9.15,color=linecolor,fill=False)
            centreSpot = plt.Circle((length/2,width/2),0.8,color=linecolor)
            leftPenSpot = plt.Circle((11,width/2),0.8,color=linecolor)
            rightPenSpot = plt.Circle((length-11,width/2),0.8,color=linecolor)

            #Draw Circles
            ax.add_patch(centreCircle)
            ax.add_patch(centreSpot)
            ax.add_patch(leftPenSpot)
            ax.add_patch(rightPenSpot)

            #Prepare Arcs
            leftArc =
Arc((11,width/2),height=18.3,width=18.3,angle=0,theta1=308,theta2=52,color=linecolor)
            rightArc =
Arc((length-11,width/2),height=18.3,width=18.3,angle=0,theta1=128,theta2=232,color=linec
olor)

            #Draw Arcs
            ax.add_patch(leftArc)
            ax.add_patch(rightArc)
            #Axis titles

    #check unity again
    elif unity == "yards":
        #check boundaries again
        if length <= 95:
            return(str("Didn't you mean meters as unity?"))
        elif length >= 131 or width >= 101:
            return(str("Field dimensions are too big. Maximum length is 130, maximum width is
100"))
        #Run program if unity and boundaries are accepted
        else:
```

```
#Create figure
fig=plt.figure()
#fig.set_size_inches(7, 5)
ax=fig.add_subplot(1,1,1)

#Pitch Outline & Centre Line
plt.plot([0,0],[0,width], color=linecolor)
plt.plot([0,length],[width,width], color=linecolor)
plt.plot([length,length],[width,0], color=linecolor)
plt.plot([length,0],[0,0], color=linecolor)
plt.plot([length/2,length/2],[0,width], color=linecolor)

#Left Penalty Area
plt.plot([18 ,18],[(width/2 +18),(width/2-18)],color=linecolor)
plt.plot([0,18],[(width/2 +18),(width/2 +18)],color=linecolor)
plt.plot([18,0],[(width/2 -18),(width/2 -18)],color=linecolor)

#Right Penalty Area
plt.plot([(length-18),length],[(width/2 +18),(width/2 +18)],color=linecolor)
plt.plot([(length-18), (length-18)],[(width/2 +18),(width/2-18)],color=linecolor)
plt.plot([(length-18),length],[(width/2 -18),(width/2 -18)],color=linecolor)

#Left 6-yard Box
plt.plot([0,6],[(width/2+7.32/2+6),(width/2+7.32/2+6)],color=linecolor)
plt.plot([6,6],[(width/2+7.32/2+6),(width/2-7.32/2-6)],color=linecolor)
plt.plot([6,0],[(width/2-7.32/2-6),(width/2-7.32/2-6)],color=linecolor)

#Right 6-yard Box
plt.plot([length,length-6],[(width/2+7.32/2+6),(width/2+7.32/2+6)],color=linecolor)
plt.plot([length-6,length-6],[(width/2+7.32/2+6),width/2-7.32/2-6],color=linecolor)
plt.plot([length-6,length],[(width/2-7.32/2-6),width/2-7.32/2-6],color=linecolor)

#Prepare Circles; 10 yards distance. penalty on 12 yards
```

```
        centreCircle = plt.Circle((length/2,width/2),10,color=linecolor,fill=False)
        centreSpot = plt.Circle((length/2,width/2),0.8,color=linecolor)
        leftPenSpot = plt.Circle((12,width/2),0.8,color=linecolor)
        rightPenSpot = plt.Circle((length-12,width/2),0.8,color=linecolor)

        #Draw Circles
        ax.add_patch(centreCircle)
        ax.add_patch(centreSpot)
        ax.add_patch(leftPenSpot)
        ax.add_patch(rightPenSpot)

        #Prepare Arcs
        leftArc =
Arc((11,width/2),height=20,width=20,angle=0,theta1=312,theta2=48,color=linecolor)
        rightArc =
Arc((length-11,width/2),height=20,width=20,angle=0,theta1=130,theta2=230,color=linecolor
)

        #Draw Arcs
        ax.add_patch(leftArc)
        ax.add_patch(rightArc)

    #Tidy Axes
    plt.axis('off')

    return fig,ax



def createPitchOld():
    #Taken from FC Python
    #Create figure
    fig=plt.figure()
    ax=fig.add_subplot(1,1,1)
```

```
linecolor='black'
#Pitch Outline & Centre Line
plt.plot([0,0],[0,90], color=linecolor)
plt.plot([0,130],[90,90], color=linecolor)
plt.plot([130,130],[90,0], color=linecolor)
plt.plot([130,0],[0,0], color=linecolor)
plt.plot([65,65],[0,90], color=linecolor)

#Left Penalty Area
plt.plot([16.5,16.5],[65,25],color=linecolor)
plt.plot([0,16.5],[65,65],color=linecolor)
plt.plot([16.5,0],[25,25],color=linecolor)

#Right Penalty Area
plt.plot([130,113.5],[65,65],color=linecolor)
plt.plot([113.5,113.5],[65,25],color=linecolor)
plt.plot([113.5,130],[25,25],color=linecolor)

#Left 6-yard Box
plt.plot([0,5.5],[54,54],color=linecolor)
plt.plot([5.5,5.5],[54,36],color=linecolor)
plt.plot([5.5,0.5],[36,36],color=linecolor)

#Right 6-yard Box
plt.plot([130,124.5],[54,54],color=linecolor)
plt.plot([124.5,124.5],[54,36],color=linecolor)
plt.plot([124.5,130],[36,36],color=linecolor)

#Prepare Circles
centreCircle = plt.Circle((65,45),9.15,color=linecolor,fill=False)
centreSpot = plt.Circle((65,45),0.8,color=linecolor)
leftPenSpot = plt.Circle((11,45),0.8,color=linecolor)
rightPenSpot = plt.Circle((119,45),0.8,color=linecolor)
```

```
    #Draw Circles
    ax.add_patch(centreCircle)
    ax.add_patch(centreSpot)
    ax.add_patch(leftPenSpot)
    ax.add_patch(rightPenSpot)

    #Prepare Arcs
    leftArc =
Arc((11,45),height=18.3,width=18.3,angle=0,theta1=310,theta2=50,color=linecolor)
    rightArc =
Arc((119,45),height=18.3,width=18.3,angle=0,theta1=130,theta2=230,color=linecolor)

    #Draw Arcs
    ax.add_patch(leftArc)
    ax.add_patch(rightArc)

    #Tidy Axes
    plt.axis('off')

    return fig,ax

def createGoalMouth():
    #Adopted from FC Python
    #Create figure
    fig=plt.figure()
    ax=fig.add_subplot(1,1,1)

    linecolor='black'

    #Pitch Outline & Centre Line
    plt.plot([0,65],[0,0], color=linecolor)
    plt.plot([65,65],[50,0], color=linecolor)
    plt.plot([0,0],[50,0], color=linecolor)

    #Left Penalty Area
    plt.plot([12.5,52.5],[16.5,16.5],color=linecolor)
    plt.plot([52.5,52.5],[16.5,0],color=linecolor)
    plt.plot([12.5,12.5],[0,16.5],color=linecolor)

    #Left 6-yard Box
    plt.plot([41.5,41.5],[5.5,0],color=linecolor)
    plt.plot([23.5,41.5],[5.5,5.5],color=linecolor)
    plt.plot([23.5,23.5],[0,5.5],color=linecolor)

    #Goal
    plt.plot([41.5-5.34,41.5-5.34],[-2,0],color=linecolor)
    plt.plot([23.5+5.34,41.5-5.34],[-2,-2],color=linecolor)
    plt.plot([23.5+5.34,23.5+5.34],[0,-2],color=linecolor)

    #Prepare Circles
    leftPenSpot = plt.Circle((65/2,11),0.8,color=linecolor)

    #Draw Circles
    ax.add_patch(leftPenSpot)

    #Prepare Arcs
    leftArc =
Arc((32.5,11),height=18.3,width=18.3,angle=0,theta1=38,theta2=142,color=linecolor)

    #Draw Arcs
    ax.add_patch(leftArc)

    #Tidy Axes
    plt.axis('off')
    return fig,ax
"""
Spyder Editor

This is a temporary script file.
"""


# The basics
import pandas as pd
import numpy as np
import json

# Plotting

# Statistical fitting of models
import statsmodels.api as sm
import statsmodels.formula.api as smf

# Decide which league to load
# Wyscout data from
https://figshare.com/collections/Soccer_match_event_dataset/4415000/2
with open('/Users/theo/fcprj/events/events_England.json') as f:
    data = json.load(f)

# Create a data set of shots.
train = pd.DataFrame(data)
pd.unique(train['subEventName'])
shots = train[train['subEventName'] == 'Shot']
shots_model = pd.DataFrame(columns=['Goal', 'X', 'Y'])

# Go through the dataframe and calculate X, Y co-ordinates.
# Distance from a line in the centre
# Shot angle.
# Details of tags can be found here: https://apidocs.wyscout.com/matches-wyid-events
for i, shot in shots.iterrows():

    header = 0
    for shottags in shot['tags']:
        if shottags['id'] == 403:
            header = 1
    # Only include non-headers
    if not (header):
        shots_model.at[i, 'X'] = 100 - shot['positions'][0]['x']
        shots_model.at[i, 'Y'] = shot['positions'][0]['y']
        shots_model.at[i, 'C'] = abs(shot['positions'][0]['y'] - 50)

        # Distance in metres and shot angle in radians.
        x = shots_model.at[i, 'X'] * 105 / 100
        y = shots_model.at[i, 'C'] * 65 / 100
        shots_model.at[i, 'Distance'] = np.sqrt(x ** 2 + y ** 2)
        a = np.arctan(7.32 * x / (x ** 2 + y ** 2 - (7.32 / 2) ** 2))
        if a < 0:
            a = np.pi + a
        shots_model.at[i, 'Angle'] = a

        # Was it a goal
        shots_model.at[i, 'Goal'] = 0
        for shottags in shot['tags']:
            # Tags contain that its a goal
            if shottags['id'] == 101:
                shots_model.at[i, 'Goal'] = 1

# Two dimensional histogram
H_Shot = np.histogram2d(shots_model['X'], shots_model['Y'], bins=50, range=[[0, 100], [0,
100]])
goals_only = shots_model[shots_model['Goal'] == 1]
```

```
H_Goal = np.histogram2d(goals_only['X'], goals_only['Y'], bins=50, range=[[0, 100], [0,
100]])

# Plot the number of shots from different points
(fig, ax) = createGoalMouth()
pos = ax.imshow(H_Shot[0], extent=[-1, 66, 104, -1], aspect='auto', cmap=plt.cm.Reds)
fig.colorbar(pos, ax=ax)
ax.set_title('Number of shots')
plt.xlim((-1, 66))
plt.ylim((-3, 35))
plt.tight_layout()
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
fig.savefig('NumberOfShots.png', dpi=None, bbox_inches="tight")

# Plot the number of GOALS from different points
(fig, ax) = createGoalMouth()
pos = ax.imshow(H_Goal[0], extent=[-1, 66, 104, -1], aspect='auto', cmap=plt.cm.Reds)
fig.colorbar(pos, ax=ax)
ax.set_title('Number of goals')
plt.xlim((-1, 66))
plt.ylim((-3, 35))
plt.tight_layout()
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
fig.savefig('NumberOfGoals.png', dpi=None, bbox_inches="tight")

# Plot the probability of scoring from different points
(fig, ax) = createGoalMouth()
pos = ax.imshow(H_Goal[0] / H_Shot[0], extent=[-1, 66, 104, -1], aspect='auto',
cmap=plt.cm.Reds, vmin=0, vmax=0.5)
fig.colorbar(pos, ax=ax)
ax.set_title('Proportion of shots resulting in a goal')
```

```
plt.xlim((-1, 66))
plt.ylim((-3, 35))
plt.tight_layout()
plt.gca().set_aspect('equal', adjustable='box')
plt.show()
fig.savefig('ProbabilityOfScoring.pdf', dpi=None, bbox_inches="tight")

#Plot a logistic curve
b=[3, -3]
x=np.arange(5,step=0.1)
y=1/(1+np.exp(-b[0]-b[1]*x))
fig,ax=plt.subplots(num=1)
plt.ylim((-0.05,1.05))
plt.xlim((0,5))
ax.set_ylabel('y')
ax.set_xlabel("x")
ax.plot(x, y, linestyle='solid', color='black')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.show()



#Get first 200 shots
shots_200=shots_model.iloc[:200]

#Plot first 200 shots goal angle
fig,ax=plt.subplots(num=1)
ax.plot(shots_200['Angle']*180/np.pi, shots_200['Goal'], linestyle='none', marker= '.',
markersize= 10.0, color='black')
ax.set_ylabel('Goal scored')
ax.set_xlabel("Shot angle (degrees)")
plt.ylim((-0.05,1.05))
ax.set_yticks([0,1])
ax.set_yticklabels(['No','Yes'])
plt.show()

#Show empirically how goal angle predicts probability of scoring
shotcount_dist=np.histogram(shots_model['Angle']*180/np.pi,bins=40,range=[0, 150])
goalcount_dist=np.histogram(goals_only['Angle']*180/np.pi,bins=40,range=[0, 150])
prob_goal=np.divide(goalcount_dist[0],shotcount_dist[0])
angle=shotcount_dist[1]
midangle= (angle[:-1] + angle[1:])/2
fig,ax=plt.subplots(num=2)
ax.plot(midangle, prob_goal, linestyle='none', marker= '.', markersize= 12, color='black')
ax.set_ylabel('Probability chance scored')
ax.set_xlabel("Shot angle (degrees)")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

b=[3, -3]
x=np.arange(150,step=0.1)
y=1/(1+np.exp(b[0]+b[1]*x*np.pi/180))
ax.plot(x, y, linestyle='solid', color='black')
plt.show()

xG=1/(1+np.exp(b[0]+b[1]*shots_model['Angle']))
shots_model = shots_model.assign(xG=xG)
shots_200=shots_model.iloc[:200]
fig,ax=plt.subplots(num=1)
ax.plot(shots_200['Angle']*180/np.pi, shots_200['Goal'], linestyle='none', marker= '.',
markersize= 12, color='black')
ax.plot(x, y, linestyle='solid', color='black')
ax.plot(x, 1-y, linestyle='solid', color='black')
loglikelihood=0
for item,shot in shots_200.iterrows():
    ang=shot['Angle']*180/np.pi
    if shot['Goal']==1:
        loglikelihood=loglikelihood+np.log(shot['xG'])
        ax.plot([ang,ang],[shot['Goal'],shot['xG']], color='red')
    else:
        loglikelihood=loglikelihood+np.log(1 - shot['xG'])
        ax.plot([ang,ang],[shot['Goal'],1-shot['xG']], color='blue')

ax.set_ylabel('Goal scored')
ax.set_xlabel("Shot angle (degrees)")
plt.ylim((-0.05,1.05))
plt.xlim((0,80))
#plt.text(45,0.2,'Log-likelihood:')
#plt.text(45,0.1,str(loglikelihood))
ax.set_yticks([0,1])
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
fig.savefig('LikelihoodExample.pdf', dpi=None, bbox_inches="tight")
plt.show()

#Make single variable model of angle
#Using logistic regression we find the optimal values of b
#This process minimizes the loglikelihood
test_model = smf.glm(formula="Goal ~ Angle" , data=shots_model,
                     family=sm.families.Binomial()).fit()
print(test_model.summary())
b=test_model.params
xGprob=1/(1+np.exp(b[0]+b[1]*midangle*np.pi/180))
fig,ax=plt.subplots(num=1)
ax.plot(midangle, prob_goal, linestyle='none', marker= '.', markersize= 12, color='black')
ax.plot(midangle, xGprob, linestyle='solid', color='black')
ax.set_ylabel('Probability chance scored')
ax.set_xlabel("Shot angle (degrees)")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.show()
fig.savefig('ProbabilityOfScoringAngleFit.pdf', dpi=None, bbox_inches="tight")


#Show empirically how distance from goal predicts probability of scoring
shotcount_dist=np.histogram(shots_model['Distance'],bins=40,range=[0, 70])
goalcount_dist=np.histogram(goals_only['Distance'],bins=40,range=[0, 70])
prob_goal=np.divide(goalcount_dist[0],shotcount_dist[0])
distance=shotcount_dist[1]
middistance= (distance[:-1] + distance[1:])/2
fig,ax=plt.subplots(num=1)
ax.plot(middistance, prob_goal, linestyle='none', marker= '.', color='black')
ax.set_ylabel('Probability chance scored')
ax.set_xlabel("Distance from goal (metres)")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
#Make single variable model of distance
test_model = smf.glm(formula="Goal ~ Distance" , data=shots_model,
                        family=sm.families.Binomial()).fit()
print(test_model.summary())
b=test_model.params
xGprob=1/(1+np.exp(b[0]+b[1]*middistance))
ax.plot(middistance, xGprob, linestyle='solid', color='black')
plt.show()
fig.savefig('Output/ProbabilityOfScoringDistance.pdf', dpi=None, bbox_inches="tight")
#Adding distance squared
squaredD = shots_model['Distance']**2
shots_model = shots_model.assign(D2=squaredD)
test_model = smf.glm(formula="Goal ~ Distance + D2" , data=shots_model,
                        family=sm.families.Binomial()).fit()
print(test_model.summary())
b=test_model.params
```

```
xGprob=1/(1+np.exp(b[0]+b[1]*middistance+b[2]*pow(middistance,2)))
fig,ax=plt.subplots(num=1)
ax.plot(middistance, prob_goal, linestyle='none', marker= '.', color='black')
ax.set_ylabel('Probability chance scored')
ax.set_xlabel("Distance from goal (metres)")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.plot(middistance, xGprob, linestyle='solid', color='black')
plt.show()
fig.savefig('/ProbabilityOfScoringDistanceSquared.pdf', dpi=None, bbox_inches="tight")
#Adding even more variables to the model.
squaredX = shots_model['X']**2
shots_model = shots_model.assign(X2=squaredX)
squaredC = shots_model['C']**2
shots_model = shots_model.assign(C2=squaredC)
AX = shots_model['Angle']*shots_model['X']
shots_model = shots_model.assign(AX=AX)
```