Predicting a Goal Before the Shot is Taken

Introduction

If you watch football, you'll likely be familiar with the term expected goals (xG). Most people understand that the higher the xG, the more goal scoring chances a team is creating. However, many people, even pundits and commentators, don't seem to understand what the metric is measuring.

The expected goal is a measure of the probability (between 1 and 0) that a single shot results in a goal. These probabilities are accumulated at the end of the 90 minutes to give a team's overall xG. This probability is based on several factors such as:

- Shot distance
- Shot angle
- Body part used for shot
- Type of assist
- Defensive pressure
- Goalkeeper position

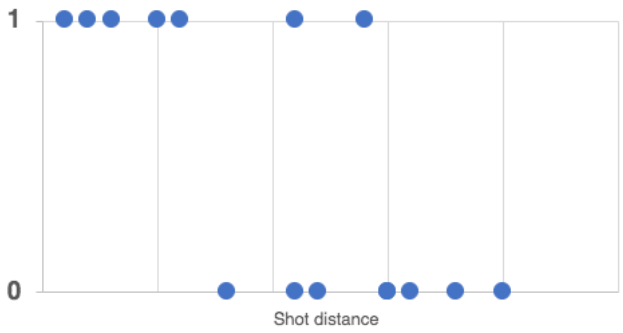
Companies such as Opta use data from over 1 million historical shots in their xG model to calculate a probability which is as accurate as possible.

In this essay I aim to
a) Understand the maths these companies use to calculate probabilities based on this data.

b) Make an xG model myself using these methods and the data I have collected.

Logistic Regression

The xG models take from a large set of independent variables, which are also known as predictors, (distance, body part, etc) to predict the dependent variable (an outcome of goal or no goal). It should be obvious from this that some form of regression will be needed, yet it should also be clear that regular linear models are not going to work.



Take, for example, this set of shots with shot distance on the x-axis and outcome on the y-axis (1 for goal, 0 for no goal). Using a linear regression will produce a regression line that goes beyond the range of between 1 and 0 as seen on the graph below.

This poses an issue since we are measuring probability therefore values outside the range $0 \leq p \leq 1$ are meaningless.
This is where the logistic regression comes in handy. Logistical regression is a statistical method used to predict the probability of a binary outcome based on predictors.

This is perfect for an xG model since we are trying to predict the probability of either scoring a goal or not. Here's how it's done.

Firstly, the probability of a goal on the y-axis is put through the logit function. The logit function is as follows:

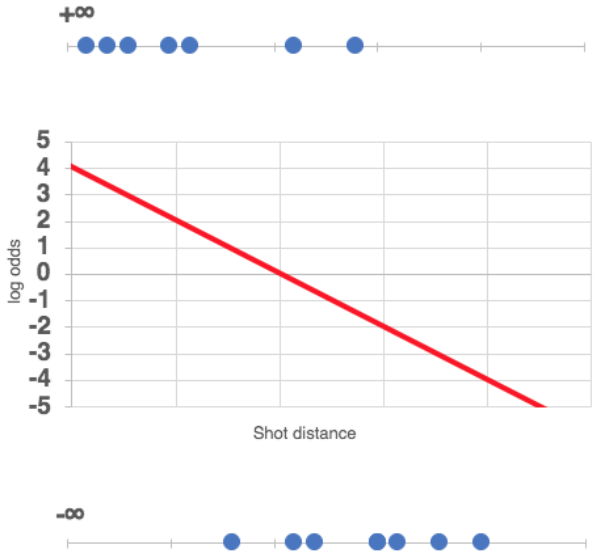
$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right), \textit{where } p \textit{ is the probability of success}$$

When inserting a probability into the logit function, we attain the "log(odds)" which are useful for calculating this regression.

If we take the limits for our range (0 and 1) and put them into the logit function, you get the following:

$$\textit{when } p = 0, \ \ln\left(\frac{p}{1-p}\right) = \ln(0)$$
$$\lim_{n \to 0}(\ln(n)) = -\infty$$
$$\textit{when } p = 1, \ \ln\left(\frac{p}{1-p}\right) = \ln\left(\frac{1}{0}\right)$$
$$= \ln(1) - \ln(0)$$
$$= -\ln(0)$$
$$\lim_{n \to 0}(-\ln(n)) = \infty$$

From this, we can see that limited range for p on the original scale extends to a range between positive and negative infinity in the log(odds) scale. This means we can plot a straight line with an infinite range on the log(odds) graph which exists only between 0 and 1 on the original graph as seen here. You can then get an equation for the log(odds) in this form: $\log(odds) = \beta_0 + X_1\beta_1$, *where* β_0 *and* β_1 *are coefficients and* X_1 *is a predictor.*



Linear regression usually uses the least squares technique where an algorithm makes an initial guess for the regression line and calculates the sum of the squared distance in the vertical axis (residuals) between each data point and the line. It then uses this to improve upon its initial regression line. This process is repeated until the algorithm finds a highly accurate estimate for the regression line.

This technique is clearly not going to work when finding the regression line for the log(odds) as the outcomes lie on either positive or negative infinity and therefore will have an infinitely large residual for any line. So, the algorithm instead uses maximum likelihood which works as follows:

1) An initial guess for the log(odds) equation is taken and transformed into probability using the inverse of the logit function.

$$\log(odds) = \ln\left(\frac{p}{1-p}\right)$$

$$e^{\log(odds)} = \frac{p}{1-p}$$

$$p = e^{\log(odds)} - pe^{\log(odds)}$$

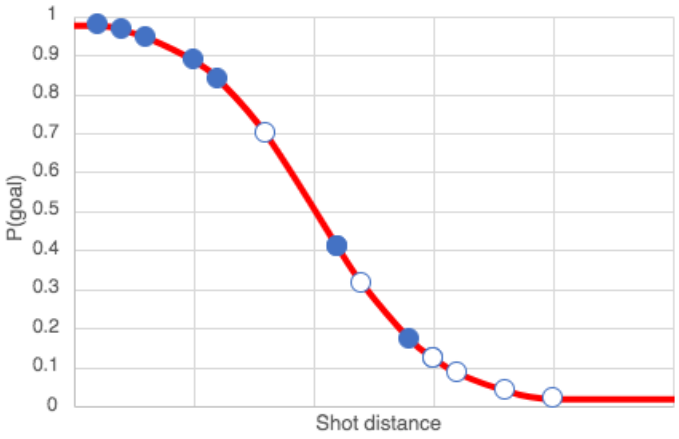
$$p + pe^{\log(odds)} = e^{\log(odds)}$$

$$p = \frac{e^{\log(odds)}}{1+e^{\log(odds)}}$$

$$p = \frac{1}{1+e^{-\log(odds)}}$$

This results in an S-shaped curve tending towards asymptotes at $y=0$ and $y=1$.

2) The data points are projected onto the curve as in this graph (blue representing a goal and white no goal).



3) Using this curve, the likelihood of our observed outcome for each shot is taken. For a goal (a success) this is just $P(goal)$. For a no goal (a failure) this is $P(no\ goal)$ or $1 - P(goal)$. The likelihoods for each individual shot are multiplied together to get the product of the likelihoods.
4) The algorithm improves upon its log(odds) equation and the process is repeated. The best fitting equation is the one with the with the greatest product of likelihoods or the maximum likelihood.

Though difficult to represent graphically, algorithms can calculate a regression equation for probabilities based on multiple predictors in the form $\log(odds) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots$
The predictors can be discrete variables as well as continuous variables. This is done by choosing one of the values to be a dummy variable and the other values to be represented by separate variables which can be "turned on or off" by setting their value to be 1 or 0 respectively.

Making my own model

Data collection

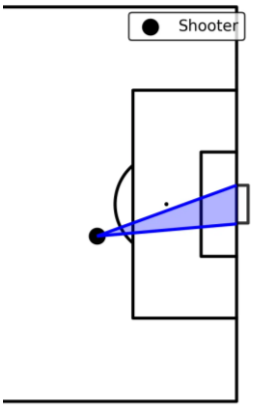
Before beginning this project, I thought that collecting data for this project would be easy since football stats and data are not difficult to come across. I was very much incorrect. After searching through the internet, I found close to no data freely available

about the conditions of each shot in a game. I found that companies such as Opta choose to keep this information proprietary so that they can sell it to data analysts. This meant I needed to be creative to get my data.
I noticed that many football websites such as Fotmob include a shot map for each game where every shot is marked by a point on a pitch and, by clicking on each shot, some information about the shot's conditions is given. So, I decided to use these maps to manually measure the shot distance and angle of each individual shot as well as using some of the given information to create my dataset. There were likely far better ways to get the data, but none came to mind. This obviously came with a handful of limitations.

- The data provided by Fotmob does not give any information about defensive pressure or goalkeeper position which are highly important for predicting goals.
- The data takes a very long time to collect meaning I only managed to get the data for a single randomly chosen Premier League match week.

With that said, I still managed to collect the following data for nearly 200 shots:

- Shot distance—the distance in metres from the ball to the middle of the goal.
- Shot angle—the angle in degrees at the point the shot is taken between the two lines going from this point to each of the goal posts as in the diagram.



- Body part of shot—I categorised this into either dominant foot, weak foot or header. A few shots in the data were described by Fotmob as "other body part" but I decided to call this a header since there were so few and would provide no useful data.

Finally, I did not use penalties in my dataset since the conditions are always the same. This means that the xG for a penalty can equal the league's penalty conversion rate (0.76) for the Premier League.

Creating the model

Before running the regression, I assigned these names to my variables:

$$X_1 = Shot\ distance$$
$$X_2 = Shot\ angle$$
$$X_3 = Weak\ foot\ (1\ or\ 0)$$
$$X_4 = Header\ (1\ or\ 0)$$

(I chose dominant foot to be my dummy variable for the body part).

$\beta_1, \beta_2, \beta_3,$ and β_4 are their corresponding coefficients for the log(odds) and β_0 is the log(odds) intercept.

By using the statsmodel library on a Python code, I performed a logistic regression on this data and got the following coefficients for the log(odds):

$$\beta_0 = -1.5617$$
$$\beta_1 = -0.0542$$
$$\beta_2 = 0.0224$$
$$\beta_3 = -0.5811$$
$$\beta_4 = -0.7912$$

These coefficients, at a first glance, seem to make sense. Coefficients for shot distance, weak foot and header are all negative as increasing those will decrease probability or scoring while shot angle has a positive coefficient since a wider angle makes it easier to score. Although given the size of my dataset, I was prepared for this model to be a flop. So, I tested my model using real examples from the Premier League.

This strong footed shot from Beto taken from 14.5m from goal with an angle of 24.3° has an xG of 0.14 according to my model and 0.14 according to Fotmob (a surprising level of accuracy).

Elliot Anderson's strong footed shot from 26.2m and an angle of 15.9° has an xG of 0.07 on my model and 0.06 on Fotmob's. I'm again pleased by the accuracy.

Here, Alexis Mac Allister's header from 10.4m at 37.4° shows an xG of 0.11 on my model but only 0.04 on Fotmob's.

These inaccuracies seem to worsen as the distance of headers, weak footed shots and tight angle shots increases. This may be because these types of shots are affected by distance differently. To put this into context, scoring a headed goal from 1m is almost as easy as scoring a strong footed goal from 1m; however, while scoring a strong footed goal from 20m is difficult, scoring a header from this distance is near impossible. This logic also applies to weak footed and tight angled shots.

In an attempt to fix this, I included Interaction variables. These variables which are composed of existing variables. For example, I could add a fifth variable which is the product of distance and angle. This means that a fifth coefficient will also be added which leads to tight angle shots being affected by distance differently to wider angled shots. I can also do the same with weak foot and header variables. I assigned the variables these names:

$X_5 = \textit{Shot distance} \cdot \textit{Shot angle}$
$X_6 = \textit{Shot distance} \cdot \textit{Weak foot}$
$X_7 = \textit{Shot distance} \cdot \textit{Header}$

I also decided to centre my continuous variables. Centring a variable is when you subtract the mean from each value.

$$x_{centered} = x - \overline{x}$$

This means that the mean of the centred variables is now 0 which has several benefits in logistic regression.

- Makes coefficients easier to interpret—$\beta_0$before represented the log(odds) when all the variables were 0 which was not at all meaningful for this data (it's

impossible for a shot to be taken from 0m with an angle of 0°). Now that the variables are centred, β_0 represents the log(odds) of a dominant footed shot from the average distance and average angle since 0 represents the means.

- Reduces collinearity of interaction variables—It is likely that the interaction variable will be strongly corelated with the variables it's composed of. This will cause problems as it increases the standard error of the coefficients making them less stable. This can be reduced by centring the variables.

With the new variables, I got these coefficients:

$$\beta_0 = -1.6067$$
$$\beta_1 = -0.0389$$
$$\beta_2 = 0.0316$$
$$\beta_3 = -0.6474$$
$$\beta_4 = -5.0971$$
$$\beta_5 = 0.0015$$
$$\beta_6 = -0.0236$$
$$\beta_7 = -0.5128$$

Initially I was sceptical of these coefficients, especially the size of coefficient for headers (β_4) but I decided to test the strength of this new model.

Mac Allister's header now has an xG of only 0.03 which makes much more sense and is closer to the Fotmob's value.

This tight angle shot from Bernardo Silva has an xG of 0.09 on my model which still seems to be high considering the angle as well the fact Fotmob gave it an xG of 0.03. This shows that my model still struggles with shots from tight angles despite my changes.

Finally, this open goal tap-in from Igor Thiago has an xG of only 0.38 even though it's an almost certain goal. This is likely down to the fact that my model uses no data about defensive pressure or goalkeeper position, which I unfortunately cannot fix.

Despite my model's flaws, I'm pleased to have created a model with a somewhat useful output, especially considering my limited data access. If I were to further improve on this model I may choose to investigate whether the log(odds) has a linear relationship with distance or if using the log of the distance, for example, would lead to a better fit.

References:

- For information on logistic regression https://www.youtube.com/watch?v=vN5cNN2-HWE
- For shot data https://www.fotmob.com/en-GB/leagues/47/fixtures/premier-league?group=by-round&round=22
- For logistic regression python code https://www.youtube.com/watch?v=pSSS64g-lP0
- For information on centring variables https://www.youtube.com/watch?v=h2nkqtLjjZY
- Dataset for shots here