Title: Bayesian Network Approach for Predictive Modeling of Football Team Performance

Description: This research uses Bayesian Networks to build a sophisticated and dynamic model for predicting football team performance. The model includes elements of uncertainty, decision-making, and conditional logic to provide more accurate and nuanced forecasts using a combination of historical and recent player statistics, including Expected Goals (xG) and Expected Assists (xA).

Also, the model makes use of sigmoid functions and conditional adjustments to account for differences in xG and xA, ensuring a balanced representation of the team's goal-scoring likelihood. The project concludes with the production of a probability distribution over potential goals scored, giving significant insights for sports analysts, managers, and fans.

How to run the project.

1. Open the football.neta file on the Netica software.
2. For the ten players, select YES or NO for any 4 or 5 players using their decision node in light blue. (The network will function as usual even if we select all players.)
3. Try to balance the xG and xA of the team so there's optimal performance.
4. Run the network
5. See the xG_total, the xG_adjusted values and check the probability distribution for each goal in the final nodes.
6. You can modify xG and xA stats at the top nodes by manually entering a numeric value. Keep it in range of [0 – 2].
7. Rerun the network with changed values, or different team.

Network Overview

In this intricate Bayesian Network model, we have incorporated data from ten different players, each of whom contributes to the predictive power of the model through four key statistics: Expected Goals (xG) from the season and xG from recent matches, and Expected Assists (xA) from the season and recent matches. All these four metrics are stored in a continuous nature node for a player. The stats are entered manually from data sites. Example nodes: [KDB_xG_season]

Understanding xG and xA:

Expected Goals (xG): This metric quantifies the probability of a shot resulting in a goal based on various factors such as distance from the goal, the angle of the shot, the type of opportunity, and more. It gives an indication of the player's goal-scoring ability and efficiency.

Expected Assists (xA): Similarly, xA measures the likelihood that a given pass will become an assist, considering several aspects like the type of pass, the location, and the receiving player's position relative to the goal. It signifies a player's playmaking ability.

Both xG and xA are pivotal in assessing a player's performance, revealing whether they are overperforming (exceeding expected metrics) or underperforming (falling short of expectations), thereby influencing team strategy and player evaluation. Theoretically, xG and xA can range from 0 to infinity based on the number of chances created. However, competitive football matches often accumulate xG of around 1 to 4, but it can vary from match to match.

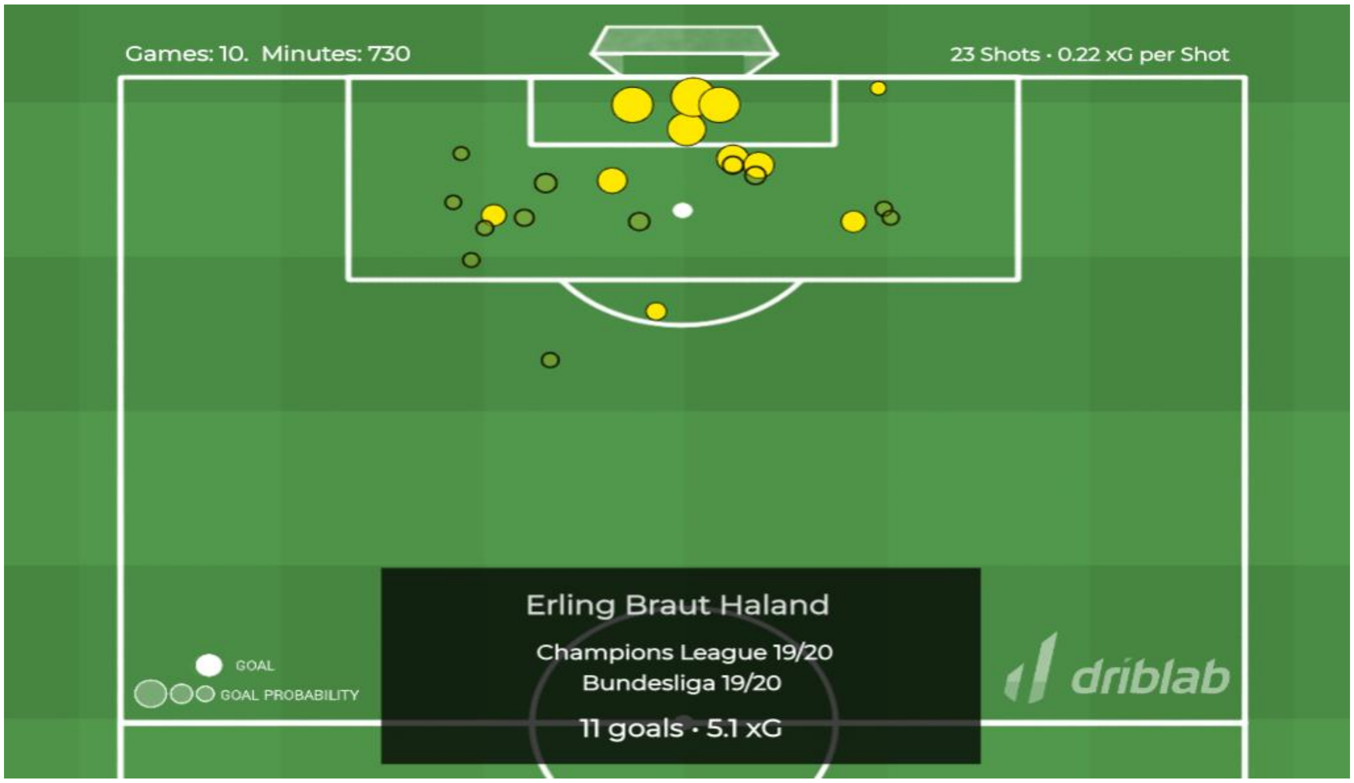

Image: Example xG evaluation of Erling Haaland from 10 matches.

The yellow dots represent the goal, and green dots represent the shots which didn't convert to goal. Both add to the xG. We can see that for an xG accumulated as 5.1, Erling Haaland scored 11 scored in those 10 matches. This means he overperformed as an average player was expected to score 5.1 goals from those same situations.

Custom Weighted Average & Player Selection:

The model leverages a custom-weighted average technique to amalgamate the recent form and overall seasonal stats of each player, formulating a more accurate representation of their expected contribution in a specific match. Nodes [Weighted_xG] and [Weighted_xA] are calculated in such a way. The equation for these nodes is:

Weighted_xA3 (Messi_xG_season, Messi_xA_recent) =

(w1 * Messi_xG_season) + (w2 * Messi_xA_recent)

Here, w1 and w2 are the two weights, with w1 + w2 being 1. For my network, w2 > w1 as I wanted to put more emphasis on the recent form of the player.

Subsequently, a subset of four or five players should be chosen through individual decision nodes, each representing a player's inclusion in the team. Example node: [Messi_selected]

Summation of xG and xA:

Once the players are selected, the model calculates the cumulative xG and xA expected from them. This accumulated data is the expected xG and xA for the team formed. Nodes: [xA_total] & [xG_total].

xG_total (xG, xG1, xG4, xG2, xG5, xG3, xG6, xG7, xG8, xG9) =

xG + xG1 + xG2 + xG3 + xG4 + xG5 + xG6 + xG7 + xG8 + xG9

Similarly, xA_total is calculated. However, we tidy up our expectations with the next step.

Conditional Adjustment & Disparity Impact:

Before transitioning to the goal probability distribution, the model applies conditional logic to adjust the xG based on its disparity with xA. This is essential because a substantial disparity between xG and xA can be detrimental for a football team. For instance, if a team's xG significantly outweighs xA, it might indicate a reliance on individual brilliance over teamwork, resulting in fewer chances created and potentially fewer goals. Conversely, a higher xA compared to xG might suggest missed opportunities and a lack of finishing ability. Nodes: [xG_adjusted]

xG_adjusted(xG_total, xA_total) =

xG_total + ((xA_total > xG_total)? 0.2: 0.8) * (xA_total - xG_total)

Here, we use a ternary operator for simple conditional logic. If (xA_total > xG_total) is true, expression 1 is taken, that is, 0.2 in this case. Else we take 0.8. What we do here is, we add a disparity factor to xG_total, which is (0.2 or 0.8) * disparity. Thus

xG_adjusted(xG_total, xA_total) = xG_total + disparity factor * disparity

Case 1: xA_total > xG_total

xG_adjusted = xG_total + 0.2 * disparity **[xG gets an added value as disparity is positive]**

Case 2: xA_total < xG_total

xG_adjusted = xG_total + 0.8 *disparity **[xG gets a reduced value as disparity is negative]**

Final Distribution & Use of Sigmoid Function:

Upon adjusting the xG in accordance with its difference with xA, the model proceeds to calculate the final probability distribution for the number of goals the team is expected to score. To achieve this, the model employs the sigmoid function, a well-known mathematical function that outputs a value between 0 and 1, making it particularly suitable for representing probabilities.

Why Sigmoid?

The sigmoid function is especially apt for this task due to its S-shaped curve, which means that as the input value (adjusted xG) increases, the output (probability of scoring goals) initially increases slowly, becomes steeper in the middle, and then slows down again as it reaches the upper limit. This characteristic ensures that even if the input varies significantly, the output remains within the 0 to 1 range, thus representing a valid probability.

Gradual Decrease in Probability:

In our model, the sigmoid function is adeptly used to depict the monotonic relationship between the expected number of goals and their respective probabilities. As the number of goals increases, the input to the sigmoid function is adjusted, resulting in a gradual decrease in the probability of achieving a higher goal count. This aligns with the real-world understanding that scoring a larger number of goals in a match is progressively less likely. The application of the sigmoid function, therefore, facilitates a realistic and nuanced representation of goal-scoring probabilities based on the team's cumulative expected goals and assists.

The sigmoid function is defined as:

$$f(x) = \frac{1}{1+e^{-x}}$$

My equation for the node of goals:

Goal_1 (xG_adjusted) =

exp(xG_adjusted – goal_factor) / (1 + exp(xG_adjusted - goal_factor))

I’ve added a goal_factor which is a numerical value ranging around the goal amount. For the sigmoid function, I wanted to take x as the difference of goal number and xG, that is, for an xG of say 3.5,

X1 for Goal 1 = 3.5 -1 = 2

X2 for Goal 2 = 3.5 -2 = 1.5

.. and similarly for next goals. This follows the logic that if xG is 3.5, probability of the team scoring one goal at least will be higher than scoring 2 goals, thus, x1 > x2. However, to accurately reflect the effect of xG in football matches, I've slightly modified goal_factor for each goal so that the probability distribution is closer to ground reality of football.

The final distribution gives us probability for each goal based on the adjusted xG from the selected players.