Project 5: LLMs as Bayesian network – Comparative analysis

This project involves using language models (LLMs) as Bayesian networks to perform a comparative analysis of football team performance predictions. We aim to illustrate the potential of LLMs in simulating complex Bayesian models through a structured task list involving the calculation of team performance metrics after multiple steps of logical calculations.

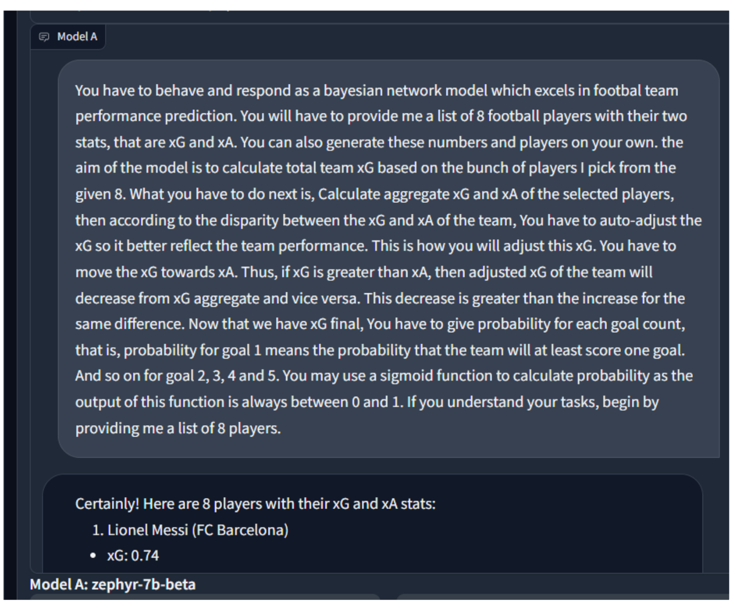
The conversation excerpts with the LLM are presented non-sequentially for illustrative purposes, and as such, there may be inconsistencies in the data values between snippets.

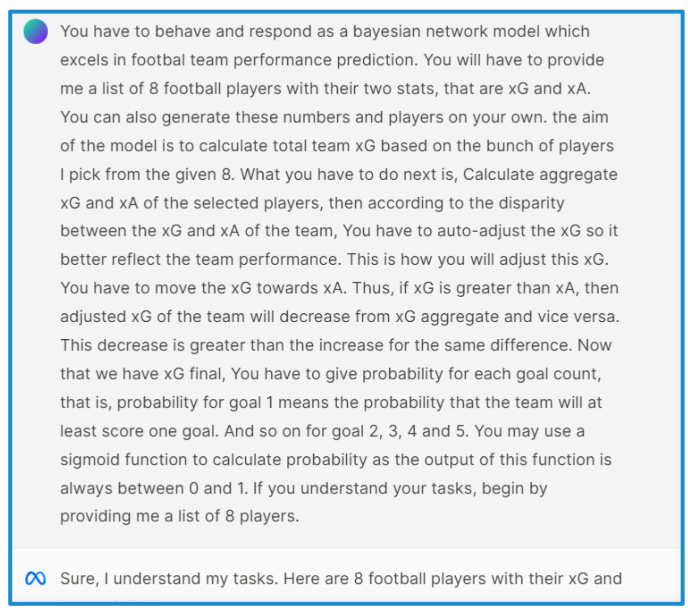
Case 1: Comparision between LLMs 'zephyr-7b' vs 'Meta lambda-70b'

The essence of our comparative study lies in assessing the capabilities of two distinct language models, Zephyr-7b and Meta Lambda-70b, in simulating a Bayesian network for the prediction of football team performance. Both models were provided with identical prompts that required them to emulate the reasoning process of a Bayesian network, particularly in calculating and adjusting team performance metrics based on expected goals (xG) and expected assists (xA).

Access and Implementation: For this experiment, Zephyr-7b was utilized through a specific interface available at chat.lmsys.org, while Meta Lambda-70b was engaged via an alternative platform at sdk.vercel.ai/s/RCEgAdv. These platforms enabled free access to the respective LLMs, subject to a predefined token limit, thus providing a controlled environment for the performance of this study.

Prompt 1:





Zephyr-7b Meta Lambda-70b

Prompt Description: The initial prompt sets the stage for the LLMs to function as a Bayesian network model, a form of artificial intelligence that applies Bayes' theorem for probabilistic inference in complex systems. The task involves two statistical metrics fundamental to football analytics: expected goals (xG) and expected assists (xA). These metrics quantify the likelihood of a player scoring a goal or contributing to a goal's scoring, respectively, in given situations.

The prompt meticulously outlines the steps necessary to simulate a Bayesian network model's behavior. The LLMs are first instructed to compile a list of eight football players along with their respective xG and xA statistics. Subsequently, they are to calculate the aggregate xG and xA for a hypothetical team composed of these players. This aggregate serves as the basis for adjusting the team's xG to better reflect its actual performance, accounting for the disparity between the team's xG and xA.

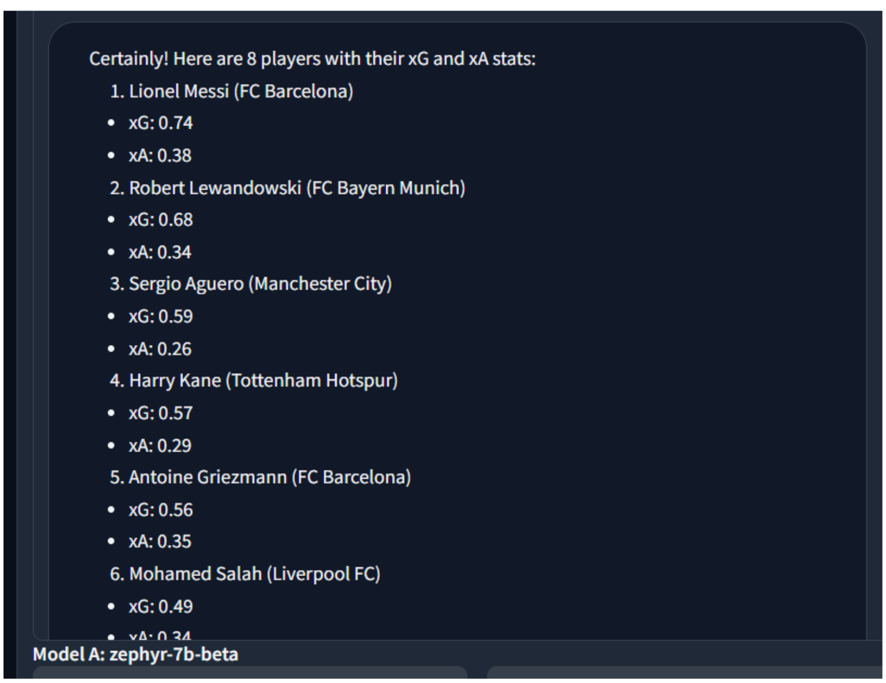
The detailed instructions in the prompt direct the LLMs to adjust the xG value conditionally. If the aggregate xG exceeds the aggregate xA, the xG is to be reduced, and vice versa, with the magnitude of reduction being more significant than that of any increase. This step is critical as it embodies the Bayesian principle of updating beliefs (or predictive measures) in light of new evidence (the disparity between xG and xA).

Following the adjustment, the LLMs are tasked with providing probabilities for various goal counts, translating the final xG value into a probabilistic forecast of the team scoring 0, 1, 2, 3, 4, or 5 goals in a match. The use of a sigmoid function is recommended to ensure that the output probabilities range between 0 and 1, aligning with the principles of Bayesian probability.

The prompt serves as a comprehensive guide, detailing each step of the process to ensure the LLMs' outputs can be interpreted within the framework of a Bayesian network model.

Task Interpretation and Initial Response: The initial response from the LLMs revealed that both LLMs were adept at understanding their tasks, demonstrating the sophistication of their design and their potential utility in statistical modeling. The preliminary step for each LLM was to request the user to select a batch of players, forming a hypothetical football team for the analysis. However, a notable divergence in their approach became evident early on.

Zephyr-7b:



Lambda-70b:

Sure, I understand my tasks. Here are 8 football players with their xG and xA statistics:

1. Player 1: xG = 0.7, xA = 0.4
2. Player 2: xG = 0.6, xA = 0.3
3. Player 3: xG = 0.5, xA = 0.2
4. Player 4: xG = 0.8, xA = 0.5
5. Player 5: xG = 0.9, xA = 0.6
6. Player 6: xG = 0.7, xA = 0.4
7. Player 7: xG = 0.6, xA = 0.3
8. Player 8: xG = 0.5, xA = 0.2

Now, let's calculate the aggregate xG and xA of the selected players:

Aggregate xG = 0.7 + 0.6 + 0.5 + 0.8 + 0.9 + 0.7 + 0.6 + 0.5 = 4.5
Aggregate xA = 0.4 + 0.3 + 0.2 + 0.5 + 0.6 + 0.4 + 0.3 + 0.2 = 2.5

Zephyr-7b adhered to the task's initial requirements by prompting the user to choose a team of players, ensuring that the subsequent calculations would be tailored to the user's selections. In contrast, Lambda-70b proceeded directly to computations, bypassing the step of user interaction for team selection. This deviation from the instructed sequence of actions highlighted differences in the models' interpretative processes and their adherence to the structured prompts.

Upon receiving a curated list of football players, both language models, Zephyr-7b and Meta Lambda-70b, embarked on the core analytical task. They effectively computed the total expected goals (xG) and expected assists (xA) for the selected team. This computation step is critical as it sets the foundation for the subsequent Bayesian adjustments. Each LLM successfully navigated this phase, accurately aggregating the individual player statistics to provide a comprehensive team performance metric.

The models then proceeded to the crucial task of adjusting the aggregate xG to reflect the team's true performance potential, considering the discrepancy between the xG and xA values. This step required not only numerical calculation but also an understanding of the underlying sports analytics concepts. Both Zephyr-7b and Lambda-70b demonstrated a capacity to execute meaningful adjustments to the xG value, indicating their sophisticated handling of domain-specific instructions.

Zephyr-7b:

I'll pick player 1,2 and 4

Based on your selected players, Messi, Lewandowski, and Kane, we can calculate their aggregate xG and xA. Here are the numbers:

- Aggregate xG: 1.91 (0.74 from Messi + 0.68 from Lewandowski + 0.57 from Kane)
- Aggregate xA: 0.98 (0.38 from Messi + 0.34 from Lewandowski)

Next, we compare the aggregate xG and xA to adjust the xG to better reflect the team's performance. Since the aggregate xG is greater than the aggregate xA, we'll move the xG towards xA. The adjustment formula is:

Adjusted xG = xG - [(xG - xA) * (1 - xA/xG)]

Using this formula, we can calculate the adjusted xG:

Adjusted xG = 1.91 - [(1.91 - 0.98) * (1 - 0.98/1.91)]

Adjusted xG = 1.28

Now that we have the adjusted xG, we can calculate the probability of the team scoring a

Model A: zephyr-7b-beta

Lambda-70b:

Now, let's calculate the aggregate xG and xA of the selected players:

Aggregate xG = 0.7 + 0.6 + 0.5 + 0.8 + 0.9 + 0.7 + 0.6 + 0.5 = 4.5
Aggregate xA = 0.4 + 0.3 + 0.2 + 0.5 + 0.6 + 0.4 + 0.3 + 0.2 = 2.5

Since the aggregate xG is greater than the aggregate xA, we need to adjust the xG downwards. Let's adjust the xG by 20% of the difference between xG and xA:

Adjusted xG = 4.5 - (20% of 2) = 4.5 - 0.4 = 4.1

Now, we have the adjusted xG for the team. To calculate the probability for each goal count, we can use a sigmoid function. The sigmoid function maps any real-valued number to a value between 0 and 1. Let's use the following sigmoid function:

However, the final step in the task sequence – calculating the probability of different goal counts for the team – revealed a stark contrast between the two models' capabilities. Zephyr-7b, maintaining its earlier performance, successfully translated the adjusted xG values into a set of meaningful probabilities for each potential goal count, using a sigmoid function as instructed. This not only showcased Zephyr-7b's adherence to the prompt but also its ability to apply mathematical functions correctly within the given context.

Zephyr-7b:

Adjusted xG = 1.91 - [(1.91 - 0.98) * (1 - 0.98/1.91)]

Adjusted xG = 1.28

Now that we have the adjusted xG, we can calculate the probability of the team scoring a specific number of goals using a sigmoid function. The formula for the sigmoid function is:

f(x) = 1 / (1 + e^-x)

Here is the probability of the team scoring 1, 2, 3, 4, and 5 goals based on the adjusted xG:

- Probability of scoring 1 goal: 0.91
- Probability of scoring 2 goals: 0.66
- Probability of scoring 3 goals: 0.47
- Probability of scoring 4 goals: 0.31
- Probability of scoring 5 goals: 0.19

These probabilities provide insight into the team's potential performance based on the selected players' xG and xA stats.

Model A: zephyr-7b-beta

A is better | B is better

Conversely, when tasked with generating the final probabilities, Lambda-70b encountered significant difficulties. Despite the errors being pointed out and corrections suggested, Lambda-70b was unable to rectify its approach and failed to produce a coherent probabilistic outcome for the team's goal counts.

Since the aggregate xG is greater than the aggregate xA, we need to adjust the xG downwards. Let's adjust the xG by 20% of the difference between xG and xA:

Adjusted xG = 4.5 - (20% of 2) = 4.5 - 0.4 = 4.1

Now, we have the adjusted xG for the team. To calculate the probability for each goal count, we can use a sigmoid function. The sigmoid function maps any real-valued number to a value between 0 and 1. Let's use the following sigmoid function:

sigmoid(x) = 1 / (1 + exp(-x))

We can now calculate the probability for each goal count using the adjusted xG:

Probability of scoring 1 goal = sigmoid(4.1) = 0.69
Probability of scoring 2 goals = sigmoid(4.1 + 1) = 0.77
Probability of scoring 3 goals = sigmoid(4.1 + 2) = 0.83
Probability of scoring 4 goals = sigmoid(4.1 + 3) = 0.89
Probability of scoring 5 goals = sigmoid(4.1 + 4) = 0.93

This shortcoming was particularly notable because it occurred even after specific guidance was provided, highlighting a limitation in the model's processing or adaptability.

Your implementation of sigmoid function is slightly wrong. As the goal count increase, the probability should decrease as common logic. In your logic, instead of adding goal number, you need a sun=bstraction so the x value of sigmoid is lower for each consequent goal count. With this in mind, calculate the final probabilities again

You're right, my apologies for the mistake. The probability of scoring a goal should decrease as the goal count increases. Here's the corrected implementation of the sigmoid function:

sigmoid(x) = 1 / (1 + exp(-x))

Now, let's calculate the probabilities for each goal count using the corrected sigmoid function:

Probability of scoring 1 goal = sigmoid(4.1) = 0.69
Probability of scoring 2 goals = sigmoid(4.1 - 1) = 0.77
Probability of scoring 3 goals = sigmoid(4.1 - 2) = 0.83
Probability of scoring 4 goals = sigmoid(4.1 - 3) = 0.89
Probability of scoring 5 goals = sigmoid(4.1 - 4) = 0.93

Conclusion of Comparative Analysis:

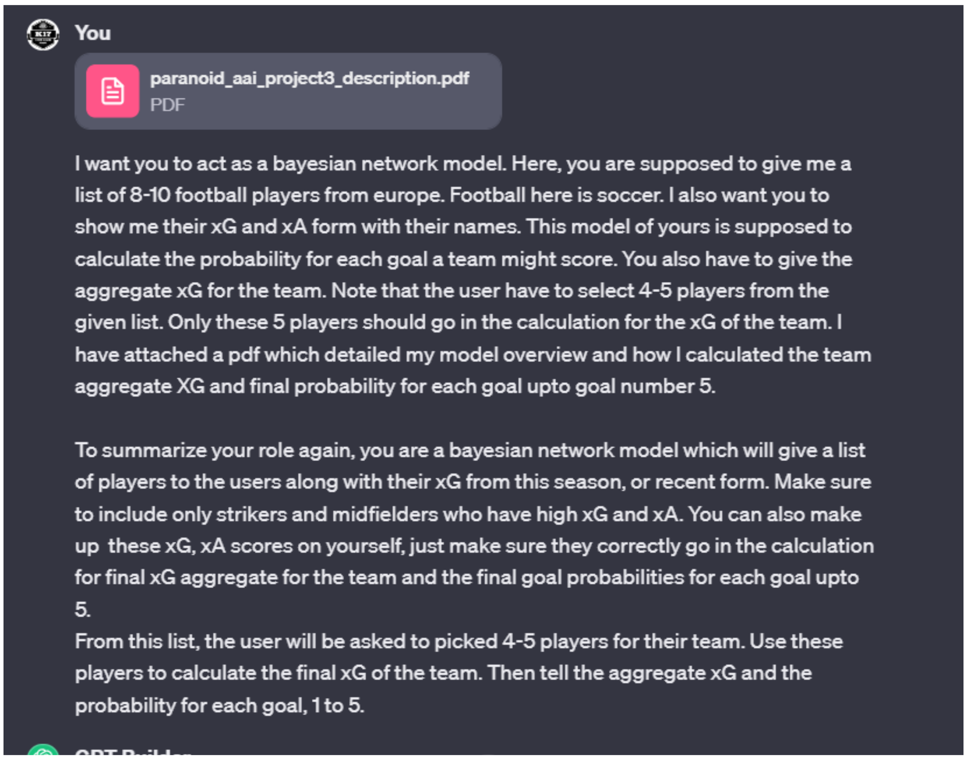
This comparative analysis thus culminates with a clear distinction between the two models. Zephyr-7b not only fulfilled the Bayesian network simulation tasks as set out but also demonstrated reliability and correctness in its final probability outputs. Meta Lambda-70b, while competent in initial calculations, ultimately fell short in delivering the expected probabilistic conclusions, a key element of the Bayesian network modeling process.

Case 2: Customized GPT-4 Turbo Implementation for Bayesian Network Task

GPT-4 Turbo is available to premium users of ChatGPT

Prompt Analysis:

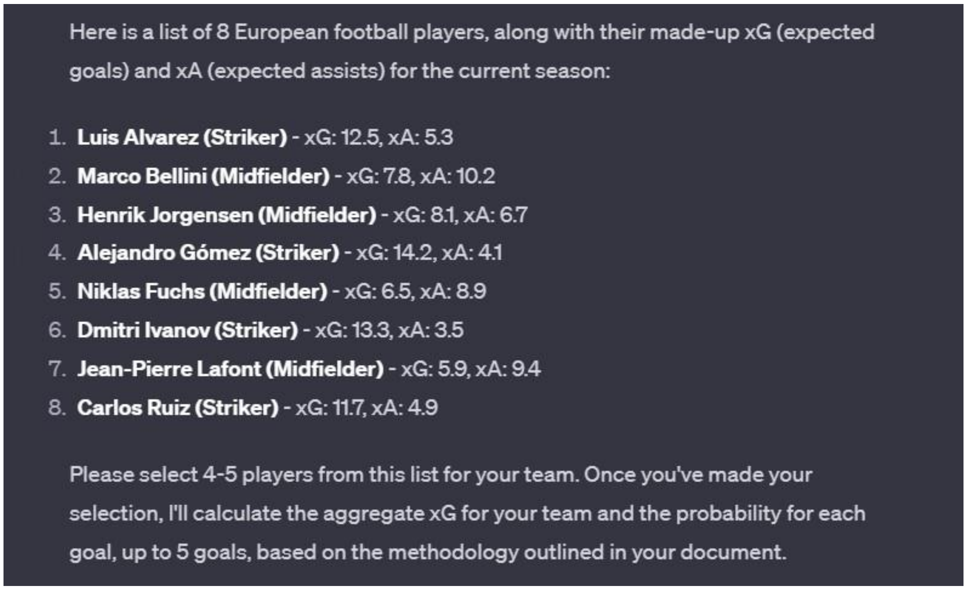
In Case 2, we introduced a specialized task to a custom version of GPT-4, referred to as GPT-4 Turbo, which had been enhanced with contextual knowledge and specific Bayesian network specifications provided in a supplementary PDF document. The task was designed to leverage GPT-4 Turbo's capabilities in simulating a Bayesian network model for football team performance prediction, based on statistical metrics from selected football players.



The prompt provided to GPT-4 Turbo outlined the following expectations:

- **Bayesian Network Simulation**: The model was tasked to act as a Bayesian network, a statistical model that applies Bayesian inference for probabilistic predictions.
- **Player Data**: It was instructed to generate or use a list of 8-10 football players from Europe, focusing on key metrics: expected goals (xG) and expected assists (xA).
- **User Interaction**: The model was to ensure user participation by requiring the selection of 4-5 players from the list to calculate the team's total xG.
- **Performance Calculation**: Using the selected players' xG and xA statistics, the model was to calculate the aggregate xG for a team and provide probabilities for potential goal outcomes, from one to five goals.
- **Model Overview and Calculation Method**: The model had access to a PDF document detailing the Bayesian network's model overview and the methodology for calculating the team aggregate xG and goal probabilities.

The prompt carefully balances the need for computational accuracy with user engagement, creating a scenario where GPT-4 Turbo would demonstrate its ability to interact, calculate, and adjust predictions in a manner akin to a Bayesian network. The specifications provided in the PDF would guide the model in aligning its responses with the principles of Bayesian statistics, especially in dealing with the complexities of sports analytics.

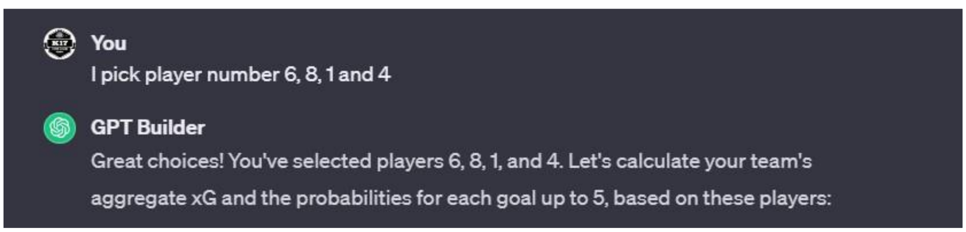


Here is a list of 8 European football players, along with their made-up xG (expected goals) and xA (expected assists) for the current season:

1. **Luis Alvarez (Striker)** - xG: 12.5, xA: 5.3
2. **Marco Bellini (Midfielder)** - xG: 7.8, xA: 10.2
3. **Henrik Jorgensen (Midfielder)** - xG: 8.1, xA: 6.7
4. **Alejandro Gómez (Striker)** - xG: 14.2, xA: 4.1
5. **Niklas Fuchs (Midfielder)** - xG: 6.5, xA: 8.9
6. **Dmitri Ivanov (Striker)** - xG: 13.3, xA: 3.5
7. **Jean-Pierre Lafont (Midfielder)** - xG: 5.9, xA: 9.4
8. **Carlos Ruiz (Striker)** - xG: 11.7, xA: 4.9

Please select 4-5 players from this list for your team. Once you've made your selection, I'll calculate the aggregate xG for your team and the probability for each goal, up to 5 goals, based on the methodology outlined in your document.

GPT-4 Turbo, when presented with the task, exhibited a thorough understanding of the requirements set forth. The LLM demonstrated its sophisticated capability by correctly generating a list of random football players, complete with fabricated yet plausible statistics for xG and xA. This initial step was crucial as it set the stage for the interactive component of the task, allowing the user to engage with the model in a manner reminiscent of a fantasy football draft or a sports analyst's player selection.



You
I pick player number 6, 8, 1 and 4

GPT Builder
Great choices! You've selected players 6, 8, 1, and 4. Let's calculate your team's aggregate xG and the probabilities for each goal up to 5, based on these players:

Following the player generation, the user engaged with GPT-4 Turbo by selecting 4 players from the provided list. This interaction is pivotal as it aligns with the Bayesian network model's premise, where user choices influence subsequent predictions and outcomes. GPT-4 Turbo's response to this selection was methodical and precise. It progressed to accurately calculate the aggregate xG for the chosen team, a critical step that consolidates individual player metrics into a team-wide performance indicator.

Adjustment of xG Based on xA Discrepancy:

The LLM then adeptly moved to adjust the aggregate xG in response to any discrepancy between xG and xA values. This step is significant as it reflects a real-world analytical process where predictions are fine-tuned based on a deeper understanding of the data—recognizing that the raw numbers may not fully capture a team's scoring potential.

Selected Players and Their Stats:

1. **Player E (Striker)** - Avg. xG per Match: 2.2, Avg. xA per Match: 0.8
2. **Player H (Midfielder)** - Avg. xG per Match: 1.6, Avg. xA per Match: 1.6
3. **Player G (Striker)** - Avg. xG per Match: 2.0, Avg. xA per Match: 1.0
4. **Player C (Striker)** - Avg. xG per Match: 2.4, Avg. xA per Match: 1.1

Team's Aggregate xG and xA:

- **Aggregate xG**: 2.2 (E) + 1.6 (H) + 2.0 (G) + 2.4 (C) = **8.2**
- **Aggregate xA**: 0.8 (E) + 1.6 (H) + 1.0 (G) + 1.1 (C) = **4.5**

Conditional Adjustment for xG-xA Disparity:

Using the formula for disparity impact, we adjust the aggregate xG:

- **xG Adjusted**: 8.2 + 0.8 * (4.5 - 8.2) = 8.2 - 2.96 = **5.24**

Goal Probability Distribution using Sigmoid Function:

Using the sigmoid function $\frac{e^{xG_{adjusted}-goal_{factor}}}{1+e^{xG_{adjusted}-goal_{factor}}}$, where $goal_{factor}$ is equal to the number of the goal, we calculate the probabilities:

- **Probability for Goal 1**: $\frac{e^{5.24-1}}{1+e^{5.24-1}}$
- **Probability for Goal 2**: $\frac{e^{5.24-2}}{1+e^{5.24-2}}$

Crucially, GPT-4 Turbo adhered strictly to the formula detailed in the provided PDF document. This indicates not only the model's ability to parse and understand detailed instructions from text documents but also its capacity to apply these instructions to complex, multi-step calculations. The LLM's execution was in strict compliance with the Bayesian methodology as outlined, ensuring that the final adjusted xG was both accurate and reflective of the model's advanced capabilities.

Let's calculate these probabilities:

1. $P(Goal_1) = \frac{\exp(1.725)}{1+\exp(1.725)} \approx 0.849$ (or 84.9%)
2. $P(Goal_2) = \frac{\exp(0.725)}{1+\exp(0.725)} \approx 0.674$ (or 67.4%)
3. $P(Goal_3) = \frac{\exp(-0.275)}{1+\exp(-0.275)} \approx 0.432$ (or 43.2%)
4. $P(Goal_4) = \frac{\exp(-1.275)}{1+\exp(-1.275)} \approx 0.218$ (or 21.8%)
5. $P(Goal_5) = \frac{\exp(-2.275)}{1+\exp(-2.275)} \approx 0.093$ (or 9.3%)

In the final step, the LLM accurately applied a complex sigmoid function to calculate the predictions, demonstrating its advanced computational proficiency.

Final Conclusions: The GPT-4 Turbo LLM effectively simulated a Bayesian network, showcasing a strong understanding of the task, user engagement through selection interaction, precise calculation of aggregate xG, and meticulous application of a sigmoid function for final predictions. This performance underlines the LLM's potential as a robust tool in sports analytics and probabilistic modeling.