Alma Mater Studiorum · University of Bologna

FantaBayesianNet[1]

Fundamentals of Artificial Intelligence and Knowledge Representation (Module 3)

Giuseppe Murro (giuseppe.murro@studio.unibo.it)
Giuseppe Boezio (giuseppe.boezio@studio.unibo.it)
Salvatore Pisciotta (salvatore.pisciotta2@studio.unibo.it)

May 14, 2021

[1]The code for the project is publicly available on GitHub

Contents

1 Introduction

The task of predicting the performance of football players is gaining increasing attention in the sports and statistical communities. In this work, we used a Bayesian Network to model relevant factors of player performance in the presence of noisy data. We apply our analysis to the 2020–2021 season in the top Italian league, Serie A, and use the player ratings provided by a popular Italian fantasy football game, the so-called Fantacalcio. Actually in literature there is a lack of published research on modeling this kind of data within a Bayesian framework. The only work done on this field is [3] which propose various hierarchical models for predicting player ratings.

Instead in this project we are focused to explore the causal relations that allow to predict if a football player can be deployed or not in a fantasy football lineup. Different kind of reasoning are done to exploit the flow of influence between variables and possible independences. The implementation of the model and the methods used for inference are based on the pgmpy [1] python library.

2 Background

Fantasy sports games typically involve roster selection and match-by-match challenges against other participants with the results determined by the collective performance of the players on the fantasy rosters. In Italy, fantasy football was popularized by the brand *Fantacalcio* [4] edited by Riccardo Albini in the 1990s and in the rest of the report we use the original denomination for referring at the Italian game.

Each player in the Italian Serie A league has an associated price determined by various factors including past performance and forecasts for the upcoming season. At the beginning of the season, the virtual managers are allocated a limited amount of virtual money with which to buy the players that will comprise their roster. Each roster will consist of 3 goalkeepers, 8 defenders, 8 midfielders and 6 forwards.

For every matchday in Serie A, virtual managers must *deploy* a fantasy team formed by 11 players divided into their respective roles according to the modules (often 4-3-3 or 3-4-3). After every match, the prominent Italian sports periodicals assign each player a rating, a so-called *grade*, on a scale from 1 to 10. In practice there is not much variability in these grades; they typically range 4 four to 8, with the majority between 5 and 6. These grades are very general and largely subjective performance ratings that do not account for significant individual events (goals, assists, yellow and red cards, etc.) in a consistent way.

As a means of systematically including specific in-game events in the ratings, Fantacalcio provides the so-called *point scoring system*. Points are added or deducted from a player's initial grade for specific positive or negative events during the match. The point scores are more variable than the grades, especially across positions (e.g. when comparing defending and attacking players). Goalkeepers suffer the most from the point scoring system, as they are deducted a point for every

goal conceded. On the other extreme, forwards (attacking players) typically receive the highest point scores because every goal scored is worth three points.

For player i in match t the total score $y_{i,t}$ is

$$y_{i,t} = G_{i,t} + P_{i,t}$$

where G is the grade and P is the point score. Table 2.1 lists the game features that contribute to a player's point score $P_{i,t}$ for a given match.

Event	**Points**
Goal	+3
Assist	+1
Penalty saved*	+3
Yellow card	-0.5
Red card	-1
Goal conceded*	-1
Own goal	-2
Missed penalty	-3

Table 2.1: Bonus/Malus points in Fantacalcio. The symbol * denotes an event only applicable to goalkeepers.

Since it is very rare for a player to participate in all matches, some $y_{i,t}$ are missing, and this may be due to different reasons:

- the player does not play in the match because of injury, disqualification, coach's decision, or some other reason;
- the player does not participate in the match for long enough for their impact to be judged by those tasked with assigning the subjective grade

For all these, the score is marked with the initials «sv».

3 Dataset

3.1 Data acquisition

The first step consists in load data, merge different data frames to get a single data frame of players and process feature in order to work with them.

Since statistics of single football players of Serie A in each mach are not directly accessible and they are distributed on different web sites, we have developed a web scraping script (accessible from the repository of this project) able to collect data from Fantacalcio.it [4] and Transfermarket.it [2] and merge them in order to obtain a single dataset.

We decided to select only forwards players, since their statistics are the most interesting from the view point of the Fantacalcio, and we picked an average of 2 players per team for a total amount of 40 players. These data are about the first 30 rounds of the 2020–2021 Serie A season.

3.2 Data preprocessing

In the initial dataset we had a lot of data, but only some of them were useful for our model. So starting from the raw dataset we applied three preprocessing operations:

- Data cleaning from useless columns and rows that refers to "postponed" matches that did not have information about players, since the game was not played.
- Data discretization since pgmpy does not support learning from continuous variable, and so all numerical values are mapped into bins
- Data transformation of the missing values in the "score" attribute into the category "sv" (meaning without a score), which is the symbolic convention in Fantacalcio that indicates the absence of score for a player.

4 Bayesian network

4.1 Structure

4.1.1 Variables

- **goal** = Goal: represents whether, at the correspondent matchday, the player has made a goal and obtained the correspondent bonus (+3)
  - True: the player has scored at least one goal
  - False:The player didn't even score one goal
- **assist** = Assist: represents whether, at the correspondent matchday, the player has made an assist and obtained the correspondent bonus (+3)
  - True: The player has made at least one assist
  - False:The player didn't even make one assist
- **time_range** = Range of minutes: minutes played by the player in the correspondent match
  - 0-15: The player played no more than 15 minutes
  - 16-45:The player has played more than 15 minutes but less than 45 minutes
  - 46-90:The player has played more than 45 minutes
- **score_range** = Range of "fantasy score": fantasy score obtained by the player at the correspondent matchday, with appropriate bonus and malus included
  - s.v. :The player has not played or has played too little time to have a vote
  - <=5 :the player had a bad game with some possible malus
  - 5.5-6.5 :the player did not play a great game or at most a sufficient game
  - 7-9.5 :the player played a great game and may have taken some bonuses as well
  - >=10 :the player played an excellent game and also took some bonuses

- **difficulty_match**= Difficulty of the match calculated in crescent order through a formula
  - 1 :easy match generally played at home
  - 2 :easy match generally played away
  - 3 :balanced match
  - 4 :difficult match generally played at home
  - 5 :difficult match generally played away
- **penalty_kicker**= Kicker of the team for penalties: represents whether the player (if at the correspondent matchday was available) is the one designed to kick the penalties
  - True: The player is the penalty kicker
  - False:The player isn't the penalty kicker
- **home_match**= : represents whether, at the correspondent matchday, the player's team played at home or away
  - True: the team played the match at home
  - False: the team played the match away from home
- **yellow_card**= Admonition: represents whether, at the correspondent matchday, the player the player got a yellow card with correspondent malus(-0.5)
  - True: the player received a yellow card
  - False: the player didn't receive a yellow card
- **red_card**= Expulsion: represents whether, at the correspondent matchday, the player the player got a red card with correspondent malus(-1)
  - True: the player received a red card
  - False: the player didn't receive a red card
- **available**= Availability: represent whether the player at the corresponding matchday was available for his team or not (e.g. for an injury or a disqualification)
  - True: the player was available
  - False: the player wasn't available
- **deployability**=Deployability: represents whether the player for the corresponding matchday will be a good choice for the fantasy team of the fantasy manager
  - True: deploy the player is a good choice
  - False: it is better, if possible, not to deploy the player

4.1.2 Network

The building step of the model started with the realization of the first network based only on the domain's knowledge. Move forward the structure was improved and developed using correlation matrix analysis that lead us to the final network presented in Figure 4.1.

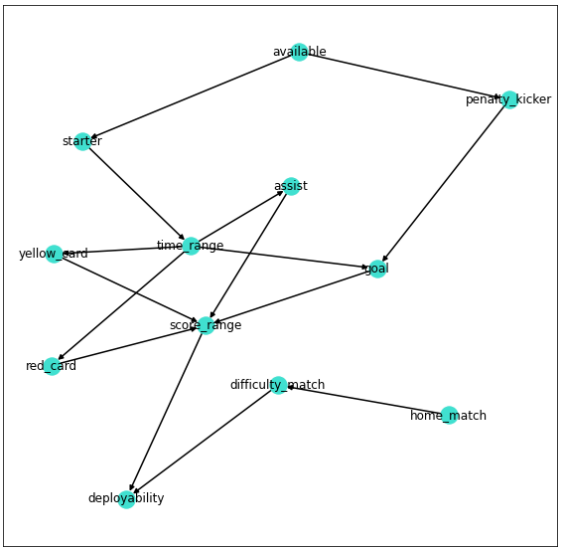

Figure 4.1: Bayesian Network

4.2 Active Trails

We have done some examples of reasoning patterns, namely causal reasoning and evidential reasoning, and possibile active-trails in the directed graph. A trail $X \leftrightharpoons Z \leftrightharpoons Y$ is active if influence can flow from X to Y via Z.

The first example consists in showing how **P(deployability)** is influenced by **available** because there is an active trail between the two variables and there is no evidence or V structure that can break it.

Then, it can be shown that **starter** in the evidence behaves as a blocking node breaking an active trail between the two aforementioned variables.

In both cases there is a flow of influence because the graph is **multiply connected** i.e. there is more than one trail which connects two nodes.

Active trails are shown in figure 4.2

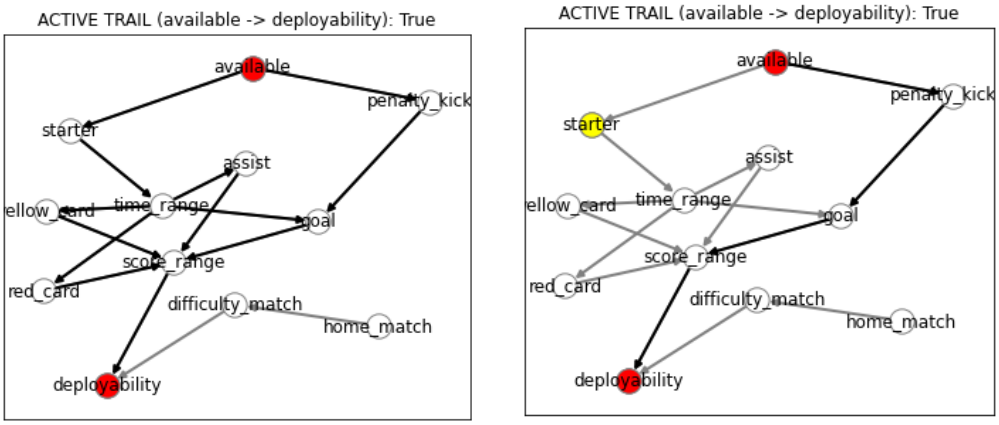

Figure 4.2: Active trails between available and deployability

The second example consists in showing conditional independencies between variables: **starter** and **penalty_kicker** break the two possible active trails between **available** and **deployability**. This is compliant with the **d-separation principle**, that allows us to determine whether a set X of variables is independent of another set Y, given a third set Z: if there is no active trail between two variables X and Y, then X and Y are d-separated. Thus:

$$P \models (available \perp deployability | starter, penalty\_kicker)$$

This means that adding **available** to the evidence gives no additional information to the query. This can be proved also comparing the two probability distributions and showing that for all combinations of values they provide the same results. Comparison is shown in figure 4.3

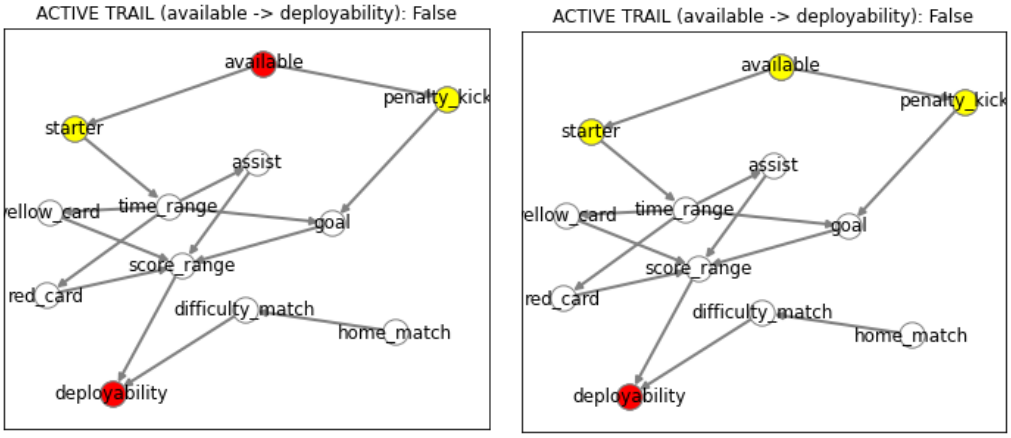

Figure 4.3: Broken active trails

In the third example we are trying to understand whether a **score range** between 7-9.5 could be due to the fact that the player made a goal or not.

It can be shown also in this case that there is a **direct cause** from **goal** to **score_range**. This means that exists an active trail between the two variables.

Looking at the probability distribution, it can be understood how the goal could be a possible explanation to the received score with a probability around 58 %.

The next step is to understand whether the goal could be a good explanation even if the player got a yellow card. In this case we put in the evidence also the **yellow card** and this variable influences the goal because **score_range** allows the activation of a V-structure, thus there is an active trail between **goal** and **yellow_card**.

An interesting thing is to highlight how the probability to have made a goal increases, probably because it is very difficult that a player can get a good score with a yellow card without a goal. Active trail is shown in figure 4.4

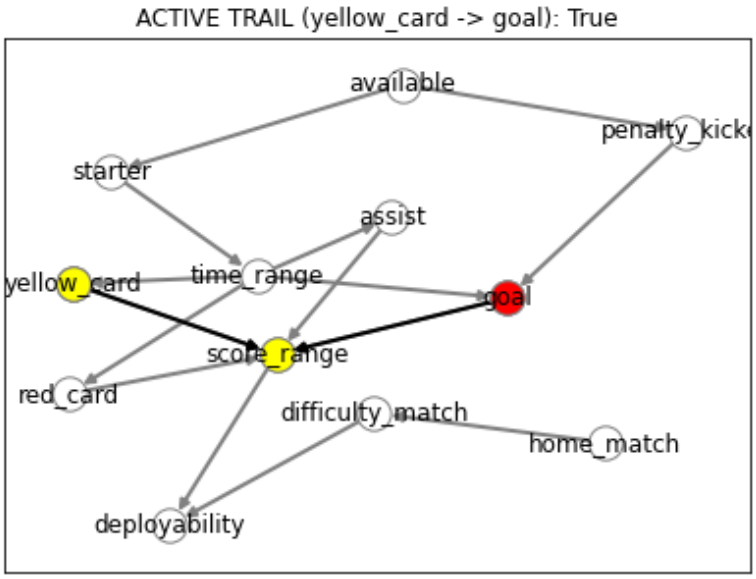

Figure 4.4: Active trail with V structure

4.3 Independencies

The V-structure between score_range and difficulty match allows to understand the independence between goal and difficulty match because this kind of structure breaks the only possible active trail between the two variables. In our domain this is quite reasonable because a good player can score a goal independently by the opposing team. This means that:

$$P \models (goal \perp difficulty\_match)$$

It has been shown in two different ways:

1. Showing that **P(difficulty_match | goal) = P(difficulty_match)** as shown in figure 4.5
2. Showing that there is no **active trail** in the graph between the two variables as showed in figure 4.6

P(difficulty_match|goal=True)

difficulty_match	phi(difficulty_match)
difficulty_match(1)	0.1101
difficulty_match(2)	0.2767
difficulty_match(3)	0.3187
difficulty_match(4)	0.2212
difficulty_match(5)	0.0734

P(difficulty_match)

difficulty_match	phi(difficulty_match)
difficulty_match(1)	0.1101
difficulty_match(2)	0.2767
difficulty_match(3)	0.3187
difficulty_match(4)	0.2212
difficulty_match(5)	0.0734

Figure 4.5: Comparison between probability distributions

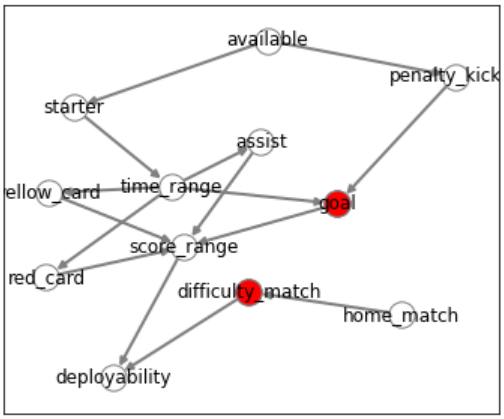

Figure 4.6: Independence between goal and difficulty match

5 Inference

5.1 Exact Inference

The query considered for a deeper analysis with exact inference is the probability of the player to be deployed given that he is a starter, more formally

$$P(Deployability|starter)$$

There are different ways to solve with exact inference this query. The first method, which is not the best one, consists of using marginalization property and global semantics to decompose the probability as the product of different probabilities iterating over the variables which are neither query variables nor evidence ones. This method is called inference by enumeration. For previous query write the whole decomposition would be too chaotic and dispersive, thus it will be provided a sketch of the steps to follow

$$\alpha P(Deployability, starter)$$

$$\alpha \sum_{hidden} P(Deployability, starter, hidden)$$

$$\alpha \sum_{hidden} P(Deployability|difficulty\,match, score\,range)P(starter|available)P(hidden|parents(hidden))$$

The query is solved using a depth first search which turns out to be not so efficient because same terms appearing in different branches are computed several times.

Better results can be obtained using a sort of dynamic programming which allows to store intermediate results avoiding repeated calculations. There are different methods in this category, the one that has been used to execute the query is called variable elimination and consists of the following steps:

- substituting each probability in the first formula with a factor which is a matrix built considering variables over which there is the summation. For example:

  $$P(Deployability|difficulty\,match, score\,range)$$

  will become $f_n(Deployability, Difficulty\,match, Score\,range)$
- Then, in right-to-left order factors are multiplied (element-wise) getting new simplified factors which allow to get an intermediate result and saving time and space for computations.

Different ordering of the variables can be tried but as heuristic a good one consists of eliminating whichever variable minimizes the size of the next factor to be constructed. It means to eliminate all variables which are ancestors neither the query variable nor evidence ones nor d-separated from both of them.

In our query there are not variables that can be eliminated because starting from deployability it is possible to reach all ancestors.

Variable elimination algorithm can be executed using different heuristics to order elimination variables. Heuristics adopted in pgmpy are the followings:

- MinFill: the number of edges that need to be added to the graph due to its elimination.
- MinNeighbors: the number of neighbors it has in the current graph.
- MinWeight: the product of weights, domain cardinality, of its neighbors.
- WeightedMinFill: the sum of weights of the edges that need to be added to the graph due to its elimination, where a weight of an edge is the product of the weights, domain cardinality, of its constituent vertices.

Considering the query:

$$P(deployability|starter = True)$$

it can be shown that query results using previously mentioned heuristics are the same but we have better time performance using MinWeight.

Following table shows different results and orderings.

Heuristic	Ordering	Time
MinFill	[red_card, score_range, home_match, difficulty_match, goal, yellow_card, time_range, penalty_kicker, available, assist]	0.068 s
MinNeighbors	[home_match, available, difficulty_match, penalty_kicker, red_card, time_range, goal, yellow_card, assist, score_range]	0.035 s
MinWeight	[available, home_match, penalty_kicker, difficulty_match, time_range, score_range, red_card, goal, yellow_card, assist]	0.031 s
WeightedMinFill	[home_match, available, penalty_kicker, difficulty_match, time_range, score_range, red_card, goal, yellow_card, assist]	0.034 s

A problem of exact inference with variable elimination is that it is NP-hard in the case of multiply connected networks (i.e. there are multiple trails to connect two variables) as the network of this project.

5.2 Approximate Inference

Starting from the results provided by the exact inference, using the variable elimination method, it is possible to analyze and compare the behavior of three approximate inference methods: the rejection sampling, the likelihood weighting and the Gibbs sampling all of them implemented using the pgmpy library.

To observe how these inference methods behave in different situations, we tested 4 interesting queries:

1. $P(goal|assist)$
   The result of this query (see Fig. 5.1) highlights that a player has a probability of around 30%

of make goal when he has already done an assist. Starting from the fact that the probability of a player to score a goal, without any evidence, is around 25% we can deduce that the probability increase due to the more comfortable approach of the forward into the match.

2. $P(penalty\_kicker\ |available, goal, \neg starter)$
   This query shows (see Fig. 5.2) that whether a player scored a goal starting the match from the bench, there is a probability of 53% that he is the penalty kicker of the team. This result highlights that it is quite probable that the goal is the consequence of a marked penalty.

Analysis of sampling methods:
As we can see from 5.1 and 5.2 since the second query has more evidence of the first one the time of the rejection sampling dramatically increase. On the contrary, likelihood weighting and Gibbs sampling maintain the same sampling times. Indeed, the method of rejection sampling is implemented by generating samples from forward sampling and rejects those that do not match our evidence and keep doing until it has as sample as the sample size. Moreover, Gibbs sampling appears poorly when the sample sizes are not big. This is due to the implementation of the sampling method in pgmpy library since it does not allow to explicit the evidence variables. For this reason, a lot of samples that are not compatible with evidence are rejected.

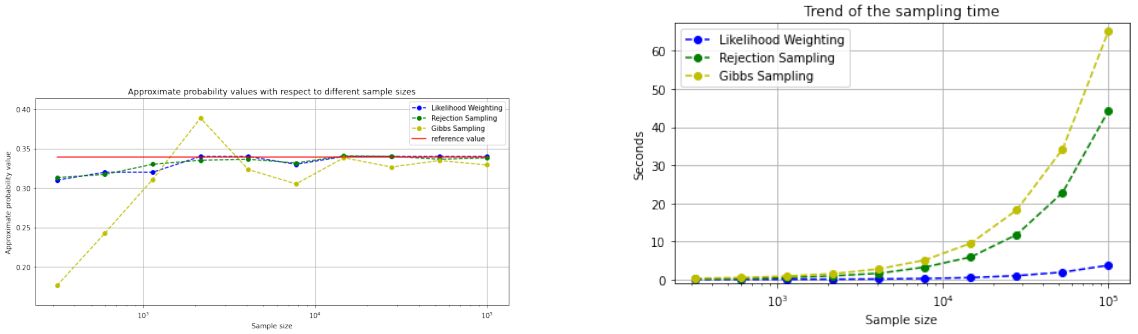

Figure 5.1: Probability values and sampling times of $P(goal|assist)$

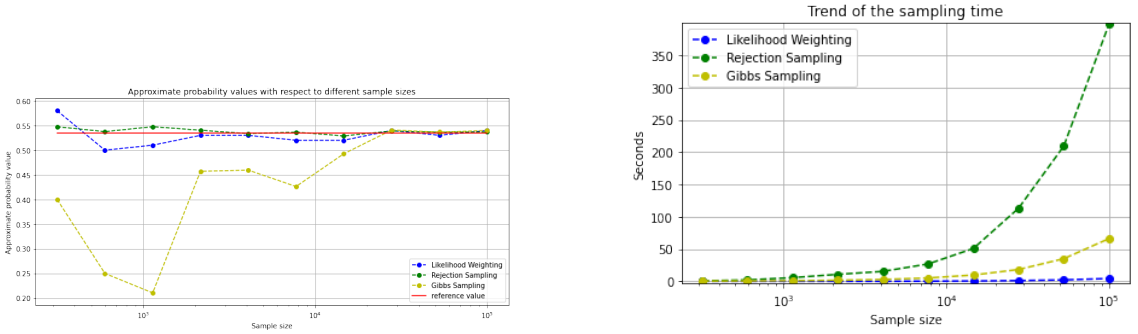

Figure 5.2: Probability values and sampling times of $P(penalty\_kicker|available, goal, \neg starter)$

3. $P(deployability|yellow\_card, assist)$
   The result of the query it is quite interesting because the malus of the yellow card (-0.5) doesn't influence the deployability. This happens because, nevertheless the player has

received an admonition he played a good game with an assist (bonus +1), so it is a good choice to deploy him in lineup.

4. $P(goal|time\_range = 0 - 15)$
   The result of this query is quite obvious, a player need more minutes to score a goal, so if you are looking for a high score in your Fantacalcio team and you know that the forward could play only few minutes, it is better not to deploy him.

Analysis of sampling methods:
As we can see from 5.3 and 5.4 the errors of likelihood weighting and Gibbs sampling are higher in these two queries with respect to previous two. This happens because they converge really slow with probabilities close to 0 (like in the forth query) and 1 (like in the third query)

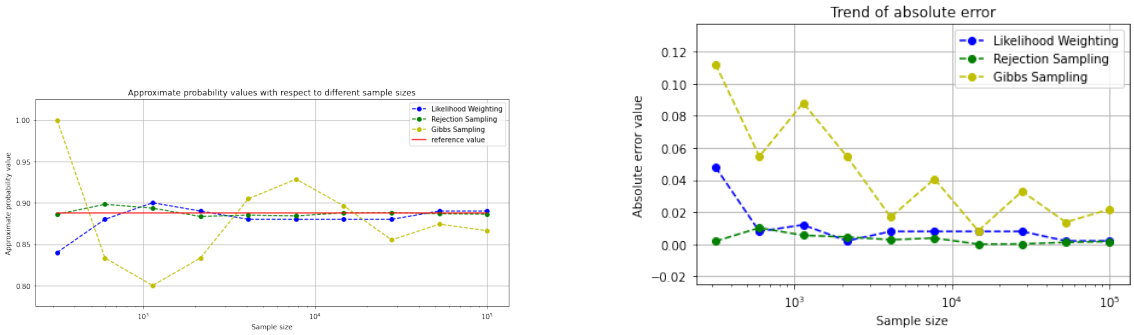

Figure 5.3: Probability values and absolute error trend of $P(deployability|yellow\_card, assist)$

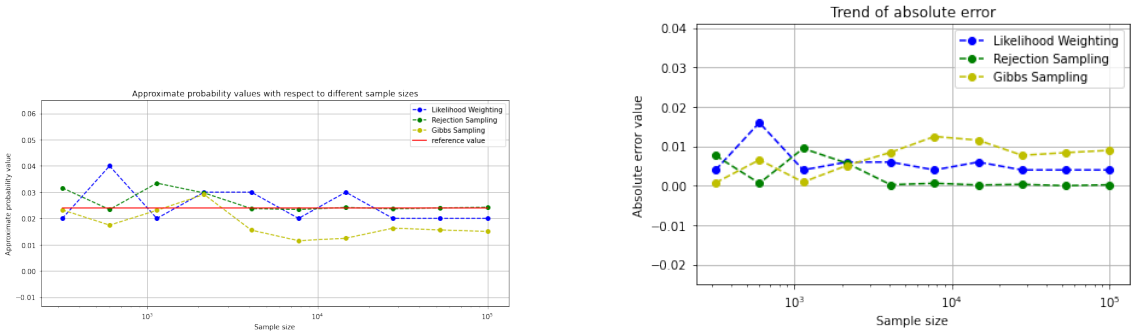

Figure 5.4: Probability values absolute error trend of $P(goal|time\_range = 0 - 15)$

6 Conclusion and future works

The results provided in this paper are reasonable admissible but since we tried to make inference about a complete stochastic domain, it found out to be really difficult to obtain good results because it was not always possible to take into account the enormous amount of variables that effectively affect the domain of our analysis and construct a model on it. A possible different analysis based on our model could be to extend the dataset of players, considering not only forwards but also midfielders and defenders in order to prove whether the model could be a good base to develop a real software that helps fantasy managers to solve their doubts in line up their fantasy football formation.

Bibliography

[1] Ankur Ankan and Abinash Panda. "Pgmpy: Probabilistic graphical models using python". In: *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. 2015. URL: https://pgmpy.org/.

[2] Transfermarkt GmbH Co. *Transfermarket*. URL: https://www.transfermarkt.it/.

[3] Gabry J.S. Egidi L. "Bayesian hierarchical models for predicting individual performance in soccer". In: *Journal of Quantitative Analysis of Sports* (2018), pp. 143–157.

[4] Quadronica s.r.l. *Fantacalcio*. URL: http://www.fantacalcio.it.