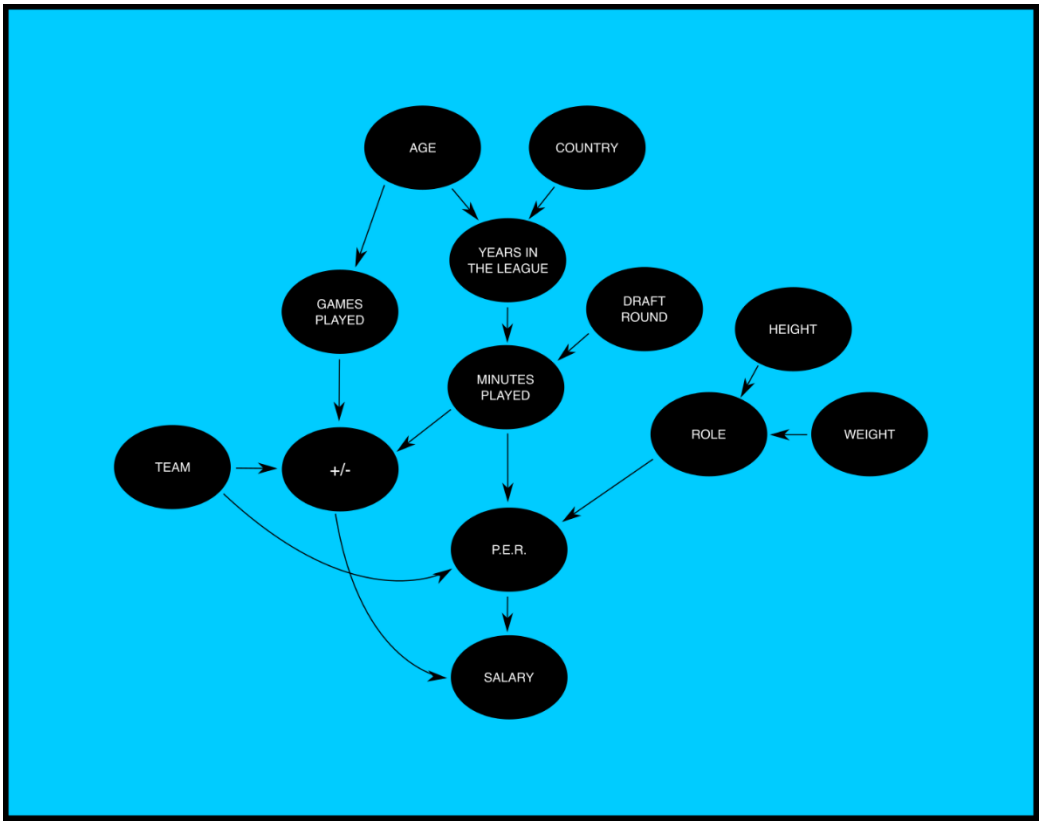


NBA SALARY PREDICTION

A BAYESIAN NETWORK TO MODEL THE SALARIES OF NBA PLAYERS

Giacomo Berselli | Fundamentals of AI and Knowledge Representation | 04/01/2022

INTRODUCTION

Determining the factors that influence National Basketball Association (NBA) owners to pay players has always been of great importance in light of financial constraints such as the NBA salary cap. Furthermore, the players themselves decided to assume teams of data scientists because interested in what variables affect their salaries, to evaluate if they were adequate or not.

This project aims to understand the relationship between a player's characteristics, his statistics, and his annual salary. Analyses were made on the performance and skills of the players, to understand whether the salaries are appropriate concerning their peers and which factors are most crucial. Finally, conclusions were drawn regarding the likelihood of improvement of a given player and the relative probability of injury, which can be of great help to teams during trade periods.

The task was done by collecting the official data of the players in the 2020-2021 regular season from the official website of the NBA and that of ESPN. Then the database was processed through a Bayesian Network, which has allowed to deal with uncertainty and partial information and to make probabilistic predictions by defining probabilities of different causes that could affect the target in a very simplified model of the domain of interest.

STATISTICS & INFORMATIONS

The following characteristics were taken into account for each player:

- `TEAM`: team with which the player has played most of the games during the regular season.
- `AGE`: player's age.
- `GAMES PLAYED`: number of games played in regular season.
- `MIN`: average number of minutes played per game.
- `+/−` : keeps track of the net changes in the score when a given player is either on or off the court. In particular, it is considered the average +/- value per game.
- `PER`: identifies the Player Efficiency Rating of a player in the 2020-2021 regular season. It is a rating system developed by John Hollinger that aims to create a summative rating of each player in the NBA. It takes into account points signed, rebounds, assists, turnovers, steals, blocks, personal fouls, and many other variables per game, and produces as output a weighted rating for each player.
- `POS`: player's role.
- `HEIGHT`: height in centimeters.
- `WEIGHT`: weight in pounds.
- `YOS`: years of service of a player in the league.
- `DRAFT`: round number associated to the player's pick.
- `COUNTRY`: country of origin.
- `SALARY`: annual salary.

The data used in the project were taken from the following sites:

- Players regular season stats: National Basketball Association (NBA) official site
- Players info: Real GM
- Players salary: ESPN & Spotrac

They have been collected in two `.csv` files to easily load them and apply the necessary preprocessing.

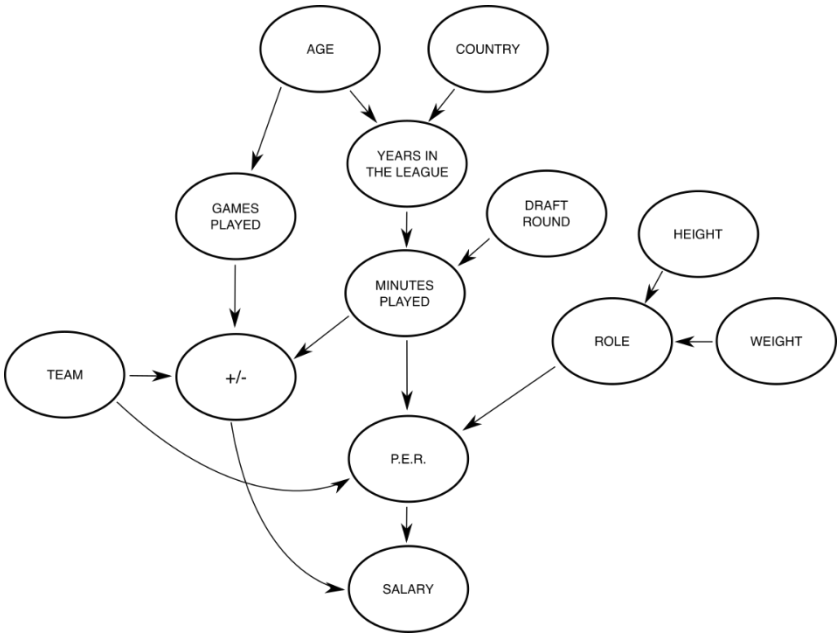
PREPROCESSING

A preprocessing step was necessary since some of the categories were not numerical and could contain many different values, leading to CPTs too large to handle, while other ones useless for our purpose. The procedure followed was to divide each category into intervals and then assign to each of them an integer, before applying the learning algorithm. Since the categories are very different from each other and there were outliers in some of them, a simple division into predefined bins all with the same size could not be effective. For this reason, each category was divided by carefully analyzing the data and taking advantage of my prior knowledge on the subject. Therefore, the dataset was manipulated as follows:

- TEAM: teams were divided based on their geographical location (Atlantic, Central, SouthEast, NorthWest, Pacific and SouthWest) and an integer was associated to each region.
- POS: players' roles in the database were multiple but what mainly changed was only the name. For this reason they were all brought back to the 5 fundamental positions, in the following way: PG/G (Point Guard) -> 1, SG/GF (Shooting Guard) -> 2, SF/F (Small Forward) -> 3, PF/FC (Power Forward) -> 4, C (Center) -> 5.
- DRAFT: identifies if a player has been drafted in a pick of the first round (DRAFT = 1), of the second round (DRAFT = 2), or is never been drafted to enter in the league (DRAFT = 3).
- COUNTRY: split US players (COUNTRY = 2) and foreign players (COUNTRY = 1)
- SALARY: the variable was already numerical so it was simply split into 8 different labels. Since most of the players in the league have a salary incredibly lower than the top 10 NBA players, more weight was given to the salaries less than 10 million USD to equally distribute the players inside the bins.
- AGE, HEIGHT, WEIGHT, MIN, GP, +/-, PER, YOS: all these variables were already numerical so the values were simply split into bins of different sizes according to need.

BAYESIAN NETWORK

Bayesian Network (BN) is a type of probabilistic graphical model that uses Bayesian inference for probability computations. Bayesian networks aim to model conditional dependence, and therefore causation, by representing it as arrows in a directed acyclic graph (DAG). Through these relationships, one can efficiently conduct inference on the random variables in the graph through the use of factors. In this case, the structure of the model has been created using both correlation analysis and my personal experience, having played basketball for more than 15 years and being a big NBA fan.



Obviously, this network version is simplified compared to the one needed in a real-world scenario. In fact, as previously mentioned, most of the seasonal statistics of a player were summarized in the variables '+/-' and 'PER', to not make the network computation too slow. Starting from the top, the figure shows that node ‘AGE’ is the parent of ‘GAMES PLAYED’ and ‘YEARS IN THE LEAGUE’. The former causal link is chosen because as a player gets older, the chances of injuries usually increase, while the latter is pretty obvious. ‘YEARS IN THE LEAGUE’ is also the child of ‘COUNTRY’, because tipically US players start to play in NBA earlier with respect to foreign ones, such as Europeans. Next, the years of service in the league are of course determinant in the minutes played, since as experience increases so do the minutes played per game. This latter variable is however influenced for the first years by the round’s pick of the draft, since the sooner a player has been chosen, the more the team has decided to invest in him. The value of the ‘+/-‘ depends on the team obviously, but it is noticed that players with more minutes and games in the season tend to have higher values than others, probably because better and more incisive and with more chances to score. The value of 'PER' also depends on the team and minutes played, but some of the values that affect it, such as rebounds, blocks, and turnovers, depend on the player's role, with the first two typically higher in players with a role of 4 or 5, and the third in play-makers. Established that the role of a

player depends on his height and weight, to conclude, the final node of the network represents the annual salary of a player, which has a causal relationship with the performances held during the past seasons, summarized by the nodes 'PER' and '+/-'.

CONDITIONAL PROBABILITY TABLE (CPT)

Each node of the network has associated a Conditional Probability Distribution (CPD) considering as given variables the direct parents. The function `BayesianNetwork.get_cpds` of PGMPY allows to calculate the CPD of a desidered variable and to show it as a table. In the figure below is shown the Conditional Probability Table (CPT) of the variable `POS`.

```
# Create 'POS' CPD
cpd = model.get_cpds('POS')
print(cpd)
```

```
+--------+-----------+-----------+-----+----------------------+
| HEIGHT | HEIGHT(1) | HEIGHT(1) | ... | HEIGHT(5)            |
+--------+-----------+-----------+-----+----------------------+
| WEIGHT | WEIGHT(1) | WEIGHT(2) | ... | WEIGHT(5)            |
+--------+-----------+-----------+-----+----------------------+
| POS(1) | 1.0       | 0.2       | ... | 0.0                  |
+--------+-----------+-----------+-----+----------------------+
| POS(2) | 0.0       | 0.2       | ... | 0.0                  |
+--------+-----------+-----------+-----+----------------------+
| POS(3) | 0.0       | 0.2       | ... | 0.0                  |
+--------+-----------+-----------+-----+----------------------+
| POS(4) | 0.0       | 0.2       | ... | 0.058823529411764705 |
+--------+-----------+-----------+-----+----------------------+
| POS(5) | 0.0       | 0.2       | ... | 0.9411764705882353   |
+--------+-----------+-----------+-----+----------------------+
```

By looking for instance to the first cell of the table, we can say that the probability for a player to be a Point Guard (PG) or a Guard (G) given that it has a height less than 180 cm and a weight less than 200 lbs is of 100%. On the other side, the probability for a player to be a Center (C/FC) given that it has a height greater than 210 cm and a weight greater than 260 lbs is approximately 94%.

INDEPENDENCE

The notion of independence is a key in the power of Bayesian networks, which can represent essentially any full joint probability distribution very concisely if each node is conditionally independent of its other predecessors given its parents. Independence assertions are usually based on knowledge of the domain and can dramatically reduce the amount of information necessary to specify the full joint distribution and so the complexity of the inference problem. Unfortunately, a clean separation of entire sets of variables by independence is quite rare. Bayesian networks leverage conditional

independence between variables, given other variables. Conditional independence assertions are brought by direct causal relationships in the domain and can allow probabilistic systems to scale up. Moreover, they are much more commonly available than absolute independence assertions.

Below it is shown the local independencies of nodes `YOS` and `+/-` :

```
# Local independencies for 'YOS' and '+/-'
print(model.local_independencies('YOS'))
print(model.local_independencies('+/-'))
```

```
(YOS ⊥ DRAFT, POS, WEIGHT, HEIGHT, GP, TEAM | AGE, COUNTRY)
(+/- ⊥ DRAFT, AGE, COUNTRY, POS, PER, YOS, WEIGHT, HEIGHT | MIN, GP, TEAM)
```

As we can see, given their parents, the nodes are independent of all the other ones which cannot be reached directly.

Another important independence property is implied by the topology of the network semantics: a node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents (so given its Markov Blanket). In this case, we can print the Markov Blanket of the node `+/-` :

```
# Markov blanket for '+/-'
model.get_markov_blanket('+/-')
```

```
['MIN', 'GP', 'PER', 'SALARY', 'TEAM']
```

In particular, the nodes `TEAM`, `GP` and `MIN` are the parents, `SALARY` is the children, while `PER` is the children's parent.

Finally, the d-separation principle allows us to determine whether a variable X is independent of another variable Y, given a third one Z. In particular, PGMPY holds two methods for this particular task. The first one is `BayesianNetwork.active_trail` which shows all the nodes reachable by an active trail starting from the node defined as parameter. It is also possible to define a set of observed nodes, to understand how the active trails change, as shown in the figure below:

```
# Get active trail nodes
print(model.active_trail_nodes('AGE'))
print(model.active_trail_nodes('AGE', observed='PER'))
```

```
{'AGE': {'PER', 'AGE', 'MIN', '+/-', 'YOS', 'SALARY', 'GP'}}
{'AGE': {'DRAFT', 'MIN', 'AGE', 'COUNTRY', 'POS', 'HEIGHT', 'GP', '+/-', 'YOS', 'SALARY', 'WEIGHT', 'TEAM'}}
```

An important aspect to notice is that by introducing `PER` as observed node, the nodes `TEAM`, `COUNTRY`, `POS`, `HEIGHT` and `WEIGHT` could be reached by an active trail starting from `AGE`, while this wasn't true previously.

The second method is instead `BayesianNetwork.is_dconnected` which checks directly if two variables passed as parameters are d-connected by looking for an active trail between the two, along the edges of the network. If there is no active trail between

the two variables, they are d-separated. With this configuration of the network, the nodes AGE and PER are d-connected, but this is no longer true if the node MIN is observed, as shown in the figure below:

```
# Check for d-separation between variables
print(model.is_dconnected('AGE', 'PER'))
print(model.is_dconnected('AGE', 'PER', observed=['MIN']))
```

```
True
False
```

INFERENCE

Unfortunately, the BayesianNetwork.get_cpds function only returns information of CPDs where the given state is made up of variables that are parents of the query variable. To compute CPDs of other variable combinations, inference methods must be used. The first fundamental function used in the project to inference different types of probabilities from the network was exact_inference. It computes analytically the posterior probability distribution for a set of query variables, given some observed event. To perform the exact inferences, I used the VariableElimination algorithm of PGMPY, which is more efficient with respect to inference by enumeration. Bayesian Networks are very useful for making predictions, following the relationship between variables in the causal direction, as well as for diagnostic reasons, when we perceive as evidence some effect that we want to explain by determining its probable cause. Unfortunately, sometimes performing exact inference is not possible due to very large network with many nodes. approximate_inference addresses this issue by approximating the posterior probability distribution with a sampling method.

SALARY INFERENCE

As already described in the introduction, the first purpose of the project is to evaluate the salaries of the players. I've started by computing the salaries obtained by players with certain characteristics, and from there go back to the other ones with a pay different from the average. At this point, the disparities between these players and the ones with an adequate salary were analyzed, to understand if exists a valid reason for such an important difference or not. Let's see step by step the results obtained.

```
query1 = exact_inference.query(['SALARY'], evidence = {'PER':6,'+/-':5}, show_progress = False)
print(query1)
```

```
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.1000 |
+-----------+---------------+
| SALARY(2) |        0.1000 |
+-----------+---------------+
| SALARY(3) |        0.0000 |
+-----------+---------------+
| SALARY(4) |        0.0000 |
+-----------+---------------+
| SALARY(5) |        0.1000 |
+-----------+---------------+
| SALARY(6) |        0.2000 |
+-----------+---------------+
| SALARY(7) |        0.0000 |
+-----------+---------------+
| SALARY(8) |        0.5000 |
+-----------+---------------+
```

As first thing, I've considered the CPT of salary for players with `PER: 6` and `+/- : 5`. The table shows that 50% of the players with these characteristics have a pay greater than 30M of dollars (`SALARY(8)`), which is the higher range of the league. This result is expected since having such a high `PER` and `+/-` means being among the best athletes in the league. Nevertheless, there are also small probabilities not only to have a salary slightly lower but also among the lowest in the NBA. In the figure below are shown the characteristics of players with a salary in the range of 5, 6 and 8.

	TEAM	AGE	GP	MIN	+/-	PER	POS	HEIGHT	WEIGHT	YOS	DRAFT	COUNTRY	SALARY
81	3	2	4	7	5	6	4	4	4	2	1	1	6
93	4	3	4	7	5	6	1	2	1	3	1	2	8
229	1	3	3	7	5	6	2	3	3	4	1	2	8
257	3	3	3	7	5	6	3	4	3	3	1	2	8
264	6	2	4	6	5	6	5	5	5	3	1	1	6
324	1	2	3	7	5	6	1	2	1	3	1	2	8
339	6	1	4	7	5	6	3	4	3	1	1	1	5
470	5	3	4	7	5	6	1	3	1	4	1	2	8

It seems that there aren't strange patterns between the features, except for a trend in the country of these players whereby the foreign ones (`COUNTRY: 1`) tend to have a slightly lower salary. By looking at the figure below, this kind of discrimination is definitely discarded, so the three players above with `PER: 6` and `+/- : 5` and a salary lower than 8 could be considered underpaid.

```
Foreign players salaries probabilities:
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.2290 |
+-----------+---------------+
| SALARY(2) |        0.2614 |
+-----------+---------------+
| SALARY(3) |        0.1329 |
+-----------+---------------+
| SALARY(4) |        0.0616 |
+-----------+---------------+
| SALARY(5) |        0.0574 |
+-----------+---------------+
| SALARY(6) |        0.1509 |
+-----------+---------------+
| SALARY(7) |        0.0558 |
+-----------+---------------+
| SALARY(8) |        0.0510 |
+-----------+---------------+
```

```
US players salaries probabilities:
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.2305 |
+-----------+---------------+
| SALARY(2) |        0.2621 |
+-----------+---------------+
| SALARY(3) |        0.1329 |
+-----------+---------------+
| SALARY(4) |        0.0614 |
+-----------+---------------+
| SALARY(5) |        0.0573 |
+-----------+---------------+
| SALARY(6) |        0.1505 |
+-----------+---------------+
| SALARY(7) |        0.0552 |
+-----------+---------------+
| SALARY(8) |        0.0500 |
+-----------+---------------+
```

The next step has been to compare the characteristics of players with `SALARY: 1`, `SALARY: 2` and `SALARY: 8`, as follows:

	TEAM	AGE	GP	MIN	+/-	PER	POS	HEIGHT	WEIGHT	YOS	DRAFT	COUNTRY	SALARY
93	4	3	4	7	5	6	1	2	1	3	1	2	8
138	3	3	1	3	5	6	4	5	4	3	3	2	1
229	1	3	3	7	5	6	2	3	3	4	1	2	8
257	3	3	3	7	5	6	3	4	3	3	1	2	8
324	1	2	3	7	5	6	1	2	1	3	1	2	8
470	5	3	4	7	5	6	1	3	1	4	1	2	8
520	3	4	1	1	5	6	5	4	3	4	3	2	2

In this case, the two players with such a low salary have simply played a few minutes in a few games of this season getting an high rating, so no problem arises.

In order to enforce the concept, I've exploited the function `map_query` of `pgmpy.inference.VariableElimination` which makes an explanation about the most likely salary given some value's nodes as evidence:

```
query4 = exact_inference.map_query(['SALARY'], evidence = {'PER':6, '+/-': 5},show_progress=False)
print(f"The most likely salary range for a player with PER = 6 and +/- = 5 is: {query4}")
```

```
The most likely salary range for a player with PER = 6 and +/- = 5 is: {'SALARY': 8}
```

We can clearly say that these three players discovered have definitely a lower salary with respect to similar players without a concrete reason. By looking into the dataset, the names associated to the their indices are respectively: `Clint Capela`, `Jonas Valanciunas` and `Luka Doncic`.

The most natural next step has been to invert the inference and search for players with a low `PER` and `+/-`.

```
query5 = exact_inference.query(['SALARY'], evidence = {'PER':2, '+/-':2}, show_progress = False)
print(query5)
```

```
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.3600 |
+-----------+---------------+
| SALARY(2) |        0.2800 |
+-----------+---------------+
| SALARY(3) |        0.1200 |
+-----------+---------------+
| SALARY(4) |        0.0800 |
+-----------+---------------+
| SALARY(5) |        0.0400 |
+-----------+---------------+
| SALARY(6) |        0.1200 |
+-----------+---------------+
| SALARY(7) |        0.0000 |
+-----------+---------------+
| SALARY(8) |        0.0000 |
+-----------+---------------+
```

Once again, I've considered the characteristics of players with `SALARY: 5` and `SALARY: 6` and with `PER: 2` and `+/- : 2`, so the ones who are potentially overpaid.

	TEAM	AGE	GP	MIN	+/-	PER	POS	HEIGHT	WEIGHT	YOS	DRAFT	COUNTRY	SALARY
11	2	3	2	4	2	2	4	4	3	4	1	2	6
287	6	2	2	5	2	2	3	3	3	2	1	2	6
410	2	4	3	6	2	2	3	3	4	3	2	2	5
444	1	2	3	4	2	2	4	4	2	2	1	2	6

The most likely salary range for players with these characteristics is instead shown below:

```
query6 = exact_inference.map_query(['SALARY'], evidence = {'PER':2, '+/-': 2},show_progress=False)
print(f"The most likely salary range for a player with PER = 2 and +/- = 2 is: {query6}")
```

```
The most likely salary range for a player with PER = 2 and +/- = 2 is: {'SALARY': 1}
```

Thus, from the table above we can see that there are two players with `GP: 2`, so their salaries may be adequate because they simply played few games in the season and were underwhelmed, perhaps because of returning from an injury. It is a different matter for the other two players with `GP: 3`. In this case, they played more than half of the games of the season and despite this, they obtained very low `PER` and `+/-` values. If in the next seasons these players were not able to do better, their salaries should be greatly reduced, as explained by `query6`. To conclude, the players discovered to be overpaid are respectively: `P.J. Tucker` and `Rodney Hood`.

The function `map_query` of PGMPY is just resulted very helpful in confirming the results, once the network has found players with an inadequate pay. Nevertheless, this is not the only possibility of use for this task. In the next few steps I will show how this

function could be used to infer directly the most appropriate salary for a particular player. For instance, I've considered the salary of three players and their stats in the 2020-2021 season, as follows:

	PLAYER	TEAM	AGE	GP	W	L	MIN	PTS	FGM	FGA	...	+/-	PER	POS	HEIGHT	WEIGHT	YOS	COLLEGE	DRAFT	COUNTRY	SALARY
9	Alex Caruso	LAL	27	58	32	26	21.0	6.4	2.3	5.3	...	2.1	11.13	G	6-4	186	3	Texas A&M	2016 NBA Draft, Undrafted	United States	$2,750,000
184	Giannis Antetokounmpo	MIL	26	61	40	21	33.0	28.1	10.3	18.0	...	6.7	29.24	F	6-11	242	7	Other	2013 Rnd 1 Pick 15	Greece	$27,528,088
399	Nikola Jokic	DEN	26	72	47	25	34.6	26.4	10.2	18.0	...	5.3	31.36	C	7-0	284	5	Other	2014 Rnd 2 Pick 11	Serbia	$29,542,010

As we can see from the figure above the salaries seem really low although they are among the best players in the league, so I've predicted the adequate ones for each of them.

The first one is `Alex Caruso`, who has the following characteristics after the preprocessing step over the dataset:

```
   TEAM AGE GP MIN +/- PER POS HEIGHT WEIGHT YOS DRAFT COUNTRY SALARY
9     5   2  3   5   5   3   1      3      1   1     3       2      2
```

Thanks to the `map_query` function, I've found that he is extremely underpaid, as shown in the figure below:

```
query7 = exact_inference.map_query(['SALARY'], evidence = {'PER':3, '+/-': 5},show_progress=False)
print(f"Correct range of salary for Alex Caruso is: {query7}")
```

```
Correct range of salary for Alex Caruso is: {'SALARY': 6}
```

Then I've considered respectively the MVP of the last two seasons `Giannis Antetokounmpo` and the one of this season `Nikola Jokic`. They have the following stats:

```
Giannis Antetokounmpo:
     TEAM AGE GP MIN +/- PER POS HEIGHT WEIGHT YOS DRAFT COUNTRY SALARY
184     2   2  4   7   6   6   4      5      4   3     1       1      7
Nikola Jokic:
     TEAM AGE GP MIN +/- PER POS HEIGHT WEIGHT YOS DRAFT COUNTRY SALARY
399     4   2  4   7   6   6   5      5      5   2     2       1      8
```

Although they have the same values of `PER` and `+/-` and Antetokounmpo won the MVP title for two seasons, he has a salary lower than Jokic.

```
query8 = exact_inference.map_query(['SALARY'], evidence = {'PER':6, '+/-': 6},show_progress=False)
print(f"The appropriate salary range for Giannis Antetokounmpo and Nikola Jokic is: {query8}")
```

```
The appropriate salary range for Giannis Antetokounmpo and Nikola Jokic is: {'SALARY': 8}
```

Once again, the `map_query` allows to find an anomaly in the salary of a player.

STATS INFERENCE

The second purpose of the project is instead more statistical and concerns an analysis of the NBA players' characteristics in general. Predictions were made on the career expectancy in the league, on the probability of injuries with age, and finally, a brief

evaluation was conducted about players' salaries concerning their role and country of origin.

The first inference that I've drawn is about how the round in which a player was selected at the draft still affects his performance years later (more than 10).

```
print("Draft round odds for players with PER = 1 and YOS = 4:")
query9 = exact_inference.query(['DRAFT'], evidence = {'PER':1, 'YOS': 4}, show_progress = False)
print(query9)
print("Draft round odds for players with PER = 5 and YOS = 4:")
query10 = exact_inference.query(['DRAFT'], evidence = {'PER':5, 'YOS': 4}, show_progress = False)
print(query10)
```

```
Draft round odds for players with PER = 1 and YOS = 4:
+----------+--------------+
| DRAFT    |   phi(DRAFT) |
+==========+==============+
| DRAFT(1) |       0.1697 |
+----------+--------------+
| DRAFT(2) |       0.1357 |
+----------+--------------+
| DRAFT(3) |       0.6946 |
+----------+--------------+
Draft round odds for players with PER = 5 and YOS = 4:
+----------+--------------+
| DRAFT    |   phi(DRAFT) |
+==========+==============+
| DRAFT(1) |       0.7118 |
+----------+--------------+
| DRAFT(2) |       0.1370 |
+----------+--------------+
| DRAFT(3) |       0.1512 |
+----------+--------------+
```

From the example above we can see that a player with a very low `PER` value who played for more than 10 years, at 69% has not been drafted. On the other hand, there is a 71% of probability that a player with the same experience and a very high `PER` value was selected in the first round of the draft, and an incredibly little probability that he was selected in the second round. This reasoning which may seem trivial is instead vitally important for teams at draft time. It shows in fact how it is very difficult for players who were not among the first picks to improve and still have good seasons many years later.

The next step has been to investigate on how NBA salaries change with respect to the role of a player.

```
print("Annual salary for players with POS = 1:")
query11 = exact_inference.query(['SALARY'], evidence = {'POS':1}, show_progress = False)
print(query11)
print("Annual salary for players with POS = 2:")
query12 = exact_inference.query(['SALARY'], evidence = {'POS':2}, show_progress = False)
print(query12)
print("Annual salary for players with POS = 3:")
query13 = exact_inference.query(['SALARY'], evidence = {'POS':3}, show_progress = False)
print(query13)
print("Annual salary for players with POS = 4:")
query14 = exact_inference.query(['SALARY'], evidence = {'POS':4}, show_progress = False)
print(query14)
print("Annual salary for players with POS = 5:")
query15 = exact_inference.query(['SALARY'], evidence = {'POS':5}, show_progress = False)
print(query15)
```

```
Annual salary for players with POS = 1:
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.2463 |
+-----------+---------------+
| SALARY(2) |        0.2775 |
+-----------+---------------+
| SALARY(3) |        0.1324 |
+-----------+---------------+
| SALARY(4) |        0.0554 |
+-----------+---------------+
| SALARY(5) |        0.0544 |
+-----------+---------------+
| SALARY(6) |        0.1508 |
+-----------+---------------+
| SALARY(7) |        0.0433 |
+-----------+---------------+
| SALARY(8) |        0.0398 |
+-----------+---------------+
Annual salary for players with POS = 2:
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.2313 |
+-----------+---------------+
| SALARY(2) |        0.2828 |
+-----------+---------------+
| SALARY(3) |        0.1366 |
+-----------+---------------+
| SALARY(4) |        0.0581 |
+-----------+---------------+
| SALARY(5) |        0.0535 |
+-----------+---------------+
| SALARY(6) |        0.1549 |
+-----------+---------------+
| SALARY(7) |        0.0449 |
+-----------+---------------+
| SALARY(8) |        0.0379 |
+-----------+---------------+
Annual salary for players with POS = 3:
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.2431 |
+-----------+---------------+
| SALARY(2) |        0.2853 |
+-----------+---------------+
| SALARY(3) |        0.1324 |
+-----------+---------------+
| SALARY(4) |        0.0564 |
+-----------+---------------+
| SALARY(5) |        0.0578 |
+-----------+---------------+
| SALARY(6) |        0.1446 |
+-----------+---------------+
| SALARY(7) |        0.0428 |
+-----------+---------------+
| SALARY(8) |        0.0376 |
+-----------+---------------+
Annual salary for players with POS = 4:
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.2176 |
+-----------+---------------+
| SALARY(2) |        0.2608 |
+-----------+---------------+
| SALARY(3) |        0.1352 |
+-----------+---------------+
| SALARY(4) |        0.0628 |
+-----------+---------------+
| SALARY(5) |        0.0588 |
+-----------+---------------+
| SALARY(6) |        0.1538 |
+-----------+---------------+
| SALARY(7) |        0.0588 |
+-----------+---------------+
| SALARY(8) |        0.0522 |
+-----------+---------------+
Annual salary for players with POS = 5:
+-----------+---------------+
| SALARY    |   phi(SALARY) |
+===========+===============+
| SALARY(1) |        0.2069 |
+-----------+---------------+
| SALARY(2) |        0.1891 |
+-----------+---------------+
| SALARY(3) |        0.1274 |
+-----------+---------------+
| SALARY(4) |        0.0783 |
+-----------+---------------+
| SALARY(5) |        0.0627 |
+-----------+---------------+
| SALARY(6) |        0.1490 |
+-----------+---------------+
| SALARY(7) |        0.0950 |
+-----------+---------------+
| SALARY(8) |        0.0916 |
+-----------+---------------+
```

It seems reasonable to say that players who play as Center (`POS: 5`) are among the most coveted in the league, having on average higher salaries than others. Then follow the Power Forward (`POS: 4`), while the players who play as 3 are the ones with the lowest salaries on average, probably due to the overabundance of players who can play in this role.

Another simple inference that could be made is about how injuries affect players over the years, as follows:

```
print("AGE 0-24:")
query16 = exact_inference.query(['GP'], evidence = {'AGE':1}, show_progress = False)
print(query16)
print("AGE 25-29:")
query17 = exact_inference.query(['GP'], evidence = {'AGE':2}, show_progress = False)
print(query17)
print("AGE 30-34:")
query18 = exact_inference.query(['GP'], evidence = {'AGE':3}, show_progress = False)
print(query18)
print("AGE 35-INF:")
query19 = exact_inference.query(['GP'], evidence = {'AGE':4}, show_progress = False)
print(query19)
```

AGE 0-24:

GP	phi(GP)
GP(1)	0.2294
GP(2)	0.2511
GP(3)	0.2727
GP(4)	0.2468

AGE 25-29:

GP	phi(GP)
GP(1)	0.1741
GP(2)	0.1940
GP(3)	0.3383
GP(4)	0.2935

AGE 30-34:

GP	phi(GP)
GP(1)	0.1047
GP(2)	0.1512
GP(3)	0.3953
GP(4)	0.3488

AGE 35-INF:

GP	phi(GP)
GP(1)	0.1818
GP(2)	0.1818
GP(3)	0.4545
GP(4)	0.1818

Younger players are the ones less prone to injuries, and it is possible to notice this by seeing how the probability is perfectly distributed among the intervals. Nevertheless, the lack of experience does not allow some of them to play all the games of the season. As age increases so do the likelihood of playing more, but the latest CPT shows that 'older' players are very unlikely to play all the games in a season. Since it was predictable, I decided to investigate further and see if the salary offered to a player takes this result into account.

```
print("Young players with lack of experience:")
query20 = exact_inference.query(['SALARY'], evidence = {'AGE': 1, 'YOS': 1}, show_progress = False)
print(query20)
print("Senior players with more than 10 years of NBA experience:")
query21 = exact_inference.query(['SALARY'], evidence = {'AGE': 4, 'YOS': 3}, show_progress = False)
print(query21)
```

Young players with lack of experience:

SALARY	phi(SALARY)
SALARY(1)	0.2451
SALARY(2)	0.2640
SALARY(3)	0.1332
SALARY(4)	0.0612
SALARY(5)	0.0555
SALARY(6)	0.1434
SALARY(7)	0.0523
SALARY(8)	0.0454

Senior players with more than 10 years of NBA experience:

SALARY	phi(SALARY)
SALARY(1)	0.2006
SALARY(2)	0.2576
SALARY(3)	0.1339
SALARY(4)	0.0636
SALARY(5)	0.0608
SALARY(6)	0.1606
SALARY(7)	0.0632
SALARY(8)	0.0597

Although older players struggle to play most of the games of a season, the experience gained over the years allows them to get a salary not only similar to that of younger ones but even slightly better.

To conclude the notebook I've made a comparison between the `approximate_inference` and the `exact_inference` techniques. In particular, some of the queries evaluated previously were considered, to understand whether approximate inference with the 'Weighted Likelihood' sampling method could be a good replacement when a bigger Bayesian Network is considered. I created a loop with an increasing number of samples and at each iteration I queried the network to get the probabilities of some of the inferences analyzed previously.

```
################## SAMPLE SIZE: 100 ##################

Query: SALARY - Evidence: {'PER': 6, '+/-': 5}
Exact probability: 0.1
Approximate probability: 0.24

Query: SALARY - Evidence: {'COUNTRY': 2}
Exact probability: 0.26
Approximate probability: 0.27

Query: GP - Evidence: {'AGE': 3}
Exact probability: 0.15
Approximate probability: 0.14

Query: SALARY - Evidence: {'AGE': 4, 'YOS': 3}
Exact probability: 0.26
Approximate probability: 0.3


################## SAMPLE SIZE: 100000 ##################

Query: SALARY - Evidence: {'PER': 6, '+/-': 5}
Exact probability: 0.1
Approximate probability: 0.08

Query: SALARY - Evidence: {'COUNTRY': 2}
Exact probability: 0.26
Approximate probability: 0.25

Query: GP - Evidence: {'AGE': 3}
Exact probability: 0.15
Approximate probability: 0.2

Query: SALARY - Evidence: {'AGE': 4, 'YOS': 3}
Exact probability: 0.26
Approximate probability: 0.28
```

In conclusion, the results show that, as the sampling size increases the approximate inference is able to correctly approximate the queries with a minimum deviation, although for some of them a size of 100.000 is unfortunately still not large enough.