A Bayesian network for heart attack prevention

Agostino Aiezzo - 982514

Introduction

Every year, in Italy, hundreds of thousands of people suffer from heart attacks. Most of the patients are saved, but still one third of them die. The possibility to reduce the occurence of this kind of event exists and involves the use of probability. The aim of this project is, in fact, to build a probabilistic model which tries to understand the relationships among different causes for heart attack cases and which of them are the most relevant. For this reason, the dataset provided by UCI and created by different medical and hospital research institutes is used. This study is intended for research purposes only, thus all rights are reserved to the legitimate owners. The dataset provides a lot of attributes, but all the experiments tend to use a subset of 14 of them. Here they are below:

Attribute	**Description**	**Possible values**
age	Age of the patient	Integers from 0 to 100
sex	Sex of the patient	0 = Female 1 = Male
cp	Chest pain	- typical angina - atypical angina - non-anginal pain - asymptomatic
trestbps	Resting blood pressure	Integers
chol	Serum cholesterol in mg/dl	Integers
fbs	Fasting blood sugar higher than 120 mg/dl	0 = False 1= True
restecg	Resting electrocardiographic results	- normal - ST-T wave abnormality - probable or definite left ventricular hypertrophy by Estes's criteria.
thalach	Maximum heart rate achieved	Integers
exang	Angina induced after exercises	0 = No 1 = Yes
oldpeak	ST depression induced by exercise relative to rest	Small real numbers

slope	The slope of the peak exercise ST segment	- up sloping - flat - downsloping
ca	Number of major vessels colored by fluoroscopy	Integers from 0 to 3
thal	Thalassemia	- normal - fixed defect - reversible defect
target	Probability of a heart attack	0 = No 1 = Yes

Fortunately, there are no missing values. Nevertheless, some data processing needs to be done in order to obtain many advantages during model construction and management.

Data bucketization

The main reason for data to be elaborated is to obtain a more compact and comprehensive model. In this way, in fact, efficiency and low computational effort are achieved. So once all the libraries needed are imported (mainly pandas, numpy and pgmpy), it's possible to have a look to the dataset:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1

Observing the data, there are attributes which are characterized by a multitude of values and this is a problem since in such a way it would be difficult to manage the model because of its ize (too big). As a consequence, what is necessary to do is to reduce the number of possible values by recasting the attributes of interest in specific intervals identified by names. This operation is known as bucketization and the parameters that need to be considered are, for example, ‘age’, ‘trestbps’, ‘chol’, ‘oldpeak’, etc. For safety, a copy of the data will be used to build a new dataset that will be adopted to design the probabilistic model.

For instance, for the ‘age’ column, 5 possible subdivisions are considered. Even if not all of them have a match, it is worth it to insert them for future extensions of the dataset.

```
df_copy['age'] = pd.cut(x=df_copy['age'],
                        bins=[0,10,18,30,65,120],
                        labels=['Baby','Minor','Young','Adult','Old'])
df_copy['age'].unique()
```

Something similar is performed for the ‘trestbps’ column.

```
df_copy['trestbps'] = pd.cut(x=df_copy['trestbps'],
                         bins=[0,50,90,130,250,500],
                         labels=['Very Low','Low','Medium','High','Very High'])
df_copy['trestbps'].unique()
```

For the sake of realism, the 'chol' column has been bucketized by using only 2 possible values. In particular, 'True' is the case in which the blood cholesterol is above 200 mg/dl; 'False' on the contrary case.

```
df_copy['chol'] = np.where((df_copy['chol'] > 200), True, #se supera i 200
                           np.where((df_copy['chol'] <= 200), False, 'No Change'))
df_copy['chol'].unique()
```

Initially, the 'thalach' column, as well as the 'chol' one, was bucketized with 'True' and 'False' values. However, in this way some relations in the Bayesian network were not expressed probably because many shades of information were lost in the process. So, for this reason the number of possible categories were expanded and thus new links among attributes have been generated.

```
df_copy['thalach'] = pd.cut(x=df_copy['thalach'],
                            bins=[0,90,130,500],
                            labels=['Low','Medium','High'])
df_copy['thalach'].unique()
```

In conclusion, also the 'oldpeak' column has been bucketized as usual. The only drawback encountered, in this case, was about some missing values, but they have been simply solved by filling them with the correspondent category.

```
df_copy['oldpeak'] = pd.cut(x=df_copy['oldpeak'],
                        bins=[0.,1.,3.,10.],
                        labels=['Small','Mid','Big'])
df_copy['oldpeak'] = df_copy['oldpeak'].fillna('Small')
df_copy['oldpeak'].unique()
```

Here below there's the final result. As it is noticeable, the dataset is not only ready to be used to build the probabilistic model, but it is also more human-readable and clean.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	Adult	1	3	High	True	1	0	High	0	Mid	0	0	1	1
1	Adult	1	2	Medium	True	0	1	High	0	Big	0	0	2	1
2	Adult	0	1	Medium	True	0	0	High	0	Mid	2	0	2	1
3	Adult	1	1	Medium	True	0	1	High	0	Small	2	0	2	1
4	Adult	0	0	Medium	True	0	1	High	1	Small	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	Adult	0	0	High	True	0	1	Medium	1	Small	1	0	3	0
299	Adult	1	3	Medium	True	0	1	High	0	Mid	1	0	3	0
300	Old	1	0	High	False	1	1	High	0	Big	1	2	3	0
301	Adult	1	0	Medium	False	0	1	Medium	1	Mid	1	1	3	0
302	Adult	0	1	Medium	True	0	0	High	0	Small	1	1	2	0

Building the Bayesian Network

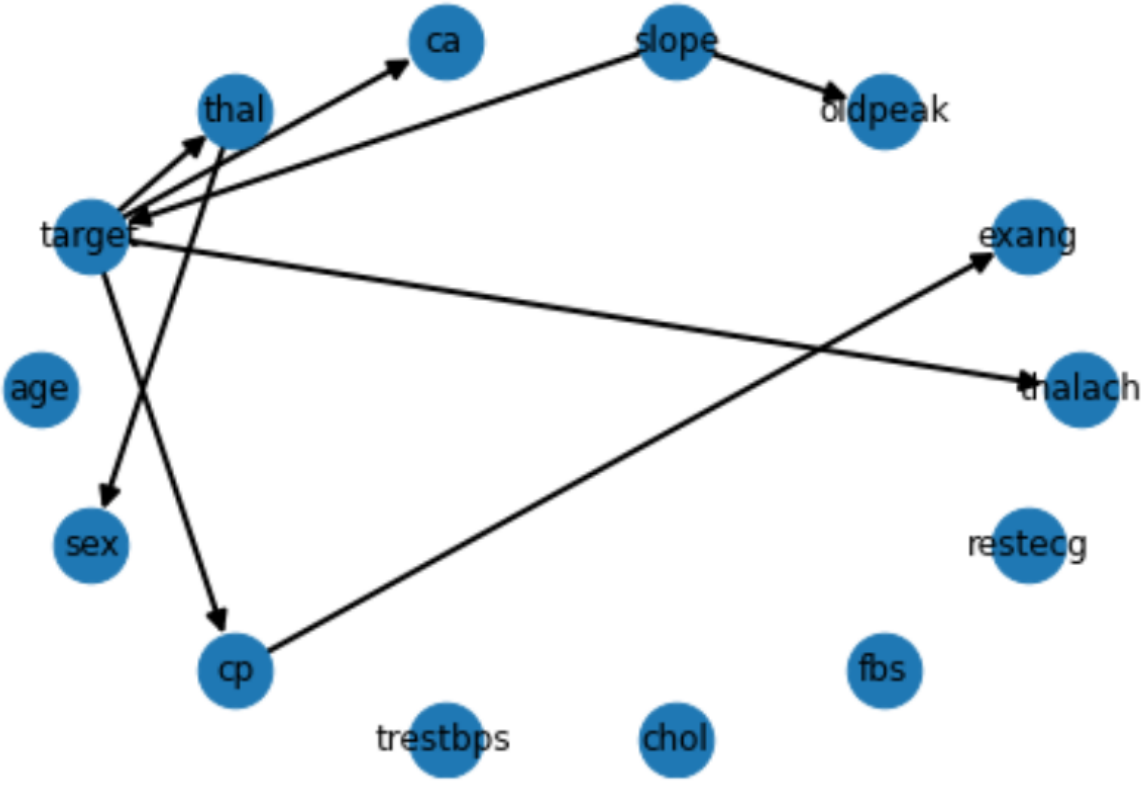
After the elaboration of the data, it's possible to build a first instance of the Bayesian network. In fact, many attempts need to be done before obtaining the best model. In particular, some considerations should be taken into account in order to have an efficient model.

To build the Bayesian network, first of all, a scoring method must be fixed. In this way, it's possible to understand how well a model is able to describe the given data set. The pgmpy library provides the 'BicScore' class. The BIC/MDL ("Bayesian Information Criterion", also "Minimal Descriptive Length") is a Log-Likelihood score. This is the one used below because it is able to avoid overfitting with an additional penalty for network complexity, unlike other classes like 'K2Score' and

'BDeuScore'. In fact, these latter seem to not care about overfitting and as a consequence the obtained graphs are quite different from the one provided by the 'BicScore' object (they have more connections).

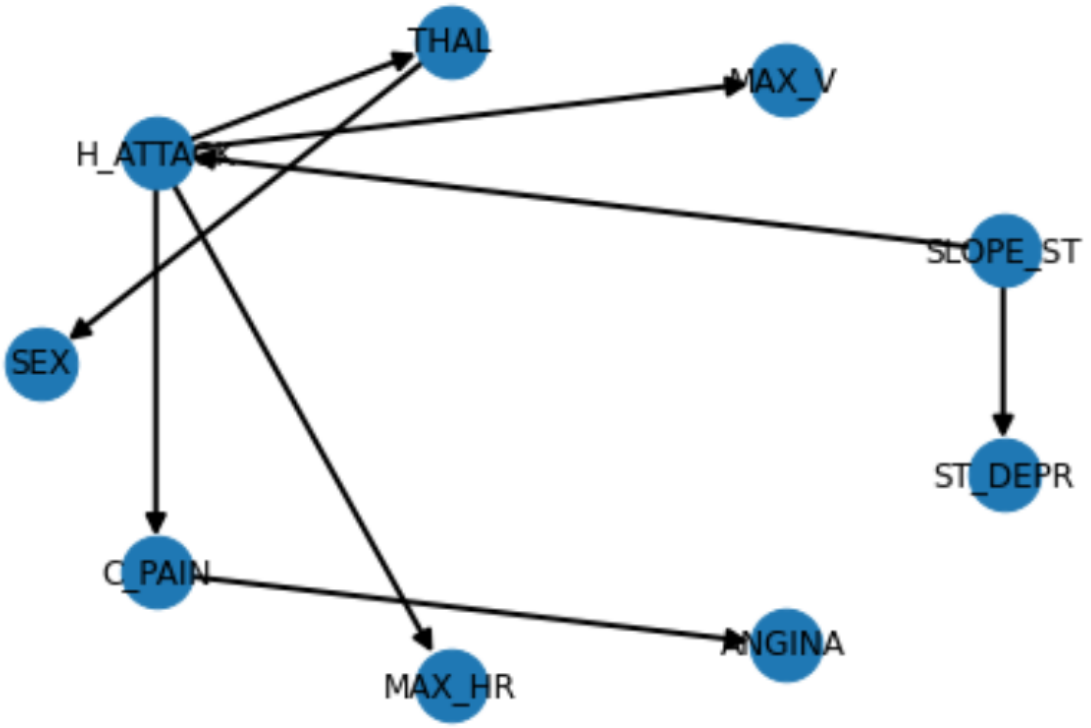
For what concerns the algorithm used for searching the best graph (always according to the score method), the 'HillClimbSearch' class is considered. In particular, it performs local hill climb search to estimate the DAG structure that has an optimal score, according to the scoring method supplied. So, it starts at an initial model *start_dag* and proceeds by step-by-step network modifications until a local maximum is reached. Also in this case, other possibilities exist like for example 'PC' or 'ExhaustiveSearch'. However, this latter, for example, is not feasible if there are more than 6 nodes (attributes), because it generates $2^{(n*(n-1))}$ graphs.

Once all this stuff has been set, it's possible to obtain a first instance of the BN:



The above first model allows us to deduce something important: there are some attributes which seem to not influence the network. It's possible to confirm this possibility if the network with the not bucketed parameters is computed. In fact, the two graphs turn out to be identical. So, as a consequence, it's possible to discard the attributes with no connections in order to have a leaner and, hypothetically, a more

efficient graph. Of course, the same scoring method and search method as before are used.



After some cuts and renaming, this is the final result. Just to be clear, the direction of the relations are considered as something like ' A is influenced by B ' and not like 'A influences B' as in fact it normally is with Bayesian networks.

So, the last step now is to compute the CPDs that are the probabilities for each of the attributes involved in the network. To do that, of course, each instance of the dataset and a 'BayesianEstimator' object are taken into account. Another possibility was to adopt a 'MaximumLikelihoodEstimator' but, after some attempts, it is less comfortable to use.

Structural analysis and inferring

Once the CPDs are available, it's straightforward to start to deduce and discover interesting patterns. In particular, it's possible to conduct a structural analysis of the network. For example, it is interesting to know about conditional independencies given certain evidence.

```
[(MAX_V ⊥ ST_DEPR, ANGINA | MAX_HR, SLOPE_ST, C_PAIN),
 (MAX_V ⊥ ANGINA | SLOPE_ST, ST_DEPR, C_PAIN),
 (MAX_V ⊥ ST_DEPR | SLOPE_ST, ANGINA, C_PAIN),
 (MAX_V ⊥ ST_DEPR | SEX, SLOPE_ST, THAL),
 (MAX_V ⊥ SEX | MAX_HR, ST_DEPR, THAL),
 (MAX_V ⊥ SEX | MAX_HR, ANGINA, THAL),
 (MAX_V ⊥ SEX, ST_DEPR | MAX_HR, SLOPE_ST, THAL),
 (MAX_V ⊥ SEX | ST_DEPR, ANGINA, THAL),
 (MAX_V ⊥ SEX | SLOPE_ST, ST_DEPR, THAL),
 (MAX_V ⊥ SEX, ST_DEPR | SLOPE_ST, ANGINA, THAL)]
```

Additionally, what is possible to do is to compute some exact inference (the exact value of probability P(X|Y)) through the 'VariableElimination' method which stores intermediate results to avoid recomputation. Trivially, it's possible to know, for example, who has the highest probability, between a man and a woman, to get a heart attack.

```
Probability to get an heart attack if I am a male
 +-------------+-----------------+
| H_ATTACK    |   phi(H_ATTACK) |
+=============+=================+
| H_ATTACK(0) |          0.5219 |
+-------------+-----------------+
| H_ATTACK(1) |          0.4781 |
+-------------+-----------------+
Probability to get an heart attack if I am a female
 +-------------+-----------------+
| H_ATTACK    |   phi(H_ATTACK) |
+=============+=================+
| H_ATTACK(0) |          0.3163 |
+-------------+-----------------+
| H_ATTACK(1) |          0.6837 |
+-------------+-----------------+
```

It turns out that females have higher chances: 68.7% against 47.8% for males.

Another interesting example may be to understand how much an angina (which is a reduction in blood flow to the heart) after a physical exercise can increase (or decrease) the probability of getting a heart attack.

```
Probability to get an heart attack caused by angina after an exercise
 +-----------+---------------+
| ANGINA    |   phi(ANGINA) |
+===========+===============+
| ANGINA(0) |        0.7717 |
+-----------+---------------+
| ANGINA(1) |        0.2283 |
+-----------+---------------+
```

It looks like that angina is not sufficient to define the occurrence of a heart attack (23%). What about if fixed defects for thalassemia and an atypical chest pain are considered? What is the probability for a heart attack in such a case?

```
Probability to get an heart attack if I have fixed thalassemia and and atypical chest pain
 +------------+-----------------+
| H_ATTACK   |   phi(H_ATTACK) |
+============+=================+
| H_ATTACK(0) |         0.3444 |
+------------+-----------------+
| H_ATTACK(1) |         0.6556 |
+------------+-----------------+
```

The probability of getting into trouble is high (65%).

In some cases, the computed network is too big to estimate exact inferences because the computational effort is too heavy. So, in alternative, it's possible to compute an approximation of these probabilities by using techniques that generate random samples and thus the probability of the query (that is actually how probability is calculated in real world experiment).

In particular, the Rejection Sampling algorithm is implemented here since it is one the simplest: it generates only those samples that have certain evidence and discard the others. However, likelihood weighting variant is preferable since it is smarter: it assigns a weight to each generated sample.

```
Approximate probability of getting an heart attack with fixed thalassemia and atypical chest pain:

0.6950127754658162
```

This is the computed result which is pretty much the same as before.

Conclusions

Surely, what is possible to deduce is that Bayesian networks are powerful tools in medical contexts because they are simple to use and allow to infer interesting insights, by combining different attributes, in an environment which is characterized by many factors and variations. Of course, the model built in this project is not reliable since many approximations and inaccuracies were committed, but the potential can be expressed with the help of an expert. In addition, other kinds of structural analysis might be thoughtful like, for example, active trails, Markov blankets, and so on.