TMBD Project 2

Students (G8) :

- Beatriz Cruz **(85578)**
- Luís Sousa **(89824)**
- Márcia Silva **(88089)**
- Roshan Poudel **(109806)**

Work Description

- Characterise the chosen Dataset
- Explain what you intend to achieve
- Describe the model used (FIS or BN) and why you think it is an acceptable representation of the problem
- Describe in markdown a particular example:
- FIS: the calculation of one rule
- BN: the calculation of one speci c probability
- Add the code with some relevant comments
- Discuss the results obtained

Bayesian Networks

Bayesian Networks are probabilistic models that define relationships between variables and can be used to calculate conditional probabilities using Bayesian inference. They consist of a directed acyclical graph in which the nodes represent the variables of our model and the edges display the relationship (or conditional dependence) of the variables. Through these relationships, one can efficiently conduct inference on the random variables in the graph through the use of factors. For example, if an edge (A, B) exists in the graph connecting random variables A and B, it means that $P(B|A)$ is a factor in the joint probability distribution, so we must know $P(B|A)$ for all values of B and A in order to conduct inference. Bayesian networks satisfy the local Markov property, which states that a node is conditionally independent of its non-descendants given its parents. This property allows us to simplify the joint distribution to a smaller form, making it also easier to infere some probabilities, since now we don't have to condition each variable to all the other previous variables, but only to its "parents". Thus, our joint distribution is calculated like this:

$$P(X_1,\dots,X_n) = \prod_{i=1}^{n} P(X_i|X_1,\dots X_{i-1}) = \prod_{i=1}^{n} P(X_i|Parents(X_i))$$

This simplification also helps reduce the amount of required computation, since most nodes will have few parents relative to the overall size of the network.

There are two types of inference we can perform using Bayesian Networks. The first one consists of evaluating the joint probability of a specific assignment of values for each variable. Since we already have the factorized form of our joint distribution function, we simply evaluate the provided conditional probabilities. If we intended to study one subset of variables, we'd have to marginalize the variables that were not inclued in that subset.

The second one consists of calculating a certain probability $P(X|Y)$ for an assignment of values of a subset of variables X given another subset of variables Y. In order to calculate this, we use the fact that $P(X|Y) = \frac{P(X,Y)}{P(Y)} = \alpha P(X,Y)$, where α is a normalization constant that we will calculate at the end such that $P(X|Y) + P(\neg X|Y) = 1$. In order to calculate $P(X,Y)$, we must marginalize the joint probability distribution over the variables that do not appear in X or Y. This is usually the inference we're most interested in performing, and it's the one we'll perform in our dataset.

In [36]:
```python
import numpy as np
import pandas as pd
import seaborn
from matplotlib import pyplot as plt

import numpy as np
from pomegranate import *
import networkx as nx
from pomegranate.utils import plot_networkx
import time
```

Dataset

The chosen dataset aims to predict whether a patient is likely to get a stroke based on the input parameters like gender, age, various diseases, and smoking status. The data contains 5110 observations with 12 attributes. Each row in the data provides relevant information about the patient. The attribute information is shown as following:

1) **id :** unique identifier. 2) **gender :** "male", "female" or "other. 3) **age :** age of each patient. 4) **hypertension :** 1 if a patient has hypertension, 0 if not. 5) **heart_disease :** 1 if a patient has heart disease, 0 if not. 6) **ever_married :** "yes" or "no". 7) **work_type :** "children", "Govt_jov", "Never_worked", "Private" or "Self-employed". 8) **Residence_type :** "Rural" or "Urban". 9) **avg_glucose_level :** average glucose level in the blood. 10) **bmi :** body mass index. 11) **smoking_status :** "formerly smoked", "never smoked", "smokes" or "unknown". This "unknown" means this type of information is unavailable for the referred patient. 12) **stroke :** 1 if a patient had a stroke, 0 if not.

Note : The dataset was downloaded from Kaggle link to dataset.

In [37]:
```python
# Reading the dataset
df = pd.read_csv("data.csv")
```

In [38]:
```python
df
```

Out[38]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_gluc
0	9046	Male	67.0	0	1	Yes	Private	Urban	
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	
2	31112	Male	80.0	0	1	Yes	Private	Rural	
3	60182	Female	49.0	0	0	Yes	Private	Urban	
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	
...	...	...	...	...	...	...	...	...	...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	
5108	37544	Male	51.0	0	0	Yes	Private	Rural	
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	

5110 rows × 12 columns

In [39]:
```python
print(f"Columns : {list(df.columns.values)}")
```
```
Columns : ['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married', 'wo
rk_type', 'Residence_type', 'avg_glucose_level', 'bmi', 'smoking_status', 'stroke']
```

Our aim

In our case, our goal is to infer whether a person is likely to have a stroke, given certain circumstances. This can easily be achieved by creating a Bayesian Network, in which the bottom and final node is the variable "stroke", and our top and middle nodes are the characterstics and risk factors of a certain person. We're interested in calculating $P(X|Y)$, where X stands for the variable "stroke" and Y is the subset of all of the other variables included in our model.

Dropping **id**

- As the id parameter is only an identifier and has no real impact on a person having a stroke, we choose to drop that column.

In [40]:
```python
df.drop(columns=["id"], inplace=True)
```

Basic Dataset Exploration

In [41]:
```python
df.nunique()
```

Out[41]:
```
gender                  3
age                   104
hypertension            2
heart_disease           2
ever_married            2
work_type               5
Residence_type          2
avg_glucose_level    3979
bmi                   418
smoking_status          4
stroke                  2
dtype: int64
```

In [42]:
```python
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             5110 non-null   object
 1   age                5110 non-null   float64
 2   hypertension       5110 non-null   int64
 3   heart_disease      5110 non-null   int64
 4   ever_married       5110 non-null   object
 5   work_type          5110 non-null   object
 6   Residence_type     5110 non-null   object
 7   avg_glucose_level  5110 non-null   float64
 8   bmi                4909 non-null   float64
 9   smoking_status     5110 non-null   object
 10  stroke             5110 non-null   int64
dtypes: float64(3), int64(3), object(5)
memory usage: 439.3+ KB
```

In [43]:
```python
df.describe()
```

Out[43]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

Checking for NaNs

In [44]:
```python
df.isna().sum()
```

Out[44]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64
```

In [45]:
```python
df.dropna(inplace=True)
print(f"NaNs left : {df.isna().sum().sum()}")
```
```
NaNs left : 0
```

- **The unknown value in smoking_status is also like a null value so, we'll be dropping those as well.**
  This unknown value would make the probabilities harder to interpret.

In [46]:
```python
df = df[df['smoking_status'] != 'Unknown']
df.reset_index(drop=True, inplace=True)
```

In [47]:
```python
df.head()
```

Out[47]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level
0	Male	67.0	0	1	Yes	Private	Urban	228.69
1	Male	80.0	0	1	Yes	Private	Rural	105.92
2	Female	49.0	0	0	Yes	Private	Urban	171.23
3	Female	79.0	1	0	Yes	Self-employed	Rural	174.12
4	Male	81.0	0	0	Yes	Private	Urban	186.21

Checking distributions

In [48]:
```python
df.columns
```

Out[48]:
```
Index(['gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

In [49]:
```python
for column in ["gender", "hypertension", "heart_disease", "ever_married", "work_type",
    print(df[column].value_counts(), end="\n\n\n")
```
```
Female    2086
Male      1339
Other        1
Name: gender, dtype: int64


0    3018
1     408
Name: hypertension, dtype: int64


0    3220
1     206
Name: heart_disease, dtype: int64


Yes    2599
No      827
Name: ever_married, dtype: int64


Private          2201
Self-employed     629
Govt_job          514
children           68
Never_worked       14
Name: work_type, dtype: int64


Urban    1745
Rural    1681
Name: Residence_type, dtype: int64


never smoked       1852
formerly smoked     837
smokes              737
Name: smoking_status, dtype: int64


0    3246
1     180
Name: stroke, dtype: int64
```

- **As we see, the Other only has 1 value so we'll be removing it.**

In [50]:
```python
df = df[df.gender != "Other"].reset_index(drop=True)
df.nunique()
```

Out[50]:
```
gender                  2
age                    73
hypertension            2
heart_disease           2
ever_married            2
work_type               5
Residence_type          2
avg_glucose_level    2903
bmi                   370
smoking_status          3
stroke                  2
dtype: int64
```

Dataset Adaptation

To turn the categorical parameters into binary variables, we used the function factorize(). And the quantitative variables were split in two intervals.

- As the average value of "age" is near 45, we decided to split that parameter in two: > 45 and ≤ 45.
- If the average glucose level is greater than 100 while fasting, it means that a patient is in prediabetes (the reference values). So, we decided to split that parameter in two: > 100 and ≤ 100.
- As obese people have more risk to have a stroke, and a patient is considered obese when their bmi is above 30, we decided to split that parameter in two: ≥ 25 and < 25.

Changing age column

In [51]:
```python
df_changed = df.copy()
```

In [52]:
```python
df_changed["age"] = (df_changed["age"] > 45).astype(int)
df_changed.rename(columns={"age":"age_above_45"}, inplace=True)
```

Changing avg_glucose_level column

In [53]:
```python
df_changed["avg_glucose_level"] = (df_changed["avg_glucose_level"] > 100).astype(int)
df_changed.rename(columns={"avg_glucose_level":"avg_glucose_level_above_100"}, inplace
```

Changing bmi column

In [54]:
```python
avg_bmi_level = 25
df_changed["bmi"] = (df_changed["bmi"] > avg_bmi_level).astype(int)
column_name = f"bmi_above_{avg_bmi_level}"
df_changed.rename(columns={"bmi":column_name}, inplace=True)
```

Applying Bayesian Networks to our dataset

Our original proposal

Initially we thought on having a Bayesian Network with 3 layers as shown below. We only kept the nodes we considered being as risk factors to a patient to have a stroke. It is expected that all the risk factor depend heavily on patient's age. However, we considered that only bmi and smoking_status should be related to gender. Obviously, as we thought that the middle nodes are all risk factors, we predicted that all of them should be related to stroke.

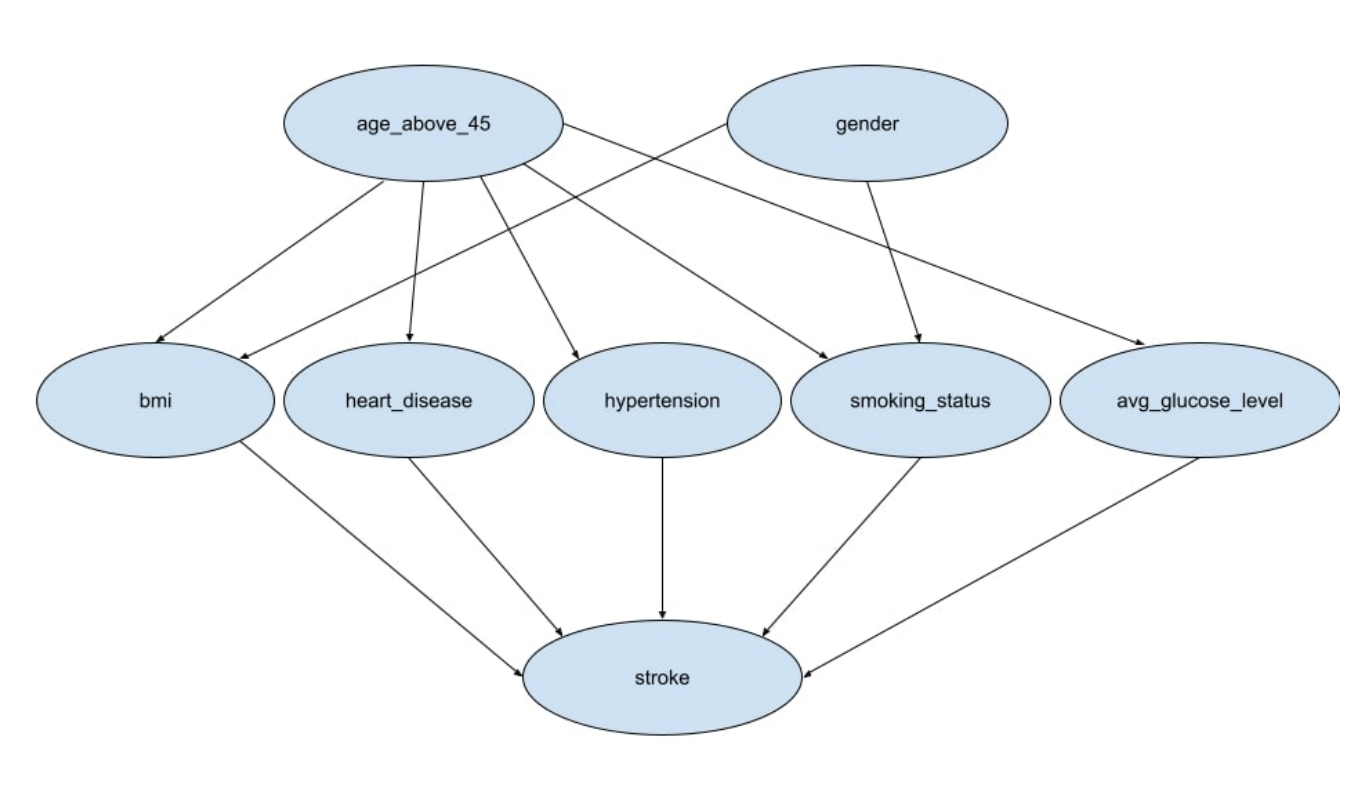


Fig : Original BN proposal

Only keeping useful nodes

In [55]:
```python
df_changed = df_changed[["gender", "age_above_45", "hypertension", "heart_disease", "
```

In [56]:
```python
df_changed
```

Out[56]:

	gender	age_above_45	hypertension	heart_disease	avg_glucose_level_above_100	bmi_above_25	smo
0	Male	1	0	1	1	1	forn
1	Male	1	0	1	1	1	n
2	Female	1	0	0	1	1	
3	Female	1	1	0	1	0	n
4	Male	1	0	0	1	1	forn
...	...	...	...	...	...	...	
3420	Male	1	1	0	0	1	n
3421	Female	1	0	0	0	0	n
3422	Female	1	0	0	1	1	n
3423	Female	0	0	0	0	1	n
3424	Male	1	0	0	1	1	forn

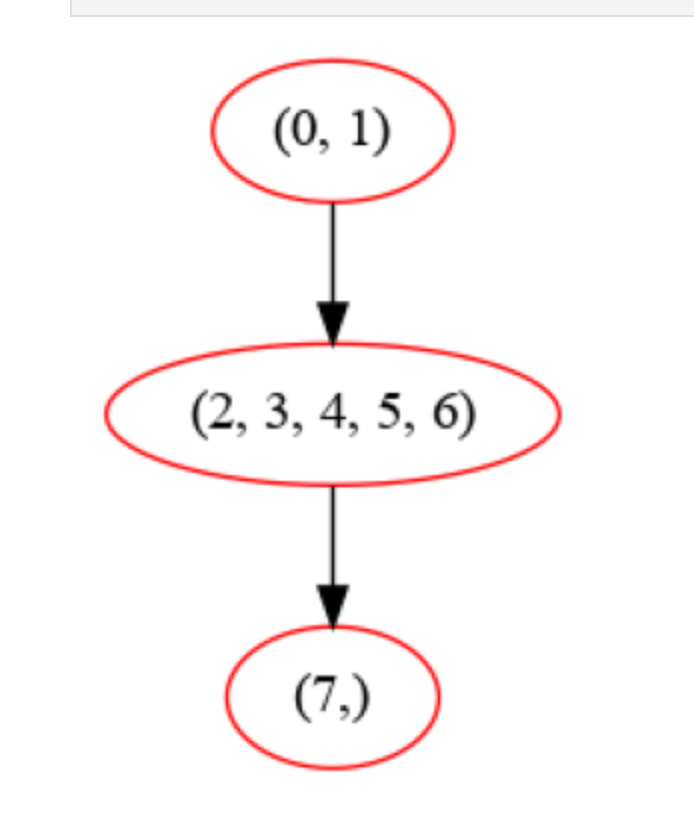
3425 rows × 8 columns

Creating BN with constraints

Using the Pomegranate library, we managed to simplify the amount of variables used, since not all of them had a significant correlation with each other, based on our data set. Since we have an idea of how the relationships between the variables of our dataset should go, we applied the notion of Constraint Graphs to our Bayesian Network, which should contain three layers: the top one consists of basic characteristics of an individual (gender, age); the middle one contains possible risk factors of a stroke (heart disease, smoking status, hypertension, etc.) and the bottom one is the variable we are interested in predicting, which is "stroke".

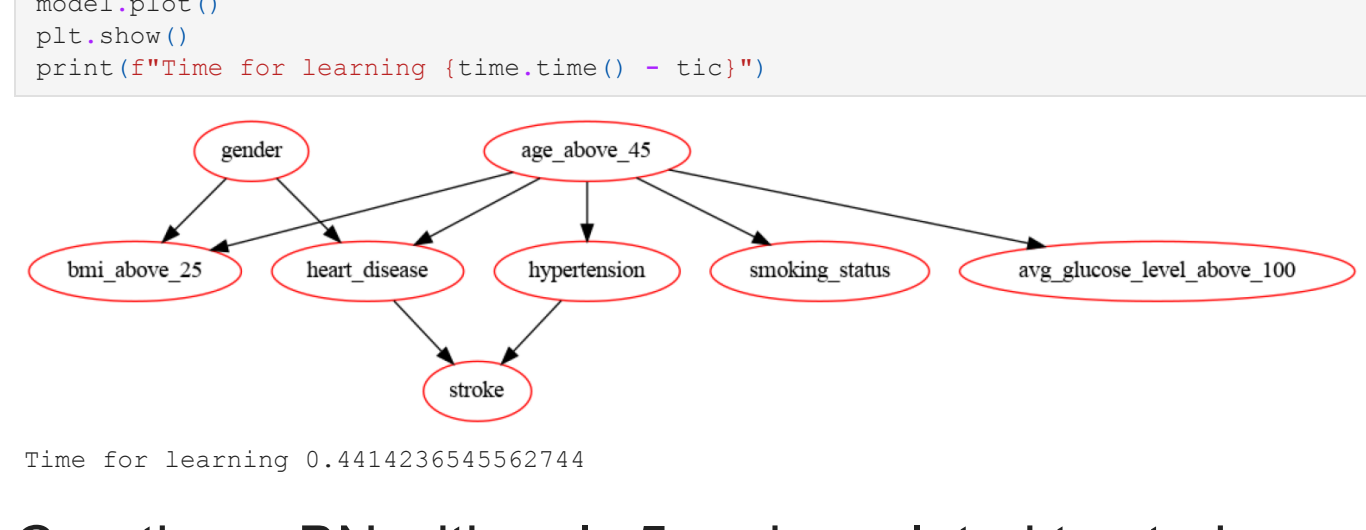
In [57]:
```python
for x,y in enumerate(df_changed.columns.values):
    print(f"{x} : {y}")
```
```
0 : gender
1 : age_above_45
2 : hypertension
3 : heart_disease
4 : avg_glucose_level_above_100
5 : bmi_above_25
6 : smoking_status
7 : stroke
```

In [58]:
```python
constrained_graph = nx.DiGraph()
layer_0 = tuple((0,1))
layer_1 = tuple((2, 3, 4, 5, 6))
layer_2 = tuple((7, ))

constrained_graph.add_edge(layer_0, layer_1)
constrained_graph.add_edge(layer_1, layer_2)
plot_networkx(constrained_graph)
plt.show()
```



The algorithm we implemented, as shown below, automatically calculated the relationships between these three layers, based on correlation notions. This way, we ended up with only two of the risk factors having a significant relationship with the variable stroke: "heart_disease" and "hypertension". These nodes are going to be known as "parents" of the node "stroke", which basically means that when we want to calculate the conditional probability of "stroke" given any subset of variables, the only ones that will be taken into account are the two parents of this node.

In [59]:
```python
tic = time.time()
model = BayesianNetwork.from_samples(df_changed, algorithm='exact',  state_names=df_cl

plt.figure(figsize=(20, 8))
model.plot()
plt.show()
print(f"Time for learning {time.time() - tic}")
```




```
Time for learning 0.4414236545562744
```

Creating a BN with only 5 nodes related to stroke

Our aim is to predict wether a person is likely to have a stroke or not, and we know that in a Bayesian Network only the parent nodes are important to calculate a conditional probabilty. For this reason, in the middle layer, we decided to only keep the two variables that our algorithm determined as variables highly related to stroke.

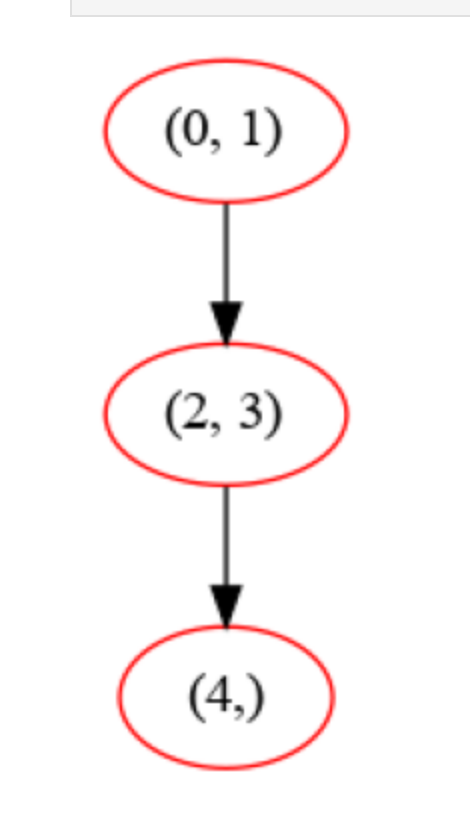
In [60]:
```python
nodes_selected = ["gender", "age_above_45", "heart_disease", "hypertension", "stroke"
df_final = df_changed[nodes_selected]
df_final.head()
```

Out[60]:

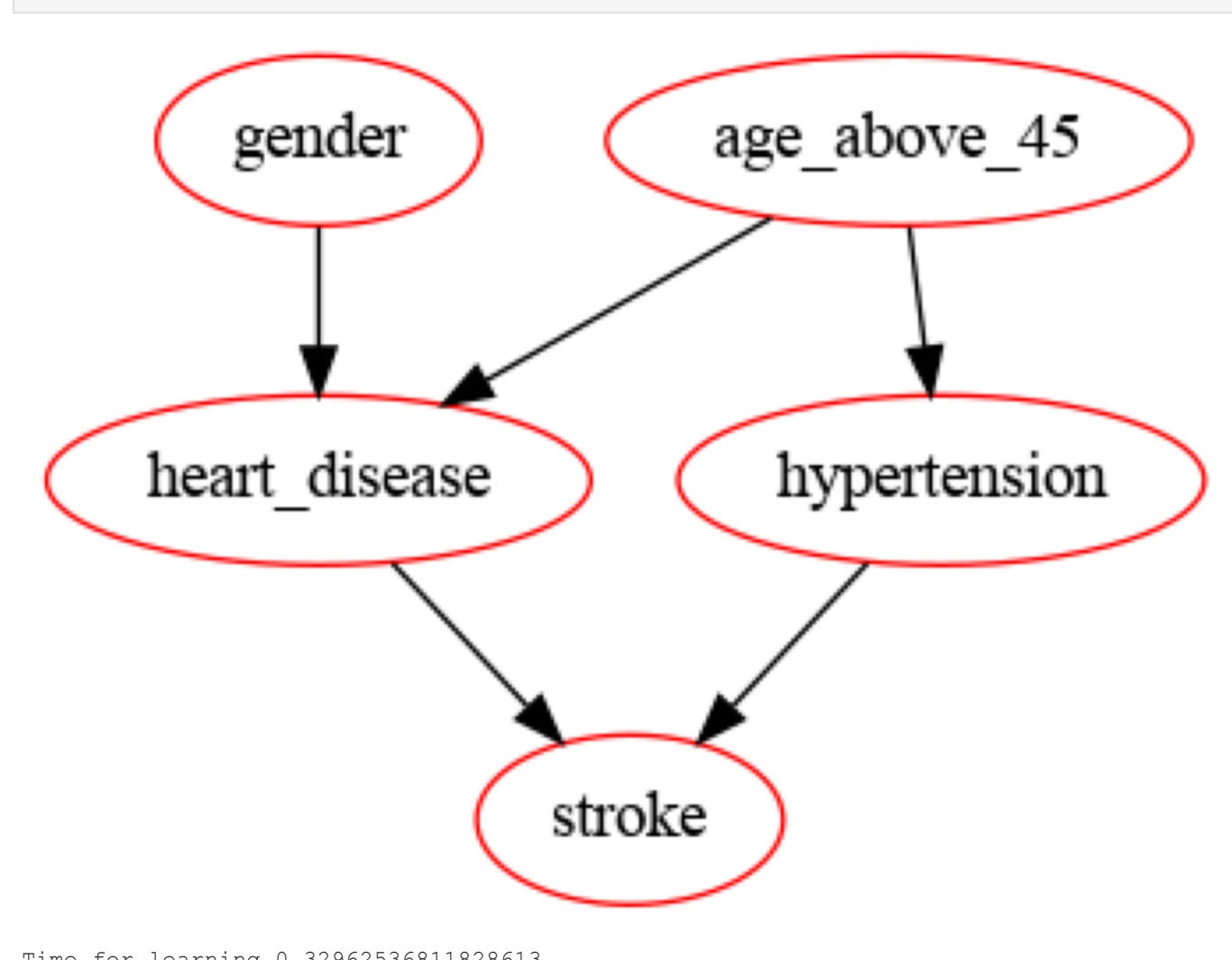
	gender	age_above_45	heart_disease	hypertension	stroke
0	Male	1	1	0	1
1	Male	1	1	0	1
2	Female	1	0	0	1
3	Female	1	0	1	1
4	Male	1	0	0	1

In [61]:
```python
for x,y in enumerate(df_final.columns.values):
    print(f"{x} : {y}")
```
```
0 : gender
1 : age_above_45
2 : heart_disease
3 : hypertension
4 : stroke
```

In [62]:
```python
constrained_graph = nx.DiGraph()
layer_0 = tuple((0,1))
layer_1 = tuple((2, 3))
layer_2 = tuple((4, ))

constrained_graph.add_edge(layer_0, layer_1)
constrained_graph.add_edge(layer_1, layer_2)
plot_networkx(constrained_graph)
plt.show()
```



In [63]:
```python
tic = time.time()
model = BayesianNetwork.from_samples(df_final, algorithm='exact',  state_names=df_fina

plt.figure(figsize=(20, 8))
model.plot()
plt.show()
print(f"Time for learning {time.time() - tic}")
```




```
Time for learning 0.32962536811828613
```

In [64]:
```python
model.bake()
```

In [65]:
```python
model.to_dict()
```

Out[65]:

```
{'class': 'BayesianNetwork',
 'name': '140192430894816',
 'structure': ((), (), (0, 1), (1,), (2, 3)),
 'states': [{'class': 'State',
   'distribution': {'class': 'Distribution',
    'dtype': 'str',
    'name': 'DiscreteDistribution',
    'parameters': [{'Female': 0.6090510948905109,
      'Male': 0.39094890510948904}],
    'frozen': False},
   'name': 'gender',
   'weight': 1.0},
  {'class': 'State',
   'distribution': {'class': 'Distribution',
    'dtype': 'int',
    'name': 'DiscreteDistribution',
    'parameters': [{'0': 0.43532846715328466, '1': 0.5646715328467153}],
    'frozen': False},
   'name': 'age_above_45',
   'weight': 1.0},
  {'class': 'State',
   'distribution': {'class': 'Distribution',
    'name': 'ConditionalProbabilityTable',
    'table': [['Female', '0', '0', '0.9969135802469136'],
     ['Female', '0', '1', '0.003086419753086421'],
     ['Female', '1', '0', '0.926391382405745'],
     ['Female', '1', '1', '0.07360861759425492'],
     ['Male', '0', '0', '0.9980732177263969'],
     ['Male', '0', '1', '0.0019267822736030833'],
     ['Male', '1', '0', '0.8536585365853658'],
     ['Male', '1', '1', '0.14634146341463414']],
    'dtypes': ['str', 'int', 'int', 'float'],
    'parents': [{'class': 'Distribution',
      'dtype': 'str',
      'name': 'DiscreteDistribution',
      'parameters': [{'Female': 0.6090510948905109,
        'Male': 0.39094890510948904}],
      'frozen': False},
     {'class': 'Distribution',
      'dtype': 'int',
      'name': 'DiscreteDistribution',
      'parameters': [{'0': 0.43532846715328466, '1': 0.5646715328467153}],
      'frozen': False}]},
   'name': 'heart_disease',
   'weight': 1.0},
  {'class': 'State',
   'distribution': {'class': 'Distribution',
    'name': 'ConditionalProbabilityTable',
    'table': [['0', '0', '0.9691482226693494'],
     ['0', '1', '0.03085177733065057'],
     ['1', '0', '0.81282316442606'],
     ['1', '1', '0.18717683557394002']],
    'dtypes': ['int', 'int', 'float'],
    'parents': [{'class': 'Distribution',
      'dtype': 'int',
      'name': 'DiscreteDistribution',
      'parameters': [{'0': 0.43532846715328466, '1': 0.5646715328467153}],
      'frozen': False}]},
   'name': 'hypertension',
   'weight': 1.0},
  {'class': 'State',
   'distribution': {'class': 'Distribution',
    'name': 'ConditionalProbabilityTable',
    'table': [['0', '0', '0', '0.9657940663176265'],
     ['0', '0', '1', '0.03420593368237347'],
     ['0', '1', '0', '0.8700564971751412'],
     ['0', '1', '1', '0.12994350282485873'],
     ['1', '0', '0', '0.8355263157894737'],
     ['1', '0', '1', '0.1644736842105263'],
     ['1', '1', '0', '0.7962962962962963'],
     ['1', '1', '1', '0.2037037037037037']],
    'dtypes': ['int', 'int', 'int', 'float'],
    'parents': [{'class': 'Distribution',
      'name': 'ConditionalProbabilityTable',
      'table': [['Female', '0', '0', '0.9969135802469136'],
       ['Female', '0', '1', '0.003086419753086421'],
       ['Female', '1', '0', '0.926391382405745'],
       ['Female', '1', '1', '0.07360861759425492'],
       ['Male', '0', '0', '0.9980732177263969'],
       ['Male', '0', '1', '0.0019267822736030833'],
       ['Male', '1', '0', '0.8536585365853658'],
       ['Male', '1', '1', '0.14634146341463414']],
      'dtypes': ['str', 'int', 'int', 'float'],
      'parents': [{'class': 'Distribution',
        'dtype': 'str',
        'name': 'DiscreteDistribution',
        'parameters': [{'Female': 0.6090510948905109,
          'Male': 0.39094890510948904}],
        'frozen': False},
       {'class': 'Distribution',
        'dtype': 'int',
        'name': 'DiscreteDistribution',
        'parameters': [{'0': 0.43532846715328466, '1': 0.5646715328467153}],
        'frozen': False}]},
     {'class': 'Distribution',
      'name': 'ConditionalProbabilityTable',
      'table': [['0', '0', '0.9691482226693494'],
       ['0', '1', '0.03085177733065057'],
       ['1', '0', '0.81282316442606'],
       ['1', '1', '0.18717683557394002']],
      'dtypes': ['int', 'int', 'float'],
      'parents': [{'class': 'Distribution',
        'dtype': 'int',
        'name': 'DiscreteDistribution',
        'parameters': [{'0': 0.43532846715328466, '1': 0.5646715328467153}],
        'frozen': False}]}]},
   'name': 'stroke',
   'weight': 1.0}]}
```


In [66]:
```python
type(model)
```

Out[66]:
```
pomegranate.BayesianNetwork.BayesianNetwork
```

In [67]:
```python
model_dict = model.to_dict()
```

In [68]:
```python
for state in model_dict["states"]:
    print(f"State : {state['name']}")

    dist = state['distribution']
    print(f"Type : {dist['name']}")

    if dist['name'] == "DiscreteDistribution":

        print("\n{:<15} |   {:<10}".format("Value","Probability"))
        print("-"*33)
        for param in dist["parameters"]:
            for v,p in param.items():
                print("{:<15} |   {:<10}".format(v, round(p,3)))


    elif dist['name'] == "ConditionalProbabilityTable":
        count = len(dist['table'][0])

        if state['name'] == "heart_disease":
            print("\n{:<12}|  {:<12}  | {:<12}  | {:<12}".format("gender", "age_above_

        elif state['name'] == "hypertension":
            print("\n{:<12}| {:<12}  | {:<12}".format("age_above_45", "hypertension",

        elif state['name'] == "stroke":
            print("\n{:<12}|  {:<12}   |   {:<12} | {:<12}".format("heart_disease", "h

        print("-"*count*15)

        for row in dist['table']:
            for i, col in enumerate(row):
                if not col.isalpha():
                    print("{:<12}".format(round(float(col),3)), end="")
                else:
                    print("{:<12}".format(col), end="")

                if i != count-1:
                    print("|     ", end="")
            print("\n")

    else:
        pass

    print("\n\n")
```
```
State : gender
Type : DiscreteDistribution

Value           |   Probability
---------------------------------
Female          |   0.609
Male            |   0.391



State : age_above_45
Type : DiscreteDistribution

Value           |   Probability
---------------------------------
0               |   0.435
1               |   0.565



State : heart_disease
Type : ConditionalProbabilityTable

gender      |  age_above_45  | heart_disease  | Probability
------------------------------------------------------------
Female      |    0.0         |    0.0         |    0.997

Female      |    0.0         |    1.0         |    0.003

Female      |    1.0         |    0.0         |    0.926

Female      |    1.0         |    1.0         |    0.074

Male        |    0.0         |    0.0         |    0.998

Male        |    0.0         |    1.0         |    0.002

Male        |    1.0         |    0.0         |    0.854

Male        |    1.0         |    1.0         |    0.146



State : hypertension
Type : ConditionalProbabilityTable

age_above_45| hypertension  | Probability
---------------------------------------------
0.0         |    0.0         |    0.969

0.0         |    1.0         |    0.031

1.0         |    0.0         |    0.813

1.0         |    1.0         |    0.187



State : stroke
Type : ConditionalProbabilityTable

heart_disease|  hypertension   |   Stroke       | Probability
------------------------------------------------------------
0.0         |    0.0         |    0.0         |    0.966

0.0         |    0.0         |    1.0         |    0.034

0.0         |    1.0         |    0.0         |    0.87

0.0         |    1.0         |    1.0         |    0.13

1.0         |    0.0         |    0.0         |    0.836

1.0         |    0.0         |    1.0         |    0.164

1.0         |    1.0         |    0.0         |    0.796

1.0         |    1.0         |    1.0         |    0.204
```

Results Analysis

- As expected, the highest probability of having a stroke happens when the patient has all the risk factors: $P(stroke = 1|heart\_disease = 1, hypertension = 1) = 0.204$. On the contrary, the highest probability of not having a stroke happens when the patient does not have either of the risk factors: $P(stroke = 0|heart\_disease = 0, hypertension = 0) = 0.966$.
- Moreover, we can perceive that when the patient only has heart_disease the probability of having a stroke is higher than when the patient only has hypertension.

Predictions

We can also use the BN model we have to make predictions.

```
In [69]: print(model.predict_proba([["Female",0,0,0,None]]))
```

```
[array(['Female', 0, 0, 0, {
                                    "class" : "Distribution",
                                    "dtype" : "int",
                                    "name" : "DiscreteDistribution",
                                    "parameters" : [
                                        {
                                            "0" : 0.9657940663176263,
                                            "1" : 0.03420593368237367
                                        }
                                    ],
                                    "frozen" : false
                                }                                     ],
      dtype=object)]
```

```
In [70]: print(model.predict_proba([["Female",0,1,1,None]]))
```

```
[array(['Female', 0, 1, 1, {
                                    "class" : "Distribution",
                                    "dtype" : "int",
                                    "name" : "DiscreteDistribution",
                                    "parameters" : [
                                        {
                                            "0" : 0.7962962962962963,
                                            "1" : 0.20370370370370378
                                        }
                                    ],
                                    "frozen" : false
                                }                                     ],
      dtype=object)]
```

- A Female with age under 45, no hypertension, no heart disease only has 3% chance of getting a stroke.
- A Female with same age characteristics but with heart disease and hypertension is 20% likely to get a stroke.

References

[1] "Home — pomegranate 0.14.6 documentation." https://pomegranate.readthedocs.io/en/latest/

[2] P. Jousilahti, E. Vartiainen, J. Tuomilehto, and P. Puska, "Sex, Age, Cardiovascular Risk Factors, and Coronary Heart Disease," Circulation, vol. 99, no. 9, pp. 1165–1172, Mar. 1999, doi: 10.1161/01.CIR.99.9.1165.