**ASSIGNMENT – 3**

**Bayesian networks**

Building a Bayesian network –

## Loading all the required packages first

library("tidyr")

library("ggplot2")

library("bnlearn")

library("Rgraphviz")

library("pROC")

## Setting no of decimal places to show

options(digits = 3)

## For this assignment I am using heart disease dataset, this is an opensource dataset from kaggle.

heart\_data <- read.csv("C:/Users/ASHISH/OneDrive/Desktop/Code Practices/R Scripts/UCI\_Heart\_Disease\_Dataset\_Combined.csv")

This is what each column in the data indicates -

Age: Age of individual. 20-80

Sex: This is the gender of the individual. It is represented as a binary value where 1 stands for male and 0 stands for female.

ChestPainType: This categorizes the type of chest pain experienced by the individual. The values are:

Value 1: Typical angina, which is chest pain related to the heart.

Value 2: Atypical angina, which is chest pain not related to the heart.

Value 3: Non-anginal pain, which is typically sharp and non-continuous.

Value 4: Asymptomatic, meaning the individual experiences no symptoms.

RestingBP: This is the individual’s resting blood pressure (in mm Hg) when they are at rest.

Cholesterol: This is the individual’s cholesterol level, measured in mg/dl.

FastingBS: This indicates whether the individual’s fasting blood sugar is greater than 120 mg/dl. It is represented as a binary value where 1 stands for true and 0 stands for false.

MaxHR: This is the maximum heart rate achieved by the individual.

ExerciseAngina: This indicates whether the individual experiences angina (chest pain) induced by exercise. It is represented as a binary value where 1 stands for yes and 0 stands for no.

# Copy the dataset

heart\_data\_copy <- heart\_data

## Analysing and pre-processing data

# view the first few rows

head(heart\_data\_copy,5)

Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR ExerciseAngina Oldpeak

1 40 1 1 140 289 0 0 172 0 0.0

2 49 0 2 160 180 0 0 156 0 1.0

3 37 1 1 130 283 0 1 98 0 0.0

4 48 0 3 138 214 0 0 108 1 1.5

5 54 1 2 150 195 0 0 122 0 0.0

HeartDisease

1 0

2 1

3 0

4 1

5 0

# view the structure

str(heart\_data\_copy)

'data.frame': 2943 obs. of 11 variables:

$ Age : int 40 49 37 48 54 39 45 54 37 48 ...

$ Sex : int 1 0 1 0 1 1 0 1 1 0 ...

$ ChestPainType : int 1 2 1 3 2 2 1 1 3 1 ...

$ RestingBP : int 140 160 130 138 150 120 130 110 140 120 ...

$ Cholesterol : int 289 180 283 214 195 339 237 208 207 284 ...

$ FastingBS : int 0 0 0 0 0 0 0 0 0 0 ...

$ RestingECG : int 0 0 1 0 0 0 0 0 0 0 ...

$ MaxHR : int 172 156 98 108 122 170 170 142 130 120 ...

$ ExerciseAngina: int 0 0 0 1 0 0 0 0 1 0 ...

$ Oldpeak : num 0 1 0 1.5 0 0 0 0 1.5 0 ...

$ HeartDisease : int 0 1 0 1 0 0 0 0 1 0 ...

# summary of the data

summary(heart\_data\_copy)

Age Sex ChestPainType RestingBP Cholesterol FastingBS

Min. :20.0 Min. :0.000 Min. :0.00 Min. : 0 Min. : 0 Min. :0.000

1st Qu.:44.0 1st Qu.:0.000 1st Qu.:0.00 1st Qu.:120 1st Qu.:204 1st Qu.:0.000

Median :54.0 Median :1.000 Median :1.00 Median :134 Median :248 Median :0.000

Mean :52.4 Mean :0.749 Mean :1.36 Mean :139 Mean :254 Mean :0.225

3rd Qu.:61.0 3rd Qu.:1.000 3rd Qu.:2.00 3rd Qu.:150 3rd Qu.:306 3rd Qu.:0.000

Max. :80.0 Max. :1.000 Max. :3.00 Max. :200 Max. :603 Max. :1.000

RestingECG MaxHR ExerciseAngina Oldpeak HeartDisease

Min. :0.000 Min. : 60 Min. :0.000 Min. :-2.60 Min. :0.000

1st Qu.:0.000 1st Qu.:124 1st Qu.:0.000 1st Qu.: 0.10 1st Qu.:0.000

Median :0.000 Median :145 Median :0.000 Median : 1.20 Median :1.000

Mean :0.627 Mean :144 Mean :0.413 Mean : 1.57 Mean :0.548

3rd Qu.:1.000 3rd Qu.:165 3rd Qu.:1.000 3rd Qu.: 2.40 3rd Qu.:1.000

Max. :2.000 Max. :202 Max. :1.000 Max. : 6.20 Max. :1.000

# dimensions

dim(heart\_data\_copy)

[1] 2943 11

# column names

names(heart\_data\_copy)

[1] "Age" "Sex" "ChestPainType" "RestingBP" "Cholesterol"

[6] "FastingBS" "RestingECG" "MaxHR" "ExerciseAngina" "Oldpeak"

[11] "HeartDisease"

#check null values

sum(is.na(heart\_data\_copy))

[1] 0

## since there are no null values there is no need to do the drop operation

## Changing the column data type into numeric for further processing since bnlearn can process factor and numeric data type easily.

for(col in names(heart\_data\_copy))

{

heart\_data\_copy[[col]] = as.numeric(heart\_data\_copy[[col]])

}

I have build Bayesian networks with pc.stable for constraint based network, hc for score based network, h2pc for hybrid model, and chow.liu for local discovery model. Because these models gave me better results that is the overall average loss was less compared to other models in the same criteria.

constraint\_model <- pc.stable(x=heart\_data\_copy)

constraint\_model

Bayesian network learned via Constraint-based methods

model:

[partially directed graph]

nodes: 11

arcs: 25

undirected arcs: 3

directed arcs: 22

average markov blanket size: 7.27

average neighbourhood size: 4.55

average branching factor: 2.00

learning algorithm: PC (Stable)

conditional independence test: Pearson's Correlation

alpha threshold: 0.05

tests used in the learning procedure: 1193

score\_based\_model <- hc(heart\_data\_copy)

score\_based\_model

Bayesian network learned via Score-based methods

model:

[ChestPainType][RestingBP][Cholesterol|ChestPainType:RestingBP]

[FastingBS|ChestPainType:RestingBP:Cholesterol][Sex|Cholesterol:FastingBS]

[Oldpeak|Sex:ChestPainType:RestingBP:Cholesterol][ExerciseAngina|Sex:Oldpeak]

[MaxHR|Cholesterol:ExerciseAngina][Age|FastingBS:MaxHR]

[HeartDisease|ChestPainType:RestingBP:FastingBS:MaxHR]

[RestingECG|RestingBP:Cholesterol:Oldpeak:HeartDisease]

nodes: 11

arcs: 25

undirected arcs: 0

directed arcs: 25

average markov blanket size: 6.18

average neighbourhood size: 4.55

average branching factor: 2.27

learning algorithm: Hill-Climbing

score: BIC (Gauss.)

penalization coefficient: 3.99

tests used in the learning procedure: 345

optimized: TRUE

hybrid\_model <- h2pc(heart\_data\_copy)

hybrid\_model

Bayesian network learned via Hybrid methods

model:

[ChestPainType][RestingBP][Cholesterol|ChestPainType:RestingBP][Sex|Cholesterol]

[Oldpeak|Sex:ChestPainType:RestingBP:Cholesterol][ExerciseAngina|Sex:Oldpeak]

[MaxHR|Cholesterol:ExerciseAngina][HeartDisease|ChestPainType:RestingBP:MaxHR]

[FastingBS|Sex:ChestPainType:RestingBP:HeartDisease]

[RestingECG|RestingBP:Cholesterol:Oldpeak:HeartDisease][Age|FastingBS:MaxHR]

nodes: 11

arcs: 24

undirected arcs: 0

directed arcs: 24

average markov blanket size: 6.18

average neighbourhood size: 4.36

average branching factor: 2.18

learning algorithm: Hybrid^2 Parent Children

constraint-based method: Hybrid Parents and Children

conditional independence test: Pearson's Correlation

score-based method: Hill-Climbing

score: BIC (Gauss.)

alpha threshold: 0.05

penalization coefficient: 3.99

tests used in the learning procedure: 3939

optimized: TRUE

local\_discovery\_model <- chow.liu(heart\_data\_copy)

local\_discovery\_model

Bayesian network learned via Pairwise Mutual Information methods

model:

[undirected graph]

nodes: 11

arcs: 10

undirected arcs: 10

directed arcs: 0

average markov blanket size: 1.82

average neighbourhood size: 1.82

average branching factor: 0.00

learning algorithm: Chow-Liu

mutual information estimator: Maximum Likelihood (Gauss.)

tests used in the learning procedure: 55

After reading the bnlearn documentation for calculating scores on different models , I was only able to calculate scores for the score based model and the hybrid model. Because both constraint based and local discovery algorithms produced partially directed graphs as shown in the below images.

graphviz.plot(constraint\_model,main=”Constraint\_based”)

A diagram of a diagram

Description automatically generated

graphviz.plot(local\_discovery\_model, main=”chow.liu”)

A diagram of a diagram

Description automatically generated with medium confidence

I used loglik-g for calculating scores of the score based and the hybrid model, because the data I have is continuous or normal variables. Loglik-g works better when you have small amounts of data. Whereas BIC and AIC requires the model to be built on high amount of data.

scores <- data.frame(

Model = c("Hill-Climbing algorithm", "H2PC algorithm"),

loglik\_g = c( score(score\_based\_model, heart\_data\_copy,"loglik-g"), score(hybrid\_model, heart\_data\_copy,"loglik-g"))

)

scores <- scores[order(-scores$loglik\_g), ]

scores

Model loglik\_g

1 Hill-Climbing algorithm -77070

2 H2PC algorithm -77077

After calculating the scores both score based and the hybrid approach had approximate scores, so to identify a better one I compared both graphs by visualizing them and I assume that HC gave a better idea on the relation between nodes.

graphviz.plot(score\_based\_model,main = "hc\_algorithm")

A diagram of a algorithm

Description automatically generated

graphviz.plot(hybrid\_model,main="h2pc")

A diagram of a diagram

Description automatically generated

For finding how well the model is working, I divided the data into training and testing datasets.

set.seed(123) # for reproducibility

## Sampling no of rows for training and testing data set

train\_indices <- sample(nrow(heart\_data\_copy), 0.7 \* nrow(heart\_data\_copy))

## Separating training set

train\_data <- heart\_data\_copy[train\_indices, ]

## Separating testing set

test\_data <- heart\_data\_copy[-train\_indices, ]

## Removing the target variable from testing set

test\_data\_1 <- subset(test\_data, select = -c(HeartDisease))

## fitting a model on the training data

model <- bn.fit(score\_based\_model,train\_data)

## predicting the model using the testing data

predicted = predict(model, node = "HeartDisease", data = test\_data\_1)

## printing the predicted values.

print(predicted)

[1] 0.7362 0.6170 0.4763 0.3459 0.5975 0.5319 0.4420 0.5100 0.6288 0.7164 0.3459

[12] 0.4696 0.5463 0.6910 0.6938 0.5531 0.5133 0.6156 0.6495 0.7380 0.7039 0.4306

[23] 0.4629 0.7543 0.6837 0.7839 0.6191 0.8316 0.5772 0.9371 0.7154 0.2314 0.3822

[34] 0.3950 0.6905 0.8250 0.8806 0.5750 0.5692 0.3835 0.5834 0.8732 0.3558 0.5730

[45] 0.3183 0.7604 0.5011 0.8828 0.8250 0.9184 0.4091 0.4192 0.7173 0.7880 0.5066

[56] 0.6950 0.7960 0.7981 0.7376 0.4360 0.7299 0.6737 0.9158 0.5154 0.4817 0.1534

[67] 0.5767 0.7166 0.3250 0.3351 0.3418 0.5504 0.7147 0.4950 0.8468 0.6185 0.6479

[78] 0.3566 0.6898 0.8075 0.5767 0.8840 0.6271 0.4461 0.8088 0.6338 0.3822 0.8686

....Omitted remaining rows

## Calculating/ evaluating loss to check how confident the model is.

mse <- mean((predicted - test\_data[["HeartDisease"]])^2)

mse

[1] 0.186

The overall loss calculated is very less which is 0.186, I feel like the model was able to predict values approximately considering it as a good model.

Predicting target variable for previous assignment dataset.

data\_jobs <- read.csv("C:/Users/ASHISH/OneDrive/Desktop/jobs\_in\_data.csv")

# Copy the dataset

data\_jobs\_copy <- data\_jobs

for(col in names(data\_jobs\_copy))

{

if(typeof(data\_jobs\_copy[[col]])=="integer")

{

data\_jobs\_copy[[col]] = as.numeric(data\_jobs\_copy[[col]])

}

else if(typeof(data\_jobs\_copy[[col]])=="character")

{

data\_jobs\_copy[[col]] = as.factor(data\_jobs\_copy[[col]])

}

}

data\_jobs\_copy <- subset(test\_data, select = -c(salary,salary\_currency,work\_setting,employee\_residence))

head(data\_jobs\_copy)

work\_year job\_title job\_category salary\_in\_usd experience\_level

1 2023 Data DevOps Engineer Data Engineering 95012 Mid-level

6 2023 Data Scientist Data Science and Research 130000 Senior

12 2023 Machine Learning Engineer Machine Learning and AI 224400 Senior

15 2023 Data Scientist Data Science and Research 36912 Mid-level

29 2023 Applied Scientist Data Science and Research 136000 Mid-level

31 2023 Machine Learning Engineer Machine Learning and AI 182200 Senior

employment\_type company\_location company\_size

1 Full-time Germany L

6 Full-time United States M

12 Full-time United States M

15 Full-time United Kingdom M

29 Full-time United States L

31 Full-time United States M

## Dividing the dataset into training and testing set

train\_indices <- sample(nrow(data\_jobs\_copy), 0.7 \* nrow(data\_jobs\_copy))

train\_data <- data\_jobs\_copy[train\_indices, ]

test\_data <- data\_jobs\_copy[-train\_indices, ]

test\_data\_1 <- subset(test\_data, select = -c(salary\_in\_usd))

new\_model <- hc(data\_jobs\_copy)

graphviz.plot(new\_model)

A diagram of a work flow

Description automatically generated

model <- bn.fit(new\_model,train\_data)

predicted = predict(model, node = "salary\_in\_usd", data = test\_data\_1)

print(predicted)

[1] 86651 211948 224279 34115 76130 299463 143926 112412 178701 140046 100565 204489 240714

[14] 199019 172000 184837 166241 173288 117471 129703 106539 166241 220070 252368 41724 31894

[27] 147306 186794 220070 110510 146311 105621 108827 209959 185096 150289 59714 115482 95592

[40] 36072 76130 408360 230147 171174 167220 220070 220070 102625 102625 102625 150488 136880

[53] 112412 114567 187498 114567 28748 28748 151560 123178 178701 78679 73263 48795 214931

[66] 167196 132201 239793 116477 90620 138508 63476 58582 249432 168986 113390 202997 127416

[79] 127101 112698 92410 161348 253716 85647 132475 82422 170156 ..omitted remaining rows

## Since the predicted values are very high, calculating mse and mae is not feasible or easy to understand so I choose rsquared since it limits the range from 0 to 1.

rsquared <- cor(predicted, test\_data[["salary\_in\_usd"]])^2

rsquared

[1] 0.983

## After calculating R^2 the overall accuracy of the predicted model is near to 1 making the model to be promising.