P SAI DEEKSHITH (1BM18CS148)

ML LAB RECORD

In [1]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
weatherinfo = pd.read_csv(r'D:\6th Sem\Machine Learning\Program 01\Data.csv')
weatherinfo
```

Out[3]:

	Weather	AirTemperature	Humidity	WaterTemperature	Wind	Goes out
0	Sunny	Moderate	Normal	Warm	Strong	Yes
1	Sunny	High	Normal	Too Warm	Strong	No
2	Rainy	Low	High	Cold	Breezy	No
3	Rainy	High	High	Warm	Breezy	Yes
4	Snowy	Moderate	Normal	Cold	Strong	Yes

In [4]:
```
data = np.array(weatherinfo)[:,:-1]
data
```

Out[4]:
```
array([['Sunny', 'Moderate', 'Normal', 'Warm', 'Strong'],
       ['Sunny', 'High', 'Normal', 'Too Warm ', 'Strong'],
       ['Rainy', 'Low', 'High', 'Cold', 'Breezy'],
       ['Rainy', 'High', 'High', 'Warm', 'Breezy'],
       ['Snowy', 'Moderate', 'Normal', 'Cold', 'Strong']], dtype=object)
```

In [6]:
```
values = np.array(weatherinfo)[:,-1]
values
```

Out[6]:
```
array(['Yes', 'No', 'No', 'Yes', 'Yes'], dtype=object)
```

In [6]:
```
values = np.array(weatherinfo)[:,-1]
values
```

Out[6]:
```
array(['Yes', 'No', 'No', 'Yes', 'Yes'], dtype=object)
```

Find S Algorithm

In [9]:
```
hypothesis = ['NULL'] * len(data[0])
print('Initial hypothesis:', hypothesis)
for j in range(0, len(values)):
    if values[j] == 'Yes':
        for i in range(0, len(data[0])):
            if hypothesis[i] == 'NULL' or hypothesis[i] == data[j][i]:
                hypothesis[i] = data[j][i]
            else:
                hypothesis[i] = '?'
    print('After', j, 'iteration in dataset, the hypothesis is:', hypothesis)
print('Final hypothesis:', hypothesis)
```

```
Initial hypothesis: ['NULL', 'NULL', 'NULL', 'NULL', 'NULL']
After 0 iteration in dataset, the hypothesis is: ['Sunny', 'Moderate', 'Normal', 'Warm', 'Strong']
After 1 iteration in dataset, the hypothesis is: ['Sunny', 'Moderate', 'Normal', 'Warm', 'Strong']
After 2 iteration in dataset, the hypothesis is: ['Sunny', 'Moderate', 'Normal', 'Warm', 'Strong']
After 3 iteration in dataset, the hypothesis is: ['?', '?', '?', 'Warm', '?']
After 4 iteration in dataset, the hypothesis is: ['?', '?', '?', '?', '?']
Final hypothesis: ['?', '?', '?', '?', '?']
```

In [ ]:

Program 2:

Candidate Elimination Algorithm

P SAI DEEKSHITH (1BM18CS148)

In [1]:
```
import pandas as pd
import numpy as np
```

In [7]:
```
data = pd.read_csv(r'D:\6th Sem\Machine Learning\Program 02\enjoysport.csv')
data = pd.DataFrame(data)
data
```

Out[7]:

	sky	airtemp	humidity	wind	water	forcast	enjoysport
0	sunny	warm	normal	strong	warm	same	yes
1	sunny	warm	high	strong	warm	same	yes
2	rainy	cold	high	strong	warm	change	no
3	sunny	warm	high	strong	cool	change	yes

In [9]:
```
concepts = np.array(data)[:,:-1]
concepts
```

Out[9]:
```
array([['sunny', 'warm', 'normal', 'strong', 'warm', 'same'],
       ['sunny', 'warm', 'high', 'strong', 'warm', 'same'],
       ['rainy', 'cold', 'high', 'strong', 'warm', 'change'],
       ['sunny', 'warm', 'high', 'strong', 'cool', 'change']],
      dtype=object)
```

In [16]:
```
target = np.array(data)[:,-1]
target
```

Out[16]:
```
array(['yes', 'yes', 'no', 'yes'], dtype=object)
```

In [21]:
```
specific_h = concepts[0].copy()
general_h = [['?' for i in range(len(specific_h))] for i in range(len(specific_h))]
print('Initially, ')
print('Specific Hypothesis : ')
print(specific_h)
print('General Hypothesis : ')
print(general_h)
print('\n')

for i, h in enumerate(concepts):
    if target[i] == "yes": # FIND S ALGO
        for x in range(len(specific_h)):
            if h[x] != specific_h[x]:
                specific_h[x] = '?'
                general_h[x][x] = '?'
            print(specific_h)
    print(specific_h)
    if target[i] == "no":
        for x in range(len(specific_h)):
            if h[x] != specific_h[x]:
                general_h[x][x] = specific_h[x]
            else:
                general_h[x][x] = '?'
    print(" steps of Candidate Elimination ALgorithm", i+1)
    print(specific_h)
    print(general_h)
indices  = [i for i, val in enumerate(general_h) if val == ['?','?','?','?','?','?']]
for i in indices:
    general_h.remove(['?','?','?','?','?','?'])

print("\n")
print("Final Specific Hypothesis :\n", specific_h)
print("\n")
```

P SAI DEEKSHITH (1BM18CS148)

```
print("Final General Hypothesis :\n", general_h )
```

```
Initially,
Specific Hypothesis :
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
General Hypothesis :
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]


['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
 steps of Candidate Elimination ALgorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination ALgorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination ALgorithm 3
```

```
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination ALgorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
 steps of Candidate Elimination ALgorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', 'same']]
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' 'warm' 'same']
['sunny' 'warm' '?' 'strong' '?' 'same']
['sunny' 'warm' '?' 'strong' '?' '?']
['sunny' 'warm' '?' 'strong' '?' '?']
 steps of Candidate Elimination ALgorithm 4
['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]


Final Specific Hypothesis :
 ['sunny' 'warm' '?' 'strong' '?' '?']


Final General Hypothesis :
 [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?',
'?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]
```

In [ ]:

P SAI DEEKSHITH (1BM18CS148)

Program 3:

Decision Tree ID3 Algorithm

In [262]:
```
import numpy as np
import pandas as pd
import math
```

In [263]:
```
data = pd.read_csv("id3.csv")
data
```

Out[263]:

	Outlook	Temperature	Humidity	Wind	Answer
0	sunny	hot	high	weak	no
1	sunny	hot	high	strong	no
2	overcast	hot	high	weak	yes
3	rain	mild	high	weak	yes
4	rain	cool	normal	weak	yes
5	rain	cool	normal	strong	no
6	overcast	cool	normal	strong	yes
7	sunny	mild	high	weak	no
8	sunny	cool	normal	weak	yes
9	rain	mild	normal	weak	yes
10	sunny	mild	normal	strong	yes
11	overcast	mild	high	strong	yes
12	overcast	hot	normal	weak	yes
13	rain	mild	high	strong	no

In [264]:
```
input = data.drop('Answer',axis=1)
input.head(3)
```

Out[264]:

	Outlook	Temperature	Humidity	Wind
0	sunny	hot	high	weak
1	sunny	hot	high	strong
2	overcast	hot	high	weak

In [265]:
```
target = data['Answer']
target.head(3)
```

Out[265]:
```
0     no
1     no
2    yes
Name: Answer, dtype: object
```

In [266]:
```
def attr_list(df):
    attributes = df.columns.tolist()[:-1]
    return attributes
print(attr_list(data))
```

```
['Outlook', 'Temperature', 'Humidity', 'Wind']
```

In [267]:
```
def attr_values(attributes,data):
    values = {}
    for i in attributes:
        values[i] = data[i].unique()
    return values
print(attr_values(attr_list(data),data))
```

P SAI DEEKSHITH (1BM18CS148)

In [267]:
```
def attr_values(attributes,data):
    values = {}
    for i in attributes:
        values[i] = data[i].unique()
    return values
print(attr_values(attr_list(data),data))
```

```
{'Outlook': array(['sunny', 'overcast', 'rain'], dtype=object), 'Temperature': array(['hot', 'mild', 'cool'], dtype=object), 'Humidit
y': array(['high', 'normal'], dtype=object), 'Wind': array(['weak', 'strong'], dtype=object)}
```

In [268]:
```
def entropy(data_n):
    p=len(data_n.where(data_n['Answer']=='yes').dropna())
    n=len(data_n.where(data_n['Answer']=='no').dropna())
    if(n==p):
        return 1
    if(n==0 or p==0):
        return 0
    e=(((-1)*p*math.log(p/(p+n),2)/(p+n)) + ((-1)*n*math.log(n/(p+n),2)/(p+n)))
    return float(e)
```

In [269]:
```
def information(attribute,att_values,data_n):
    p = len(data_n.where(data_n['Answer']=='yes').dropna())
    n = len(data_n.where(data_n['Answer']=='no').dropna())
    i=0
    for x in att_values:
        p_val = len(data_n.where((data_n[attribute]==x) & (data_n['Answer']=='yes')).dropna())
        n_val = len(data_n.where((data_n[attribute]==x) & (data_n['Answer']=='no')).dropna())
        i += ((p_val+n_val)/(p+n))*(entropy(data_n.where(data_n[attribute]==x).dropna()))
    return i
```

In [270]:
```
class Node:
```

In [270]:
```
class Node:
    def __init__(self,attribute):
        self.attribute=attribute
        self.children=[]
        self.answer = {}
```

In [271]:
```
NG = ['NG','NG','NG']
root = Node(NG)
print(root.attribute)
```

```
['NG', 'NG', 'NG']
```

In [272]:
```
def Compute_tree(df,root):
    if(len(set(df[df.columns.tolist()[-1]])) == 1):
        return set(df[df.columns.tolist()[-1]])
    e = entropy(df)
    attributes = attr_list(df)
    values = attr_values(attributes,df)
    gain_of_attr = {}
    gain = 0
    best_attr = attributes[0]
    for a in attributes:
        gain_of_attr[a] = e - information(a,values[a],df)
        gain,best_attr = (gain_of_attr[a],a) if(gain_of_attr[a] > gain) else (gain,best_attr)
    node = Node(best_attr)
    node.children = values[best_attr]
    for x in node.children:
        node.answer[x]=Compute_tree(df.where(df[best_attr]==x).dropna(),root)
    return node
r=Compute_tree(data,root)
```

In [273]:
```
#print(r.attribute,r.children,r.answer['sunny'].attribute)
```

P SAI DEEKSHITH (1BM18CS148)

In [273]:
```
#print(r.attribute,r.children,r.answer['sunny'].attribute)
print((" ")*(len(r.children[0])+5),r.attribute)
for c in r.children:
    if(r.answer[c]=={'yes'} or r.answer[c]=={'no'}):
        print(c,r.answer[c],end="    ")
    else:
        print(c,end="     ")
```

```
           Outlook
sunny     overcast {'yes'}   rain
```

In [250]:
```
def Display_tree(node,lable):
    if(node=={'yes'} or node=={'no'}):
        return
    print((" ")*(24),"",str(node.attribute).upper(),"(",lable,")")
    print("Brnches of :",node.attribute,"\n")
    for c in node.children:
        if(node.answer[c]!={'yes'} and node.answer[c]!={'no'}):
            print(c,end="\n")
        else:
            print(c,"\n",node.answer[c],end="\n")
        print()
    for c in node.children:
        if(node.answer[c]!={'yes'} and node.answer[c]!={'no'}):
                Display_tree(node.answer[c],"child of "+str(node.attribute).upper()+" through the branch : "+c)
```

In [251]:
```
Display_tree(r,"root")
```

```
                            OUTLOOK ( root )
Brnches of : Outlook

sunny
```

n [251]:
```
Display_tree(r,"root")
```

```
                            OUTLOOK ( root )
Brnches of : Outlook

sunny

overcast
 {'yes'}

rain

                            HUMIDITY ( child of OUTLOOK through the branch : sunny )
Brnches of : Humidity

high
 {'no'}

normal
 {'yes'}

                            WIND ( child of OUTLOOK through the branch : rain )
Brnches of : Wind

weak
 {'yes'}

strong
 {'no'}
```

P SAI DEEKSHITH (1BM18CS148)

Program 4:

Bayesian Classifier

Naive Bayes Classifier

In [36]:
```
import pandas as pd

data = pd.read_csv('PlayTennis.csv')
data.head()
```

Out[36]:

	PlayTennis	Outlook	Temperature	Humidity	Wind
0	No	Sunny	Hot	High	Weak
1	No	Sunny	Hot	High	Strong
2	Yes	Overcast	Hot	High	Weak
3	Yes	Rain	Mild	High	Weak
4	Yes	Rain	Cool	Normal	Weak

In [40]:
```
y = list(data['PlayTennis'].values)
X = data.iloc[:,1:].values

print(f'Target Values: {y}')
print(f'Features: \n{X}')
```

```
Target Values: ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
Features:
[['Sunny' 'Hot' 'High' 'Weak']
 ['Sunny' 'Hot' 'High' 'Strong']
 ['Overcast' 'Hot' 'High' 'Weak']
 ['Rain' 'Mild' 'High' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Strong']
 ['Overcast' 'Cool' 'Normal' 'Strong']
 ['Sunny' 'Mild' 'High' 'Weak']
 ['Sunny' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Mild' 'Normal' 'Weak']
 ['Sunny' 'Mild' 'Normal' 'Strong']
 ['Overcast' 'Mild' 'High' 'Strong']
 ['Overcast' 'Hot' 'Normal' 'Weak']
 ['Rain' 'Mild' 'High' 'Strong']]
```

In [41]:
```
y_train = y[:8]
y_val = y[8:]

X_train = X[:8]
X_val = X[8:]

print(f"Number of instances in training set: {len(X_train)}")
print(f"Number of instances in testing set: {len(X_val)}")
```

```
Number of instances in training set: 8
Number of instances in testing set: 6
```

In [42]:
```
class NaiveBayesClassifier:


    def __init__(self, X, y):

        self.X, self.y = X, y
```

P SAI DEEKSHITH (1BM18CS148)

```
def __init__(self, X, y):

    self.X, self.y = X, y

    self.N = len(self.X)

    self.dim = len(self.X[0])

    self.attrs = [[] for _ in range(self.dim)]

    self.output_dom = {}

    self.data = []

    for i in range(len(self.X)):
        for j in range(self.dim):
            if not self.X[i][j] in self.attrs[j]:
                self.attrs[j].append(self.X[i][j])

        if not self.y[i] in self.output_dom.keys():
            self.output_dom[self.y[i]] = 1

        else:
            self.output_dom[self.y[i]] += 1

        self.data.append([self.X[i], self.y[i]])


def classify(self, entry):

    solve = None
    max_arg = -1

    for y in self.output_dom.keys():
```

```
            for i in range(self.dim):
                cases = [x for x in self.data if x[0][i] == entry[i] and x[1] == y]
                n = len(cases)
                prob *= n/self.N

            if prob > max_arg:
                max_arg = prob
                solve = y

        return solve
```

[46]:
```
nbc = NaiveBayesClassifier(X_train, y_train)

total_cases = len(y_val)

good = 0
bad = 0
predictions = []

for i in range(total_cases):
    predict = nbc.classify(X_val[i])
    predictions.append(predict)

    if y_val[i] == predict:
        good += 1
    else:
        bad += 1

print('Predicted values:', predictions)
print('Actual values:', y_val)
print()
print('Total number of testing instances in the dataset:', total_cases)
print('Number of correct predictions:', good)
```

```
good = 0
bad = 0
predictions = []

for i in range(total_cases):
    predict = nbc.classify(X_val[i])
    predictions.append(predict)

    if y_val[i] == predict:
        good += 1
    else:
        bad += 1

print('Predicted values:', predictions)
print('Actual values:', y_val)
print()
print('Total number of testing instances in the dataset:', total_cases)
print('Number of correct predictions:', good)
print('Number of wrong predictions:', bad)
print()
print('Accuracy of Bayes Classifier:', good/total_cases)
```

```
Predicted values: ['No', 'Yes', 'No', 'Yes', 'Yes', 'No']
Actual values: ['Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']

Total number of testing instances in the dataset: 6
Number of correct predictions: 4
Number of wrong predictions: 2

Accuracy of Bayes Classifier: 0.6666666666666666
```

P SAI DEEKSHITH (1BM18CS148)

Program 5:

Bayesian Network

Baysian Network

In [2]:
```
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
```

In [13]:
```
heartDisease = pd.read_csv('heart.csv')
heartDisease.head(10)
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
5	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
6	62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
7	57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
8	63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
9	53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515	0.990099	149.607261	0.326733	1.039604	1.600660	0.683168		
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198	0.994971	22.875003	0.469794	1.161075	0.616226	0.944808		
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000		
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000		
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000		
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000		
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000		

In [7]:
```
model = BayesianModel([('age','heartdisease'),('sex','heartdisease'),('exang','heartdisease'),('cp','heartdisease'),('heartdisease','restecg'),('heartdisease','chol')])
print('Learning CPD using Maximum likelihood estimators')
```

```
Learning CPD using Maximum likelihood estimators
```

In [8]:
```
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)
```

In [9]:
```
print('Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model)
```

```
Inferencing with Bayesian Network:
```

In [24]:
```
print('1. Probability of HeartDisease given evidence = restecg : 1')
q1 = HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restecg':1})
print(q1)
```

```
Finding Elimination Order: :   0%|          | 0/5 [00:00<?, ?it/s]
  0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: chol:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: age:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: cp:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: exang:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: sex: 100%|██████████| 5/5 [00:00<00:00, 155.84it/s]
```

P SAI DEEKSHITH (1BM18CS148)

```
Finding Elimination Order: :   0%|          | 0/5 [00:00<?, ?it/s]
  0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: chol:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: age:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: cp:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: exang:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: sex: 100%|██████████| 5/5 [00:00<00:00, 155.84it/s]
```

```
1. Probability of HeartDisease given evidence = restecg : 1
+----------------+---------------------+
| heartdisease   |   phi(heartdisease) |
+================+=====================+
| heartdisease(0) |              0.1016 |
+----------------+---------------------+
| heartdisease(1) |              0.0000 |
+----------------+---------------------+
| heartdisease(2) |              0.2361 |
+----------------+---------------------+
| heartdisease(3) |              0.2017 |
+----------------+---------------------+
| heartdisease(4) |              0.4605 |
+----------------+---------------------+
```

3]:
```
print('Tuples with \'restecg = 1\' in the database are:')
heartDisease[heartDisease['restecg'] == 1]
```

```
Tuples with 'restecg = 1' in the database are:
```

3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
231	55	0	4	180	327	0	1	117	1	3.4	2	0	3	2
257	76	0	3	140	197	0	1	116	0	1.1	2	0	3	0
282	55	0	4	128	205	0	1	130	1	2.0	2	1	7	3

Out[18]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
231	55	0	4	180	327	0	1	117	1	3.4	2	0	3	2
257	76	0	3	140	197	0	1	116	0	1.1	2	0	3	0
282	55	0	4	128	205	0	1	130	1	2.0	2	1	7	3
285	58	1	4	114	318	0	1	140	0	4.4	3	3	6	4

In [20]:
```
print('2. Probability of HeartDisease given evidence = cp : 2')
q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2})
print(q2)
```

```
Finding Elimination Order: : 100%|██████████| 5/5 [00:37<00:00,  7.56s/it]
Finding Elimination Order: :   0%|          | 0/5 [00:00<?, ?it/s]
  0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: chol:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: restecg:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: age:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: exang:   0%|          | 0/5 [00:00<?, ?it/s]
Eliminating: sex: 100%|██████████| 5/5 [00:00<00:00, 142.78it/s]
```

```
2. Probability of HeartDisease given evidence = cp : 2
+----------------+---------------------+
| heartdisease   |   phi(heartdisease) |
+================+=====================+
| heartdisease(0) |              0.3742 |
+----------------+---------------------+
| heartdisease(1) |              0.2018 |
+----------------+---------------------+
| heartdisease(2) |              0.1375 |
+----------------+---------------------+
```

```
+----------------+---------------------+
| heartdisease(2) |              0.1375 |
+----------------+---------------------+
| heartdisease(3) |              0.1541 |
+----------------+---------------------+
| heartdisease(4) |              0.1323 |
+----------------+---------------------+
```

In [23]:
```
print('Tuples with \'cp = 2\' in the database are:')
heartDisease[heartDisease['cp'] == 2].head(10)
```

```
Tuples with 'cp = 2' in the database are:
```

Out[23]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
5	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
11	56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
13	44	1	2	120	263	0	0	173	0	0.0	1	0	7	0
16	48	1	2	110	229	0	0	168	0	1.0	3	0	7	1
19	49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
22	58	1	2	120	284	0	2	160	0	1.8	2	0	3	1
42	71	0	2	160	302	0	0	162	0	0.4	1	2	3	0
50	41	0	2	105	198	0	0	168	0	0.0	1	1	3	0
53	44	1	2	130	219	0	2	188	0	0.0	1	0	3	0