Pattern Recognition Assignment 1

N. Balasubramaniam

2020506020

Aim:

To analyze and use various machine learning algos

- Naïve Bayes
- Bayesian Belief Network

These methods are used to predict what contraceptive method used

Dataset:

This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey

This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

Attribute Info:

1. Wife's age (numerical)
2. Wife's education (categorical) 1=low, 2, 3, 4=high
3. Husband's education (categorical) 1=low, 2, 3, 4=high
4. Number of children ever born (numerical)
5. Wife's religion (binary) 0=non-Islam, 1=Islam
6. Wife's now working? (binary) 0=Yes, 1=No
7. Husband's occupation (categorical) 1, 2, 3, 4
8. Standard-of-living index (categorical) 1=low, 2, 3, 4=high
9. Media exposure (binary) 0=Good, 1=Not good
10. Contraceptive method used (class attribute) 1=No-use, 2=Long-term, 3=Short-term

Data Set Characteristics:	Multivariate	**Number of Instances:**	1473	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	9	**Date Donated**	1997-07-07
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	244395

Dataset Sample:

```
24,2,3,3,1,1,2,3,0,1
45,1,3,10,1,1,3,4,0,1
43,2,3,7,1,1,3,4,0,1
42,3,2,9,1,1,3,3,0,1
36,3,3,8,1,1,3,2,0,1
19,4,4,0,1,1,3,3,0,1
38,2,3,6,1,1,3,2,0,1
21,3,3,1,1,0,3,2,0,1
27,2,3,3,1,1,3,4,0,1
45,1,1,8,1,1,2,2,1,1
38,1,3,2,1,0,3,3,1,1
42,1,4,4,1,1,1,3,0,1
44,4,4,1,1,0,1,4,0,1
42,2,4,1,1,0,3,3,0,1
```

Naïve Bayes

Algo:

1. Import libraries and data set
2. Pre-process the data
3. Calculate the probabilities of each feature
4. Create a new sample
5. Predict what class the new sample falls under

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
ds = pd.read_csv('cmc.data.csv')
ds.head()
```

Out[2]:

	24	2	3	3.1	1	1.1	2.1	3.2	0	1.2
0	45	1	3	10	1	1	3	4	0	1
1	43	2	3	7	1	1	3	4	0	1
2	42	3	2	9	1	1	3	3	0	1
3	36	3	3	8	1	1	3	2	0	1
4	19	4	4	0	1	1	3	3	0	1

In [3]:

```
ds.columns = ["wife_age", "wife_education", "husband_education", "number_of_children", "
wife_religion", "wife_work", "husband_occupation", "standard_of_living", "media_exposure"
, "contraceptive_method_used"]
ds.head()
```

Out[3]:

	wife_age	wife_education	husband_education	number_of_children	wife_religion	wife_work	husband_occupation	standard_
0	45	1	3	10	1	1	3	
1	43	2	3	7	1	1	3	
2	42	3	2	9	1	1	3	
3	36	3	3	8	1	1	3	
4	19	4	4	0	1	1	3	

In [4]:

```
ds.shape
```

Out[4]:

```
(1472, 10)
```

In [5]:

```
# drop wife age
ds = ds.drop('wife_age', axis=1)
ds.head()
```

Out[5]:

	wife_education	husband_education	number_of_children	wife_religion	wife_work	husband_occupation	standard_of_living	
0	1	3	10	1	1	3	4	
1	2	3	7	1	1	3	4	
2	3	2	9	1	1	3	3	
3	3	3	8	1	1	3	2	
4	4	4	0	1	1	3	3	

In [6]:

```
ds['wife_education'].unique()
```

Out[6]:

```
array([1, 2, 3, 4], dtype=int64)
```

In [7]:

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1472 entries, 0 to 1471
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   wife_education              1472 non-null   int64
 1   husband_education           1472 non-null   int64
 2   number_of_children          1472 non-null   int64
 3   wife_religion               1472 non-null   int64
 4   wife_work                   1472 non-null   int64
 5   husband_occupation          1472 non-null   int64
 6   standard_of_living          1472 non-null   int64
 7   media_exposure              1472 non-null   int64
 8   contraceptive_method_used   1472 non-null   int64
dtypes: int64(9)
memory usage: 103.6 KB
```

In [8]:

```
no_of_records = ds.shape[0]
no_of_features= ds.shape[1]

method_1 = ds[ds['contraceptive_method_used'] == 1].shape[0]
method_2 = ds[ds['contraceptive_method_used'] == 2].shape[0]
method_3 = ds[ds['contraceptive_method_used'] == 3].shape[0]

method1_percentage = (method_1/no_of_records)*100
method2_percentage = (method_2/no_of_records)*100
method3_percentage = (method_3/no_of_records)*100

print("No of records: ", no_of_records)
print("No of features: ", no_of_features)
print("No of records for method 1: ", method_1)
print("No of records for method 2: ", method_2)
print("No of records for method 3: ", method_3)

print("Percentage of method 1: ", method1_percentage)
print("Percentage of method 2: ", method2_percentage)
print("Percentage of method 3: ", method3_percentage)
```

```
No of records:  1472
No of features:  9
No of records for method 1:  628
No of records for method 2:  333
No of records for method 3:  511
Percentage of method 1:  42.66304347826087
Percentage of method 2:  22.622282608695652
Percentage of method 3:  34.71467391304348
```

In [9]:

```
dso = ds.copy()
```

In [10]:

```
values = {}
```

```
for feature in ds.columns:
    values[feature] = []
values
```

Out[10]:

```
{'wife_education': [],
 'husband_education': [],
 'number_of_children': [],
 'wife_religion': [],
 'wife_work': [],
 'husband_occupation': [],
 'standard_of_living': [],
 'media_exposure': [],
 'contraceptive_method_used': []}
```

In [11]:

```
for feature in ds.columns:
    for i in ds[feature].unique():
        values[feature].append(i)
values
```

Out[11]:

```
{'wife_education': [1, 2, 3, 4],
 'husband_education': [3, 2, 4, 1],
 'number_of_children': [10, 7, 9, 8, 0, 6, 1, 3, 2, 4, 5, 12, 11, 13, 16],
 'wife_religion': [1, 0],
 'wife_work': [1, 0],
 'husband_occupation': [3, 2, 1, 4],
 'standard_of_living': [4, 3, 2, 1],
 'media_exposure': [0, 1],
 'contraceptive_method_used': [1, 2, 3]}
```

In [12]:

```
p1 = ds[ds['contraceptive_method_used'] == 1].sum()/len(ds)
p2 = ds[ds['contraceptive_method_used'] == 2].sum()/len(ds)
p3 = ds[ds['contraceptive_method_used'] == 3].sum()/len(ds)

p1, p2, p3
```

Out[12]:

```
(wife_education                1.139946
 husband_education             1.400136
 number_of_children            1.252038
 wife_religion                 0.375679
 wife_work                     0.311141
 husband_occupation            0.938859
 standard_of_living            1.260190
 media_exposure                0.050272
 contraceptive_method_used     0.426630
 dtype: float64,
 wife_education                0.781929
 husband_education             0.828804
 number_of_children            0.845788
 wife_religion                 0.174592
 wife_work                     0.165761
 husband_occupation            0.416440
 standard_of_living            0.784647
 media_exposure                0.006793
 contraceptive_method_used     0.452446
 dtype: float64,
 wife_education                1.037364
 husband_education             1.201087
 number_of_children            1.163723
 wife_religion                 0.300272
 wife_work                     0.272418
 husband_occupation            0.782609
 standard_of_living            1.088995
 media_exposure                0.016984
```

```
contraceptive_method_used    1.041440
dtype: float64)
```

In [13]:

```
dsp1 = ds[ds['contraceptive_method_used'] == 1].copy()
dsp2 = ds[ds['contraceptive_method_used'] == 2].copy()
dsp3 = ds[ds['contraceptive_method_used'] == 3].copy()

dsp1
```

Out[13]:

	wife_education	husband_education	number_of_children	wife_religion	wife_work	husband_occupation	standard_of_livin
0	1	3	10	1	1	3	
1	2	3	7	1	1	3	
2	3	2	9	1	1	3	
3	3	3	8	1	1	3	
4	4	4	0	1	1	3	
...	...	...	...	...	...	...	
1208	4	4	6	1	1	1	
1209	3	3	3	1	1	3	
1210	1	2	5	1	0	2	
1211	2	4	3	1	1	3	
1212	4	4	0	0	0	2	

628 rows × 9 columns

In [14]:

```
dsp1.drop('contraceptive_method_used', axis=1, inplace=True)
dsp2.drop('contraceptive_method_used', axis=1, inplace=True)
dsp3.drop('contraceptive_method_used', axis=1, inplace=True)

dsp1
```

Out[14]:

	wife_education	husband_education	number_of_children	wife_religion	wife_work	husband_occupation	standard_of_livin
0	1	3	10	1	1	3	
1	2	3	7	1	1	3	
2	3	2	9	1	1	3	
3	3	3	8	1	1	3	
4	4	4	0	1	1	3	
...	...	...	...	...	...	...	
1208	4	4	6	1	1	1	
1209	3	3	3	1	1	3	
1210	1	2	5	1	0	2	
1211	2	4	3	1	1	3	
1212	4	4	0	0	0	2	

628 rows × 8 columns

In [15]:

```
list_p1 = {}
for feature in dsp1:
    temp = dsp1.groupby(feature)
    p_temp = temp.size().apply(lambda x: x/len(dsp1))
    list_p1[feature] = p_temp
list_p1
```

Out[15]:

```
{'wife_education': wife_education
 1    0.164013
 2    0.278662
 3    0.278662
 4    0.278662
 dtype: float64,
 'husband_education': husband_education
 1    0.049363
 2    0.157643
 3    0.254777
 4    0.538217
 dtype: float64,
 'number_of_children': number_of_children
 0     0.151274
 1     0.227707
 2     0.181529
 3     0.109873
 4     0.090764
 5     0.070064
 6     0.055732
 7     0.028662
 8     0.046178
 9     0.007962
 10    0.014331
 11    0.009554
 12    0.006369
 dtype: float64,
 'wife_religion': wife_religion
 0    0.119427
 1    0.880573
 dtype: float64,
 'wife_work': wife_work
 0    0.270701
 1    0.729299
 dtype: float64,
 'husband_occupation': husband_occupation
 1    0.251592
 2    0.316879
 3    0.410828
 4    0.020701
 dtype: float64,
 'standard_of_living': standard_of_living
 1    0.127389
 2    0.186306
 3    0.291401
 4    0.394904
 dtype: float64,
 'media_exposure': media_exposure
 0    0.882166
 1    0.117834
 dtype: float64}
```

In [16]:

```
list_p2 = {}
for feature in dsp2:
    temp = dsp2.groupby(feature)
    p_temp = temp.size().apply(lambda x: x/len(dsp2))
    list_p2[feature] = p_temp
list_p2
```

Out[16]:

```
{'wife education': wife education
```

```
1    0.027027
2    0.111111
3    0.240240
4    0.621622
dtype: float64,
'husband_education': husband_education
1    0.030030
2    0.048048
3    0.150150
4    0.771772
dtype: float64,
'number_of_children': number_of_children
1     0.138138
2     0.168168
3     0.210210
4     0.186186
5     0.108108
6     0.081081
7     0.057057
8     0.027027
9     0.009009
10    0.006006
11    0.006006
13    0.003003
dtype: float64,
'wife_religion': wife_religion
0    0.228228
1    0.771772
dtype: float64,
'wife_work': wife_work
0    0.267267
1    0.732733
dtype: float64,
'husband_occupation': husband_occupation
1    0.468468
2    0.237237
3    0.279279
4    0.015015
dtype: float64,
'standard_of_living': standard_of_living
1    0.027027
2    0.090090
3    0.270270
4    0.612613
dtype: float64,
'media_exposure': media_exposure
0    0.96997
1    0.03003
dtype: float64}
```

In [17]:

```
list_p3 = {}
for feature in dsp3:
    temp = dsp3.groupby(feature)
    p_temp = temp.size().apply(lambda x: x/len(dsp3))
    list_p3[feature] = p_temp
list_p3
```

Out[17]:

```
{'wife_education': wife_education
1    0.078278
2    0.236791
3    0.303327
4    0.381605
dtype: float64,
'husband_education': husband_education
1    0.005871
2    0.123288
3    0.275930
4    0.594912
```

```
dtype: float64,
'number_of_children': number_of_children
0     0.003914
1     0.170254
2     0.207436
3     0.232877
4     0.152642
5     0.107632
6     0.058708
7     0.023483
8     0.017613
9     0.015656
11    0.005871
13    0.001957
16    0.001957
dtype: float64,
'wife_religion': wife_religion
0    0.135029
1    0.864971
dtype: float64,
'wife_work': wife_work
0    0.215264
1    0.784736
dtype: float64,
'husband_occupation': husband_occupation
1    0.238748
2    0.285714
3    0.457926
4    0.017613
dtype: float64,
'standard_of_living': standard_of_living
1    0.078278
2    0.160470
3    0.307241
4    0.454012
dtype: float64,
'media_exposure': media_exposure
0    0.951076
1    0.048924
dtype: float64}
```

In [18]:

```
def predict(newSample):
    p1 = 0
    p2 = 0
    p3 = 0
    for feature in newSample:
        p1 += np.log(list_p1[feature][newSample[feature]])
        p2 += np.log(list_p2[feature][newSample[feature]])
        p3 += np.log(list_p3[feature][newSample[feature]])
    p1 += np.log(method1_percentage)
    p2 += np.log(method2_percentage)
    p3 += np.log(method3_percentage)
    return np.argmax([p1, p2, p3])+1
```

Function to predict the method

In [19]:

```
def predict_print(newSample):
    print("Predicted method: ", predict(newSample))
```

In [21]:

```
# testing new samples

newSample = {'wife_education': 4, 'husband_education': 4, 'number_of_children': 3, 'wife
_religion': 1, 'wife_work': 1, 'husband_occupation': 2, 'standard_of_living': 3, 'media_
exposure': 0}
```

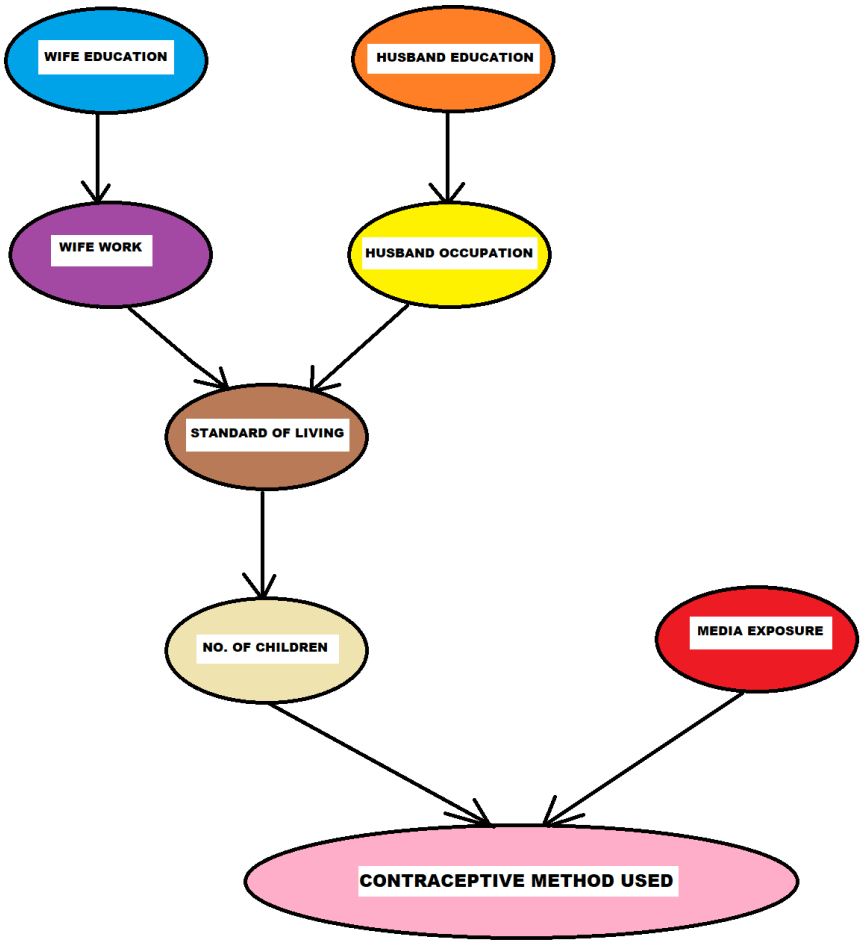
```
predict_print(newSample)

newSample2 = {'wife_education': 1, 'husband_education': 1, 'number_of_children': 3, 'wif
e_religion': 1, 'wife_work': 1, 'husband_occupation': 3, 'standard_of_living': 4, 'media
_exposure': 0}

predict_print(newSample2)
```

```
Predicted method:  3
Predicted method:  1
```

Bayesian Belief Network

Algo:

1. Import libraries and data set
2. Pre-process the data
3. Calculate the probabilities of each feature
4. Calculate relational probability of each feature
5. Calculate probability where 2 parents also exist
6. Finally calculate the final probability



In [1]:

```
import pandas as pd
```

In [2]:

```
ds = pd.read_csv('cmc.data.csv')
ds.head()
```

Out[2]:

	24	2	3	3.1	1	1.1	2.1	3.2	0	1.2
0	45	1	3	10	1	1	3	4	0	1
1	43	2	3	7	1	1	3	4	0	1
2	42	3	2	9	1	1	3	3	0	1
3	36	3	3	8	1	1	3	2	0	1
4	19	4	4	0	1	1	3	3	0	1
...	...	...	...	...	...	...	...	...	...	...
1467	33	4	4	2	1	0	2	4	0	3
1468	33	4	4	3	1	1	1	4	0	3
1469	39	3	3	8	1	0	1	4	0	3
1470	33	3	3	4	1	0	2	2	0	3
1471	17	3	3	1	1	1	2	4	0	3

1472 rows × 10 columns

In [3]:

```
ds.columns = ["wife_age", "wife_education", "husband_education", "number_of_children", "
wife_religion", "wife_work", "husband_occupation", "standard_of_living", "media_exposure"
, "contraceptive_method_used"]
ds.head()
```

Out[3]:

	wife_age	wife_education	husband_education	number_of_children	wife_religion	wife_work	husband_occupation	standard_
0	45	1	3	10	1	1	3	
1	43	2	3	7	1	1	3	
2	42	3	2	9	1	1	3	
3	36	3	3	8	1	1	3	
4	19	4	4	0	1	1	3	

In [4]:

```
ds.shape
```

Out[4]:

```
(1472, 10)
```

In [5]:

```
# print max and min no of children
print("Max no of children: ", ds['number_of_children'].max())
print("Min no of children: ", ds['number_of_children'].min())
```

```
Max no of children:  16
```

```
Min no of children:  0
```

In [6]:

```
# split age 0: young 1:middle 2:old

ds['wife_age'] = pd.cut(ds['wife_age'], bins=[0, 25, 40, 200], labels=[0, 1, 2])
```

In [7]:

```
# group children as 0-3, 4-6, 7-10, 10+
ds['number_of_children'] = pd.cut(ds['number_of_children'], bins=[-1, 3, 6, 10, 200], la
bels=[0, 1, 2, 3])
```

In [8]:

```
ds.head()
```

Out[8]:

	wife_age	wife_education	husband_education	number_of_children	wife_religion	wife_work	husband_occupation	standard_
0	**2**	**1**	**3**	**2**	**1**	**1**	**3**	
1	**2**	**2**	**3**	**2**	**1**	**1**	**3**	
2	**2**	**3**	**2**	**2**	**1**	**1**	**3**	
3	**1**	**3**	**3**	**2**	**1**	**1**	**3**	
4	**0**	**4**	**4**	**0**	**1**	**1**	**3**	

In [9]:

```
# check if nan values is present

ds.isnull().values.any()
```

Out[9]:

```
False
```

In [10]:

```
# probability of wife age
age_rows = ds['wife_age'].cat.categories
age_dict = {}

for i in age_rows:
    age_dict[i] = len(ds[ds['wife_age'] == i]) / len(ds)

age_dict
```

Out[10]:

```
{0: 0.24116847826086957, 1: 0.5509510869565217, 2: 0.2078804347826087}
```

In [11]:

```
# print datatypes of all columns
ds.dtypes
```

Out[11]:

```
wife_age                  category
wife_education               int64
husband_education            int64
number_of_children        category
wife_religion                int64
wife_work                    int64
husband_occupation           int64
standard_of_living           int64
media_exposure               int64
```

```
media_exposure                int64
contraceptive_method_used     int64
dtype: object
```

In [12]:

```
# probability of wife education
education_rows = ds['wife_education'].unique()
education_dict = {}
education_rows.sort()

for i in education_rows:
    education_dict[i] = len(ds[ds['wife_education'] == i]) / len(ds)

education_dict
```

Out[12]:

```
{1: 0.10326086956521739,
 2: 0.2262228260869565,
 3: 0.27853260869565216,
 4: 0.3919836956521739}
```

In [13]:

```
# probability of husband education
husband_education_rows = ds['husband_education'].unique()
husband_education_dict = {}
husband_education_rows.sort()

for i in husband_education_rows:
    husband_education_dict[i] = len(ds[ds['husband_education'] == i]) / len(ds)

husband_education_dict
```

Out[13]:

```
{1: 0.029891304347826088,
 2: 0.12092391304347826,
 3: 0.23845108695652173,
 4: 0.610733695652174}
```

In [14]:

```
# probability of wife religion

religion_rows = ds['wife_religion'].unique()
religion_dict = {}
religion_rows.sort()

for i in religion_rows:
    religion_dict[i] = len(ds[ds['wife_religion'] == i]) / len(ds)

religion_dict
```

Out[14]:

```
{0: 0.14945652173913043, 1: 0.8505434782608695}
```

In [15]:

```
# probability of wife work depends on wife education

work_rows = ds['wife_work'].unique()
work_dict = {}
prob_work = {}

for i in work_rows:
    ls = {}
    temp = 0
    for j in education_rows:
        ls[j] = len(ds[(ds['wife_work'] == i) & (ds['wife_education'] == j)]) / len(ds[d
s['wife_education'] == j])
```

```
        temp += ls[j] * education_dict[j]
    work_dict[i] = ls
    prob_work[i] = temp

print(work_dict)
print(prob_work)
```

```
{1: {1: 0.7697368421052632, 2: 0.7747747747747747, 3: 0.7829268292682927, 4: 0.7053726169
844021}, 0: {1: 0.23026315789473684, 2: 0.22522522522522523, 3: 0.21707317073170732, 4: 0
.29462738301559793}}
{1: 0.7493206521739131, 0: 0.250679347826087}
```

In [16]:

```
# probability of husband occupation depends on husband education

occupation_rows = ds['husband_occupation'].unique()
occupation_dict = {}
prob_occupation = {}

for i in occupation_rows:
    ls = {}
    temp = 0
    for j in husband_education_rows:
        ls[j] = len(ds[(ds['husband_occupation'] == i) & (ds['husband_education'] == j)]
) / len(ds[ds['husband_education'] == j])
        temp += ls[j] * husband_education_dict[j]
    occupation_dict[i] = ls
    prob_occupation[i] = temp

print(prob_occupation)
# pretty print occupation_dict
import pprint
pprint.pprint(occupation_dict)
```

```
{3: {1: 0.5, 2: 0.550561797752809, 3: 0.5498575498575499, 4: 0.3025583982202447}, 2: {1:
0.4090909090909091, 2: 0.37640449438202245, 3: 0.3504273504273504, 4: 0.24026696329254726
}, 1: {1: 0.022727272727272728, 2: 0.033707865168539325, 3: 0.08547008547008547, 4: 0.443
82647385984425}, 4: {1: 0.06818181818181818, 2: 0.03932584269662921, 3: 0.014245014245014
245, 4: 0.013348164627363738}}
{3: 0.39741847826086957, 2: 0.28804347826086957, 1: 0.2961956521739131, 4: 0.018342391304
347824}
{1: {1: 0.022727272727272728,
     2: 0.033707865168539325,
     3: 0.08547008547008547,
     4: 0.44382647385984425},
 2: {1: 0.4090909090909091,
     2: 0.37640449438202245,
     3: 0.3504273504273504,
     4: 0.24026696329254726},
 3: {1: 0.5,
     2: 0.550561797752809,
     3: 0.5498575498575499,
     4: 0.3025583982202447},
 4: {1: 0.06818181818181818,
     2: 0.03932584269662921,
     3: 0.014245014245014245,
     4: 0.013348164627363738}}
```

In [32]:

```
# probability of standard of living depends on wife work and husband work

living_rows = ds['standard_of_living'].unique()
living_dict = {}
prob_living = {}
# sort living rows
living_rows.sort()

for i in living_rows:
    ls = {}
    temp = 0
```

```
    for j in work_rows:
        ls2 = {}
        for k in occupation_rows:
            ls2[k] = len(ds[(ds['standard_of_living'] == i) & (ds['wife_work'] == j) & (
ds['husband_occupation'] == k)]) / len(ds[(ds['wife_work'] == j) & (ds['husband_occupati
on'] == k)])
            temp += ls2[k] * prob_work[j] * prob_occupation[k]
        ls[j] = ls2
    living_dict[i] = ls
    prob_living[i] = temp

# print(living_dict)

# pretty print living_dict
for i in living_dict:
    print(i, ":")
    for j in living_dict[i]:
        print("\t", j, ":")
        for k in living_dict[i][j]:
            print("\t\t", k, ": ", living_dict[i][j][k])
```

```
1 :
  1 :
   3 :  0.1434878587196468
   2 :  0.11326860841423948
   1 :  0.018461538461538463
   4 :  0.1875
  0 :
   3 :  0.09090909090909091
   2 :  0.043478260869565216
   1 :  0.009009009009009009
   4 :  0.18181818181818182
2 :
  1 :
   3 :  0.1986754966887417
   2 :  0.1715210355987055
   1 :  0.08
   4 :  0.1875
  0 :
   3 :  0.23484848484848486
   2 :  0.19130434782608696
   1 :  0.02702702702702703
   4 :  0.09090909090909091
3 :
  1 :
   3 :  0.33774834437086093
   2 :  0.313915857605178
   1 :  0.24
   4 :  0.25
  0 :
   3 :  0.3409090909090909
   2 :  0.2
   1 :  0.26126126126126126
   4 :  0.09090909090909091
4 :
  1 :
   3 :  0.3200883002207506
   2 :  0.40129449838187703
   1 :  0.6615384615384615
   4 :  0.375
  0 :
   3 :  0.3333333333333333
   2 :  0.5652173913043478
   1 :  0.7027027027027027
   4 :  0.6363636363636364
```

In [18]:

```
# find probability of no of children

children_rows = ds['number_of_children'].unique()
children_dict = {}
```

```
prob_children = {}

for i in children_rows:
    ls = {}
    temp = 0
    for j in living_rows:
        ls[j] = len(ds[(ds['number_of_children'] == i) & (ds['standard_of_living'] == j)
]) / len(ds[ds['standard_of_living'] == j])
        temp += ls[j] * prob_living[j]
    children_dict[i] = ls
    prob_children[i] = temp

print(children_dict)
```

```
{2: {1: 0.11627906976744186, 2: 0.08296943231441048, 3: 0.09534883720930233, 4: 0.0701754
3859649122}, 0: {1: 0.6356589147286822, 2: 0.5851528384279476, 3: 0.6116279069767442, 4:
0.6257309941520468}, 1: {1: 0.23255813953488372, 2: 0.31877729257641924, 3: 0.27906976744
186046, 4: 0.29385964912280704}, 3: {1: 0.015503875968992248, 2: 0.013100436681222707, 3:
0.013953488372093023, 4: 0.01023391812865497}}
```

In [19]:

```
# find probability of media exposure, does not depend on anything

media_rows = ds['media_exposure'].unique()
media_dict = {}
prob_media = {}

for i in media_rows:
    media_dict[i] = len(ds[ds['media_exposure'] == i]) / len(ds)
    prob_media[i] = media_dict[i]

print(media_dict)
```

```
{0: 0.9259510869565217, 1: 0.07404891304347826}
```

In [21]:

```
ds.head()
```

Out[21]:

	wife_age	wife_education	husband_education	number_of_children	wife_religion	wife_work	husband_occupation	standard_
0	2	1	3	2	1	1	3	
1	2	2	3	2	1	1	3	
2	2	3	2	2	1	1	3	
3	1	3	3	2	1	1	3	
4	0	4	4	0	1	1	3	

In [27]:

```
# probability of contraceptive method
# depends on media_exposure and no_of_children


method_rows = ds['contraceptive_method_used'].unique()
method_dict = {}
prob_method = {}

for i in method_rows:
    ls = {}
    temp = 0
    for j in media_rows:
        ls2 = {}
        for k in children_rows:
            ls2[k] = len(ds[(ds['contraceptive_method_used'] == i) & (ds['media_exposure
'] == j) & (ds['number_of_children'] == k)]) / len(ds[(ds['media_exposure'] == j) & (ds[
```

```
'number_of_children'] == k)])
            temp += ls2[k] * prob_media[j] * prob_children[k]
        ls[j] = ls2
    method_dict[i] = ls
    prob_method[i] = temp

print(prob_method)
```

```
{1: 0.42422090100141174, 2: 0.2270201346784538, 3: 0.3487589643201346}
```

In [30]:

```
# print percentage of each contraceptive method

for i in prob_method:
    print(i, ": ", prob_method[i] * 100, "%")
```

```
1 :  42.422090100141176 %
2 :  22.70201346784538 %
3 :  34.87589643201346 %
```

Result:

Thus, Naïve Bayes and Bayesian Belief Network were tested on the given dataset and the results were observed.