1. Write a program to construct a Bayesian Belief network considering medical data. Use this model to demonstrate the diagnosis of patients.

4. Write a program to construct a Bayesian network to diagnose heart disease prediction.

This for 1 and 4

```
import pandas as pd
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination

data = pd.read_csv("D:\Downloads\heartdisease.csv")
heart_disease = pd.DataFrame(data)
print(heart_disease.head())

model = BayesianModel([
    ('age', 'Lifestyle'),
    ('Gender', 'Lifestyle'),
    ('Family', 'heartdisease'),
    ('diet', 'cholestrol'),
    ('Lifestyle', 'diet'),
    ('cholestrol', 'heartdisease'),
    ('diet', 'cholestrol')
])
#Building Bayesian Network
model.fit(heart_disease, estimator=MaximumLikelihoodEstimator)
print(model.get_cpds('age'))
#inference Bayesian Network

#Till here 1st one


HeartDisease_infer = VariableElimination(model)

print('For Age enter SuperSeniorCitizen:0, SeniorCitizen:1, MiddleAged:2, Youth:3,
Teen:4')
print('For Gender enter Male:0, Female:1')
print('For Family History enter Yes:1, No:0')
print('For Diet enter High:0, Medium:1')
print('for LifeStyle enter Athlete:0, Active:1, Moderate:2, Sedentary:3')
print('for Cholesterol enter High:0, BorderLine:1, Normal:2')

q = HeartDisease_infer.query(variables=['heartdisease'], evidence={
    'age': int(input('Enter Age: ')),
    'Gender': int(input('Enter Gender: ')),
    'Family': int(input('Enter Family History: ')),
    'diet': int(input('Enter Diet: ')),
    'Lifestyle': int(input('Enter Lifestyle: ')),
    'cholestrol': int(input('Enter Cholestrol: '))
})
```

```
print(q)
```

OUTPUT:

```
g: BayesianModel has been renamed to BayesianNetwork. Please use BayesianNetwork class, BayesianModel will be removed in futur
e.
  warnings.warn(
+--------+----------+
| age(0) | 0.210526 |
+--------+----------+
| age(1) | 0.157895 |
+--------+----------+
| age(2) | 0.157895 |
+--------+----------+
| age(3) | 0.210526 |
+--------+----------+
| age(4) | 0.263158 |
+--------+----------+
For Age enter SuperSeniorCitizen:0, SeniorCitizen:1, MiddleAged:2, Youth:3, Teen:4
For Gender enter Male:0, Female:1
For Family History enter Yes:1, No:0
For Diet enter High:0, Medium:1
for LifeStyle enter Athlete:0, Active:1, Moderate:2, Sedentary:3
for Cholesterol enter High:0, BorderLine:1, Normal:2
Enter Age: 0
Enter Gender: 0
Enter Family History: 1
Enter Diet: 1
Enter Lifestyle: 3
Enter Cholestrol: 0
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.0000 |
+-----------------+---------------------+
| heartdisease(1) |              1.0000 |
+-----------------+---------------------+
PS C:\Users\Sudharshanan N\Desktop>
```

5. Implement decision problem for any real-world application

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
from dtreeviz.trees import*
#Visualizing the DataFrame using Pandas
data = pd.read_csv('diabetes.csv')
data.index+=1
data.head()
# Feature extraction
X = data.drop(columns='Outcome')
Y = data['Outcome']
```

```
# Splitting the dataset into the Training set and Test set
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3,random_state
=123)

#Training the Decision Tree Classification model on the training set
clf= DecisionTreeClassifier(criterion='entropy',max_depth=5, random_state=0)
clf.fit(X_train, Y_train)
Y_pred= clf.predict(X_test)
# Checking Accuracy using sklearn.metrics
print("Accuracy: ",accuracy_score(Y_test, Y_pred))
```

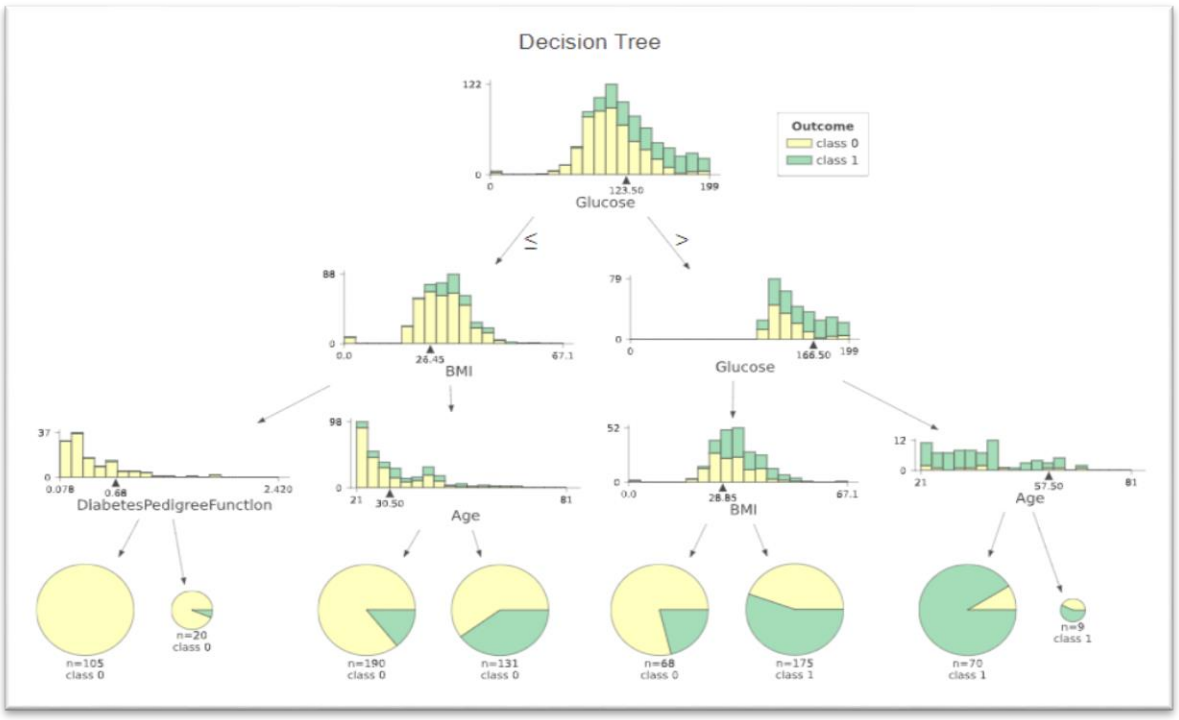
Output: Accuracy: 0.7705627705627706

```
# Decision Tree in Text Representation
feature_cols = ['Pregnancies','Glucose' ,'BloodPressure', 'SkinThickness','Insulin
', 'BMI', 'DiabetesPedigreeFunction','Age']
text_representation = tree.export_text(clf,feature_names=feature_cols)
print(text_representation)
```

```
|--- Glucose <= 123.50
|   |--- BMI <= 26.45
|   |   |--- DiabetesPedigreeFunction <= 0.68
|   |   |   |--- class: 0
|   |   |--- DiabetesPedigreeFunction >  0.68
|   |   |   |--- class: 0
|   |--- BMI >  26.45
|   |   |--- Age <= 30.50
|   |   |   |--- class: 0
|   |   |--- Age >  30.50
|   |   |   |--- class: 0
|--- Glucose >  123.50
|   |--- Glucose <= 166.50
|   |   |--- BMI <= 28.85
|   |   |   |--- class: 0
|   |   |--- BMI >  28.85
|   |   |   |--- class: 1
|   |--- Glucose >  166.50
|   |   |--- Age <= 57.50
|   |   |   |--- class: 1
|   |   |--- Age >  57.50
|   |   |   |--- class: 1
```

```
# Visualizing Decision Tree using dtreeviz
```

```
viz = dtreeviz(clf,X,Y,target_name='Outcome',feature_names=feature_cols,title="Dec
ision Tree")
viz
```



3. Write a program to implement the naïve Bayesian classifier for a sample dataset and compute the accuracy

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,ConfusionMatrixDisplay
from sklearn.naive_bayes import GaussianNB
data = pd.read_csv("D:\Downloads\spam.csv")
data.index+=1
data.head()
# Feature Extraction
X = data['EmailText'].values
y = data['Label'].values
# coversion to lower case and removal of stop words using TFIDF VECTORIZER
tfvec=TfidfVectorizer(stop_words='english')
X=tfvec.fit_transform(X).toarray()
#Splitting the dataset into the Training set and Test set
```

```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2,
random_state=0)
#Training the SVM Classification model on the training set
clf = GaussianNB().fit(X_train,y_train)
y_pred = clf.predict(X_test)
Accuracy = accuracy_score(y_test, y_pred)
print("ACCURACY: ",Accuracy)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Copyright (C) Microsoft Corporation. All rights r

Install the latest PowerShell for new features an

PS C:\Users\Sudharshanan N\Desktop>  & 'C:\Users\
udharshanan N\.vscode\extensions\ms-python.python
03' '--' 'C:\Users\Sudharshanan N\Desktop\Ai.py'
ACCURACY:  0.8762331838565023
PS C:\Users\Sudharshanan N\Desktop>
```

6. Implement approximate inferences in Bayesian network.

Approximate Inference is done by using Rejection Sampling technique.

Rejection Sampling

1. **It Generates samples from the prior distribution (conditional pr) specified by the network**
2. **Then, it rejects all those do not match evidence**

DrawBack

It rejects so many samples….

```
from pgmpy.models import BayesianNetwork
from pgmpy.factors.discrete import TabularCPD
from pgmpy.factors.discrete import State
from pgmpy.sampling import BayesianModelSampling
student = BayesianNetwork([('diff', 'grade'), ('intel', 'grade')])
cpd_d = TabularCPD('diff', 2, [[0.6], [0.4]])
cpd_i = TabularCPD('intel', 2, [[0.7], [0.3]])
cpd_g = TabularCPD('grade', 3, [[0.3, 0.05, 0.9, 0.5], [0.4, 0.25,0.08, 0.3],
[0.3, 0.7, 0.02, 0.2]],
                   ['intel', 'diff'], [2, 2])
student.add_cpds(cpd_d, cpd_i, cpd_g)
inference = BayesianModelSampling(student)
evidence = [State(var='diff', state=0)]
print(inference.rejection_sample(evidence=evidence, size=2))
```

OUTPUT:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Sudharshanan N\Desktop>  & 'C:\Users\Sudharshanan N\AppData\Local\Programs\Python\Python39\python.exe' 'c:\Users\Sudharshanan N\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '56614' '--' 'C:\Users\Sudharshanan N\Desktop\Ai-2.py'
100%|██████████| 2/2 [00:00<00:00, 40.53it/s]
   diff  grade  intel
0     0      0      1
1     0      0      1
PS C:\Users\Sudharshanan N\Desktop>
```

2. Implement Bayesian parameter estimation in a real world problem.

```
import pandas as pd
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import BayesianEstimator
data = pd.DataFrame(data={'A': [0, 0, 1], 'B': [0, 1, 0], 'C': [1, 1, 0]})
model = BayesianNetwork([('A', 'C'), ('B', 'C')])
estimator = BayesianEstimator(model, data)
cpd_C = estimator.estimate_cpd('C', prior_type="dirichlet",
                               pseudo_counts=[[1, 1, 1, 1],
                                              [2, 2, 2, 2]])
print(cpd_C)
```

Output:

```
+--------------+-----------+
PS C:\Users\Sudharshanan N> & "C:/Users/Sudharshanan N/AppData/Local/Programs/
c:/Users/Sudharshanan N/Desktop/Ai.py"
+------+------+------+------+--------------------+
| A    | A(0) | A(0) | A(1) | A(1)               |
+------+------+------+------+--------------------+
| B    | B(0) | B(1) | B(0) | B(1)               |
+------+------+------+------+--------------------+
| C(0) | 0.25 | 0.25 | 0.5  | 0.3333333333333333 |
+------+------+------+------+--------------------+
| C(1) | 0.75 | 0.75 | 0.5  | 0.6666666666666666 |
+------+------+------+------+--------------------+
PS C:\Users\Sudharshanan N>
```