EX.NO: 4 IMPLEMENT BAYESIAN NETWORKS

Aim:

The aim of implementing Bayesian Networks is to model the probabilistic relationships between a set of variables. A Bayesian Network is a graphical model that represents the conditional dependencies between different variables in a probabilistic manner.It is a powerful tool for reasoning under uncertainty and can be used for a wide range of applications, including decision making, risk analysis, and prediction.

Algorithm:

1. Define the variables: The first step in implementing a Bayesian Network is to define the variables that will be used in the model. Each variable should be clearly defined and its possible states should be enumerated.
2. Determine the relationships between variables: The next step is to determine the probabilistic relationships between the variables. This can be done by identifying the causal relationships between the variables or by using data to estimate the conditional probabilities of each variable given its parents.
3. Construct the Bayesian Network: The Bayesian Network can be constructed by representing the variables as nodes in a directed acyclic graph (DAG). The edges between the nodes represent the conditional dependencies between the variables.
4. Assign probabilities to the variables: Once the structure of the Bayesian Network has been defined, the probabilities of each variable must be assigned. This can be done by using expert knowledge, data, or a combination of both.
5. Inference: Inference refers to the process of using the Bayesian Network to make predictions or draw conclusions. This can be done by using various inference algorithms, such as variable elimination or belief propagation.
6. Learning: Learning refers to the process of updating the probabilities in the Bayesian Network based on new data. This can be done using various learning algorithms, such as maximum likelihood or Bayesian learning.
7. Evaluation: The final step in implementing a Bayesian Network is to evaluate its performance. This can be done by comparing the predictions of the model to actual data and computing various performance metrics, such as accuracy or precision.

Program:

```
Import numpy as np
import csv
import pandas as pd
from pgmpy.models import BayesianModel
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

#read Cleveland Heart Disease data heartDisease
= pd.read_csv('heart.csv') heartDisease =
heartDisease.replace('?',np.nan)

#display the data
print('Few examples from the dataset are given below')
print(heartDisease.head())

#Model Bayesian Network
Model=BayesianModel([('age','trestbps'),('age','fbs'),
('sex','trestbps'),('exang','trestbps'),('trestbps','heartdise
ase'),('fbs','heartdisease'),('heartdisease','restecg'),
('heartdisease','thalach'),('heartdisease','chol')])

#Learning CPDs using Maximum Likelihood Estimators
print('\n Learning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

# Inferencing with Bayesian Network
print('\n Inferencing with Bayesian Network:')
HeartDisease_infer = VariableElimination(model)

#computing the Probability of HeartDisease given Age
print('\n 1. Probability of HeartDisease given Age=30')
q=HeartDisease_infer.query(variables=['heartdisease'],evidence
={'age':28})
print(q['heartdisease'])

#computing the Probability of HeartDisease given cholesterol
print('\n 2. Probability of HeartDisease given cholesterol=100')
q=HeartDisease_infer.query(variables=['heartdisease'],evidence
={'chol':100})
print(q['heartdisease'])
```

Output:

age sex cptrestbps ...slope cathalheartdisease0
63 1 1 145 ... 3 0 6 0
1 67 1 4 160 ... 2 3 3 2
2 67 1 4 120 ... 2 2 7 1
3 37 1 3 130 ... 3 0 3 0
4 41 0 2 130 ... 1 0 3 0
[5 rows x 14 columns]
Learning CPD using Maximum likelihood estimators
Inferencing with Bayesian Network:
1. Probability of HeartDisease given Age=28

heartdisease	phi(heartdisease)
heartdisease_0	0.6791
heartdisease_1	0.1212
heartdisease_2	0.0810
heartdisease_3	0.0939
heartdisease_4	0.0247

2. Probability of HeartDisease given cholesterol=100

heartdisease	phi(heartdisease)
heartdisease_0	0.5400
heartdisease_1	0.1533
heartdisease_2	0.1303
heartdisease_3	0.1259
heartdisease_4	0.0506

Result:

Thus the program is executed successfully and output is verified.

EX.NO: 5 BUILD REGRESSION MODEL

Aim:

The aim of building a regression model is to predict a continuous numerical outcome variable based on one or more input variables. There are several algorithms that can be usedto build regression models, including linear regression, polynomial regression, decision trees,random forests, and neural networks.

Algorithm:

1. Collecting and cleaning the data: The first step in building a regression model is to gather the data needed for analysis and ensure that it is clean and consistent. This may involve removing missing values, outliers, and other errors.
2. Exploring the data: Once the data is cleaned, it is important to explore it to gain an understanding of the relationships between the input and outcome variables. This may involve calculating summary statistics, creating visualizations, and testing for correlations.
3. Choosing the algorithm: Based on the nature of the problem and the characteristics of the data, an appropriate regression algorithm is chosen.
4. Preprocessing the data: Before applying the regression algorithm, it may be necessary to preprocess the data to ensure that it is in a suitable format. This may involve standardizing or normalizing the data, encoding categorical variables, or applying feature engineering techniques.
5. Training the model: The regression model is trained on a subset of the data, using an optimization algorithm to find the values of the model parameters that minimize the difference between the predicted and actual values.
6. Evaluating the model: Once the model is trained, it is evaluated using a separate test dataset to determine its accuracy and generalization performance. Metrics such as mean squared error, R-squared, or root mean squared error can be used to assess the model's performance.
7. Improving the model: Based on the evaluation results, the model can be refined by adjusting the model parameters or using different algorithms.
8. Deploying the model: Finally, the model can be deployed to make predictions on new data.

Program:

```
import pandas as pd
importnumpy as np
importmatplotlib.pyplot as plt
fromsklearn.linear_model import LinearRegression
fromsklearn.model_selection import train_test_split
fromsklearn.metrics import mean_squared_error, r2_score

# Load the dataset
df = pd.read_csv('dataset.csv')

# Split the dataset into training and testing setsX
= df[['feature1', 'feature2', ...]]
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the regression model
reg = LinearRegression()
reg.fit(X_train, y_train)

# Make predictions on the test set
y_pred = reg.predict(X_test)

# Evaluate the model
print('Mean squared error: %.2f' % mean_squared_error(y_test, y_pred))
print('Coefficient of determination: %.2f' % r2_score(y_test, y_pred))

# Plot the results
plt.scatter(X_test['feature1'], y_test, color='black')
plt.plot(X_test['feature1'], y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

Output:

```
Coefficients: [ 0.19246454 -0.07720843  0.02463994]
Mean squared error: 18.10
Coefficient of determination: 0.87
```

Result:

Thus the program for build regression models is executed successfully and output is verified.

.