13. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set.

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

# Read Cleveland Heart Disease data
heartDisease = pd.read_csv('C:/Users/kamle/Downloads/ML/prac8/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', 'heartdisease'),('fbs', 'heartdisease'), ('heartdisease', 'restecg'),('heartdisease', 'thalach'),
('heartdisease', 'chol')])
#estimator = MaximumLikelihoodEstimator(model, data)
# Learning CPDs using Maximum Likelihood Estimators
print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)

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

# Computing the Probability of HeartDisease given Age
print('\n1. Probability of HeartDisease given Age=28')
```

```
q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'age': 28})

print(q['heartdisease'])


# Computing the Probability of HeartDisease given cholesterol

print('\n2. Probability of HeartDisease given cholesterol=100')

q = HeartDisease_infer.query(variables=['heartdisease'], evidence={'chol': 100})

print(q['heartdisease'])
```

14. Write a program to implement Support Vector Machines (SVM) and Principal Component Analysis (PCA)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Apply PCA to reduce the dimensionality of the dataset
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

# Train an SVM classifier on the reduced dataset
svm = SVC(kernel='linear')
svm.fit(X_train_pca, y_train)

# Make predictions on the test set
y_pred = svm.predict(X_t
```

15. Write a Program to implement Principle Component Analysis.

```
import numpy as np

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt


# Sample data: 5 samples with 3 features

data = np.array([

    [2.5, 2.4, 1.5],

    [0.5, 0.7, 0.8],

    [2.2, 2.9, 2.1],

    [1.9, 2.2, 1.8],

    [3.1, 3.0, 2.9]

])


# Standardize the data (mean = 0, variance = 1)

data_mean = np.mean(data, axis=0)

data_std = np.std(data, axis=0)

standardized_data = (data - data_mean) / data_std


# Apply PCA

pca = PCA(n_components=2)  # Reduce to 2 dimensions

principal_components = pca.fit_transform(standardized_data)


# Print the principal components

print("Principal Components:\n", principal_components)


# Explained variance

print("Explained Variance Ratio:\n", pca.explained_variance_ratio_)


# Plotting the principal components

plt.scatter(principal_components[:, 0], principal_components[:, 1], c='blue', marker='o')
plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA of Sample Data')

plt.show()
```