Group Name: **Agile**

Github link: https://github.com/M-Shreya/AI_Lab_Assignments

Lab 4

Problem Statement:

Case of Medical Diagnosis

Thyroid Disease Dataset from UCI Machine Learning Repository

Dataset Description: https://archive.ics.uci.edu/dataset/102/thyroid+disease

Features: T3,T4,TSH(thyroid stimulating hormone), Goiter presence, etc.

Labels: Hyperthyroid, Hypothyroid, Normal Thyroid Function Build and train a Bayesian Network as a classifier (labels)

Expected accuracy >=85%

Solution:

To implement a Bayesian Network classifier for the thyroid disease dataset from the UCI Machine Learning Repository, you will need to:

1. Download and prepare the dataset.
2. Preprocess the data (if needed).
3. Build and train a Bayesian Network classifier.
4. Evaluate the model's accuracy.

Below is an implementation using Python with the pgmpy library to create a Bayesian Network. If pgmpy is not already installed, you can install it using:

bash

pip install pgmpy pandas scikit-learn

Implementation:

python

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

# Load the dataset
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/thyroid-disease/new-thyroid.data'
column_names = ['Class', 'T3', 'T4', 'TSH']
data = pd.read_csv(url, names=column_names, delim_whitespace=True)

# Convert class labels to string for clarity
class_map = {1: 'Hyperthyroid', 2: 'Hypothyroid', 3: 'Normal'}
data['Class'] = data['Class'].map(class_map)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(data.drop(columns=['Class']), data['Class'],
test_size=0.3, random_state=42)

# Add labels to training set
X_train['Class'] = y_train
```

```
# Define the structure of the Bayesian Network
# You might want to adjust this according to the domain knowledge
model = BayesianNetwork([('T3', 'Class'), ('T4', 'Class'), ('TSH', 'Class')])

# Fit the model using Maximum Likelihood Estimation
model.fit(X_train, estimator=MaximumLikelihoodEstimator)

# Inference
inference = VariableElimination(model)

# Predict function using the trained model
def predict(model, X_test):
    y_pred = []
    for _, sample in X_test.iterrows():
        query = model.map_query(variables=['Class'], evidence=sample.to_dict())
        y_pred.append(query['Class'])
    return y_pred

# Perform predictions
y_pred = predict(inference, X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Expected output should show an accuracy >= 85%
```

Summary:

1. *Dataset*: This code uses the new-thyroid.data subset from the UCI Thyroid Disease dataset. The url should point to the correct dataset file. Ensure that you adjust the file path if you use a different subset.

2. *Model Structure*: The Bayesian Network's structure is defined as BayesianNetwork([('T3', 'Class'), ('T4', 'Class'), ('TSH', 'Class')]). You might need to adjust this according to the domain knowledge.

3. *Dependencies*: Make sure to have the pgmpy, pandas, and scikit-learn libraries installed in your environment.

4. *Evaluation*: This code prints the accuracy of the model. Ensure the accuracy is above 85% by possibly tweaking the model structure and features.

If the accuracy is lower than expected, you may need to explore other structures for the Bayesian Network or perform feature engineering on the dataset to improve performance.g