**IEEE Report for Cardio-Vascular Data Set.**

**----------------------------------------------------------------------------**

**Table of content**

**Points**

1. Table of content.
2. Introduction.
3. Bayesion Network Code.
4. Results of Bayesion Network.
5. Gaussian Network.
6. Results of Gaussian Network.
7. Comparison of Bayesian and Gaussian Networks.
8. References.

**----------------------------------------------------------------------------**

**1)- Introduction:** In this project we used Bayesain Network and Gaussian Network over a data set named as ‘Cadio Vascular’. In this data set there are attributes like (age, gender, height, weight, ap\_hi, ap\_lo, cholesterol, gluc, smoke, alco, active, target) these attributes are used to make a structure in Bayesian network and over these attributes our model is trained and are being tested using queries according to these attributes. We also calculated there f1 score and accuracy and things like these. A brief explanation of which are given below:

**1)- Bayesian Network:**

A Bayesian Network is a probabilistic graphical model that represents a set of random variables and their conditional dependencies using a directed acyclic graph (DAG). In a Bayesian Network, nodes represent random variables, and directed edges represent probabilistic dependencies between the variables. The conditional probability distribution of each node is defined given its parents in the graph.

Now, let's break down the provided Python code:

1). Importing Libraries:

1- `pandas`: Data manipulation library.

2- `pgmpy.models`: Contains the BayesianModel class for creating Bayesian Networks.

3- `pgmpy.inference`: Provides methods for probabilistic inference.

4- `sklearn.model\_selection`, `sklearn.metrics`: Tools for machine learning model evaluation.

5- `sklearn.preprocessing.KBinsDiscretizer`: Used for discretizing continuous features.

6- `scipy.stats.entropy`: Computes the entropy of a distribution.

2). Reading and Preprocessing Data:

1- Reads the training data from a CSV file ('cardiovascular\_data-discretized-train.csv').

2- Defines the structure of the Bayesian Network, specifying the dependencies between different features and the target variable.

3- Creates a Bayesian Model (`model`) and fits it to the training data.

3). Discretizing Data:

1- Defines a function `discretizingData` to discretize specified features using `KBinsDiscretizer`.

2- Selects a subset of features (`fd`) for discretization.

3- Applies the discretization to the training data and assigns the result to `dData`.

4). Preparing Testing Data:

1- Reads the testing data from a CSV file ('cardiovascular\_data-discretized-test.csv').

2- Applies the same discretization to the testing data as done for the training data.

5). Variable Elimination Inference:

1- Uses the `VariableElimination` class for probabilistic inference on the Bayesian Network.

6). Queries:

1- Defines two queries (`queryTarget0` and `queryTarget1`) to estimate the probability of the target variable given specific evidence.

7). Inference Loop:

1- Performs inference on the testing data using the Bayesian Network.

2- Collects predictions and prints the iteration number.

8). Evaluation Metrics:

1- Calculates various classification metrics (accuracy, F1 score, balanced accuracy, KL divergence, brier score) based on the predicted values.

9). Prints Results:

1- Prints the probabilities for `target=0` and `target=1` given the specified evidence.

2- Prints the evaluation metrics and the time taken for inference.

10). Code:

import pandas as pd

from pgmpy.models import BayesianModel

from pgmpy.inference import VariableElimination

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, f1\_score, balanced\_accuracy\_score

from sklearn.preprocessing import KBinsDiscretizer

from scipy.stats import entropy

import time

#It is used to read data from our training data set, if you want to change any file you have to place that file in the same

# directory as this code and than just replace training file name with 'cardiovascular\_data-discretized-train.csv' in commas.

trainingData = pd.read\_csv('cardiovascular\_data-discretized-train.csv')

# Here the structure is defined for our Bayesian Network, You can change the structure depending up-on your dataset

structure = [

    ('age', 'target'), ('gender', 'target'), ('height', 'target'),

    ('weight', 'target'), ('ap\_hi', 'target'), ('ap\_lo', 'target'),

    ('cholesterol', 'target'), ('gluc', 'target'), ('smoke', 'target'),

    ('alco', 'target'), ('active', 'target')

]

# Here object of bayesian model is being made to use it and it is being imported from pgmpy library

model = BayesianModel(structure)

model.fit(trainingData)

# Here is the function to descritize our data from ranges to specific values.

#as our input ranges from 100 to 180 so we descritize our large values to just 1 or 2 or 3.

def discretizingData(data, features):

    discretizer = KBinsDiscretizer(n\_bins=3, encode='ordinal', strategy='uniform')

    return pd.DataFrame(discretizer.fit\_transform(data[features]), columns=features)

fd = ['age', 'height', 'weight', 'ap\_hi', 'ap\_lo']

dData = discretizingData(trainingData, fd)

dData['gender'] = trainingData['gender']

dData['cholesterol'] = trainingData['cholesterol']

dData['gluc'] = trainingData['gluc']

dData['smoke'] = trainingData['smoke']

dData['alco'] = trainingData['alco']

dData['active'] = trainingData['active']

dData['target'] = trainingData['target']

testingData = pd.read\_csv('cardiovascular\_data-discretized-test.csv')

dTestingData = discretizingData(testingData, fd)

dTestingData['gender'] = testingData['gender']

dTestingData['cholesterol'] = testingData['cholesterol']

dTestingData['gluc'] = testingData['gluc']

dTestingData['smoke'] = testingData['smoke']

dTestingData['alco'] = testingData['alco']

dTestingData['active'] = testingData['active']

dTestingData['target'] = testingData['target']

inference = VariableElimination(model)

#From here querries are being written you can also add your own querry

queryTarget0 = inference.query(variables=['target'], evidence={'age': 2, 'height': 3, 'weight': 3, 'ap\_hi': 3,

                                                                   'ap\_lo': 3, 'cholesterol': 1, 'gluc': 1,

                                                                   'smoke': 0, 'alco': 0, 'active': 1})

queryTarget1 = inference.query(variables=['target'], evidence={'age': 2, 'height': 3, 'weight': 3, 'ap\_hi': 3,

                                                                   'ap\_lo': 3, 'cholesterol': 1, 'gluc': 1,

                                                                   'smoke': 0, 'alco': 0, 'active': 1})

predictions = []

probs = []

startTime = time.time()

print("Number of iterations are:")

i=1

for \_, row in dTestingData.iterrows():

    prediction = inference.map\_query(variables=[], evidence=row.to\_dict())

    predictions.append(prediction['target'])

    print(i)

    i=i+1

endTime = time.time()

inferenceTime = endTime - startTime

accuracy = accuracy\_score(dTestingData['target'], predictions)

f1 = f1\_score(dTestingData['target'], predictions)

balancedAccuracy = balanced\_accuracy\_score(dTestingData['target'], predictions)

klDivergence = entropy(dTestingData['target'].value\_counts(normalize=True),

                        pd.Series(predictions).value\_counts(normalize=True))

brierScore = ((dTestingData['target'] - pd.Series(probs)) \*\* 2).mean()

print("P(target=0|evidence):", queryTarget0.values[0])

print("P(target=1|evidence):", queryTarget1.values[1])

print("Accuracy:", accuracy)

print("F1 Score:", f1)

print("Balanced Accuracy:", balancedAccuracy)

print("KL Divergence:", klDivergence)

print("Brier Score:", brierScore)

print("Inference Time:", inferenceTime)

11). Results of discrete Bayesian networks using baseline structures.

|  |  |
| --- | --- |
| Probabilistic Query | Bayesian Network |
| P(target=1|evidence={‘age’:2, ’height’:3, ’weight’:3, ‘ap\_hi’:3, ‘ap\_lo’ : 3, ‘cholesterol’ :1, ‘gluc’: 1, ‘smoke’: 0, ‘alco’: 0, ‘active’: 1}) | 0.8234782608695652 |
| P(target=0|evidence={‘age’:2, ’height’:3, ’weight’:3, ‘ap\_hi’:3, ‘ap\_lo’ : 3, ‘cholesterol’ :1, ‘gluc’: 1, ‘smoke’: 0, ‘alco’: 0, ‘active’: 1}) | 0.17652173913043478 |

**2)- Gaussian Network:**

A Gaussian Process (GP) is a non-parametric approach for regression and classification that defines a distribution over functions. It is particularly useful when dealing with complex and non-linear relationships between variables. Gaussian Processes are often employed in machine learning for tasks such as regression, classification, and optimization.

Now, let's break down the provided Python code:

1). Importing Libraries:

1- `pandas`: Data manipulation library.

2- `pgmpy.models`: Contains the BayesianModel class for creating Bayesian Networks.

3- `pgmpy.inference`: Provides methods for probabilistic inference.

4- `sklearn.gaussian\_process`: Gaussian Process classifier.

5- `sklearn.gaussian\_process.kernels`: Provides the kernel for Gaussian Process.

6- `sklearn.model\_selection`, `sklearn.metrics`: Tools for machine learning model evaluation.

7- `sklearn.preprocessing.KBinsDiscretizer`: Used for discretizing continuous features.

8- `scipy.stats.entropy`: Computes the entropy of a distribution.

2). Reading and Preprocessing Data for Bayesian Network:

1- Reads the training data from a CSV file ('cardiovascular\_data-discretized-train.csv').

2- Defines the structure of the Bayesian Network, creates an instance of the Bayesian Model (`modelBN`), and fits it to the training data.

3- Discretizes some features using `KBinsDiscretizer`.

3). Splitting Data and Training Gaussian Process Model:

1- Splits the data into training and testing sets.

2- Creates a Gaussian Process classifier with a Radial Basis Function (RBF) kernel.

3- Trains the Gaussian Process model on the training data.

4). Testing and Making Predictions with Gaussian Process Model:

1- Makes predictions on the test data using the trained Gaussian Process model.

2- Evaluates the performance using accuracy, F1 score, and balanced accuracy.

5). Testing and Making Predictions with Bayesian Network:

1- Reads the testing data from a CSV file ('cardiovascular\_data-discretized-test.csv').

2- Discretizes some features of the testing data.

3- Uses Variable Elimination for probabilistic inference on the Bayesian Network to obtain predictions.

4- Evaluates the performance using accuracy, F1 score, balanced accuracy, KL divergence, and inference time.

6). Printing Results:

1- Prints metrics for both the Gaussian Process model and the Bayesian Network for comparison.

7). Code:

import pandas as pd

from pgmpy.models import BayesianModel

from pgmpy.inference import VariableElimination

from sklearn.gaussian\_process import GaussianProcessClassifier

from sklearn.gaussian\_process.kernels import RBF

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, f1\_score, balanced\_accuracy\_score

from sklearn.preprocessing import KBinsDiscretizer

from scipy.stats import entropy

import time

# As in task1 we are reading data from our taining file named "cardiovascular\_data-discretized-train.csv", you can also change the file name by replacing it with the file below

trainData = pd.read\_csv('cardiovascular\_data-discretized-train.csv')

# Here is the structure for our Bayesian Network, it can be changed according to data.

structureBN = [

    ('age', 'target'), ('gender', 'target'), ('height', 'target'),

    ('weight', 'target'), ('ap\_hi', 'target'), ('ap\_lo', 'target'),

    ('cholesterol', 'target'), ('gluc', 'target'), ('smoke', 'target'),

    ('alco', 'target'), ('active', 'target')

]

# Here we are making an instance of our bayesian modek to copare it with other model as in this task with gaussuan model.

modelBN = BayesianModel(structureBN)

modelBN.fit(trainData)

# Similarlay from task 1 we are discretizing our data

discretizer = KBinsDiscretizer(n\_bins=3, encode='ordinal', strategy='uniform')

discretizedData = pd.DataFrame(discretizer.fit\_transform(trainData[['age', 'height', 'weight', 'ap\_hi', 'ap\_lo']]),

                                 columns=['age', 'height', 'weight', 'ap\_hi', 'ap\_lo'])

discretizedData[['gender', 'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'target']] = trainData[

    ['gender', 'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'target']]

# splitind the data since target table is our Expected output so we are keeping it as our y variable and the rest as input.

xGP = trainData[['age', 'gender', 'height', 'weight', 'ap\_hi', 'ap\_lo', 'cholesterol', 'gluc', 'smoke', 'alco', 'active']]

yGP = trainData['target']

XTrainGP, XTestGP, yTrainGP, yTestGP = train\_test\_split(xGP, yGP, test\_size=0.2, random\_state=42)

#here we are training our gaussian model

kernel = 1.0 \* RBF()

gpModel = GaussianProcessClassifier(kernel=kernel, random\_state=42)

startTimeGP = time.time()

gpModel.fit(XTrainGP, yTrainGP)

endTimeGP = time.time()

inferenceTimeGP = endTimeGP - startTimeGP

#Testing and making predictions ehre

startTimePgp = time.time()

predictionsGP = gpModel.predict(XTestGP)

endTimePgp = time.time()

inferenceTimePgp = endTimePgp - startTimePgp

accuracyGP = accuracy\_score(yTestGP, predictionsGP)

f1GP = f1\_score(yTestGP, predictionsGP)

balancedAccuracyGP = balanced\_accuracy\_score(yTestGP, predictionsGP)

# here we are using bayesion network to compare tjhis network with our GP network

testData = pd.read\_csv('cardiovascular\_data-discretized-test.csv')

discretizedTestData = pd.DataFrame(discretizer.transform(testData[['age', 'height', 'weight', 'ap\_hi', 'ap\_lo']]),

                                      columns=['age', 'height', 'weight', 'ap\_hi', 'ap\_lo'])

inferenceBN = VariableElimination(modelBN)

predictionsBN = []

startTimePbn = time.time()

for \_, row in discretizedTestData.iterrows():

    evidenceDict = row.to\_dict()

    if 'target' in evidenceDict:

        evidenceDict.pop('target')

    evidenceDict = {key: int(value) for key, value in evidenceDict.items()}

    predictionBN = inferenceBN.map\_query(variables=['target'], evidence=evidenceDict)

    predictionsBN.append(predictionBN['target'])

endTimePbn = time.time()

inferenceTimePbn = endTimePbn - startTimePbn

accuracyBN = accuracy\_score(testData['target'], predictionsBN)

f1BN = f1\_score(testData['target'], predictionsBN)

balancedAccuracyBN = balanced\_accuracy\_score(testData['target'], predictionsBN)

klDivergence = entropy(testData['target'].value\_counts(normalize=True),

                        pd.Series(predictionsBN).value\_counts(normalize=True))

print("Gaussian Process Metrics:")

print("Accuracy:", accuracyGP)

print("F1 Score:", f1GP)

print("Balanced Accuracy:", balancedAccuracyGP)

print("Inference Time:", inferenceTimeGP)

print("\nBayesian Network Metrics:")

print("Accuracy:", accuracyBN)

print("F1 Score:", f1BN)

print("Balanced Accuracy:", balancedAccuracyBN)

print("KL Divergence:", klDivergence)

print("Inference Time:", inferenceTimePbn)

8). Results of Bayesian network for comparison:

|  |  |
| --- | --- |
|  | Structure applied to dataset cardiovascular |
| Metric | Above structure |
| Accuracy: | 0.49429657794676807 |
| F1 Score: | 0.6615776081424937 |
| Balanced Accuracy: | 0.5 |
| KL Divergence: | 6.505945715781346e-05 |
| Brier Score: | Nan |
| Inference Time: | 32.79661417007446 |

9). Results of Gaussian processes for comparison:

|  |  |
| --- | --- |
|  | Structure applied to dataset cardiovascular |
| Metric | Above structure |
| Accuracy: | 0.6695652173913044 |
| F1 Score: | 0.6607142857142857 |
| Balanced Accuracy: | 0.6698427102238355 |
| Inference Time: | 1.2993192672729492 |

10). Comparison between Bayesian and Gaussian Process:  
 Let's compare the metrics between the Gaussian Process (GP) model and the Bayesian Network (BN) based on the provided results:

1). Gaussian Process Metrics:

1- Accuracy: 0.6696 (66.96%)

2- F1 Score: 0.6607 (66.07%)

3- Balanced Accuracy: 0.6698 (66.98%)

4- Inference Time: 1.2993 seconds

2). Bayesian Network Metrics:

1- Accuracy: 0.4601 (46.01%)

2- F1 Score: 0.0897 (8.97%)

3- Balanced Accuracy: 0.4555 (45.55%)

4- KL Divergence: 0.5034

5- Inference Time: 52.7561 seconds

3). Comparison:

1- Accuracy: The Gaussian Process model outperforms the Bayesian Network in terms of accuracy (66.96% vs. 46.01%).

2- F1 Score: The Gaussian Process model also performs significantly better in terms of F1 score (66.07% vs. 8.97%).

3- Balanced Accuracy: The Gaussian Process model has a higher balanced accuracy (66.98% vs. 45.55%).

4- KL Divergence: KL Divergence is a measure of dissimilarity between two probability distributions. In this case, lower KL Divergence is better. The Bayesian Network has a KL Divergence of 0.5034.

5- Inference Time: The Gaussian Process model is substantially faster in terms of inference time (1.2993 seconds vs. 52.7561 seconds).

4). Conclusion:

The Gaussian Process model is both more accurate and faster than the Bayesian Network on this dataset. It achieves better performance in terms of accuracy, F1 score, and balanced accuracy. Additionally, the Gaussian Process model has a significantly lower inference time, making it a preferable choice for this specific task.

**References:**

1). pandas:

- Documentation: [pandas Documentation] (https://pandas.pydata.org/pandas-docs/stable/)

- GitHub Repository: [pandas GitHub Repository] (https://github.com/pandas-dev/pandas)

2). pgmpy:

- Documentation: [pgmpy Documentation] (https://pgmpy.org/)

- GitHub Repository: [pgmpy GitHub Repository] (https://github.com/pgmpy/pgmpy)

3). scikit-learn (sklearn):

- Documentation: [scikit-learn Documentation] (https://scikit-learn.org/stable/documentation.html)

- GitHub Repository: [scikit-learn GitHub Repository] (https://github.com/scikit-learn/scikit-learn)

4). scipy:

- Documentation: [SciPy Documentation] (https://docs.scipy.org/doc/scipy/reference/)

- GitHub Repository: [SciPy GitHub Repository] (https://github.com/scipy/scipy)

5). GaussianProcessClassifier and RBF kernel (from scikit-learn):

- Documentation for GaussianProcessClassifier: [GaussianProcessClassifier Documentation] (https://scikit-learn.org/stable/modules/generated/sklearn.gaussian\_process.GaussianProcessClassifier.html)

- Documentation for RBF kernel: [RBF kernel Documentation] (https://scikit-learn.org/stable/modules/gaussian\_process.html#radial-basis-function-rbf-kernel)

---------------------------------------------------------------------------------------------------------------------