Bayesian Reasoning and Learning Assignment 2

Ning Wei Zhou (852757814)

Open University of the Netherlands

1 Background

Statistical uncertainty arises from finite sampling variability, missing data and measurement noise, and can severely bias downstream inference if not properly acknowledged. Conventional analysis pipelines often treat point estimates as exact quantities, overlooking how minor data imperfections propagate through complex model dependencies and yield overconfident or misleading conclusions. A robust inferential framework must therefore estimate parameters in tandem with a rigorous quantification and propagation of uncertainty across all interconnected variables.

Bayesian networks furnish such a framework by representing random variables as nodes in a directed acyclic graph and encoding their relationships via local conditional probability distributions Meekes et al. [2015]. Exploiting conditional-independence properties, these models provide a compact factorization of high-dimensional joint distributions, render exact or approximate inference tractable, and support systematic belief updating as new evidence becomes available. In this paper, we undertake a comprehensive evaluation of constraint-based and score-based structure learning algorithms for Bayesian networks under varying conditions.

2 Research Questions

In the first stage of our study, we specify a simple Bayesian network and generate synthetic datasets of size $n \in \{100, 500, 1000, \}$ by sampling from its conditional distributions. We then apply the Mutual Information–based Inference of Causality (MIIC) constraint-based algorithm and a Greedy Hill–Climbing (GHC) search to recover both the network's structure and its parameters. Finally, we evaluate each method's ability to reconstruct the original graph topology from the simulated data.

In the second stage, we apply both algorithms to a real-world breast cancer dataset, assessing their skill at modeling complex dependencies and calibrating uncertainty. We quantify structural accuracy with the Structural Hamming Distance (SHD) and evaluate predictive performance using ROC curves and F1 scores. This two-stage framework enables a rigorous comparison of score-based and constraint-based structure-learning methods, motivating the following research questions:

Research Question 1: How do the MIIC constraint-based algorithm and the GHC score-based algorithm compare in their ability to recover the true structure and local parameter values of the handcrafted Bayesian network from synthetic data?

Research Question 2: How effectively does the GHC algorithm recover the dependency structure of a Bayesian network learned from breast cancer data, as measured by the SHD relative to a manually constructed reference network?

Research Question 3: How do the manually constructed and greedily learned Bayesian network models compare, when the "Breast Cancer" variable is dichotomized into "no cancer" versus "cancer", in terms of classification performance measured by the area under the ROC curve?

Research Question 4: How does incorporating expert-derived structural constraints into the GHC learning process influence the learned network's alignment with the manually constructed reference and its binary classification performance (AUC) on the breast cancer dataset?

3 Framework

A *Bayesian network* is a graphical model that encodes a joint probability distribution over a collection of n random variables by means of a directed acyclic graph (DAG). Formally, a Bayesian network is

$$G = (V, E) \quad \text{with} \quad V = \{X_1, X_2, \ldots, X_n\}, \quad E \subseteq V \times V$$

Where each node $X_i \in V$ corresponds to a random variable, and each directed edge $(X_j \to X_i) \in E$ indicates that X_j is a *parent* of X_i. The acyclicity of G ensures that no variable can be an ancestor of itself. The defining property of a Bayesian network is the factorization of its joint distribution according to the graph structure. Let $\mathrm{Pa}(X_i)$ denote the set of parents of X_i in G; then the joint distribution over $X = (X_1, \ldots, X_n)$ decomposes as

$$P(X_1, X_2, \ldots, X_n) \;=\; \prod_{i=1}^{n} P\big(X_i \;\big|\; \mathrm{Pa}(X_i)\big). \tag{1}$$

Within the extant literature, two principal families of methods have been proposed for learning Bayesian network structure from data. Constraint-based approaches reconstruct the network by conducting conditional-independence tests among variables or by integrating expert domain knowledge, whereas score-based approaches recast structure learning as an optimization problem in which candidate graphs are evaluated according to a likelihood-driven score Beretta et al. [2017].

Under the experimental conditions outlined above, we first constructed a manual Bayesian network to model students' academic performance and generated corresponding datasets by sampling from its conditional distributions. The inferred structure is depicted in Figure 1. In parallel, we display the provided manual Bayesian Network, resulting in the network shown in Figure 2. The complete source code for this project is presented in Appendix A.

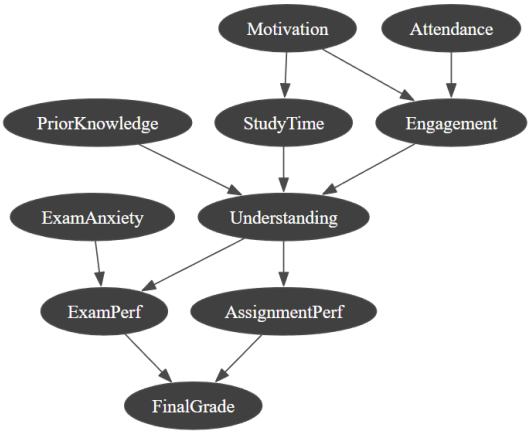

Figure 1: Bayesian network for academic performance

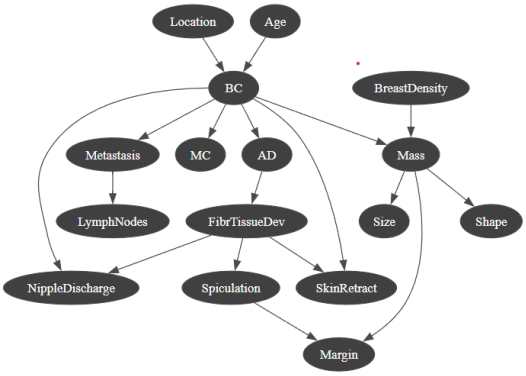

Figure 2: Bayesian network for breast cancer development

To assess the fidelity of the inferred networks, we employ two complementary distance measures that capture discrepancies in both edge existence and orientation. Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two Bayesian networks on the same node set V. We will measure their dissimilarity in two ways.

3.1 Hamming distance

Define the *skeleton* of a DAG G as the undirected edge set

$$\mathrm{Skel}(G) \;=\; \big\{\{i, j\} \subset V : (i \to j) \in E(G) \;\vee\; (j \to i) \in E(G)\big\}.$$

Then the Hamming distance is simply the size of the symmetric difference of the skeletons:

$$d_H(G_1, G_2) \;=\; \big|\,\mathrm{Skel}(G_1) \,\Delta\, \mathrm{Skel}(G_2)\big|. \tag{2}$$

This counts how many (undirected) edges appear in one network but not the other.

3.2 Structural Hamming distance

The Structural Hamming distance (SHD) further penalizes orientation mismatches. Equivalently, it is the minimum number of atomic operations, edge insertion, deletion, or reversal, required to transform G_1 into G_2:

$$\mathrm{SHD}(G_1, G_2) \;=\; \min\bigl\{\#\text{adds} + \#\text{deletes} + \#\text{reversals} \;\bigm|\; G_1 \to G_2\bigr\}. \tag{3}$$

In practice, one often computes SHD by comparing the directed edge sets E_1 and E_2, counting each missing or extra arc as one edit and each reversed arc as one additional edit.

3.3 ROC, AUC and F1

In our breast cancer dataset, we evaluate how well the classifier performs by using three common tools: the ROC curve, the AUC score, and the F_1–score. The ROC curve is a plot that shows how good the model is at telling apart healthy patients from those with cancer. It does this by comparing how often it correctly identifies healthy cases (true positives) versus how often it mistakenly labels cancer cases as healthy (false positives), across different decision settings. The AUC (area under the ROC curve) gives us a single number that summarizes this performance. A higher AUC means the model is better at making the right call. The F_1–score is a way to measure how reliable the model's "healthy" predictions are. It combines two things: how many of those predictions are actually correct (precision), and how many healthy cases the model successfully finds (recall). The F_1–score balances these two to give a clearer picture of overall accuracy. The original diagnostic label

$$Y \;\in\; \{\text{No, InSitu, Invasive}\}$$

comprises three mutually exclusive categories. To focus exclusively on the "No" outcome, we define the one-vs-rest indicator

$$Y_{\mathrm{No}}(x) = \begin{cases} 1, & Y(x) = \text{No}, \\ 0, & Y(x) \in \{\text{InSitu, Invasive}\}. \end{cases}$$

All performance metrics, ROC/AUC and F_1, are computed with $Y_{\mathrm{No}} = 1$ as the positive class and $Y_{\mathrm{No}} = 0$ as the negative class.

Since any error in predicting "No" necessarily corresponds to assigning an instance to one of the cancer subtypes (and conversely), the entries for the binary indicator Y_{No} completely characterize all misclassifications among the three original categories. For the precise mathematical formulations of the AUC and F_1–score, the reader is referred to Appendix C.

4 Results

Table 1 summarizes the Hamming distance (HD) and structural Hamming distance (SHD) obtained by applying Greedy Hill Climbing (GHC) and MIIC to the Academic Performance Bayesian network across sample sizes of 100, 500, and 1000. As sample size increases, GHC's SHD rises from 5 to 12, reflecting a degradation in its ability to recover the true DAG. In contrast, MIIC achieves zero Hamming distance for datasets of 500 and 1000 observations and maintains consistently lower SHD values, demonstrating superior scalability and robustness in reconstructing the underlying structure.

In the context of the Breast Cancer network, we first applied the greedy hill-climbing (GHC) algorithm to learn an unconstrained Bayesian network, then constructed a Guided Bayesian Network by enforcing directed arcs from *Location* and *Age* into the *BC* node, and finally initialized the GHC search with the expert-specified model to obtain the Expert-Guided Bayesian Network; when comparing these three GHC-based variants—the unconstrained learner, the learner augmented with structural constraints, and the learner initialized with the expert model—we observed that all methods achieved an identical Hamming distance of 4, while both structural constraint incorporation and expert initialization reduced the structural Hamming distance (SHD) from 17 to 9, thereby demonstrating the benefit of embedding prior knowledge to guide the search toward the original network topology.

Academic Performance			
Data Size (n)	Algorithm	Hamming (HD)	Structural Hamming (SHD)
100	GHC	3	5
100	MIIC	3	9
500	GHC	2	11
500	MIIC	0	6
1000	GHC	3	12
1000	MIIC	0	7
Breast Cancer			
Network Variant	Algorithm	Hamming (HD)	Structural Hamming (SHD)
GHC Learned	GHC	4	17
GHC with Constraints	GHC	4	9
Expert-initialized GHC	GHC	4	9

Table 1: Evaluation of GHC and MIIC on the Academic Performance network and of GHC variants on the Breast Cancer network.

4.1 Graph Difference Visualization

Figure 3 illustrates the structural differences between the expert-provided Bayesian network and the data-driven network learned for breast cancer classification. To facilitate interpretation, each edge in the learned model is assigned one of four mutually exclusive categories, as defined below:

Correct: An arc present in both the expert and learned networks with the same orientation.
Reversed: An arc whose orientation in the learned network is opposite to that specified in the expert model.
Missing: An arc included in the expert network but not recovered by the learning algorithm.
Overflow: An additional arc introduced by the learning algorithm that does not appear in the expert network.

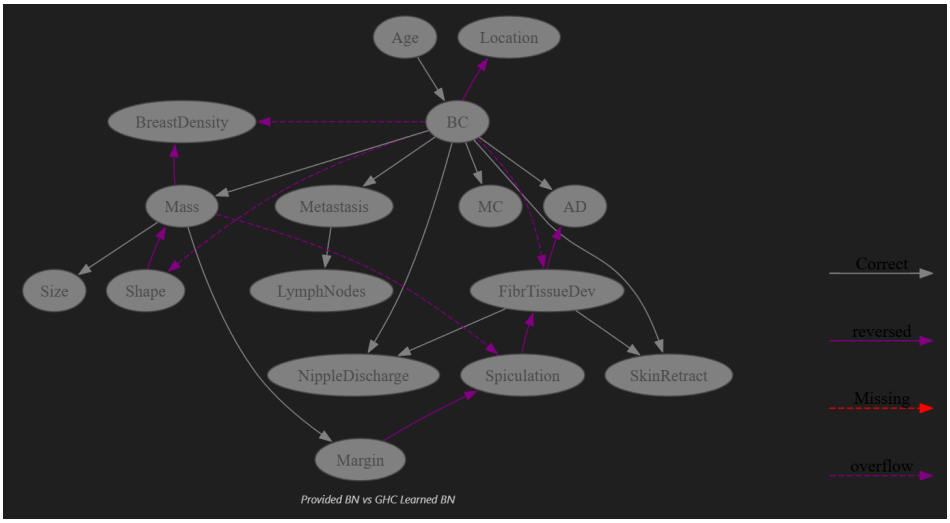

Figure 3: Visual comparison of structural differences between the expert-provided and learned Bayesian networks for breast cancer classification. Edges are colored according to the categories defined in Section 4.1.

4.2 Breast Cancer Evaluation

To assess each Breast Cancer model's ability to predict the "No" breast cancer outcome, we computed the ROC, AUC and the F1-score. Figures 4–7 display the ROC curves for all four approaches, and summarizes their AUC and F1-score. The incorporation of structural constraints yielded systematic

improvements in AUC performance. Notably, when structure learning was initialized with the expert-provided network, the resulting Expert Guided Bayesian Network surpassed the original expert model in discriminatory power.

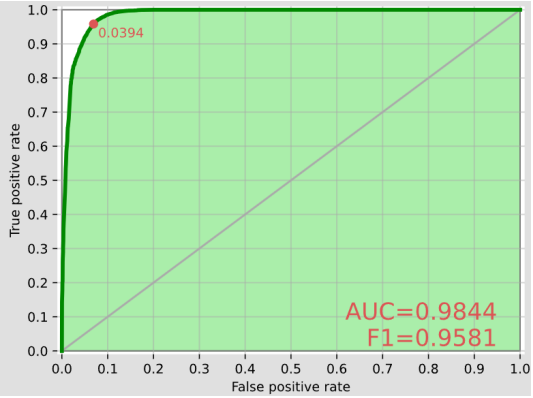

Figure 4: Provided Bayesian Network

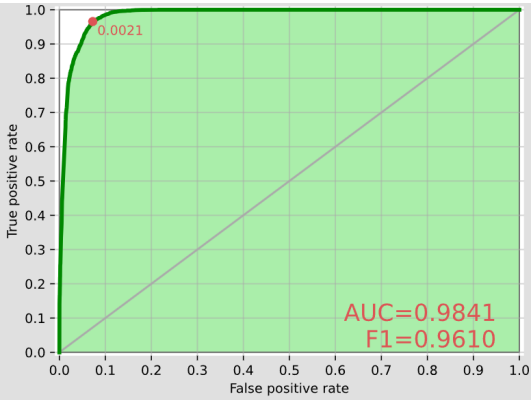

Figure 5: GHC Learned Bayesian Network

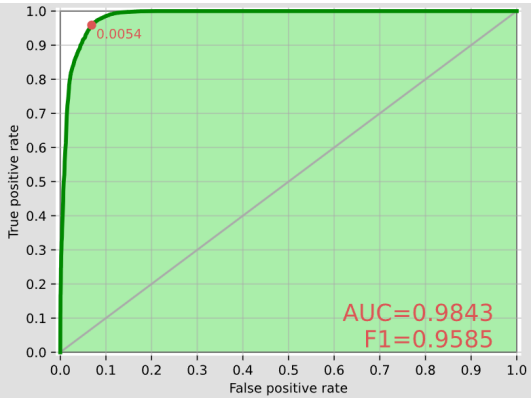

Figure 6: Guided Bayesian Network

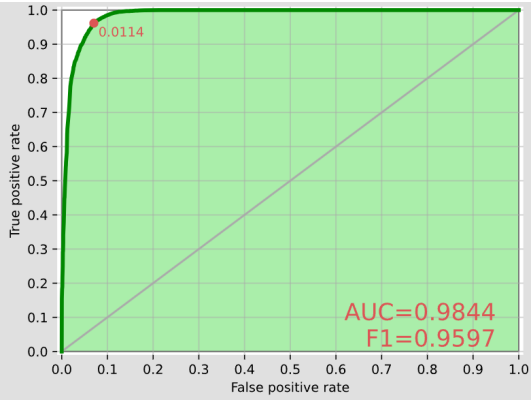

Figure 7: Expert Guided Bayesian Network

4.3 Conclusion

The comparative analysis shows that MIIC consistently produces network topologies closer to the true model than GHC. As sample size increases, MIIC maintains structural fidelity, while GHC diverges, indicating MIIC's robustness against overfitting.

Unconstrained GHC yields a Hamming distance of 4 and a SHD of 17. When initialized with expert-provided arcs or structural constraints, SHD drops to 9, demonstrating the value of domain knowledge in improving edge orientation and reducing false positives.

In predictive terms, the expert network performs strongly, and the GHC-learned model closely matches its discrimination while slightly improving precision–recall balance. Seeding GHC with the expert network preserves top AUC and further boosts performance, while applying only structural constraints yields intermediate results.

Overall, these findings support a hybrid approach that blends expert input with algorithmic learning to produce Bayesian networks that are both structurally accurate and predictively robust.

5 Discussion

This study was inherently limited by the scope of the classroom assignment, which constrained our investigation to a set of predefined research questions. Consequently, ROC curves, AUC metrics, and F_1-scores were not computed for the synthetic dataset, and validation of the breast cancer dataset was restricted to a narrow subset of evaluation metrics and experimental configurations.

The coursework framework also precluded the exploration of alternative model variants and more comprehensive evaluation protocols, any of which might have yielded improved performance or deeper insights. Future work should address these limitations by increasing sample size, applying dimensionality-reduction techniques, employing rigorous train–test splits or cross-validation procedures, and evaluating a broader range of modeling approaches to bolster predictive accuracy and robustness. Additionally, rather than focusing solely on the highest F_1-score, separate evaluation of precision and recall should be considered, with an emphasis on achieving higher recall to reduce false negatives in medical applications.

6 Appendix

A Python Codes

```
import numpy as np
import pandas as pd
import pyagrum as gum
import pyagrum.lib.notebook as gnb
import pyagrum.causal as csl
import pyagrum.causal.notebook as cslnb
import pyagrum.lib.bn_vs_bn as gcm
from pyagrum.lib.bn2roc import showROC, showPR, showROC_PR
import itertools
import networkx as nx
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# 1. Create the Bayesian network
bn = gum.BayesNet("StudentPerformance")

# 2. Add nodes (0 = low/poor, 1 = high/good), except FinalGrade has 4 states
for name, states in [
    ("Motivation", 2),
    ("PriorKnowledge", 2),
    ("StudyTime", 2),
    ("Attendance", 2),
    ("Engagement", 2),
    ("Understanding", 2),
    ("AssignmentPerf", 2),
    ("ExamAnxiety", 2),
    ("ExamPerf", 2),
    ("FinalGrade", 4)    # 0=F,1=C,2=B,3=A
]:
    bn.add(name, states)

# 3. Define the DAG structure
arcs = [
    ("Motivation", "StudyTime"),
    ("Motivation", "Engagement"),
    ("PriorKnowledge", "Understanding"),
    ("Attendance", "Engagement"),
    ("StudyTime", "Understanding"),
    ("Engagement", "Understanding"),
    ("Understanding", "AssignmentPerf"),
    ("Understanding", "ExamPerf"),
    ("ExamAnxiety", "ExamPerf"),
    ("AssignmentPerf", "FinalGrade"),
    ("ExamPerf", "FinalGrade")
]
for p, c in arcs:
    bn.addArc(p, c)

# 4. Define CPTs

# 4.1 Root nodes
bn.cpt("Motivation").fillWith([0.4, 0.6])
```

```
bn.cpt("PriorKnowledge").fillWith([0.5, 0.5])
bn.cpt("Attendance").fillWith([0.3, 0.7])
bn.cpt("ExamAnxiety").fillWith([0.6, 0.4])

# 4.2 StudyTime | Motivation
cpt = bn.cpt("StudyTime")
cpt[{"Motivation": 0, "StudyTime": 1}] = 0.3
cpt[{"Motivation": 0, "StudyTime": 0}] = 0.7
cpt[{"Motivation": 1, "StudyTime": 1}] = 0.8
cpt[{"Motivation": 1, "StudyTime": 0}] = 0.2

# 4.3 Engagement | Motivation, Attendance
cpt = bn.cpt("Engagement")
for mot in [0, 1]:
    for att in [0, 1]:
        if mot + att == 0:
            p = 0.2
        elif mot + att == 1:
            p = 0.5
        else:
            p = 0.9
        cpt[{"Motivation": mot, "Attendance": att, "Engagement": 1}] = p
        cpt[{"Motivation": mot, "Attendance": att, "Engagement": 0}] = 1 - p

# 4.4 Understanding | PriorKnowledge, StudyTime, Engagement
cpt = bn.cpt("Understanding")
for pk in [0, 1]:
    for st in [0, 1]:
        for eg in [0, 1]:
            score = pk + st + eg
            p = {0: 0.1, 1: 0.3, 2: 0.7, 3: 0.9}[score]
            cpt[{"PriorKnowledge": pk, "StudyTime": st,
                 "Engagement": eg, "Understanding": 1}] = p
            cpt[{"PriorKnowledge": pk, "StudyTime": st,
                 "Engagement": eg, "Understanding": 0}] = 1 - p

# 4.5 AssignmentPerf | Understanding
cpt = bn.cpt("AssignmentPerf")
cpt[{"Understanding": 0, "AssignmentPerf": 1}] = 0.4
cpt[{"Understanding": 0, "AssignmentPerf": 0}] = 0.6
cpt[{"Understanding": 1, "AssignmentPerf": 1}] = 0.9
cpt[{"Understanding": 1, "AssignmentPerf": 0}] = 0.1

# 4.6 ExamPerf | Understanding, ExamAnxiety
cpt = bn.cpt("ExamPerf")
for und in [0, 1]:
    for anx in [0, 1]:
        if und == 0 and anx == 1:
            p = 0.2
        elif und == 1 and anx == 0:
            p = 0.8
        else:
            p = 0.5
        cpt[{"Understanding": und, "ExamAnxiety": anx, "ExamPerf": 1}] = p
        cpt[{"Understanding": und, "ExamAnxiety": anx, "ExamPerf": 0}] = 1 - p
```

```
# 4.7 FinalGrade | AssignmentPerf, ExamPerf
cpt = bn.cpt("FinalGrade")
for ap in [0,1]:
    for ep in [0,1]:
        if ap==1 and ep==1:
            probs = [0.0,0.1,0.4,0.5]
        elif ap==1:
            probs = [0.0,0.3,0.5,0.2]
        elif ep==1:
            probs = [0.0,0.2,0.4,0.4]
        else:
            probs = [0.5,0.3,0.1,0.1]

        for state, pr in enumerate(probs):
            # use cpt[ap, ep, state] rather than cpt[[ap,ep,state]]
            cpt[ap, ep, state] = pr

# Display interactive SVG in the notebook
gnb.showBN(bn)

# Assuming 'bn' is your Bayesian network
generator = gum.BNDatabaseGenerator(bn)
nr = 10000  #fill in the amount of datasameples you want. We filled in 100, 500, 1000 an
generator.drawSamples(nr)
df = generator.to_pandas()

# Save it to a CSV file
df.to_csv(f'data_sample_{nr}.csv', index=False)

df_MIIC = pd.read_csv("data_sample_1000.csv") #Adjust this number accordingly to 100, 50

# ----- Structure Learning -----
# When learning the structure, simply pass the dataframe to BNLearner.
bn_MIIC = gum.BNLearner(df_MIIC)

# Instruct the learner to use the MIIC algorithm.
bn_MIIC.useMIIC()

# Learn the structure of the BN.
bn_structure_MIIC = bn_MIIC.learnBN()

# Display the interactive graph of the learned Bayesian network
gnb.showBN(bn_structure_MIIC)

# 6. Compute Structural Hamming Distance
cmp = gcm.GraphicalBNComparator(bn, bn_structure_MIIC)
print("hamming:", cmp.hamming())
# gnb.showBNDiff(expert_bn, learned_bn)
gnb.flow.add(gnb.getBNDiff(bn, bn_structure_MIIC, size="8!"), "manual-bn-vs-learned-bn")
gnb.flow.add(gcm.graphDiffLegend())
gnb.flow.display()

df_GHC = pd.read_csv("data_sample_1000.csv") #Adjust this number accordingly to 100, 500

# ----- Structure Learning -----
# When learning the structure, simply pass the dataframe to BNLearner.
```

```
bn_GHC = gum.BNLearner(df_GHC)

# Instruct the learner to use the MIIC algorithm.
bn_GHC.useGreedyHillClimbing()

# Learn the structure of the BN.
bn_structure_GHC = bn_GHC.learnBN()
print("Learned BN Structure:")
print(bn_structure_GHC)

# Display the interactive graph of the learned Bayesian network
gnb.showBN(bn_structure_GHC)

# 6. Compute Structural Hamming Distance
cmp = gcm.GraphicalBNComparator(bn, bn_structure_GHC)
print("hamming:", cmp.hamming())
# gnb.showBNDiff(expert_bn, learned_bn)
gnb.flow.add(gnb.getBNDiff(bn, bn_structure_GHC, size="8!"), "manual bn vs learned bn")
gnb.flow.add(gcm.graphDiffLegend())
gnb.flow.display()

# 1. Load expert BN from bc.net
expert_bn = gum.loadBN("bc.net")

# 2. Read your data
df_med = pd.read_csv("bc.csv")

# 3. Instantiate the learner on your DataFrame
learner = gum.BNLearner(df_med)

# 4. Tell the learner to use Greedy Hill-Climbing
learner.useGreedyHillClimbing()#.useNMLCorrection().useScoreBDeu()

# 5. Trigger the structure learning and get back a BayesianNetwork
learned_bn = learner.learnBN()

# 6. Compute Structural Hamming Distance
cmp = gcm.GraphicalBNComparator(expert_bn, learned_bn)
print("hamming:", cmp.hamming())

gum.loadBN("bc.net")
gnb.showBN(expert_bn)
gnb.showBN(learned_bn)

# gnb.showBNDiff(expert_bn, learned_bn)
gnb.flow.add(gnb.getBNDiff(expert_bn, learned_bn, size="8!"), "Provided BN vs GHC Learned
gnb.flow.add(gcm.graphDiffLegend())

gnb.flow.display()

# showBNDiff itself will add & render the diff
gnb.showBNDiff(
    expert_bn,
    learned_bn,
    size="8!",
)
```

```
showROC(expert_bn, df_med, "BC", "No", show_progress=False)
showROC(learned_bn, df_med, "BC", "No", show_progress=False)
# Create learner from data (file or DataFrame)
learner = gum.BNLearner(df_med)

# Choose algorithm
learner.useGreedyHillClimbing()

## Apply constraints
learner.addMandatoryArc("Location", "BC")
learner.addMandatoryArc("Age", "BC")     # force an arc

# Learn the BN
guided_bn_arc = learner.learnBN()

# Visualise
gnb.show(guided_bn_arc)

# 6. Compute Structural Hamming Distance
cmp = gcm.GraphicalBNComparator(expert_bn, guided_bn_arc)
print("hamming:", cmp.hamming())

# gnb.showBNDiff(expert_bn, learned_bn)
gnb.flow.add(gnb.getBNDiff(expert_bn, guided_bn_arc, size="8!"), "Difference")
gnb.flow.add(gcm.graphDiffLegend())
gnb.flow.display()
showROC(guided_bn_arc, df_med, "BC", "No", show_progress=False)

# Create learner from data (file or DataFrame)
learner = gum.BNLearner(df_med, expert_bn)

# Choose algorithm
learner.useGreedyHillClimbing()

## Apply constraints
# learner.addMandatoryArc("Location", "BC")
# learner.addMandatoryArc("Age", "BC")     # force an arc

# Learn the BN
guided_bn = learner.learnBN()

# Visualise
gnb.show(guided_bn)

# 6. Compute Structural Hamming Distance
cmp = gcm.GraphicalBNComparator(expert_bn, guided_bn_arc)
print("hamming:", cmp.hamming())

# gnb.showBNDiff(expert_bn, learned_bn)
gnb.flow.add(gnb.getBNDiff(expert_bn, guided_bn_arc, size="8!"), "Difference")
gnb.flow.add(gcm.graphDiffLegend())
gnb.flow.display()
showROC(guided_bn, df_med, "BC", "No", show_progress=False)
```

B Bayesian Networks

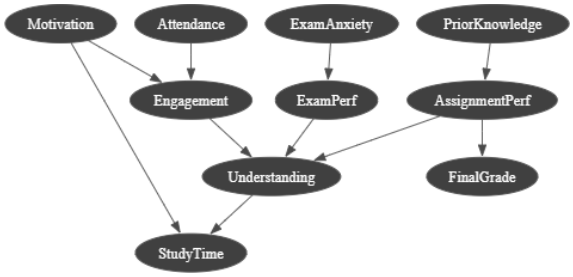

Figure 8: MIIC BN for Academic Performance, $n = 100$

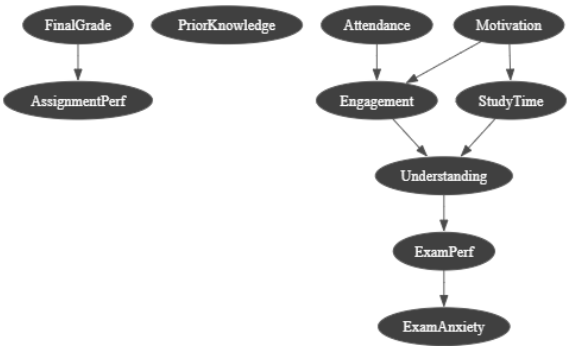

Figure 9: GHC BN for Academic Performance, $n = 100$

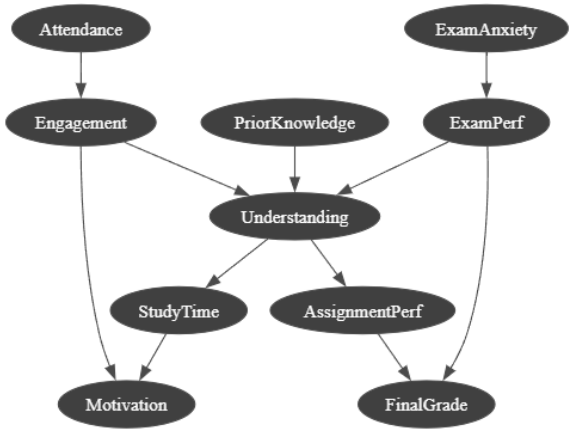

Figure 10: MIIC BN for Academic Performance, $n = 500$

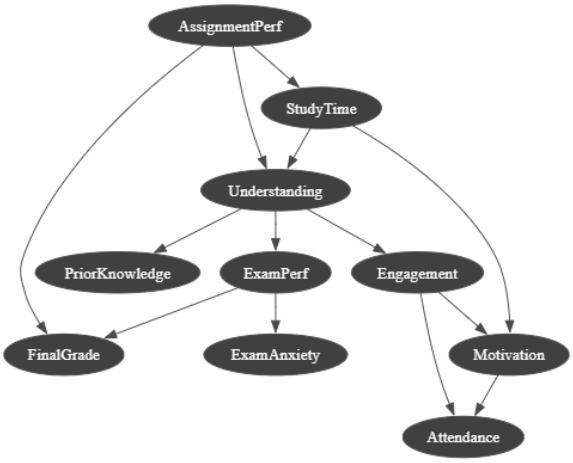

Figure 11: GHC BN for Academic Performance, $n = 500$

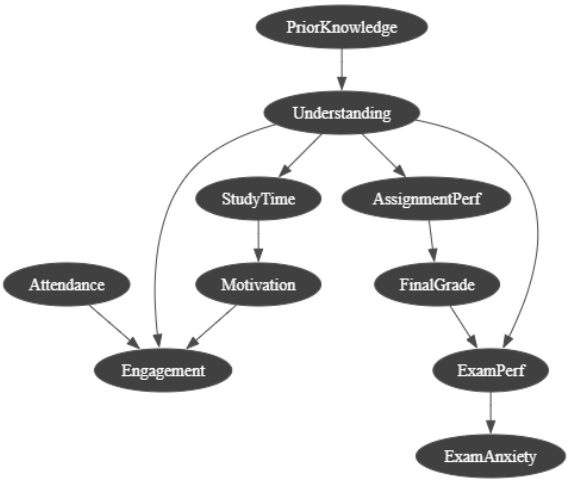

Figure 11: MIIC BN for Academic Performance, $n = 1000$

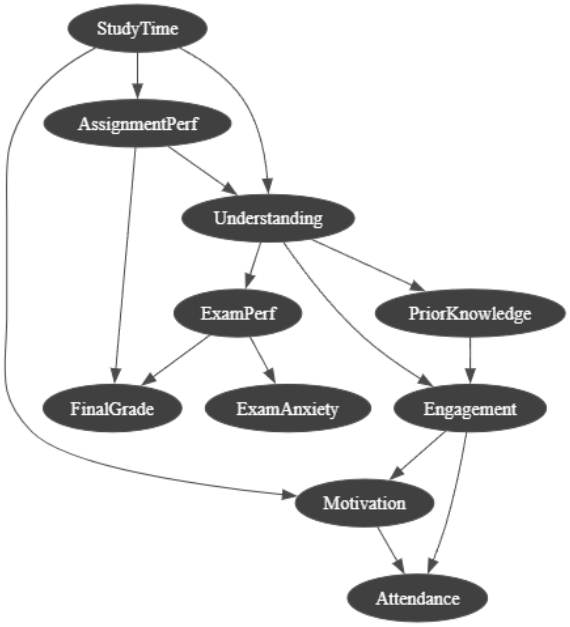

Figure 12: GHC BN for Academic Performance, $n = 1000$

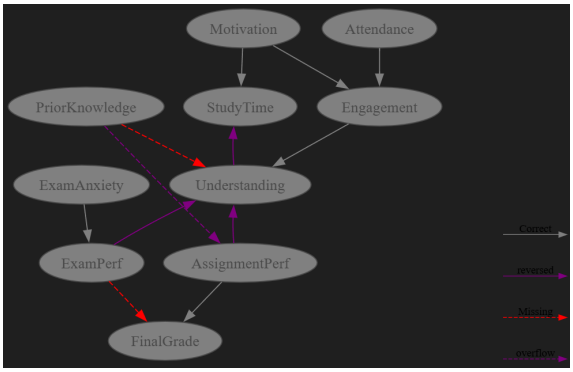

Figure 13: MIIC BN, Difference Comparison, $n = 100$

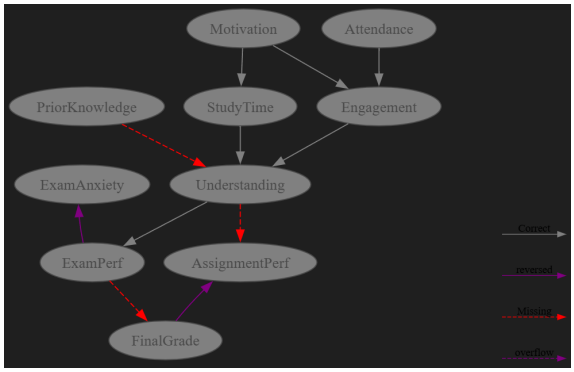

Figure 14: GHC BN, Difference Comparison, $n = 100$

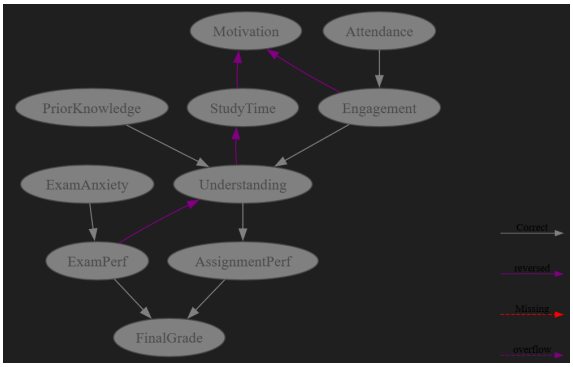

Figure 14: MIIC BN, Difference Comparison, $n = 500$

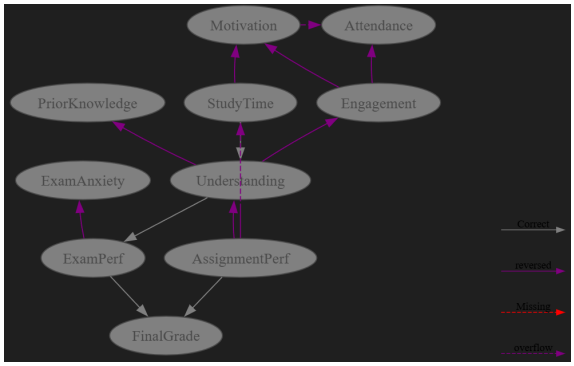

Figure 15: GHC BN, Difference Comparison, $n = 500$

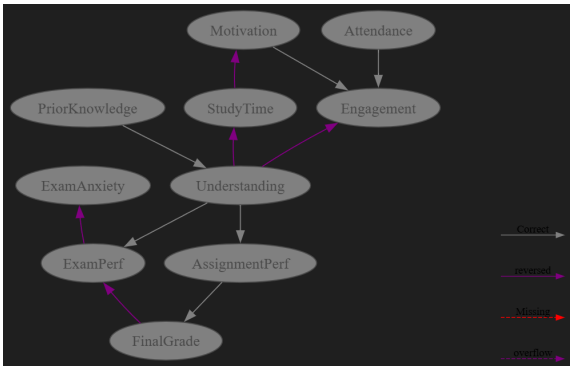

Figure 16: MIIC BN, Difference Comparison, $n = 1000$

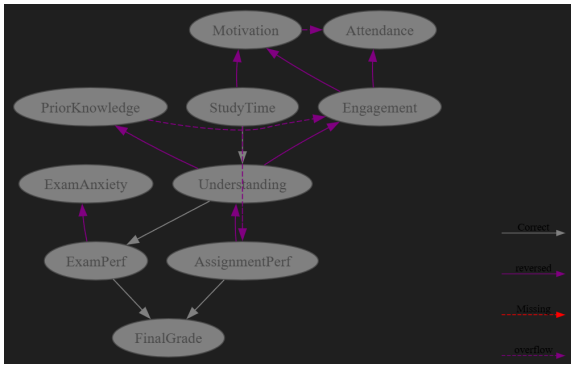

Figure 17: GHC BN, Difference Comparison, $n = 1000$

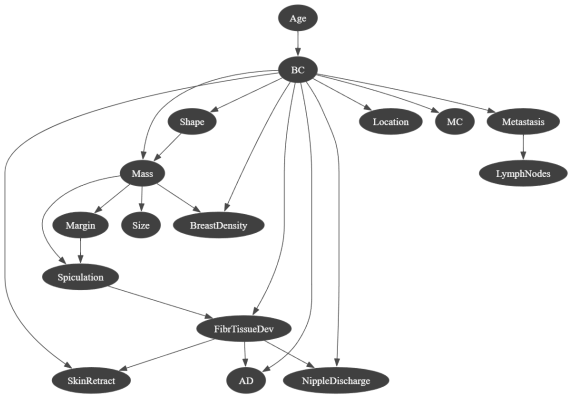

Figure 17: Unconstrained greedy hill-climbing

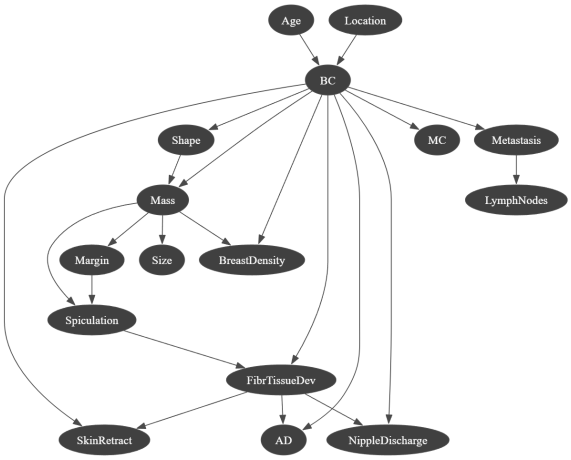

Figure 18: Greedy hill-climbing with expert-imposed constraints

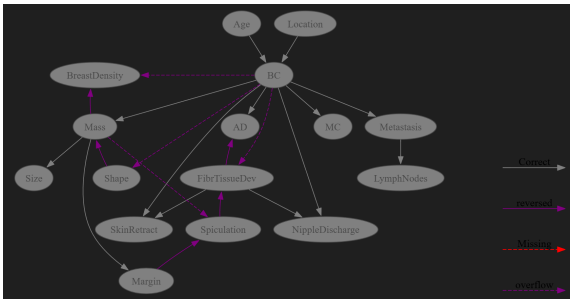

Figure 19: Difference resulting from constrained learning

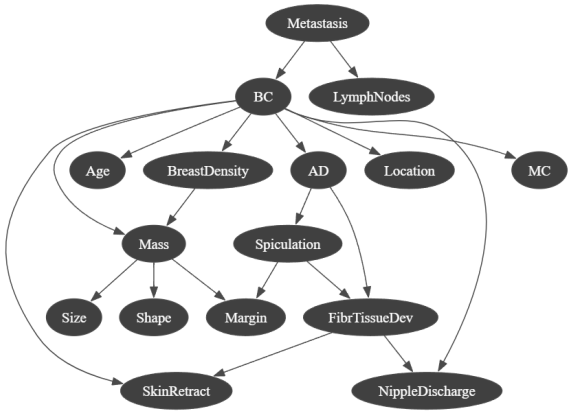

Figure 20: Expert-initialized greedy hill-climbing

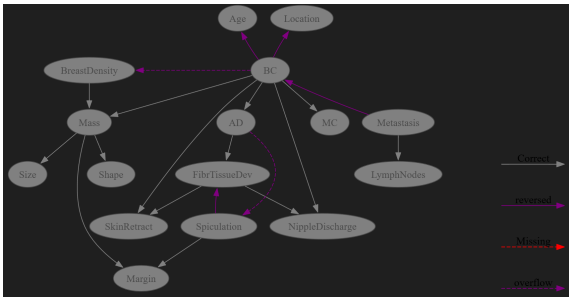

Figure 21: Difference from expert-initialized learning

C Mathematical Formula's

The receiver operating characteristic (ROC) curve characterizes a binary classifier's trade-off between sensitivity and specificity as its decision threshold t varies. Define:

$$\mathrm{TPR}(t) \;=\; \frac{\mathrm{TP}(t)}{\mathrm{TP}(t)+\mathrm{FN}(t)}, \qquad \mathrm{FPR}(t) \;=\; \frac{\mathrm{FP}(t)}{\mathrm{FP}(t)+\mathrm{TN}(t)},$$

where $\mathrm{TP}(t), \mathrm{FP}(t), \mathrm{TN}(t), \mathrm{FN}(t)$ are the counts of true positives, false positives, true negatives, and false negatives at threshold t. Plotting the points $(\mathrm{FPR}(t), \mathrm{TPR}(t))$ for all $t \in \mathbb{R}$ yields the ROC curve, which begins at $(0,0)$ and ends at $(1,1)$.

The area under the ROC curve (AUC) measures overall classifier discrimination. In the continuous limit,

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR})\,\mathrm{dFPR}.$$

When the ROC is sampled at $\{(\mathrm{FPR}_i, \mathrm{TPR}_i)\}_{i=1}^{n}$, a trapezoidal approximation is

$$\mathrm{AUC} \approx \sum_{i=1}^{n-1} \bigl(\mathrm{FPR}_{i+1} - \mathrm{FPR}_i\bigr)\,\frac{\mathrm{TPR}_{i+1}+\mathrm{TPR}_i}{2}.$$

An AUC of 0.5 indicates random performance; values closer to 1.0 signify stronger class separation.

Precision and recall quantify different facets of positive-class accuracy:

$$\mathrm{Precision} \;=\; \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \qquad \mathrm{Recall} \;=\; \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}.$$

Precision is the fraction of predicted positives that are correct, while recall (identical to TPR) is the fraction of actual positives captured by the classifier. The F_1–score harmonizes these into a single metric via their harmonic mean:

$$F_1 = 2\,\frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}} = \frac{2\,\mathrm{TP}}{2\,\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}.$$

A high F_1–score reflects a balance between precision and recall, penalizing extreme imbalances more than the arithmetic mean would.

References

Stefano Beretta, Mauro Castelli, Ivo Goncalves, Roberto Henriques, and Daniele Ramazzotti. Learning the structure of bayesian networks: A quantitative assessment of the effect of different algorithmic schemes. *arXiv preprint arXiv:1704.08676*, 2017. doi: 10.48550/arXiv.1704.08676. URL `https://arxiv.org/abs/1704.08676`. Submitted 27 Apr 2017 (v1); revised 3 Aug 2018 (v2); cs.LG.

Michelle Meekes, Steven Renooij, and Linda C. van der Gaag. Symbolic and quantitative approaches to reasoning with uncertainty. In *Proceedings of the Thirteenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2015)*, volume 9161 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 2015. ISBN 978-3-319-20806-0. doi: 10.1007/978-3-319-20807-7_33.