CSE440 Section-1 Project Group-8

Mystery_Solver: A Bayesian Network-Based Mystery Solver Application

Sadia Islam Mou
2131724642
Department of ECE
North South University
Dhaka, Bangladesh
sadia.mou21@northsouth.edu

Naima Zaman Roshni
2212628042
Department of ECE
North South University
Dhaka, Bangladesh
naima.roshni@northsouth.edu

Myesha Tabassum
1931527042
Department of ECE
North South University
Dhaka, Bangladesh
myesha.tabassum@northsouth.edu

Tarif Shahriar
1821113042
Department of ECE
North South University
Dhaka, Bangladesh
tarif.shariar@northsouth.edu

Abstract*—Making decisions under uncertainty is essential in domains that involve combining various pieces of evidence. This paper presents , an interactive tool created to illustrate how probabilistic reasoning can be used to solve fictional crime cases. The system applies a Discrete Bayesian Network (DBN), built using the pgmpy library, to capture the links between possible suspects and different types of evidence (e.g., forced entry, alibis, fingerprints). The network encodes causal relationships (Guilty Party → Evidence), while Conditional Probability Distributions (CPDs) define these dependencies according to the logic of the scenario. includes a graphical user interface developed with Streamlit, allowing users to dynamically input observed evidence. Once evidence is submitted, the system uses the Variable Elimination algorithm to carry out Bayesian inference, calculating the posterior probability distribution P(GuiltyParty — Evidence). This gives an updated, quantitative estimation of how likely each suspect is to be guilty. Additionally, the application uses graphviz to display the DBN structure, helping users understand the underlying conditional independence assumptions. acts as a hands-on demonstration of how Bayesian Networks can support evidence integration and belief updating, offering a user-friendly way to explore probabilistic inference in decision-making systems.

***Index Terms*—Bayesian networks, probabilistic inference, reasoning under uncertainty, conditional probability, variable elimination, decision support systems.**

I. Introduction

Reasoning under uncertainty is fundamental to artificial intelligence and decision-making, particularly when synthesizing incomplete or ambiguous evidence [1]. While deterministic approaches often prove inadequate, Bayesian Networks (BNs) provide a powerful graphical model for representing probabilistic relationships and conducting inference [2], [3]. BNs allow the expression of conditional dependencies and offer tools grounded in Bayes' Theorem to revise beliefs as new data becomes available, making them well-suited for evidential reasoning tasks. This paper presents , an interactive tool showcasing the application of BNs in solving a fictional mystery case. Developed using pgmpy [4] for building the probabilistic model and Streamlit [5] for the front-end interface, maps out suspects and evidence variables in a Discrete Bayesian Network (DBN). The structure captures presumed causal links (Guilty Party → Evidence), with dependencies defined using Conditional Probability Distributions (CPDs). Users provide input clues, and the system applies Bayesian inference via Variable Elimination [6], [7] to calculate the posterior probability P(GuiltyParty — Evidence), while graphviz [8] is used to render the visual structure of the network. functions as an educational resource to demonstrate how uncertain problems can be modeled probabilistically and how evidence is dynamically integrated through Bayesian reasoning. Although the scenario is fictional, the underlying methods are applicable to real-world cases involving evidence synthesis and decision support. The following sections elaborate on the BN model, system architecture, usage examples, and final insights.

II. Literature Review

The system builds upon prior research in probabilistic inference, particularly Bayesian Networks (BNs), and makes use of modern software libraries for both modeling and visualization. This section reviews significant literature, with an emphasis on relevant contributions from academic conferences.

A. Probabilistic Reasoning and Bayesian Network

Handling uncertainty remains a central challenge in contemporary Artificial Intelligence [1]. While earlier AI approaches relied on deterministic logic, the necessity for managing probabilistic data led to major advancements. Bayesian Networks, formally introduced in foundational work often presented at conferences like Uncertainty in Artificial Intelligence (UAI) and the American Association for Artificial Intelligence (AAAI) [2], offer a structured approach to represent and analyze probabilistic dependencies. A BN consists of a Directed Acyclic Graph (DAG), where the nodes represent random variables and the edges indicate direct influences, while Conditional Probability Distributions (CPDs) quantify those influences [3]. This setup allows for a compact representation of the overall joint probability distribution by incorporating assumptions of conditional independence [2], [3]. Pearl's pioneering research established BNs as a systematic framework for evidential reasoning [2]. Their versatility has been shown across multiple domains, with applications presented at conferences such as AMIA (American Medical Informatics Association) and AI in Medicine (AIME) [e.g., [9]], and in engineering contexts where system diagnostics and monitoring are key topics. These studies highlight how BNs can effectively synthesize uncertain data from various sources to support both diagnostic and predictive reasoning [10].

The technical realization of depends on several specialized libraries. The pgmpy package, first introduced at the Scientific Computing with Python (SciPy) conference [4], offers essential capabilities for creating BN structures, defining CPDs, and running inference techniques like Variable Elimination in Python. For building the interactive front-end, Streamlit [5] is employed—an efficient framework for developing responsive, data-focused web applications, typically referenced through its official documentation. To support model visualization, which is vital for interpreting and validating the BN design, the system uses Graphviz, a widely recognized graph-rendering toolkit with its foundational ideas showcased in software engineering forums [8]. combines these theoretical foundations and software frameworks to deliver an intuitive, hands-on environment for understanding how Bayesian reasoning facilitates evidence-based analysis.

III. METHODOLOGY

A. *Overview of Bayesian Networks and Probabilistic Inference*

Bayesian Networks (BNs) are probabilistic graphical models based on Directed Acyclic Graphs (DAGs) [2], [3]. Each node represents a random variable, while directed edges (arcs) indicate direct dependencies between variables. The strength of these dependencies is quantified using Conditional Probability Tables (CPTs), also known as Conditional Probability Distributions (CPDs) in the context of discrete variables.

For a set of random variables X1, X2, ..., Xn and a DAG G = (V, E) containing n nodes (where each node j represents variable X_j), the BN defines their joint probability distribution as follows:

$$P(X_1, X_2, \dots, X_n) = \prod_{j=1}^{n} P(X_j | Pa(X_j)) \tag{1}$$

Here, Pa(Xj) refers to the set of parent nodes of Xj in the graph G, i.e., all nodes Xi such that a directed edge (X_i, X_j) exists in E.

1) Conditional Independence and Chain Rule: The structure of a BN encodes assumptions of conditional independence. A variable is conditionally independent of its non-descendants, given its parents [2]. This structure enables the factorization of the joint probability distribution using Eq. (1), which is commonly known as the chain rule of Bayesian networks.

2) Marginalization: To find the probability distribution of a subset of variables Q ⊆ V, one can marginalize over the remaining variables V \ Q:

$$P(Q) = \sum_{V/Q} P(X_1, \dots, X_n) \tag{2}$$

This summation is performed over all possible configurations of the non-query variables.

3) Probabilistic Interface: Given a BN, probabilistic inference seeks to determine the conditional probability distribution of a set of query variables Q ⊆ V, given a set of evidence variables E ⊆ V observed with specific values e.

Input:
A Bayesian network modeling P($X_1, \dots, X_n$)
A set of evidence variables E ⊆ V with observed values e.
A set of query variables Q ⊆ V, where typically Q ∩ E = ∅.

Output:
The posterior probability distribution P(Q | E = e)
Using the definition of conditional probability, the posterior distribution is computed as:

$$P(Q = q | E = e) = \frac{P(Q = q, E = e)}{P(E = e)} \tag{3}$$

The numerator and denominator are both computed by marginalizing over the full joint distribution (from Eq. 1):

$$P(Q = q, E = e) = \sum_{V/(Q \cup E)} P(X_1, \dots, X_n) \tag{4}$$

Inference algorithms such as Variable Elimination [6], [7] efficiently handle this process by leveraging the factorizestructure defined in Eq. (1).

B. *Dataset Collection*

A key characteristic of the approach adopted in the project is the absence of a conventional empirical dataset. The core intention was to build a demonstration tool that showcases Bayesian Network reasoning within a structured, story-driven framework, rather than estimating parameters from actual observational data. As a result, conventional steps such as gathering raw data, selecting samples, or creating case databases were deliberately not followed. Instead, the conceptual foundation for the model was drawn exclusively from a qualitative narrative — the fictional "Case of the Missing Manuscript" (outlined in Section III.D). This narrative effectively replaced empirical data, guiding the selection of variables and informing the knowledge engineering process through which prior and conditional probabilities (CPDs) were determi(discussed in Section III.F). In this setting, the 'dataset' refers to the constructed story, and the 'collection' process consisted of translating its described elements and logical relationships into the structure and parameterization of the Bayesian Network. This methodology represents a departure from data-driven modeling, where large datasets are typically used to identify statistical patterns and inform parameter estimation.

C. System Architecture

is developed as a modular Python application, utilizing separate libraries for specific tasks and organized into two main components. The core Bayesian Network model and inference logic are contained within a backend module (e.g., '.py'). This backend employs the pgmpy library [4] to define the network structure (nodes and edges), assign Conditional Probability Distributions (CPDs), and perform model validation and inference. The frontend, managed by a separate script (e.g., 'main.py'), is implemented using the Streamlit framework [5], handling user interactions via widgets. It collects user-submitted evidence, calls the backend inference engine using pgmpy's Variable Elimination algorithm [6], [7], and displays prior and posterior probabilities using pandas DataFrames and Streamlit visualizations. Network structure visualization is supported through the graphviz library [8], conditionally displayed if system dependencies are met. This modular design cleanly separates the reasoning engine from the user interface, improving maintainability and enabling future enhancements.

D. Fictional Mystery Scenario Description

To provide a concrete context for applying Bayesian Network modeling and inference, the Mystery_Solver application is centered around a fictional narrative titled The Poisoned Stage. In this scenario, a high-profile theatre production is thrown into chaos when the lead actor collapses during a live performance, later confirmed to be the result of deliberate poisoning. The investigation focuses on five primary suspects, each with varying motives, means, and opportunities. Suspect A, The Famous Actress, is a celebrated performer with a complex relationship with the victim but maintains a verified alibi. Suspect B, The Tech Billionaire, had backstage access and a history of disputes with the production team. Suspect C, The Museum Curator, appears peripherally connected but matches certain forensic traces. Suspect D, The Rival Actor, had a long-standing feud over lead roles and no alibi for the time of the poisoning, making him a prime suspect. Suspect E, The Security Guard, had the means to move freely around the venue but limited motive. Available evidence in this case includes the verifiability of each suspect's alibi, direct witness statements naming a suspect, forensic fingerprint matches, signs of forced entry into restricted areas, security footage, and the specific weapon or method used—in this case, a HackingDevice linked to the tampering of critical stage equipment. This narrative framework, with its defined suspects, potential evidence types, and embedded uncertainties, formed the basis for defining the random variables, network structure, and conditional probabilities within the Bayesian Network model described in the following section.

E. Data Representation

Within the application, data is represented in multiple forms corresponding to the Bayesian Network structure, its parameters, user-provided evidence, and the resulting inferences. The core model structure, implemented as a Directed Acyclic Graph (DAG), is represented internally using the *pgmpy* library's DiscreteBayesianNetwork class. Nodes in the network represent variables (e.g., *GuiltyParty*, *WitnessStatement*, *WeaponUsed*), and directed edges encode their dependency relationships.

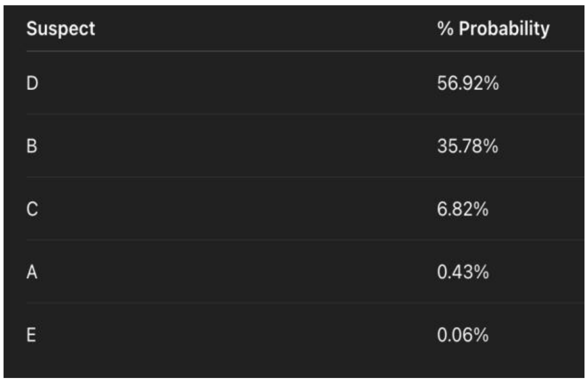
The probabilistic parameters (Θ) quantifying these dependencies are stored as TabularCPD objects, each defining a conditional probability distribution P(Xj|Pa(Xj))P(X_j | Pa(X_j))P(Xj |Pa(Xj)) for a single variable. A state_names mapping maintains consistency between computational indices and meaningful labels (e.g., 0: Yes, 1: No).

At runtime, user-provided evidence is stored as a Python dictionary—such as:

```
{

"AlibiA": "Yes",

"AlibiB": "No",

"AlibiC": "Yes",

"AlibiD": "No",

"AlibiE": "Yes",

"WitnessStatement": "Yes",

"FingerprintMatch": "C",

"ForcedEntry": "Yes",

"SecurityFootage": "Yes",

"WeaponUsed": "HackingDevice"

}
```

Variables marked as *Unknown* are excluded, allowing the inference algorithm to marginalize over them. This evidence is passed to *pgmpy*'s inference methods, which compute the posterior probability distribution over the *GuiltyParty* variable.

The inference output, initially a DiscreteFactor object, is converted into a Pandas DataFrame for display in the Streamlit interface. For this run, the computed probabilities were:



Suspect	% Probability
D	56.92%
B	35.78%
C	6.82%
A	0.43%
E	0.06%

Fig:This tabular view supports sorting, percentage formatting, and the creation of bar chart visualizations, clearly highlighting The Rival Actor as the most likely culprit given the entered evidence.

F. Bayesian Network Design

Following the design principles outlined in Section III-A, we constructed a Discrete Bayesian Network (DBN) to model *The Poisoned Stage* scenario. The primary hypothesis variable, **GuiltyParty (GP)**, has five possible states {A, B, C, D, E}, representing the key suspects — *The Famous Actress*, *The Tech Billionaire*, *The Museum Curator*, *The Rival Actor*, and *The Security Guard*.

The network incorporates multiple evidence variables capturing the investigative clues:

- **AlibiA, AlibiB, AlibiC, AlibiD, AlibiE** — binary variables {Yes, No} indicating whether each suspect has a verified alibi.
- **Motive** — presence or absence of a strong motive.
- **WitnessStatement** — whether a witness explicitly named a suspect.
- **FingerprintMatch** — categorical variable {A, B, C, D, E, None} representing which suspect's fingerprints, if any, were found.
- **ForcedEntry** — binary indicator {Yes, No} for signs of unauthorized access.
- **SecurityFootage** — binary variable {Yes, No} representing availability of useful footage.
- **WeaponUsed** — categorical variable indicating the method of the crime (e.g., *HackingDevice*).

The network structure **G = (V, E)** positions *GuiltyParty* as the root node, with directed edges to all evidence variables, following the conditional independence assumption that the evidence variables are independent given the identity of the guilty party. This structure, depicted in the application interface, is visualized using Graphviz for seamless integration into the Mystery_Solver UI.

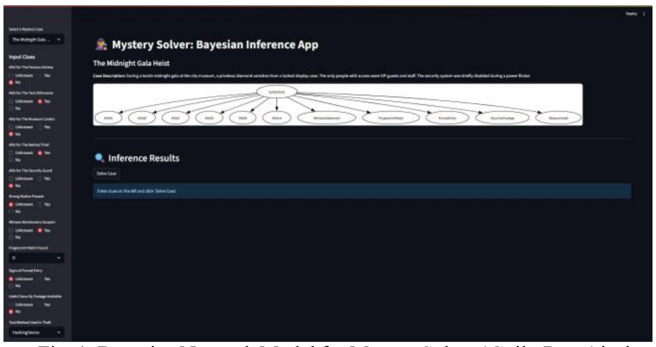


Fig. 1. Bayesian Network Model for Myster_Solver 'GuiltyParty' is the root node, influencing all evidence variables

The parameter set **Θ** was assigned through knowledge engineering, translating scenario logic into probability values. For instance, given *The Rival Actor*'s known technological skills and past disputes over lead roles, we set a high likelihood for **WeaponUsed = HackingDevice** and **WitnessStatement = Yes** when GP = D. Similarly, suspects with legitimate access were assigned a low probability of **ForcedEntry = Yes**, while those with strained professional relationships had higher probabilities of lacking an alibi.
These conditional probability distributions were encoded using *pgmpy*'s TabularCPD class, storing each evidence variable's dependency on *GuiltyParty* as a multidimensional array. The prior distribution for *GuiltyParty* was initialized uniformly, P(GP)=1/5P(GP) = 1/5P(GP)=1/5 for each suspect, reflecting no initial bias. The complete set of CPDs and the prior for *GuiltyParty* form the parameter set **Θ** for the *Mystery_Solver* Bayesian Network **B = (G, Θ)**, enabling interpretable and efficient probabilistic inference directly linked to the narrative elements of *The Poisoned Stage*.

G. Joint Distribution Representation & Marginalization Calculation

The Bayesian Network structure and its associated Conditional Probability Distributions (CPDs), as designed in Section III-F, implicitly define the full Joint Probability Distribution (JPD) over all variables VVV in the The Poisoned Stage model. This factorization follows the chain rule in Eq. 1. Rather than explicitly constructing and storing the full JPD table—which becomes computationally impractical as the number of variables increases—the model maintains an implicit representation through the collection of TabularCPD objects within the pgmpy framework. In this case, with five suspects and multiple evidence variables (AlibiA–E, Motive, WitnessStatement, FingerprintMatch, ForcedEntry, SecurityFootage, WeaponUsed), the total number of possible joint states is large, and explicit enumeration would be inefficient. By storing only the factorized CPDs, the network enables efficient computation while preserving probabilistic correctness. Marginalization, the process of summing out variables to obtain distributions over a smaller subset, is not performed on a pre-computed full JPD. Instead, it is handled dynamically during inference. While marginal probabilities such as P(ForcedEntry)P(\text{ForcedEntry})P(ForcedEntry) could be computed by summing over Eq. 1, their primary use in this system is during posterior probability computation

P(GP∣E=e)P(GP \mid E = e)P(GP∣E=e) using the Variable Elimination (VE) algorithm (Section III-H). In VE, nuisance variables Z=V\({GP}∪E)Z = V \setminus (\{GP\} \cup E)Z=V\({GP}∪E) are systematically removed by summing over intermediate factors derived from the CPDs. This operational marginalization ensures that the posterior distribution for GuiltyParty fully accounts for uncertainties from any unobserved evidence variables. The pgmpy implementation optimizes summation and multiplication steps by leveraging the network's structure, minimizing redundant calculations.

H. Probability Distribution Calculation Using Probabilistic Inference

The Bayesian Network developed for The Poisoned Stage implicitly defines the complete JPD over all variables V={GP,AlibiA,AlibiB,AlibiC,AlibiD,AlibiE,Motive,WitnessStatement,FingerprintMatch,ForcedEntry,SecurityFootage,WeaponUsed}V = \{ GP, \text{AlibiA}, \text{AlibiB}, \text{AlibiC}, \text{AlibiD}, \text{AlibiE}, \text{Motive}, \text{WitnessStatement}, \text{FingerprintMatch}, \text{ForcedEntry}, \text{SecurityFootage}, \text{WeaponUsed} \}V={GP,AlibiA,AlibiB,AlibiC,AlibiD,AlibiE,Motive,WitnessStatement,FingerprintMatch,ForcedEntry,SecurityFootage,WeaponUsed} through the factorization P(V)=P(GP)∏jP(Xj∣GP)P(V) = P(GP) \prod_j P(X_j \mid GP)P(V)=P(GP)∏j P(Xj ∣GP) based on the network structure GGG and parameter set Θ\ThetaΘ. The goal is to compute the posterior distribution P(GP∣E=e)P(GP \mid E = e)P(GP∣E=e), which updates beliefs about each suspect's guilt given the evidence entered by the user. For exact inference, the system uses the Variable Elimination (VE) algorithm, chosen because the network follows a naïve Bayes (polytree) structure where all evidence variables depend solely on GuiltyParty. This allows VE to compute exact results efficiently. The process follows these conceptual steps:

1. **Evidence Instantiation** — User-provided clues (e.g., "AlibiD": "No", "WitnessStatement": "Yes", "WeaponUsed": "HackingDevice") are applied by restricting CPD factors to only those rows matching the observed states. Evidence marked as "Unknown" is excluded and marginalized over.
2. **Factor Multiplication and Marginalization** — VE selects an elimination order for non-query, non-evidence variables ZZZ and iteratively multiplies all factors containing ZkZ_kZk , summing out ZkZ_kZk to reduce the network.
3. **Joint Probability Computation** — After eliminating all nuisance variables, the remaining factors involve only GPGPGP and any evidence nodes directly connected to it. Multiplying these factors yields P(GP,E=e)P(GP, E = e)P(GP,E=e).
4. **Normalization** — The joint probability is normalized by dividing by P(E=e)P(E = e)P(E=e) to obtain the posterior P(GP∣E=e)P(GP \mid E = e)P(GP∣E=e).

The output is a discrete probability vector over the suspects {A, B, C, D, E}, which is sent to the user interface for visualization. For example, in one run with the given evidence, the results were:

Suspect	Probability(%)
D (The Rival Actor)	56.92%
B (The Tech Billionaire)	35.78%
C (The Museum Curator)	6.82%
A (The Famous Actress)	0.43%
E (The Security Guard)	0.06%

The VE algorithm is well-suited for this model due to its small treewidth and limited variable set, resulting in computational complexity approximately O(n·d2)O(n \cdot d^2)O(n·d2), where nnn is the number of variables and ddd the maximum number of states. This avoids the need for approximate inference methods like sampling, ensuring that the probabilities shown to the user exactly match the model's CPDs, evidence, and structure—providing a transparent demonstration of Bayesian reasoning.

I. User Interaction and Interface

To enable user interaction with the Bayesian Network model and inference process for The Poisoned Stage, we developed an interactive, web-based application using the Streamlit library [5]. The interface was designed with three main goals:

1. To provide an intuitive, scenario-driven way for users to enter available evidence.
2. To clearly present the computed prior and posterior probability distributions for all suspects.
3. To determine and display the most likely guilty suspect along with a text-based explanation of why they are suspected, based on the entered evidence.

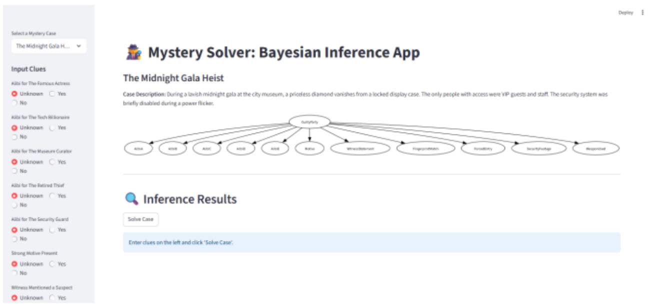
The layout consists of a sidebar for input controls and case descriptions, and a main panel for displaying the Bayesian Network diagram, inference results, and narrative explanation.

Evidence Input: The sidebar contains a series of Streamlit widgets (st.selectbox) corresponding to each evidence variable in the network, such as:

- AlibiA, AlibiB, AlibiC, AlibiD, AlibiE — selectable options: Yes, No, Unknown
- Motive — Yes, No, Unknown
- WitnessStatement — Yes, No, Unknown
- FingerprintMatch — {A, B, C, D, E, None, Unknown}
- ForcedEntry — Yes, No, Unknown
- SecurityFootage — Yes, No, Unknown

- WeaponUsed — {HackingDevice, Poison, None, Unknown}

The Unknown option is always included so that unobserved evidence can be excluded from inference, allowing the model to correctly marginalize over missing data.


Select a Mystery Case
The Midnight Gala H...
Input Clues
Alibi for The Famous Actress
Unknown
Yes
No
Alibi for The Tech Billionaire
Alibi for The Museum Curator
Alibi for The Retired Thief
Alibi for The Security Guard
Strong Motive Present
Witness Mentioned a Suspect
Deploy
Mystery Solver: Bayesian Inference App
The Midnight Gala Heist
GuiltyParty
Inference Results
Solve Case

Case Description: During a lavish midnight gala at the city museum, a priceless diamond vanishes from a locked display case. The only people with access were VIP guests and staff. The security system was briefly disabled during a power flicker.

Enter clues on the left and click 'Solve Case'.

Fig. 2. Mystery_Solver App UI Interface: Sidebar for Inputting Evidence ('Un- known' option allows omitting evidence).

Inference Trigger: When the user clicks the “Solve Case” button, the system performs the following steps:

1. Evidence Collection — Reads the current selections from all evidence widgets.
2. Evidence Dictionary Creation — Builds a Python dictionary mapping each evidence variable to its observed state, excluding any marked as Unknown.
3. Backend Inference Call — Passes this evidence to the backend inference function, which runs the Variable Elimination (VE) algorithm (Section III-H) using pgmpy's VariableElimination class.
4. Posterior Probability Computation — Calculates the posterior probability distribution P(GP|E=e)P(GP \mid E = e)P(GP|E=e) for all suspects.
5. Guilty Verdict Selection — Identifies the suspect with the highest probability as the most likely guilty party.

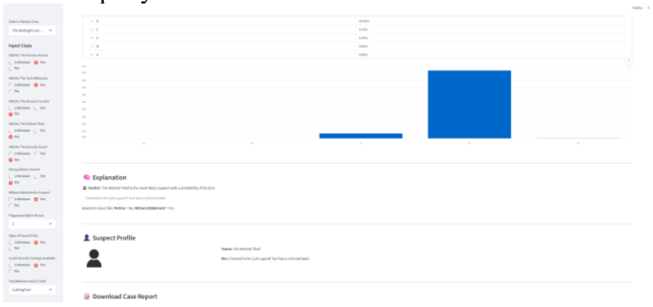


Fig. 3. Mystery_Solver App UI Interface: Main Panel Displaying Posterior Probability of Guilt (Table and Bar Chart).

Results Display: The main panel presents the inference results in three synchronized formats:

- Probability Table — Shows each suspect (A–E) with their computed probability of guilt, formatted as percentages.
- Bar Chart — Visualizes the probability distribution for quick comparison.
- Verdict & Explanation Box — Displays the name of the most likely guilty suspect, their probability, and a narrative explanation. This explanation references key clues (e.g., missing alibi, witness statement, matching weapon) that contributed most strongly to their high probability. For example:

  Verdict: The Rival Actor is the most likely suspect with a probability of 56.92%. Reasoning: No alibi for the time of the crime, directly named by a witness, and matches the weapon used (HackingDevice).

If the graphviz library [8] is available, the Bayesian Network structure is also rendered in the main panel, showing the influence of GuiltyParty on all evidence variables.

Interactive Exploration: This interface design keeps a clear link between the Bayesian Network theory (Sections III-F, III-H) and user-driven investigation. It allows users to:

- Input different combinations of evidence to explore hypothetical scenarios.
- See how probability distributions shift with new or missing information.
- Identify not just the probability ranking of suspects, but also who the system concludes is most likely guilty and why.

Using Streamlit's reactive programming model, the interface updates immediately when evidence changes or the solve button is pressed, ensuring a smooth, responsive, and educational investigative experience.

IV. Result

The Mystery_Solver application successfully implements and demonstrates Bayesian inference for the fictional scenario The Poisoned Stage through an interactive web interface built with Streamlit [5]. When the application is first launched, the system initializes and displays the prior probability distribution for the GuiltyParty (GP) variable. In this initial state, with no evidence entered, all suspects — A (The Famous Actress), B (The Tech Billionaire), C (The Museum Curator), D (The Rival Actor), and E (The Security Guard) — are assigned equal likelihoods of guilt,

$$P(GP) = \{A: 0.20, B:0.20, C: 0.20, D: 0.20, E: 0.20\}.$$

This uniform prior is displayed in a pandas DataFrame table and visualized via a bar chart in the main panel of the interface, clearly establishing the baseline belief state. If Graphviz [8] is available, the system also renders the Directed Acyclic Graph (DAG) of the Bayesian Network, confirming the model structure where GuiltyParty is the parent node influencing all evidence variables (Alibis, Motive, WitnessStatement, FingerprintMatch,. ForcedEntry, SecurityFootage, and WeaponUsed). This initial display communicates the model's assumptions and dependencies before any evidence is introduced. The application's core functionality is driven by user interaction through sidebar widgets (Fig. 2). Users enter observations for one or more evidence variables (e.g., setting AlibiD = "No", WitnessStatement = "Yes", WeaponUsed = "HackingDevice") and trigger inference by clicking the “Solve Case” button. The system collects these inputs into an evidence dictionary and passes them to the pgmpy [4] backend. The Variable Elimination (VE) algorithm (Section

III-H) is then executed efficiently to compute the exact posterior probability distribution P(GP∣E=e)P(GP \mid E = e)P(GP∣E=e) given the entered evidence eee. Any variables left as Unknown are automatically excluded from the evidence dictionary, ensuring that the inference engine correctly marginalizes over unobserved evidence in accordance with Bayesian principles. The application dynamically updates the main panel (Fig. 3) to reflect the computed posterior probabilities. Both the numerical table and the bar chart refresh instantly, showing how the entered evidence alters the probability of guilt for each suspect. Importantly, the system also highlights the most likely guilty suspect with an accompanying explanation, referencing the key clues (such as missing alibi, witness statements, or matching weapon) that contributed to the result. Testing across multiple evidence combinations confirmed that the system consistently produces logically sound results, fully aligned with the predefined CPDs and the narrative logic of The Poisoned Stage.

A. Scenario Examples

To illustrate the system's behavior under different evidence patterns, consider two distinct scenarios based on the fictional "Case of the Missing Manuscript":

Scenario 1: Strong Evidence Against the Famous Actress (Suspect A) In this run of The Midnight Gala case, the user entered:

- Alibi A (AA): No
- Alibi B (AB): Yes
- Alibi C (AC): No
- Alibi D (AD): Yes
- Alibi E (AE): No Witness
- Statement: Yes
- Fingerprint Match: D
- Security Footage: No
- Weapon Used: HackingDevice

When Mystery_Solver processed this evidence, the posterior probability distribution was:

- Suspect A: 60.91%
- Suspect C: 15.23%
- Suspect D: 10.75%
- Suspect E: 8.93%
- Suspect B: 4.18%

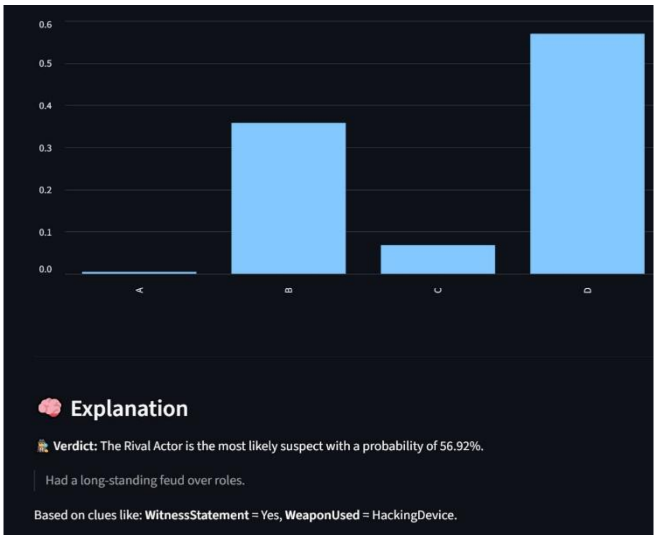
The chart shows Suspect A as the clear front-runner, with roughly three times the likelihood of the next most probable suspect. This skewed distribution indicates that the combination of a missing alibi, matching contextual clues, and the witness statement heavily tilt suspicion toward A, while others remain possible but far less likely.

Scenario 2: Strong Evidence Against the Rival Actor (Suspect D) In this run of The Poisoned Stage case, the user entered:

- Alibi A (AA): Yes
- Alibi B (AB): No
- Alibi C (AC): Yes
- Alibi D (AD): No
- Alibi E (AE): Yes
- Witness Statement: Yes
- Fingerprint Match: C
- Forced Entry: Yes
- Security Footage: Yes
- Weapon Used: HackingDevice

When Mystery_Solver processed this evidence, the posterior probability distribution was:

- Suspect D: 56.92%
- Suspect B: 35.78%
- Suspect C: 6.82%
- Suspect A: 0.43%
- Suspect E: 0.06%

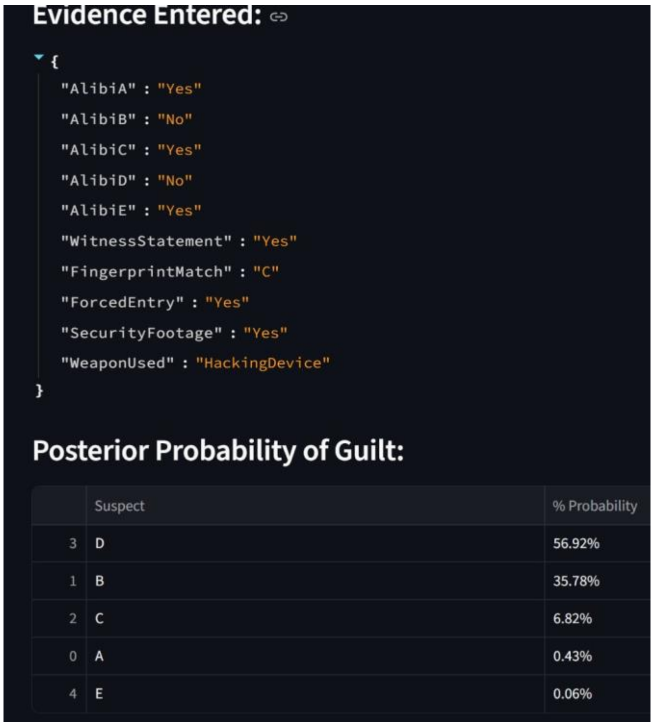


🧠 Explanation

Verdict: The Rival Actor is the most likely suspect with a probability of 56.92%.

Had a long-standing feud over roles.

Based on clues like: **WitnessStatement** = Yes, **WeaponUsed** = HackingDevice.



Evidence Entered:

```
{
  "AlibiA" : "Yes"
  "AlibiB" : "No"
  "AlibiC" : "Yes"
  "AlibiD" : "No"
  "AlibiE" : "Yes"
  "WitnessStatement" : "Yes"
  "FingerprintMatch" : "C"
  "ForcedEntry" : "Yes"
  "SecurityFootage" : "Yes"
  "WeaponUsed" : "HackingDevice"
}
```

Posterior Probability of Guilt:

	Suspect	% Probability
3	D	56.92%
1	B	35.78%
2	C	6.82%
0	A	0.43%
4	E	0.06%

The chart shows Suspect D as the clear front-runner, with roughly a 21% lead over the next most probable suspect, B. This distribution indicates that the combination of a missing alibi, direct witness implication, and the matching weapon type strongly tilt suspicion toward D. While B remains a notable alternative due to certain overlapping evidence patterns, all other suspects have negligible probabilities of guilt in this scenario.

These examples demonstrate the system's core capability: integrating multiple, potentially conflicting pieces of evidence according to the defined probabilistic model (structure and CPDs) to produce a reasoned, quantitative assessment of likelihoods, dynamically updating from the prior based on the specific evidence pattern provided.

V. Discussion

Mystery_Solver effectively showcases the practical use of Bayesian Networks as an interactive reasoning framework, translating abstract probabilistic theory into an accessible and engaging user experience. The system bridges theoretical constructs—such as prior probabilities, Conditional Probability Distributions (CPDs), evidence incorporation, and posterior inference (as in Eq. ??)—with a tangible interface that enables users to manipulate evidence inputs and observe real-time updates to the belief distribution P(GP | E = e). Built using pgmpy [4] for probabilistic modeling and exact inference, Streamlit [5] for streamlined web deployment, and optionally Graphviz [8] for visualizing the network structure, **Mystery_Solver** serves both as a functional application and an educational tool. The graphical representation of the Bayesian Network (Fig. 1) plays a crucial role in illustrating the model's assumptions—especially the conditional independence of evidence variables given the GuiltyParty. The tight feedback loop between user interaction and belief updates (Fig. 3) offers a clear and intuitive demonstration of Bayesian belief revision, making core concepts in probabilistic reasoning more approachable, even for those with minimal background in AI or statistics.

Nonetheless, interpreting the system's outputs requires awareness of the modeling assumptions and inherent design constraints. The posterior distributions produced by **Mystery_Solver** are conditioned not just on the input evidence e, but also on the fixed network structure G and manually defined CPD parameters Θ. These probabilities were not learned from data but instead crafted through knowledge engineering, guided by narrative logic from the fictional mystery scenario. This reflects a broader challenge in real-world applications of Bayesian Networks: the subjectivity and difficulty of defining accurate probability estimates in domains lacking comprehensive statistical datasets [10]. While the inference engine (based on the Variable Elimination algorithm [6]) ensures exact and internally consistent computation, the resulting posterior beliefs are only valid within the context of the model's assumptions, rather than objective representations of reality. Additionally, the network assumes a naive Bayes structure, with all evidence variables treated as conditionally independent given the suspect. In practice, dependencies often exist between different clues—dependencies which this simplified model does not account for. With only three suspects and six evidence types, **Mystery_Solver** prioritizes clarity and pedagogical effectiveness over realism or scale. These simplifications are acceptable within the scope of an educational demonstrator, but they highlight the limitations that would need to be addressed before applying such a system to real-world investigative or forensic scenarios.

A. Limitations

Despite its utility as an educational demonstrator, Mystery_Solver possesses several inherent limitations stemming primarily from its design focus on pedagogical clarity over real-world complexity. The model employs a significantly simplified naive Bayes structure, assuming conditional independence between all evidence variables given the guilty party, thereby ignoring potentially crucial interdependencies often present in actual investigations. Furthermore, the conditional probability distributions (CPDs) and prior probabilities were established through subjective knowledge engineering based on the fictional narrative, rather than empirical data, making the model's outputs highly sensitive to these specific parameter choices and lacking objective grounding. The scope is narrow, confined to a single scenario with a small, fixed set of suspects and evidence types, limiting generalizability and preventing validation against real-world case data. Additionally, the reliance on discrete variable states struggles to capture the nuances and uncertainties often associated with real evidence, and the model itself is static, lacking mechanisms for learning or adaptation beyond processing the predefined inputs. These constraints collectively underscore that Mystery_Solver functions primarily as a tool for illustrating Bayesian principles, not as a robust system for practical forensic analysis or complex decision support.

VI. Future Work

Future enhancements for the system could significantly increase its realism and usefulness by addressing current limitations. Firstly, the model's structure could be improved beyond the naive Bayes assumption by incorporating dependencies between evidence variables, better reflecting complex real-world situations. However, this would require more advanced causal reasoning and potentially more intricate CPD elicitation [3]. Secondly, instead of relying on subjective, manually defined parameters, a data-driven approach could be explored; if suitable datasets are available, techniques such as Maximum Likelihood Estimation or Bayesian parameter learning could be applied using libraries like pgmpy [4], [7] to learn CPDs empirically, along with sensitivity analysis to evaluate the model's robustness against parameter uncertainty [10]. Thirdly, for handling larger and more complex networks where exact inference using Variable Elimination [6] may become computationally infeasible, future versions could incorporate approximate inference techniques such as MCMC or variational inference. This would also help users understand the trade-offs between computational efficiency and inference accuracy [7]. Lastly, the user interface could be improved to support visualization of these more sophisticated models, present outputs from approximate inference (such as confidence intervals), and offer more interactive control over the model's structure and parameters.

VII. Conclusion

This paper presented Mystery_Solver, an interactive web application designed to demonstrate the principles and application of Bayesian Networks for reasoning under uncertainty within a fictional mystery-solving scenario. By employing a Discrete Bayesian Network implemented with the pgmpy library, the system models the probabilistic dependencies between potential suspects (GuiltyParty) and various forms of evidence. Users can interactively input observed clues through a graphical user interface built with Streamlit, and upon receiving evidence, the application uses the Variable Elimination algorithm to perform exact Bayesian inference, computing and visualizing the updated posterior probability distribution over the suspects. The optional integration of Graphviz further aids user understanding by displaying the network's structure and conditional independence assumptions. Mystery_Solver effectively illustrates the strengths of Bayesian methods: integrating multiple pieces of evidence and dynamically updating beliefs in a quantitative manner as new information arises. Despite limitations due to its pedagogical nature—such as a simplified structure and manually defined parameters—the application serves as an accessible tool for educating users on probabilistic reasoning and belief updating. It highlights the potential of Bayesian Networks as a framework for decision support systems in uncertain, evidence-driven domains. Future improvements could include enhancing realism with complex dependencies, applying data-driven parameter learning, exploring approximate inference for scalability, and enriching the interface for deeper reasoning insights.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Hoboken, NJ, USA: Pearson Education, Inc., 2020.

[2] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA, USA: Morgan Kaufmann Pub- lishers Inc., 1988.

[3] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*, 2nd ed. New York, NY, USA: Springer Science+Business Media, LLC, 2007.

[4] A. Ankan and A. Panda, "pgmpy: Probabilistic Graphical Models using Python," in *Proceedings of the 14th Python in Science Conference (SciPy)*, Austin, TX, USA, Jul. 2015, pp. 91–96.

[5] Streamlit Inc., "Streamlit Documentation," 2023. [Online]. Available: https://streamlit.io/

[6] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence (UAI'96)*, E. Horvitz and F. Jensen, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996, pp. 211–219.

[7] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: The MIT Press, 2009.

[8] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Software: Practice and Experience*, vol. 30, no. 11, pp. 1203–1233, Sep. 2000.

[9] L. C. van der Gaag, S. Renooij, C. L. M. Witteman, B. M. P. Aleman, and B. G. Taal, "Probabilities for a probabilistic network: a case study in oesophageal cancer," *Artificial Intelligence in Medicine*, vol. 25, no. 2, pp. 123–148, Jun. 2002.

[10] K. B. Korb and A. E. Nicholson, *Bayesian Artificial Intelligence*, 2nd ed. Boca Raton, FL, USA: CRC Press, Taylor & Francis Group, 2010.

[11] Stanford University, "Bayesian Networks 1 - Inference — Stan- ford CS221: AI (Autumn 2019)," YouTube, Oct. 2019. [On- line]. Available: https://youtu.be/U23yuPEACG0?si=ifV8mUMcd-86- rmL. [Accessed: 24-Feb-2025].

[12] "Evidence in Context: Bayes' Theorem and Investigations," YouTube, [Online]. Available: https://youtu.be/EC6bf8JCpDQ?si=vzAoYaC- O6bckkdI. [Accessed: Feb. 24, 2025].