School of Engineering and Applied Science (SEAS), Ahmedabad University

Probabilistic Graphical Models (CSE 516)

Project Report

Group Name: Quantum Bayesian Network (G9)
Team Members:

- Freya Shah (AU2120184)
- Prapti Patel(AU2240241)
- Dhairya Doshi (AU2240031)
- Ved Raval (AU2240068)

I. Summary

Bayesian inference is traditionally computationally challenging because it requires summing or integrating over a vast number of possible outcomes, a task that is known to be NP-hard. Bayesian inference has found significant applications in fields such as finance, where it is used to model uncertainty and make predictions based on limited data.

Quantum computing is an emerging technology with the potential to solve complex and computationally intensive problems that classical computers struggle with. In 2014, Low et al. proposed a quantum Bayesian inference algorithm that has been theoretically proven to achieve quadratic speedup over classical rejection sampling methods. The key principle behind this quantum speedup is amplitude amplification, a variant of Grover's algorithm. This technique leverages quantum superposition and entanglement in a clever way to accelerate the inference process.

In this paper, we recreate the theoretical results presented by Low et al. (2014) and extend their work with practical applications using real-world financial datasets. It is important to note that the original paper did not provide any code or practical demonstrations. To address this, we apply the quantum Bayesian inference algorithm to finance-related datasets, such as ExxonMobil's and Microsoft's 10-year financial data. These datasets are used to construct Bayesian networks with four and eight nodes. We also reference the implementation of the quantum Bayesian network as described by Borujeni et al. (2020). We implement structural learning algorithms, like Hill's algorithm, to create the corresponding Bayes network from the datasets.

For classical inference, we apply exact inference methods since our datasets are not large enough to necessitate approximate methods. We have also developed a general quantum code framework that could be adapted for larger datasets in future work. For quantum inference, we construct the corresponding quantum circuit for a given Bayesian network. This quantum circuit is then simulated using IBM's Qiskit simulator to compute marginal probabilities, assuming an ideal quantum setup (i.e., no noise).

We compare the results obtained from both quantum and classical Bayesian inference methods. Our findings demonstrate low error rates for both 4-node and 8-node networks, showcasing the scalability potential of the quantum approach. Additionally, we conduct an in-depth statistical analysis by running the quantum circuit 1,000 times to ensure consistency in the results, as quantum circuits can produce different outcomes across different runs.

II. Introduction

Bayesian inference is a well-known NP-hard problem, widely applied in fields such as finance due to its versatility. However, classical Bayesian inference methods are computationally intensive, often making them impractical for complex problems. This paper explores Quantum Bayesian Inference, which has been shown to offer quadratic speedup over traditional classical sampling techniques. In this study, we replicate and extend the results from a theoretical paper by implementing the corresponding code, which was not provided in the original work. Using two finance-related datasets, one with 4 nodes and the other with 8 nodes, we apply a structural learning algorithm to construct Bayesian networks. We then perform both classical and quantum inference on these networks and compare the results. The quantum inference consistently achieves an error rate of only 0.001% for both networks, demonstrating high accuracy and scalability potential. While exact inference is employed in this work due to the relatively small size of the datasets, this approach holds promise for scaling to larger models in future research, where classical sampling methods may struggle to achieve the same quadratic speedup.

A. Background

Here we discuss some of the necessary background needed for this paper.

A..1 Bayesian Networks

A **Bayesian Network (BN)** is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). This model enables the representation of complex probabilistic relationships among a large number of variables, which makes Bayesian Networks widely used in machine learning, bioinformatics, natural language processing, and many other fields.

Key Components of Bayesian Networks The main components of a Bayesian Network are:

1. **Nodes (Variables):** Each node in a Bayesian Network represents a random variable, which can be either discrete or continuous. For example, a node might represent an event such as *Rain* or *Traffic*.
2. **Edges (Dependencies):** Directed edges between nodes represent dependencies between the variables. If there is an edge from node A to node B, A is a *parent* of B, and B is *conditionally dependent* on A.
3. **Conditional Probability Distribution (CPD):** Each node has an associated conditional probability distribution that quantifies the effect of its parents on that node. For example, if X has parents A and B, its CPD is $P(X \mid A, B)$.

Joint Probability Representation A Bayesian Network expresses the joint probability distribution over a set of variables $X = \{X_1, X_2, \ldots, X_n\}$ in a factored form:

$$P(X) = \prod_{i=1}^{n} P(X_i \mid \mathrm{Pa}(X_i))$$

where $\mathrm{Pa}(X_i)$ denotes the set of parent nodes of X_i. This factorization allows efficient computation of the joint distribution, leveraging conditional independencies encoded in the network structure.

A..2 Structural Learning Algorithms

Structural learning refers to the process of determining the structure of a Bayesian Network, specifically identifying which variables are conditionally dependent and how they are connected. Structural learning algorithms are essential when the structure of the network is unknown, and they aim to create an accurate representation of dependencies among variables.

Types of Structural Learning Algorithms There are two primary categories of structural learning algorithms:

1. **Constraint-Based Algorithms:** These algorithms determine the structure by testing conditional independence relationships between variables. Some key steps include:

- *Independence Tests:* Check for conditional independence relationships between pairs of variables.
- *Edge Orientation:* Orient edges based on discovered dependencies and independencies.

Examples of constraint-based algorithms include the PC (Peter-Clark) algorithm, which iteratively removes edges between variables until a dependency structure emerges based on conditional independence tests.

2. **Score-Based Algorithms:** These algorithms use a scoring function to evaluate candidate network structures, selecting the one that best fits the data. Common scoring methods include:
   - *Bayesian Information Criterion (BIC):* A metric balancing model complexity with data fit.
   - *Bayesian Dirichlet Equivalent (BDe):* A scoring function that uses prior distributions on the structure.

   Score-based algorithms often involve heuristic or search methods like greedy search, hill climbing, or simulated annealing to identify the highest-scoring structure.

A..3 Bayesian Inference

Bayesian Inference is the process of updating beliefs or probabilities of variables in light of new evidence. In Bayesian Networks, inference aims to compute the posterior distribution of a subset of variables given observed values of others.
Key Inference Tasks in Bayesian Networks The primary tasks in Bayesian inference include:

1. **Marginalization:** Calculating the marginal probability of a subset of variables.
2. **Conditional Probability Queries:** Computing the probability of a variable conditioned on evidence.
3. **Maximum A Posteriori (MAP) Estimation:** Determining the most probable assignment for a set of variables given observed data.

Bayesian inference is generally computationally intensive, especially for networks with many variables and dependencies. Various algorithms, including exact and approximate methods, are used to perform inference efficiently.

A..4 Quantum Computing Basics

In this section, we offer a concise overview of qubits and the various quantum gates employed to execute transformations on them.
A qubit, short for quantum bit, is a fundamental unit of information in quantum computing, analogous to a classical bit in traditional computing. While a classical bit is restricted to one of two states, either 0 or 1, a qubit can exist in both states simultaneously. This characteristic of a qubit is referred to as quantum superposition. In quantum computing, the Dirac notation is used to represent the two basis states as $|0\rangle$ and $|1\rangle$. In general, any two orthonormal states can be used as the basis states but the commonly used basis states or computational basis are $|0\rangle$ and $|1\rangle$. The vector representations is given as,

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad\qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{1}$$

A general pure state of a qubit is a superposition, which is linear combination of the two basis states written as $|\Psi\rangle = c_1|0\rangle + c_2|1\rangle$, where c_1 and c_2 are complex numbers, which represent the probability amplitudes corresponding to $|0\rangle$ and $|1\rangle$ respectively. When a measurement is made, the qubit collapses to one of the two basis states. The probabilities of observing the qubit in $|0\rangle$ and $|1\rangle$ states are computed as the inner product of their probability amplitudes and their complex conjugates, represented as $c_1^\dagger c_1 = |c_1|^2$ and $c_2^\dagger c_2 = |c_2|^2$ respectively, where $c_1^\dagger$ and $c_2^\dagger$ are the complex conjugates of c_1 and c_2 respectively. Since a qubit can be either in $|0\rangle$ or in $|1\rangle$, the sum of their probabilities is equal to one ($|c_1|^2 + |c_2|^2 = 1$).

When multiple qubits are utilized in computing, their combined state can be determined by taking the tensor product of the individual qubits. For instance, if $|\Psi_1\rangle = a_1|0\rangle + a_2|1\rangle$ and $\Psi_2 = b_1|0\rangle + b_2|1\rangle$ represent two qubits with real-valued probability amplitudes, their joint state is expressed as $|\Psi_1\rangle \otimes |\Psi_2\rangle = a_1b_1|00\rangle + a_1b_2|01\rangle + a_2b_1|10\rangle + a_2b_2|11\rangle$. However, the joint state does not always have to be a product state (the tensor product of the individual states). There exist joint system states that cannot be decomposed into products of individual states, which are known as entangled states. Entanglement represents a form of quantum correlation that surpasses any classical correlation possible.

The states of these qubits are manipulated using quantum gates. Quantum gates are mathematical operations performed on the qubits to change their probability amplitudes to gain the desired computations. The quantum gates are similar to classical gates (such as the AND gate) acting on classical bits. There are elementary quantum gates that are needed for universal quantum computation. These gates can be used to make other complex gates. More information on the quantum gates can be found here
Rotational Gates: These gates, denoted as $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$, apply rotations around the X, Y, and Z axes of the Bloch sphere, respectively. They allow for continuous control over the qubit's state, enabling the transformation of qubits into superpositions or other desired states based on the angle θ. For instance, $R_x(\theta)$ rotates the qubit around the X-axis by an angle θ, changing its amplitude and phase.

Controlled Gates: These gates operate on two or more qubits, where the operation on a target qubit is conditioned on the state of a control qubit. The most common controlled gate is the Controlled-NOT (CNOT) gate, which flips the target qubit if the control qubit is in the state $|1\rangle$. Controlled gates are essential for creating entanglement between qubits and implementing quantum algorithms that require conditional operations.

Pauli-X Gate: Also known as the NOT gate or Pauli-X gate, the X gate flips the state of a single qubit from $|0\rangle$ to $|1\rangle$ and vice versa. Mathematically, it can be represented by the matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

B. Motivation

Bayesian inference is considered NP-hard due to the computational challenges associated with performing inference in complex probabilistic models. Specifically, the problem arises because computing the posterior distribution often involves integrating or summing over an exponential number of possible states or outcomes. Intuitively, these are the problems that are at least as hard as the NP-complete problems.
The precise definition here is that a problem X is NP-hard if there is an NP-complete problem Y, such that Y is reducible to X in polynomial time. However since any NP-complete problem can be reduced to any other NP-complete problem in polynomial time, all NP-complete problems can be reduced to any NP-hard problem in polynomial time. Then, if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP problems in polynomial time.

Bayesian inference is used in finance because it helps make better decisions by combining past knowledge with new data. It is great for dealing with uncertainty, which is everywhere in financial markets. For example, it can help manage risk, improve investment strategies, and make better predictions about future market behavior. It's especially useful when data is incomplete or noisy, and because financial conditions change all the time, Bayesian methods are good at adapting to new information as it comes in.
Classical Bayesian inference falls short in finance due to the field's inherent complexity, non-stationarity, and high-dimensional nature. Financial data often exhibit non-Gaussian characteristics like heavy tails and sudden jumps, which are challenging to model with traditional Bayesian methods. Additionally, classical approaches face scalability issues, struggling to handle the large volumes and rapid pace of financial data.

Current implementations of Bayesian networks encounter significant challenges, particularly the high computational costs associated with analyzing large numbers of nodes (random variables) in both forward and inverse analyses. One promising approach to mitigate these challenges involves leveraging the principles of quantum computing, or quantum-assisted computing. Quantum computers utilize the concepts of "superposition," which allows quantum systems to exist in multiple states simultaneously, and "entanglement," which

establishes correlations between quantum states that classical systems cannot replicate. Several quantum algorithms have demonstrated substantial computational advantages over their classical counterparts, with Shor's algorithm and Grover's algorithm being particularly noteworthy examples.

In Bayesian inference, the principle of amplitude amplification, an extension of Grover's algorithm, can be applied to achieve a quadratic speedup compared to classical inference methods (Low et al., 2014). Amplitude amplification increases the probability, or amplitude, of a desired outcome in a quantum system, making it more likely to be observed upon measurement. In classical computation, if you want to increase the chances of finding a specific solution in a search problem, you would typically repeat the process multiple times, linearly scaling the probability of success. However, quantum algorithms like amplitude amplification achieve a quadratic speedup, making them highly efficient.

In (Low et al., 2014), it has been theoretically demonstrated that the quantum rejection sampling algorithm, which utilizes the principles of quantum mechanics to transform a Bayesian network problem into a quantum circuit for inference, achieves a quadratic speedup compared to its classical counterpart. The key to this enhancement lies in the application of amplitude amplification, which effectively harnesses these quantum principles to optimize the sampling process. We describe the detailed process to convert Bayesian network to quantum circuit for quantum inference in Section G.

C. Contribution

In this paper, we identify two Bayesian networks in finance that possess significant inference capabilities. Recognizing that Bayesian inference is NP-hard, our goal is to transform this problem to achieve quantum inference, for which a quadratic speedup has been theoretically demonstrated (Low et al., 2014).

We first selected two finance-related problems and preprocessed the data to develop corresponding Bayesian networks. We also employed a structural learning algorithm for this purpose. Following that, we performed classical inference on these two Bayesian networks, which contained 4 and 8 nodes, respectively. To understand quantum inference, we meticulously reconstructed the methodologies outlined in papers (Borujeni et al., 2020) and (Low et al., 2014), gaining a comprehensive understanding of their approaches and developing code that was not provided in the original publications. This involved a detailed examination of how each Bayesian network is converted into its corresponding quantum circuit and how inference is obtained by simulating the quantum circuit.

Although the Bayesian networks analyzed in this study can be easily solved using classical inference methods, our work lays the groundwork for applying quantum inference techniques to larger networks, where classical methods may become infeasible.

III. Methodology

A. Proposed BayesNet

We have chose BNs for our problem statement. Classical Bayesian networks (BNs) are generally preferred over Markov networks (also known as Markov random fields) in several domains, including finance, for a few key reasons:

Directed vs. Undirected Models

- **Bayesian networks** use directed acyclic graphs (DAG) to display dependencies between different variables and, because of this, are effective in displaying *causal relations*. In finance, most relationships occur as causal (e.g., economic policies that impact interest rates, which correspondingly affect stock prices), and BNs capture these relationships very naturally.
- **Markov networks**, on the other hand, employ undirected graphs to represent relationships. They imply contingency without indicating direct causation. This lack of direction can make it harder to

interpret cause-and-effect relationships in finance, where directionality (e.g., the effect of economic indicators on stock prices) is often essential for predictive analysis.

Interpretability and Causal Inference

- When there is a need to establish a *cause-and-effect relationship*, Bayesian networks are more understandable, which is highly relevant in financial modeling. Investors and analysts must not only know the likelihood of specific events occurring but also understand *how one financial variable affects another*. These causal relationships can be more easily described in Bayesian networks than in traditional statistical models due to the higher interpretability of the former.
- In **Markov networks**, dependency relationships are usually reciprocal, which makes it challenging to determine which factors influence others directly. Because of this nature, they are less useful in financial modeling, where the relative impact of macroeconomic factors on asset prices is based on directional effects.

Data Requirements and Computational Efficiency

- Bayesian networks are generally faster for learning and inference than Markov networks, as they require fewer parameters due to conditional independence assumptions. In finance, where datasets are often large and computational resources may be limited, BNs are advantageous because they can approximate the corresponding model with fewer parameters than competitors. This reduces the risk of overfitting and makes training easier.
- In contrast, **Markov networks** often require more parameters to capture similar relational features due to their undirected framework. This can be an issue in finance, as they are computationally more demanding and prone to overfitting, especially when sample sizes are not large enough.

Application to Quantitative Finance

- In quantitative finance, Bayesian networks are effective for *risk management, asset valuation, and forecasting*—areas where the direction of certain factors, such as the influence of macroeconomic conditions on asset returns, is critical. For instance, in portfolio management, Bayesian networks assist in determining how external factors impacting one or multiple assets can influence investment decisions.
- Markov networks, on the other hand, are commonly used in fields such as image processing, where spatial or grid-like mutual dependencies are more important. However, their application is more limited in finance due to the reasons mentioned above.

B. Dataset

B..1 ExxonMobil 10-Year Financial and Market Data Analysis

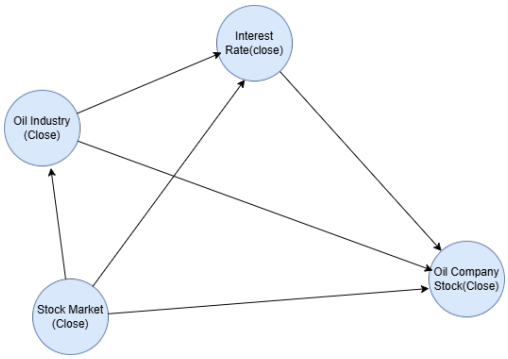

Figure 1: 4 Node Bayesian Network for ExxonMobil Finincial Data

This dataset gives a 10-year historical picture of the ExxonMobil stock in relation to some of the greatest economic indicators to enable long term assessments of the company in the light of other market and industrial structures. Below is a breakdown of each metric included in the dataset:

- **Date**: It maps one entry against a specific trading day, for a span of 10 years. This enables to analyze both short term fluctuates and long term trends in stock and economic metrics.
- **Oil Company Stock Price (Close) (USD)**: The closing stock price of ExxonMobil as an indication of the value of the business as evaluated in the market every day. It is this metric that is of importance in analyzing ExxonMobil shares .
- **Interest Rate (Close) (%)**: The closing interest rate for each day, represents the cost of borrowing. within the economy. ExxonMobil's operations, financing and stock price would be affected by interest rates. It also impacts company and consumers borrowing costs.
- **Stock Market (Close) (USD)**: SP, tends to be an indicator of where a particular market may be heading in the trade and the next year.
- **Oil Industry (Close) (USD)**: The closing value of an index representing the oil industry. This metric provides insight into the performance of ExxonMobil relative to other companies within the oil sector, revealing industry-specific trends and challenges over time.

With 10 years of data, this dataset allows for a comprehensive analysis of ExxonMobil's performance, both in absolute terms and relative to market and industry benchmarks. By examining these metrics, analysts can identify long-term trends, assess the impact of macroeconomic factors, and gain insights into ExxonMobil's resilience and adaptability within the oil industry.

B..2 Microsoft 10-year Financial Data

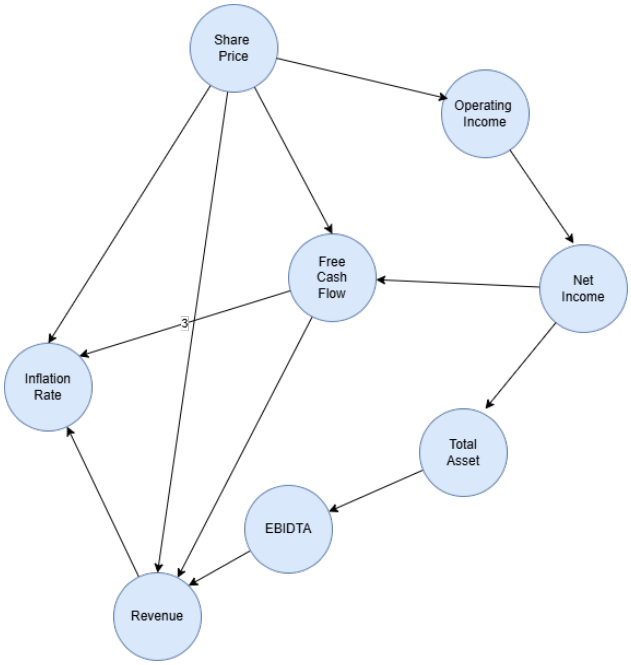

Figure 2: 8 Node Bayesian Network for Microsoft Finincial Data

The information presented in this dataset is a digest essence of Microsoft company's short-term financial and market data that describes several days of the indicator, which gives the necessary glimpse into the company's stability, profitability, and the public sentiment regarding the shares. Here's a breakdown of what each metric represents for Microsoft:

- **Timestamp**: The dates presented suggest precise days, which makes it comfortable to analyze short–term outcomes and fluctuation in shares' prices.
- **Inflation Rate (%)**:Connected with the overall economic situation in the Microsoft's operations and its ability to address the inflation rate.
- **Revenue (USD)**: Microsoft's total revenue from its products and services, or the rate at which it can sustain or expand the product's consumption.
- **EBITDA (USD)**:Earnings before interest Taxes, Depreciation, and Amortization is a measure of Microsoft business operating profit which provides a perspective on the company's ability to generate profits from its business operation exclusive of other influences.
- **Net Income (USD)**:The last of all profit after all the costs incurred has been taken, showing the ability of Microsoft to earn to the basics in terms of profit.
- **Market Cap (USD)**:Market capitalization is a valuation of the company in the market since it is calculated with the help of share price and number of outstanding shares. This enable one to determine the magnitude of Microsoft and its image in the market.
- **Operating Income (USD)**: The recreational operations' earnings before taxes and interests, showing the revenues Microsoft gets from the normal business functioning.
- **Free Cash Flow (USD)**: Operating cash flows available to invest in capital assets that indicate Microsoft's ability to fund growth, pay dividends or repay debts.
- **Total Assets (USD)**: always remain lower than their cost: Microsoft uses this approach to value its owned resources that can cover liabilities and measure its financial strength.
- **Share Price (USD)**: Microsofts daily price index on its stocks, therefore, as a measure of the company market value and investors perception of the firm.

Collectively, such information allows for the precise triangulation of Microsoft's financial vectors and market position within a specified timeframe. This also gives information concerning how those factors with macroeconomic connotations such as inflation may affect the company's statistics. It was useful for financial analysts and strategists to use this data in order to study performance trends and growth possibilities, as well as the Microsoft position on the market.

C. Process Flow Diagram

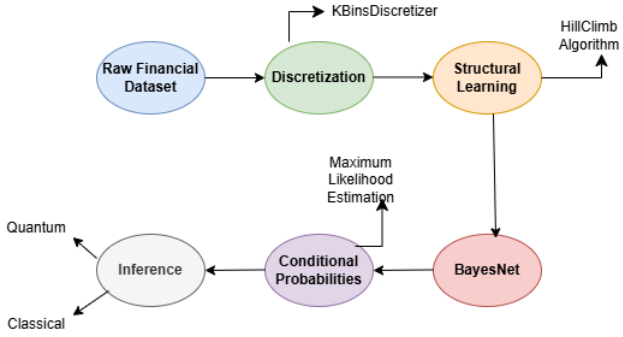

Figure 3: Process Flow diagram for the Classical Implementaton

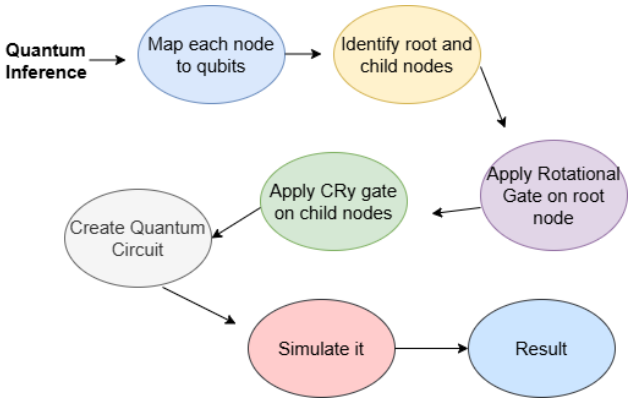

Figure 4: Process Flow diagram for the Quantum Implementation

D. Learning

The Hill Climb (HC) algorithm is a popular search algorithm used in Bayesian Network structure learning. It is essentially a local search optimization technique that starts with an initial structure (often an empty network) and iteratively improves it by making small changes. In Bayesian networks, this means adding, removing, or reversing edges between nodes to maximize a scoring criterion, such as the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC). Below is the step-by-step working process:

- **Initialization:** Start with an empty network or a simple initial structure.
- **Iteration:** At each step, consider all possible changes to the network:
  - Add an edge between two nodes if it doesn't already exist.
  - Remove an existing edge between two nodes.
  - Reverse an existing edge's direction between two nodes.
- **Score and Evaluate:** Each possible change is scored using a predefined criterion (such as BIC). The algorithm picks the change that maximally increases the network score or minimally decreases it if no positive change is possible.
- **Stopping Condition:** The process continues until no single modification yields an improvement in the score. The resulting structure is considered a local maximum with respect to the scoring function.

The PC (Peter-Clark) algorithm is another well-known method for learning the structure of Bayesian networks, particularly useful for constraint-based learning. However, the PC algorithm may not be suitable in this case due to the nature of the dataset and the type of dependencies involved. Here are key reasons why the PC algorithm would not work effectively in this case:

- **Data Sensitivity:** The PC algorithm relies heavily on independence tests (testing if two variables don't affect each other) to figure out which edges should connect nodes. These tests work well if you have a large, clean dataset, but they can be unreliable when data is limited or noisy. With our dataset sampled and discretized (turning continuous values into "High" and "Low" categories), the PC algorithm might struggle to identify genuine connections and could make errors
- **Discretization Challenges:** Since we turned continuous data into just two categories, we lose some subtlety in the data. PC relies on independence tests that require well-behaved data distributions, and this type of transformation can sometimes hide real dependencies or introduce false ones, making the PC algorithm's output less reliable. Hill Climb, however, is more flexible with these simpler, categorical data points, as it focuses on scoring the whole network instead of relying on fine-grained independence tests.

- **Structure Complexity:** In financial datasets, variables like stock prices and economic indicators are usually very interconnected, leading to more dense relationships. PC is better for simpler structures, where only a few variables interact, but it can struggle with densely connected data like ours. Hill Climb, by optimizing the network incrementally, is better suited to capture the more complex web of dependencies we expect in financial data.

In short, Hill Climb fits our needs because it builds a structure based on overall score improvements, making it more robust to discretization. The PC algorithm, on the other hand, can miss these dense relationships in smaller or noisier datasets, making it less reliable for this kind of financial modeling.

E. Computing Conditional Probabilities

After, getting the Bayesnet with the help of Hill Climb algorithm, the next task is to get the conditional probabilities, which are used to get classical and quantum inferences. To get the conditional probabilities we use, MaximumLikelihoodEstimator provided by the pgmpy library. Here are the necessary steps:

- Import necessary modules, `pgmpy` for model estimation and `matplotlib` with `networkx` for visualization.
- Estimate CPDs (conditional probability distributions) using maximum likelihood estimation based on the data.
- For each variable, print the estimated probabilities. This part formats the output for readability, distinguishing between multi-dimensional and single-dimensional CPDs.
- Finally, we create and display a directed graph of the Bayesian Network structure, which shows the relationships between variables.

Figure 6 and 5 show the Bayesnet of 4 nodes and 8 nodes respectively, with the obtained conditional probabilities by using MLE.

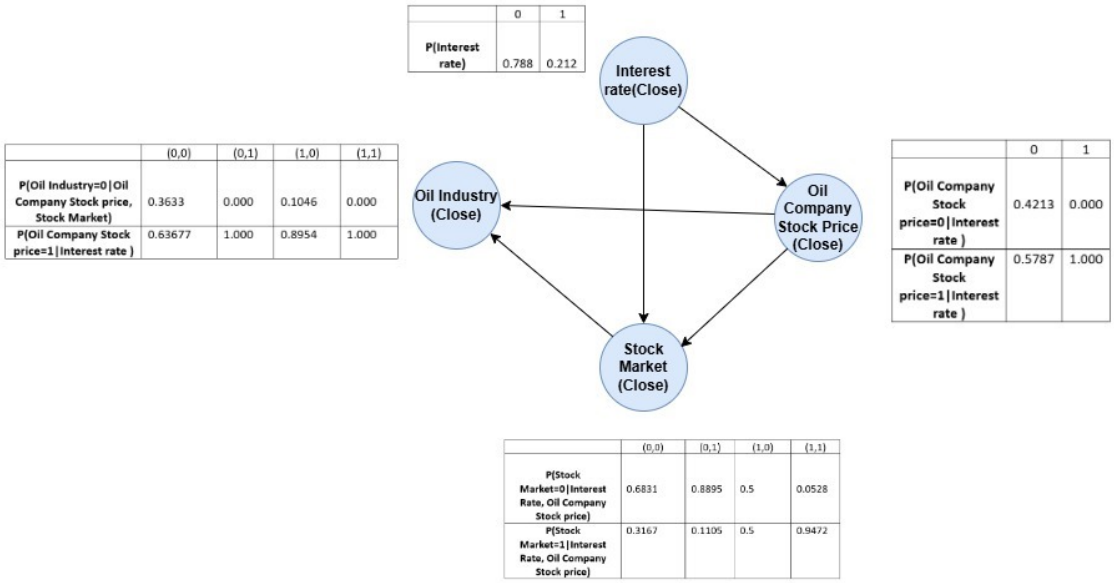

Figure 5: Bayesnet with Conditional Probabilities for 4-node

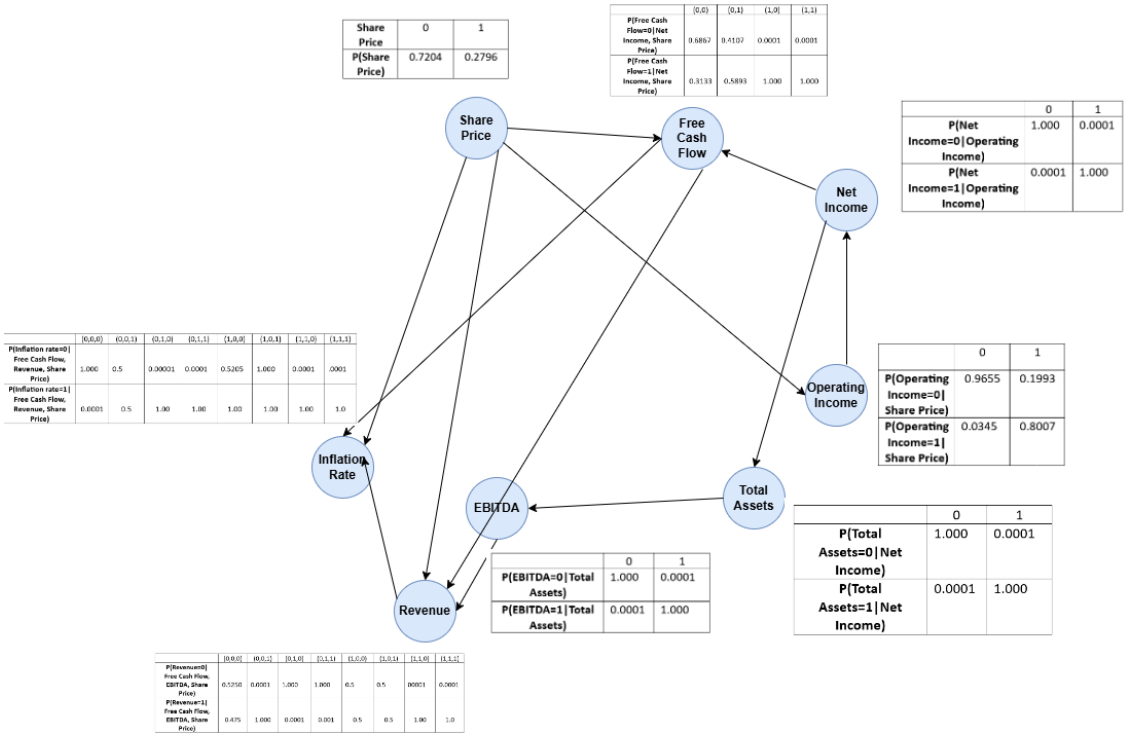

Figure 6: Bayesnet with Conditional Probabilities for 8-node

F. Classical Inference

The steps to infer probabilities in a classical Bayesian network are detailed below.

1. Model each node in a Bayesian network by a separate stochastic variable.
2. The marginal probability of a node is defined as its conditional probability with respect to the rest of nodes of the network.
3. Do computations using such probabilities in order to find the joint probabilities and marginal probabilities.

In this project, each node represents a two-state variable, similar to the binary states (0 and 1) in the quantum section. These states correspond to outcomes within the dataset. For a Bayesian network with n nodes, each node V_i (where $i \in n$) is associated with its own probability distribution, where each state is defined by the observed dataset.

Then, we single out endogenous nodes, which have no other nodes that depend on them (they are the root nodes) and exogenous nodes, which are required by other nodes (they are called child nodes). For the root nodes, marginal probabilities are computed directly from the dataset: Number of occurrences of state 0 divided by the total size of the data is $P(V_i = 0)$, and same for state 1 is $P(V_i = 1)$. The calculation of these marginal probabilities is further described in Section F..1.

That for the child nodes, the probabilities are conditional probabilities which means they depend on the states of other parent nodes they have. These conditional probabilities, which are symbolized as $P(V_i|\Pi_{V_i})$, where Π_{V_i} define the parent nodes for the V_i node, are obtained from the dataset by computing the frequency of each state combination. They are stored in conditional probability tables (CPTs) and these enable us to deduce the state probabilities of the child nodes.

F..1 Marginal Probability Computation for Root Nodes

To compute the marginal probabilities for a root node V_i, we examine the frequency of its states across all data entries. These frequencies are normalized to obtain probabilities for states 0 and 1, respectively. Given that a node V_i has two states, we express these probabilities as $P(V_i = 0)$ and $P(V_i = 1)$. This approach enables us to directly compute the probability distribution for each root node without needing conditional calculations, as root nodes have no dependencies.

F..2 Conditional Probability Computation for Child Nodes

For each child node V_i with parent nodes Π_{V_i}, we calculate conditional probabilities $P(V_i = 0|\Pi_{V_i})$ and $P(V_i = 1|\Pi_{V_i})$ by counting the occurrences of each combination of parent states in the dataset. Let $\Pi^*_{V_i}$ represent a particular configuration of the parent nodes' states. Then, for each possible state of V_i, the conditional probability is calculated as the frequency of V_i in each parent state configuration, normalized by the total occurrences of $\Pi^*_{V_i}$ in the dataset:

$$P(V_i|\Pi^*_{V_i}) = \frac{\text{Frequency of } V_i \text{ given } \Pi^*_{V_i}}{\text{Total occurrences of } \Pi^*_{V_i}} \tag{2}$$

These conditional probabilities are crucial for inferring the state of each child node, as they provide insight into how the child node's states depend on its parent nodes within the network.

F..3 Classical Network Inference Process

After determining the marginal and conditional probabilities, inference in the classical Bayesian network is performed by leveraging the joint probability distribution, which is factored to allow for the computation of marginal and conditional probabilities for each node. For example, when calculating the marginal probability of a child node, a sum is taken over the joint probabilities of all possible configurations of its parent nodes. This approach enables the evaluation of each node's state and the interactions and dependencies between nodes within the network.

Exact inference is achieved by recursively calculating the marginal probability for each child node, beginning with the root nodes (which have direct probabilities). The process proceeds by working through the child nodes using conditional probability tables (CPTs). This recursive method, in combination with the use of CPTs, ensures that all dependencies within the network are correctly accounted for, providing an accurate and comprehensive inference.

G. Quantum Inference

The steps to convert a Bayesian network to a quantum circuit are mentioned below
1. Map each node in a Bayesian network to qubits in a quantum cricuit
2. Map the marginal/conditional probabilities of each node to the probability amplitudes associated with various states of the qubit(s).
3. Realize the required probability amplitudes of quantum states using (controlled) rotation gates.

The nodes in the Bayesian networks presented in this paper have two states (0 and 1). Since a qubit can represent these two states ($|0\rangle$ and $|1\rangle$), we can map each node to a distinct qubit. Suppose there are n nodes in the Bayesian network; then, each node i (where $i \in n$) is mapped to a qubit q_i. The initial state of each qubit is set to $|0\rangle$.

Next, we identify the root nodes and the child nodes within the given Bayesian network. For all qubits corresponding to the root nodes, we need to apply rotation gates (R_Y) with angles (θ_i) that produce superposed quantum states, whose probabilities correspond to the probabilities of the respective root nodes. The calculation of these angles is discussed in Section G..1.

For the qubits corresponding to the child nodes, we will have different rotation angles conditioned on the root nodes upon which the child nodes depend. Based on the combinations of parent node values, each

combination will yield a distinct rotation angle. These conditional rotations are implemented using controlled rotation gates, and their angles depend on the conditional probabilities of the child nodes. The angles for these rotations are represented as $\theta_{C,ij}$, where C denotes the child node and i, j represent the parent nodes, which can take on combinations of 0s and 1s.

G..1 Rotation angle computation

We can represent a two-state node of a bayesian network using a single qubit. For the root nodes we apply an R_Y gate with an appropriate angle. The probabilities of the root node are mapped to the probabilities (and thus probability amplitudes) of the basis states, $|0\rangle$ and $|1\rangle$. Let θ_{V_i} represent the rotation angle associated with a two-state root node, V_i. Given the initial state of a qubit as $|0\rangle$, the application of $R_Y(\theta)$ will transform $|0\rangle$ to $\cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)|1\rangle$. Thus, the probabilities associated with the $|0\rangle$ and $|1\rangle$ states are equal to $\cos^2\left(\frac{\theta}{2}\right)$ and $\sin^2\left(\frac{\theta}{2}\right)$ respectively. Then, $P(V_i = 0)$ and $P(V_i = 1)$ represent the probabilities of states 0 and 1 of V_i. Thus, we compute the rotational angle as,

$$\theta_{V_i} = 2 \times \tan^{-1}\sqrt{\frac{P(|1\rangle)}{P(|0\rangle)}} = 2 \times \tan^{-1}\sqrt{\frac{P(V_i = 1)}{P(V_i = 0)}} \tag{3}$$

In Eq. (3), $P(|0\rangle)$ and $P(|1\rangle)$ represent the probabilities of a qubit to be in $|0\rangle$ and $|1\rangle$ respectively. Since we map the nodal probabilities to the probabilities of quantum states, $P(|0\rangle)$ and $P(|1\rangle)$ are replaced with $P(V_i = 0)$ and $P(V_i = 1)$ respectively. Therefore, two-state root nodes can be represented using an R_Y gate with a rotation angle of $2 \times \tan^{-1}\sqrt{\frac{P(V_i = 1)}{P(V_i = 0)}}$.

We can also compute the rotation angles associated with conditional probabilities of two-state child nodes using Eq. (3). Let V_i and Π_{V_i} represent a child node and set of its parent nodes respectively. For each combination of parent node values, $\Pi_{V_i} = \Pi^*_{V_i}$, we have probabilities for $V_i = 0$ and $V_i = 1$ denoted as $P(V_i = 0|\Pi_{V_i} = \Pi^*_{V_i})$ and $P(V_i = 1|\Pi_{V_i} = \Pi^*_{V_i})$ respectively. The rotation angle associated with V_i when $\Pi_{V_i} = \Pi^*_{V_i}$, which is denoted by $\theta_{V_i,\Pi^*_{V_i}}$ can be calculated as

$$\theta_{V_i,\Pi^*_{V_i}} = 2 \times \tan^{-1}\sqrt{\frac{P(V_i = 1|\Pi_{V_i} = \Pi^*_{V_i})}{P(V_i = 0|\Pi_{V_i} = \Pi^*_{V_i})}} \tag{4}$$

The conditional probabilities for child nodes are implemented using controlled rotations. Since controlled rotation gates are not considered elementary gates, they must be broken down into a combination of single-qubit and two-qubit elementary gates.

G..2 Representing two-state child nodes with multiple two-state parent nodes

Let us consider the representation of child nodes with one parent node, and then we consider the case with multiple parent nodes. A two-state child node with one parent node can be represented using two CR_Y gates. This is based on the assumption that the parent node has two states. The rotation angles are computed using Eq.(4), based on the parent node values.

For the case of multiple parents, let n represent the number of parent nodes for a child node, V_i. The conditional probabilities of V_i can be stated using C^nR_Y gate where the n control qubits are the n qubits corresponding to the n parent nodes and the target qubit represents the child node. For example, consider a child node C with $n = 2$; in this scenario, we would apply a CCR_Y or C^2R_Y gate to show the conditional probabilities of C. However, C^nR_Y is not an elementary gate and will need to be decomposed into elementary gates.

In the paper (Borujeni et al., 2020), the authors detail the decomposition of such multi-controlled gates, which are not elementary and may occasionally require additional qubits, referred to as ancillary qubits. Fortunately, we can bypass this manual decomposition because the library we employ for our coding, Qiskit,

manages it automatically. Developed by the IBM Quantum team, Qiskit is a comprehensive quantum software development toolkit that facilitates the creation and simulation of quantum circuits. With the recent introduction of the multi-controlled rotational gate (MCRY Gate), we can directly implement this functionality for child nodes that have more than one parent.

G..3 Quantum Circuit Generation

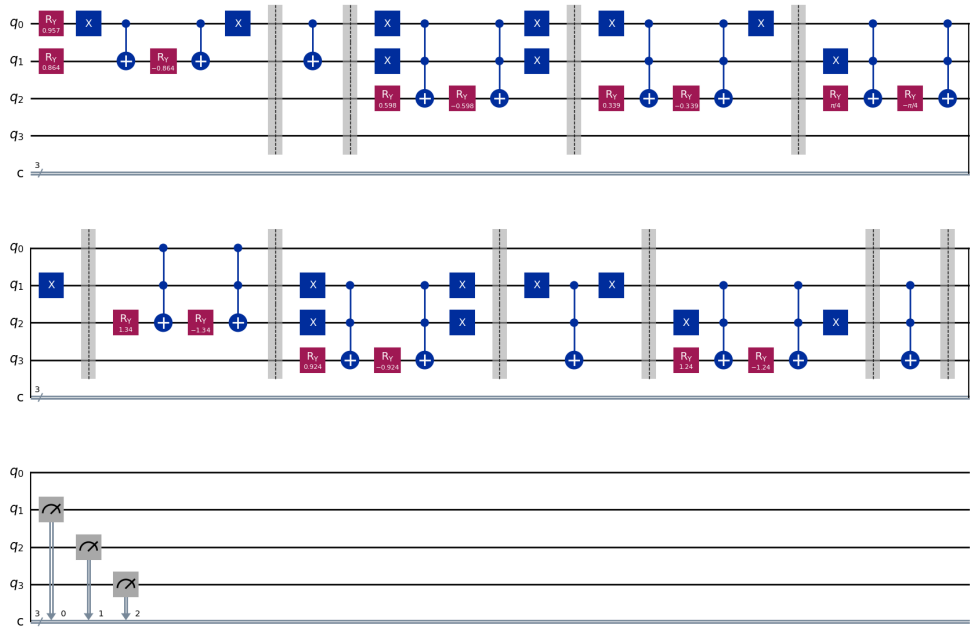

Figure 7: Quantum circuit representing the 4-node Bayesian network as outlined in Section B..1. The circuit illustrates the relationships between parent and child nodes, showcasing the application of controlled gates based on probabilities.

Building upon the previously described steps, we can construct a quantum circuit tailored for any given Bayesian network featuring nodes with two distinct states. We present an example of the quantum circuit generated for the four-node Bayesian network, as outlined in Section B..1. The circuit in Figure 7 demonstrates the application of the specified gates and showcases how the relationships and dependencies between the nodes are represented in the quantum circuit.

In Figure 7, we first apply the rotational gate (R_Y Gate) on the root node i.e interest rate. Then based on the conditional probabilities of the corresponding child nodes, we apply the multicontrolled rotational gate. Since Oil Company Stock Price depends on Interest Rate, we apply CR_Y Gate. For Stock Market and Oil Industry child nodes we use CCR_Y Gate. We also change the qubit state from 0 to 1 and vice-versa using X Gate and CNOT Gate based on the conditional probability table. The theta values for all rotational and controlled-rotational gate is calculated using the method described in Section G..1.

G..4 Simulating the Quantum Circuits

Once the quantum circuit is generated, we execute it to obtain the marginal probabilities for all child nodes in a single run using IBM Qiskit's AER Simulator. The AER Simulator is a powerful tool that allows researchers to simulate quantum circuits on classical computers, providing a platform to validate quantum algorithms without needing access to physical quantum hardware. It efficiently models the behavior of quantum systems for less qubits, allowing for the exploration of circuit performance under various conditions, including ideal and noisy scenarios. Note that in this simulation, we use the ideal conditions and do not account for any noisy scenario.

In contrast, running circuits on actual quantum computers introduces complexities such as inherent noise, qubit decoherence, and gate fidelity, which can lead to varied outcomes across multiple executions. Moreover, utilizing real quantum computers typically incurs additional costs, such as fees associated with accessing cloud-based quantum services, and users may experience queue times due to demand.

We have opted to use the AER Simulator for our test runs due to its simplicity, speed, and absence of additional costs. In quantum computing, obtaining reliable results typically requires executing multiple shots of a circuit to accumulate sufficient statistical data. Consequently, we run our quantum circuit multiple times, referred to as "shots," and calculate the statistical average of the outcomes to derive our final results. We use 256 shots for all simulations. This approach not only enhances the reliability of our findings but also streamlines the testing process, allowing us to efficiently explore and validate our quantum algorithms.

IV. Results and Inferences

A. Node-4 Bayesian Network

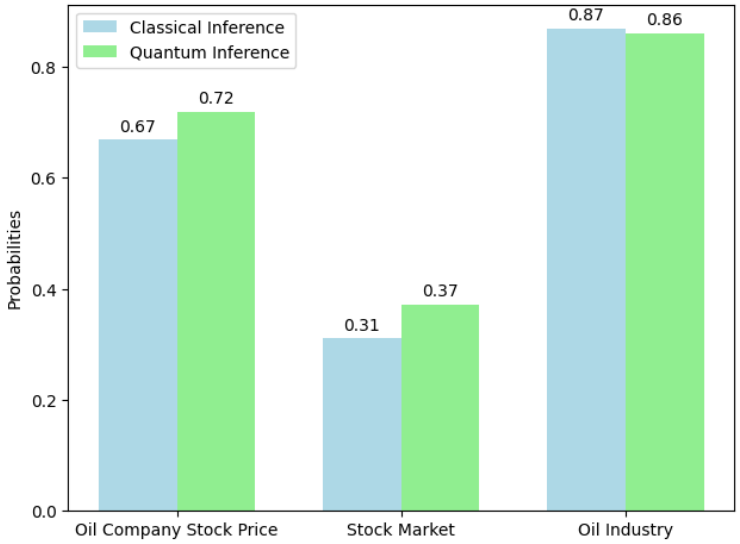

Figure 8: Marginal probability results for the 4-node Bayesian network as described in Section B..1.

Figure 8, shows the marginal probabilities obtained for the child nodes in the 4-node Bayesian network described in Section B..1. This is obtained after running the quantum circuit as shown in Figure 7 using the Qiskit AER simulator. The obtained probabilities are similar to those obtained using classical inference.

As mentioned earlier, we executed 256 shots of the quantum circuit to estimate the marginal probabilities. Since these probabilities are derived from data, variations may occur across different runs of the circuit. To assess this variability, we ran the circuit $r = 1000$ times, collecting marginal probability values from each execution. Given the simulation results from r runs, we compute the $(1 - \alpha)$ confidence intervals of the estimated marginal probabilities and check if the marginal probabilities from classical inference fall within the estimated intervals. We consider the case of the Oil Company Stock Price node and look at the statistical data across multiple runs on the quantum circuit. The analysis for the Oil Company Stock Price node produced several key results as shown in Figure 9. The classical probability was found to be 0.668, while the mean probability derived from 1000 quantum circuit runs was 0.66788. The 95% confidence interval for the estimated marginal probabilities ranged from 0.66596 to 0.66980, indicating that the classical estimate falls within this interval. Additionally, the standard deviation of the quantum circuit results was 0.03097, reflecting some variability in the outcomes, and the percent error was remarkably low at 0.001%, demonstrating a high level of accuracy in the quantum estimation compared to the classical benchmark.

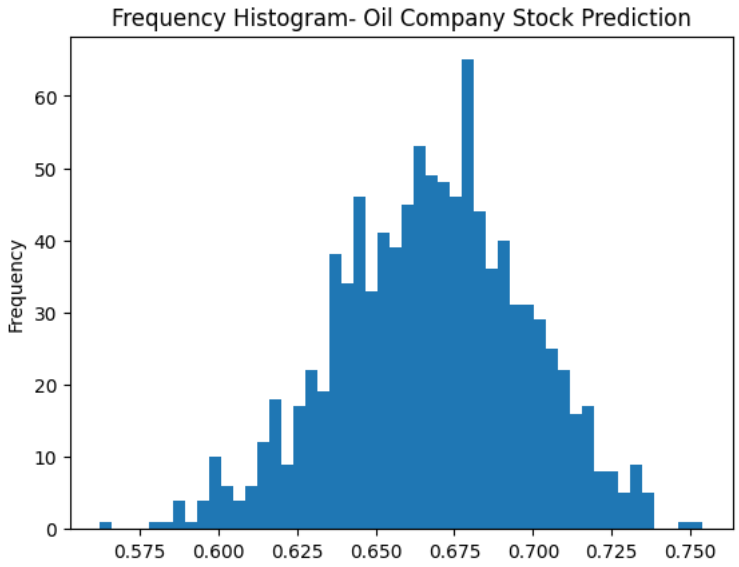

Figure 9: Frequency histogram after multiple runs for Oil Company Stock Prediction node.

B. Node-8 Bayesian Network

A similar analysis is conducted for the 8-node Bayesian Network as described in Section B..2. We compare the marginal probabilities of all child nodes, as shown in Figure 10. We also conduct statistical analysis for the Net income node by running on the quantum circuit 1000 times in Figure 11. Consistent with the 4-node case, we observe an extremely low error rate of 0.001%, demonstrating high measurement accuracy. These results indicate a precise and reliable value with minimal deviation from the expected mean. Moreover, this analysis confirms that the error rate remains consistently low even when scaling the Bayesian Network.

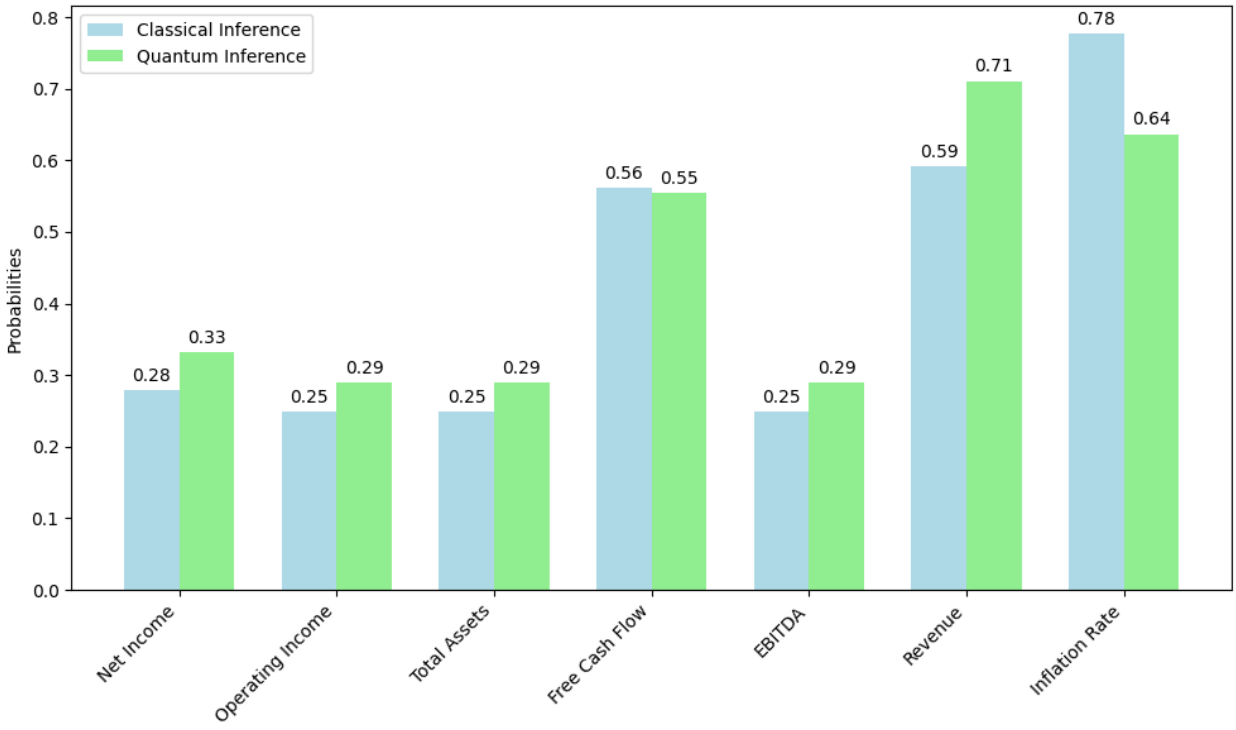

Figure 10: Marginal probability results for the 8-node Bayesian network as described in Section B..2.

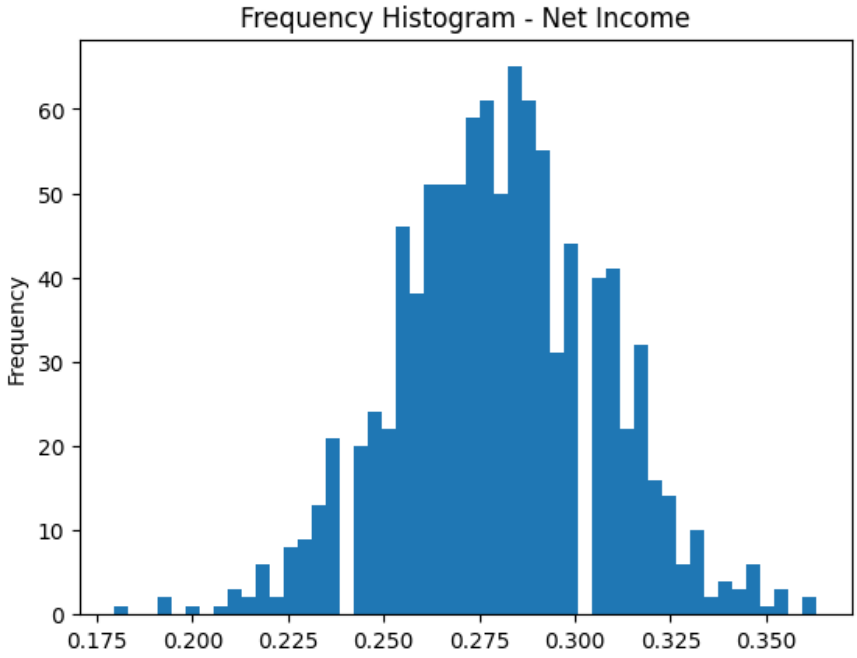

Figure 11: Frequency histogram after multiple runs for Net Income node.

References

[1] Adhitama, Ria Puan, Dewi Retno Sari Saputro, and Sutanto Sutanto. "HILL CLIMBING ALGORITHM ON BAYESIAN NETWORK TO DETERMINE PROBABILITY VALUE OF SYMPTOMS AND EYE DISEASE." BAREKENG: Jurnal Ilmu Matematika dan Terapan 16.4 (2022): 1271-1282.

[2] Piot, Mélanie, et al. "Bayesian Network structure learning algorithm for highly missing and non imputable data: Application to breast cancer radiotherapy data." Artificial Intelligence in Medicine 147 (2024): 102743.

[3] Low, G. H., Yoder, T. J., Chuang, I. L. (2014). Quantum inference on Bayesian networks. Physical Review A, 89(6). https://doi.org/10.1103/physreva.89.062315

[4] Borujeni, S. E., Nannapaneni, S., Nguyen, N. H., Behrman, E. C., Steck, J. E. (2020, April 29). Quantum circuit representation of Bayesian networks. arXiv.org. https://arxiv.org/abs/2004.14803

[5] Moreira, Catarina, and Andreas Wichert. Are quantum-like Bayesian networks more powerful than classical Bayesian networks?. Journal of Mathematical Psychology 82 (2018): 73-83.

[6] https://stackoverflow.com/questions/1857244/what-are-the-differences-between-np-np-complete-and-np-hard