Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en en Inteligencia Artificial

Trabajo Fin de Máster

Interactive Structural Learning for Discrete Bayesian Network Classifiers

Author: Iván Eugenio Tello López
Tutors: Pedro Larrañaga Múgica y Concha Bielza Lozoya

Madrid, July 2023

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en en Inteligencia Artificial

Título: Interactive Structural Learning for Discrete Bayesian Network Classifiers

July 2023

Author: Iván Eugenio Tello López
Tutors: Pedro Larrañaga Múgica y Concha Bielza Lozoya
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Acknowledgments

Thanks, primarily to my mentors, Pedro and Concha, who have helped and guided me throughout this work, teaching me what research work is like. Without them, this would not have been possible.

To my fellow research group members for always being there to help me and creating a comfortable working environment.

To my parents, who are the cornerstone of my life and have been supporting me since forever. And to my best friend, even though we are no longer studying together, she is always there to support me and help me.

This work was partially supported by the Spanish Ministry of Science and Innovation through the PID2019-109247GB-I00 and TED2021-131310B-I00 projects.

Resumen

Debido al auge de la Inteligencia Artificial, el uso de ella se ha vuelto cada vez más cotidiano, utilizándola incluso para tomar decisiones en campos como el sanitario o el financiero, en los que es necesario entender los resultados o predicciones que proporciona el modelo. Debido a esto, surgen dos ramas en las que dividir los modelos dentro de la .Explainable AI"(XAI): los modelos interpretables y los modelos explicables.

Los modelos explicables son aquellos que necesitan de otro modelo u otras técnicas para entender las decisiones tomadas por este. En cambio, los modelos interpretables son entendibles directamente por el ser humano. Es por eso que muchos autores indican que el futuro debería ser el estudio de los modelos encuadrados en la segunda categoría. Uno de los modelos más importantes de esta categoría son las redes Bayesianas.

En este trabajo, se presenta la implementación de una interfaz gráfica y de diferentes técnicas para mejorar la interpretabilidad intrínseca de las redes Bayesianas, en concreto de las redes Bayesianas aplicadas a la clasificación supervisada: los clasificadores Bayesianos.

Además de esto, se propone un nuevo método para encontrar el manto de Markov óptimo de la variable clase, esto es importante ya que el manto de Markov indica cuáles son las variables que afectan más a la variable clase permitiendo así descartar el resto, ya que no aportan información extra, esta característica se usa sobre todo en problemas de selección de variables, un proceso indispensable para problemas que utilicen muchas variables. Para encontrarlo, se ha hecho uso de los algoritmos de estimación de distribución (EDAs en su acrónimo en lengua inglesa), se han escogido este tipo de algoritmos ya que en espacios de búsqueda muy grandes, las búsquedas exhaustivas son intratables por lo que aunque no aseguren alcanzar el óptimo, por las ventajas que ofrencen en términos de rendimiento y de computación los EDAs pueden ser una buena solución a este problema.

Además al nuevo modelo se le aplicaran diferentes técnicas de interpretabilidad además de incluir nuevas técnicas para facilitar al usuario el entendimiento de la evolución de las soluciones a lo largo de las generaciones.

Abstract

Due to the rise of Artificial Intelligence, its use has become more and more commonplace, even being used to make decisions in fields such as healthcare or finance, where it is necessary to understand the results or predictions provided by the model. Because of this, there are two branches into which models can be divided: interpretable models and explainable models.

Explainable models are those that need another model or other techniques to understand the decisions made by the model. On the other hand, interpretable models are directly understandable by humans. This is why some authors indicate that the future should be the study of models of the second category. One of the most important models in this category are Bayesian networks.

In this work, we present the implementation of a graphical interface with interaction and different techniques to improve the intrinsic interpretability of Bayesian networks, specifically Bayesian networks applied to supervised classification: Bayesian classifiers.

In addition to this, a new method to find the optimal Markov blanket of the class variable. This is important since the Markov blanket indicates which variables affect the class variable the most, thus allowing to discard the rest, since they do not provide extra information. This feature is used mainly in variable selection problems, an indispensable process for problems that use many variables. To find it To find it, we have made use of Estimation of Distribution Algorithms (EDAs) is proposed. This type of algorithms has been chosen because in very large search spaces, exhaustive searches are intractable, so although they do not ensure reaching the optimum, due to the advantages they offer in terms of performance and computation, EDAs can be a good solution to this problem.

In addition, different interpretability techniques will be applied to the new model and new techniques will be included to facilitate the user's understanding of the evolution of the solutions throughout the generations.

Table of contents

List of figures

Chapter 1

Introduction

Over the past few years, Artificial Intelligence (AI) has gained relevance across numerous fields that are embracing emerging information technologies. Furthermore, the availability of vast datasets and improved computing power has enabled significant progress in addressing complex computational tasks, leading to the development of increasingly complex models and technologies. However, this complexity often hampers our ability to comprehend and interpret these models effectively. That is why ensuring transparency and interpretability of the AI models has emerged as a critical concern.

For addressing this problem, explainability aims to bridge the gap between AI's complex inner workings and human comprehension by providing interpretable and meaningful explanations for AI-based decisions. To achieve this objective AI systems must elucidate the reasoning, factors, and rationale behind their outputs, enabling humans to understand and evaluate its behavior. This transparency is essential for building trust, assuring safety, and facilitating the effective collaboration between humans and AI particularly in high-stakes domains such as healthcare, finance, criminal justice, and others.

Achieving this transparency is not an easy task, especially in black-box models. This term refers to models whose internal workings and decision-making processes are opaque for interpretation, making it challenging to comprehend how and why specific decisions or predictions are made. This transparency problem is usually addressed by post-hoc explanations.

Post-hoc explanations refers to the retrospective analysis and interpretation of the decision-making process of an already trained and deployed machine learning or AI model. It focuses on providing explanations and insights into the model's outputs and reasoning after the study has been concluded and predictions are already done.

Unlike other methods or techniques, post-hoc explanations are applied to existing black-box models to uncover their internal workings and provide understandable justifications for their outputs by analyzing the model's behavior by examining its internal representations, such as feature importance. These methods aim to identify which input features or factors have influenced the model's decision-making process the most.

Although these explanations offer insights into the model's decisions, they are limited to considering the input and output alone. As a result, they do not offer deeper insights into the inner workings of the model beyond the outcomes, which is why many authors such as Rudin (2019), consider this branch of thinking unuseful and likely to perpetuate bad practices.

Consequently, the presence of more naturally transparent models, known as interpretable models, is needed, especially in high-stakes domains. We will focus throughout this work on one of such models, Bayesian networks, which hold significant importance on AI field.

1.1. Motivation

Interpretability in AI holds significant importance, especially in high-stakes domains where the consequences of AI system decisions can have direct impact on individuals and society as a whole. In such domains, it is essential to understand which is the process an AI system did to get a particular decision or prediction. This understanding allows stakeholders, including users, regulators, and domain experts, to evaluate the system's reliability, fairness, and potential biases.

Additionally, interpretability helps build trust and acceptance of AI systems making them more similar to human thinking, so users are more likely to adopt AI technologies when they are able to understand and comprehend the reasoning behind their outputs.

In the context of supervised classification, where the goal is to predict the class or category of a given input based on labeled training data, interpretability plays a critical role. When deploying AI models for tasks like fraud detection or medical diagnosis, it is essential to have insights into the factors influencing the model's predictions.

Interpretability enables the identification of key features, relationships, and making explanations about the prediction that contribute to the classification outcome. This information not only enhances the model's interpretability but also enables domain experts to validate and potentially refine the model, improving its overall accuracy and reliability.

Among the various machine learning models, Bayesian networks stand out for their innate transparency, making them particularly promising in the pursuit of interpretable AI. The key strength of Bayesian networks lies in their ability to provide transparent and interpretable explanations for their predictions thanks to their innate probabilistic nature.

The graphical structure of the Bayesian network visually helps the user to understand the relationships between variables, making the understanding of how information flows and influences the final classification easier.

By leveraging the principles of probability theory, Bayesian networks enable the identification of causal relationships, the exploration of feature dependencies, and the impact of evidence on the final prediction. This inherent transparency makes Bayesian networks an ideal choice in domains where understanding the decision-making process is of utmost importance.

Not only Bayesian networks are transparent but also they hold an advantage over other AI models: the integration of domain knowledge and expert input with the data. Prior beliefs and domain-specific constraints can be explicitly incorporated into the network structure and the local probability distributions, allowing for a better outcome, improving performance and granting user interaction.

In conclusion, the ongoing debate around the choice between using explainable or adopting interpretable models persists as the demand for transparency and interpretability in AI systems continues growing.

Within this paradigm, Bayesian networks, with their innate interpretability and ability to model complex dependencies, hold great potential for providing interpretable and trustworthy solutions.

Which is why, this work aims to show the advantages of using them over explainable machine learning models and other natural transparent machine learning models.

1.2. Fundamentals

1.2.1. Introduction to Bayesian networks

Bayesian networks (Pearl (1988)) are a specific type of probabilistic graphical model. A probabilistic graphical model is a tool used to visualize and work with variables, particularly with the conditional independences among them.

Two variables that are conditionally independent given another variable have no direct impact between them.

Probabilistic graphical models are usually represented with a graph structure. This structure is composed of a set of nodes, that reperesents the variables, and a set of edges, they can be directed or undirected.

In particular, Bayesian network only allow a specific graph structures which is a directed acyclic graphs (DAGs), that means all edges must be directed and cyclic connections are not permitted within the graph.

A Bayesian network encodes a joint probability distribution (JPD). In general given a set of n random variables $\mathrm{X} = \{X_1, ..., X_n\}$ the JPD factorises as:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i | Pa(X_i)),$$

where $Pa(X_i)$ is the set of the parents of X_i. The JPD is the product of the probabilities of each variable given its parents' values. In practice assuming complete dependence among variables would make the JPD intractable, requiring an excessive number of parameters (i.e. $2^n - 1$ in binary domains).

That is why taking advantage of conditional independence between variables is needed.

Formally, an event α is independent of an event β in P, denoted:

$$P \models (\alpha \perp \beta), \text{ if } P(\alpha|\beta) = P(\alpha) \text{or if} P(\beta) = 0$$

.

Independence is a strong property so it is not so commonly given, which is why we need another concept that can occur more often such as conditional independence, which is when two events are independent given an additional event.

The definition would be: An event α is conditionally independent of event β given event γ in P, denoted

$$P \models (\alpha \perp \beta|\gamma), \text{ if } P(\alpha|\beta \cap \gamma) = P(\alpha|\gamma) \text{or if} P(\beta \cap \gamma) = 0$$

.

One of the main advantages of conditional independence is that the computation cost of JPD is reduced.

An efficient way to identify condit. independences in the graph is using d-separation criterion (Koller and Friedman (2009)), which is defined as:

A set Z of nodes d-separates X and Y if in all undirected paths between any node of X and any node of Y in the Bayesian network, there is an intermediate node C such that either:

- C is a converging connection in the path and C and its descendants do not belong to Z
- C is not a converging connection and it belongs to Z

Given the d-separation theorem of Verma and Pearl (1990), it states that if there is a d-separation in the graph, then there is a conditional independence in the JPD. This implication is not bidirectional.

1.2.2. Markov blanket

The Markov blanket of a variable is the set of variables that, when known, provide all the information needed to make the variable independent of the remaining variables in the network.

Formally, the Markov blanket of a variable consists of the union of three sets of variables: its parents, its children, and the parents of its children, also known as spouses.

These sets encapsulate the direct influences and relevant information that are required to determine the variable's state or behavior. By conditioning on the Markov blanket, all other variables in the network become conditional independent of the variable of interest.

The concept of the Markov blanket has significant implications for inference and reasoning in Bayesian networks. It allows for efficient computations by reducing the number of variables that need to be considered when performing probabilistic inference or updating beliefs about a specific variable.

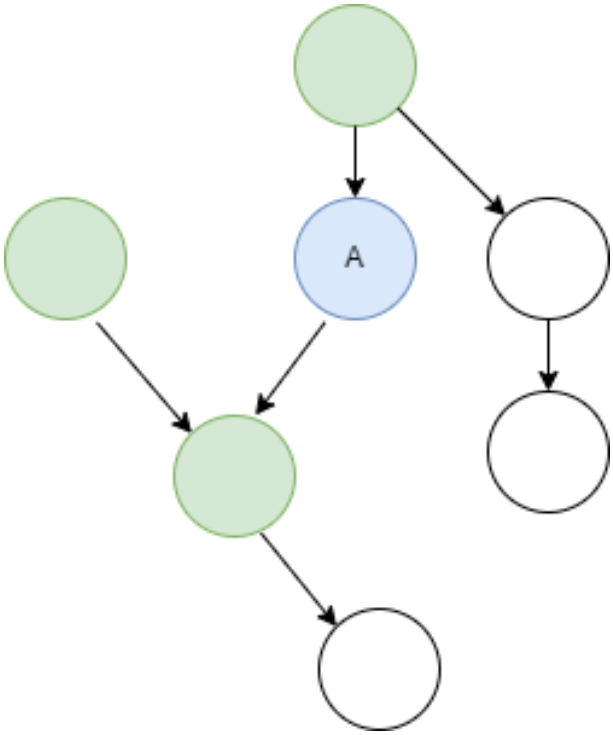

Figure 1.1: Representation of the Markov Blanket as green nodes of the node A

In terms of conditional independence, the Markov blanket provides a property for determining whether two variables are conditionally independent given a set of other variables. According to the concept of d-separation explained before, two variables are conditionally independent if they are disjointed or separated when conditioning on the Markov blanket of both variables, reducing the space search from all network to only both of Markov blankets.

1.2.3. Inference

Inference in Bayesian networks refers to the process of reasoning about unobserved variables given observed evidence on some variables. The primary goal is to get probabilities of interest, such as the probability of a particular event occurring, given the evidence available.

Inference methods can be divided into two main types:

- Exact inference: These methods aim to calculate the precise probabilities of interest by leveraging the network's structure and conditional probability tables.
- Approximate inference: As Bayesian networks grow in size and complexity, exact inference becomes computationally intractable. To address this issue, approximate inference methods are employed.

The approximate inference method known as likelihood weighting will be explained in detail as it will be utilized and implemented in this work.

1.2.3.1. Likelihood weighting

The likelihood weighting method Fung and Chang (1990) is a sampling-based inference technique to estimate the posterior probabilities or distributions of unobserved variables given observed evidence.

The likelihood weighting method involves the following steps:

1. First step is the initialization of the evidenced variables with their respective observed values, while ignoring unobserved variables.
2. After specifying the evidenced values, given an ancestral order (topological ordering of variables in the network based on their dependencies. It represents a sequence of variables where each variable comes before its descendants in the DAG), generate samples for all unobserved variables when their parents are already sampled. For each sample, assign values to the variables based on their conditional probability distribution given the values of their parents.
3. Calculate the weight for each sample based on the likelihood of the evidence given the sampled values. This weight represents the likelihood of the sample being consistent with the observed evidence.
4. Aggregate the samples and their corresponding weights to estimate the posterior probability or distributions of the unobserved variables. This can be done by summing the weights for each combination of values of the unobserved variables and normalizing them to obtain probabilities.

The key idea behind likelihood weighting is to assign higher weights to the samples that are consistent with the observed evidence. By incorporating these weights, the method focuses on the relevant parts of the probability space and provides accurate estimations of the posterior probabilities.

1.2.4. Supervised classification

Supervised classification is a major task in machine learning that aims to assign input data points to predefined categories or classes.

It is a predictive modeling approach where the learning algorithm is trained on a labeled dataset, consisting of input samples along with their corresponding class labels, which its main objective is to learn a model that can accurately generalize from the training data to make predictions on unseen or future instances.

The process of supervised classification involves several components. First of all, the input data, also known as predictive features or attributes, describe the characteristics or properties of the instances which are going to be classified. These features can take different forms, such as numerical values or categories.

The labeled training dataset serves as the basis for learning the classification model. Each instance in the training data is associated with a known class label, indicating the class it belongs. The model is trained by extracting patterns and relationships from the feature values and their corresponding class labels.

However, supervised classification needs the availability of labeled training data, which may imply efforts in data collection and annotation. As the rest of machine learning tasks, the quality of the data that is going to be used to train the model has a direct impact on its results.

1.2.5. Bayesian network classifiers

Bayesian network classifiers (Bielza and Larrañaga (2014)) are Bayesian networks designed to solve supervised classification problems . Compared to the other models, Bayesian networks hold advantages such as: They offer an explicit graphical and interpretable representation (which is one of the features that we will exploit throughout this project). In addition decision theory is applicable to them.

Let $\boldsymbol{X} = (X_1, ..., X_n)$ be a vector of predictor random variables or features and C be the class variable with $c \in \Omega_C$. A Bayesian classifier assigns the most probable a posteriori (MAP) class value to a given instance $\boldsymbol{x} = (x_1, ..., x_n)$, that is:

$$arg\, max_c\, p(c|\boldsymbol{x}) = arg\, max_c\, p(\boldsymbol{x}, c)$$

Friedman et al. (1997) proposed that Bayesian network classifiers approximate $p(\boldsymbol{x}, c)$ with a factorization according to a Bayesian network.

Each of the Bayesian network classifiers has different factorization of $p(\boldsymbol{x}, c)$. The models we are going to see in this work are divided into two different types:

- Naive Bayes and Tree-augmented naive Bayes (TAN) have $p(\boldsymbol{x}, c) = p(c)p(x_1, ..., x_n|c)$ factorization, because these models do not allow predictor features to be a parent of the class variable.
- On the other hand, the Markov blanket based classifiers, have the next factorization:

$$p(\boldsymbol{x}, c) = p(c|\boldsymbol{pa}(c)) \prod_{i=1}^{n} p(x_i|\boldsymbol{pa}(x_i))$$

1.3. Objectives

The main objective of this work is the development of a software program that allows end-users to have an interactive interface with the main objective of improving user's understanding of Bayesian networks classifiers.

Users expected to use this interface are users with a certain level of understanding and familiarity with the concepts, principles, and techniques related to Bayesian networks.

For achieving this objective, there is a series of tasks to be done:

- Understand and comprehend the different methods of the current state of the art for interpretability of Bayesian networks, specifically for Bayesian network classifiers.

- Study and develop of a visual interface with the tkinter[1] python library, to improve the interpretability of the models designed and used, showing the steps of their construction.
- In relation with the previous task, the implementation of interactive mechanisms to allow the user change parameters and compare their influence on the results. For example fixing evidence to see how this impacts on the probabilities of the unobserved variables (not evidenced).
- Create a new method of supervised classification in Bayesian networks using Estimation of distribution algorithms (EDAs) for searching the best Markov Blanket based classifier and study the results of its performance and compare it with state-of-the-art algorithms.

[1] Official tkinter documentation: https://docs.python.org/3/library/tkinter.html

Chapter 2

State of the Art

2.1. Interpretability in Bayesian networks

The explainable artificial intelligence approach tries to give explanation usually about black-box models using another model to explain the decision made by the first one. These explanations are usually not faithful. Relying on black-box models instead of prioritizing the development of inherently interpretable models is a practice that may perpetuate negative consequences and have significant societal repercussions and a risk of propagating suboptimal practices.

Unlike black-box models, interpretable models do not require post-hoc explanations to be understood by humans. Models such as logical rules, linear models, and decision trees are commonly regarded as interpretable due to their inherent transparency and comprehensibility. These models allow for direct interpretation and understanding of their underlying logic, making them more accessible and interpretable to human users.

However, there is an ongoing debate regarding the interpretability of certain models, such as Bayesian networks. While some authors argue that Bayesian networks can be considered interpretable, differing perspectives exist within the research community. This discrepancy arises due to varying interpretations and criteria for what constitutes interpretability.

Despite the debate, Bayesian networks possess characteristics that can facilitate understanding and provide insights into the relationships between variables, leading some authors to classify them as interpretable models (as explained by Mihaljević et al. (2021)).For example, Arrieta et al. (2020) claimed that Bayesian networks can be classified as interpretable models, since they fulfill that they can be categorized as simulatable, decomposable, and algorithmically transparent.

It is also important to acknowledge that under certain circumstances, such as when dealing with excessively complex or unwieldy variables, a Bayesian network may lose these first two properties. These challenges arise due to the increased intricacy and difficulty in simulating and decomposing the model, which can impact its overall transparency.

In addition to the content covered in the previous paragraph, it is important to take into account that the internal mechanisms and knowledge base of Bayesian networks can hinder the understanding to end users. The explanation of probabilistic reasoning in Bayesian networks presents challenges, often resulting in counter-intuitive or seemingly incorrect outcomes. This is why, the need for explanations becomes crucial.

Lacave and Díez (2002) presented a review of the different methods for generating explanations for Bayesian networks.

They give the next definition about explanations: explaining consists of exposing something in such way that it is **understandable** for the receiver of the explanation and also **satisfactory**.

This explanation possess ten distinct properties that can be grouped into three primary categories, namely content (corresponding to what is being explained), communication (involving interaction with the user), and adaptation (relating to the intended recipient of the explanation).

With this properties, Lacave and Díez (2002) proposed a division based on where we put the focus of the explanation. Separating explanations in three different clauses: the evidence propagated, the knowledge base (explanation of model) and the reasoning process.

Explanation of evidence: Within this context, an explanation, denoted as w, encompasses the assignment of values to all variables within a specific subset W of the network's variables. As abductive methods focus solely on unobserved variables, given that the values of observed variables are known with certainty, their objective is to identify the most probable explanation (MPE), which corresponds to as maximum a-posteriori probability (MAP). The MPE, denoted as **w***, corresponds to the configuration with

$$\mathbf{w^*} = \arg\max p(w|e)$$

where **e** represents the available evidence. Sometimes the k most probable explanations are sought. When W encompasses all unobserved variables, this process is referred to as total abduction; otherwise, it is known as partial abduction.

Primarily, the methods belonging to this section aim to identify the MPEs without providing a justification for their higher probability compared to alternative explanations. Therefore, the purpose of these methods is descriptive rather than explanatory.

Obtaining the MPE is a NP-complete problem (Shimony (1994)). However, as Kwisthout (2011) explained, there are polynomial time algorithms for obtaining the MPE if certain parameters are known to be small (i.e. treewidth)

Bodlaender et al. (2002) presented an algorithm to determine whether the MPE surpasses a predefined probability threshold, denoted as q. Notably, in this algorithm, q is considered a fixed parameter and is not part of the input. The algorithm exhibits a runtime of: $O(2^{\frac{\log q}{\log(1-q)}}n)$ with n representing the number of variables.

Specifically, when q is treated as a constant parameter, the algorithm's runtime scales linearly with n. Additionally, as q increases, the running time decreases, thereby rendering the problem tractable for instances where the MPE exhibits a high probability.

Moreover, the problem can be straightforwardly extended to a functional problem variant, where the algorithm returns the most probable assignment instead of a simple true or false outcome.

The difference between abduction and inference is caused by states with the highest probabilities for a set of nodes that may not always correspond to the most probable joint value assignment for those nodes.

Explanation of model: The act of explaining the model involves presenting the information stored within the knowledge base (the reasoning process made by the system to obtain a conclusion and the evidence propagation if there is an evidence) in various formats, such as verbal explanations, graphical representations, or step-by-step frames.

Explaining the model serves in providing novice users with domain-specific knowledge for educational purposes. This enables users who are new to the domain to acquire foundational knowledge and enhance their understanding of the subject matter.

The most straightforward and intuitive approach to represent the information encapsulated in a Bayesian network is by visually displaying the corresponding DAG. This graphical representation offers a clear and concise visualization of the Bayesian network structure, allowing for a better understanding of the interconnections among variables.

There are several methods to generate static explanations such as translating the qualitative and quantitative information of the network into linguistic expressions, known as verbal descriptions of the model (Henrion and Druzdzel (1990)).

Explanation of reasoning: Explanations in this context obtain how probable a conclusion is given an evidence in terms of justifying the process that has been followed to reach that solution. In the case that the expected results do not occur, this is useful to be able to find the causes that have provoked them, and finally to theorize possible solutions that are generated by changing some of the variables of the initially observed set.

In this section, explanations can divided into micro level and macro level. Micro level explanations focus only on one variable at each moment, One method within this category is the representation of variations in the probability of a node through linguistic or graphical explanations. Other methods focus on the study of the impact of evidence or providing explanations for local updates in polytrees (Pearl (1988)).

At macro level, explanations are divided into two different categories, those that focus on quantitative analysis and the ones that centre on qualitative analysis.

In quantitative analysis, explanations focus on statistical analysis and quantify relationships, patterns, trends, and associations within the data. Usually this models focus on probabilities or other parameters such as the fitness function (Echegoyen et al. (2011)).

In qualitative analysis, in particular, Druzdzel (1996) proposed an explanation to determine the qualitative impact of each finding F on a specific variable of interest, V, and identify the paths from F to V.

There are three types of explanations related to the three types of elementary qualitative inferences:

- Predictive inference: proceeds from causes to effects, following the direction of the links.
- Diagnostic inference: goes from effects to causes, opposite to the direction of the arcs.
- Intercausal inference: analyzes the qualitative impact of evidence for variable A on another variable B when both variables influence a third variable C.

There are more examples of explanations found within explanations of reasoning such as contrastive explanations (Miller (2021)), this explanations aim to answer why an event X has happened in terms of hypothesized non-occurring events ("Why did X happened rather than Y?"), and counterfactual explanations (Koopman (2020)) , counterfactual explanations consider cases that have already happened and study what would happen if a different decision had been made at some point of the process, and study the relation of this alternative possible scenarios and compare them with the one that actually happened and study its consequences.

This categorisation is extended by Derks and De Waal (2020) to include a new type of explanation: explanation of decisions.

Explanation of decisions: In situations where the end user has to make a decision with the information he gathered from the network, especially in cases where that decision may endanger human lives, doubts and questions arise as to whether the end users are prepared to make the decision or if further preparations are required to enable them to make the decision.

In order to solve these doubts that arise in users and obtain an explanation, a method known as the s̈ame-decision probabilityẅas proposed by Choi et al. (2012). This method aims to provide an explanation of decisions by assigning a confidence level to each decision. This confidence level represents the probability that a decision is made based on unobserved variables. The method uses a decision threshold, which aids in determining the confidence associated with each decision.

Among the explanations of the evidence, there is a method known as MAP-explanation. It is the outcome of computing MAP hypothesis. When responding to a MAP query, all nodes that do not serve as hypothesis nodes or evidence nodes are marginalized. Nevertheless, these intermediate nodes may contain valuable information regarding the stability of the MAP explanation. It is crucial to determine whether the MAP explanation remains consistent regardless of the specific values assigned to these intermediate nodes.

This consideration helps assess the robustness and reliability of the MAP explanation, ensuring its validity across various scenarios and configurations of the intermediate nodes, the concept to obtain the relevance of the intermediate nodes, was introduced by Kwisthout (2021),and it is known as MAP-independence.

In contrast to much of the existing literature in this field, the concept of relevance, specifically associated with MAP-independence, focuses exclusively on intermediate nodes. This notion is motivated by its significant application in explainable AI, particularly in assessing the stability of MAP explanations.

With this new concept, Valero-Leal et al. (2022) introduced a new category known as **support methods**, which covers methodologies designed to enhance and evaluate the quality of explanations within Bayesian networks. These methods aim to provide improved insights into the underlying mechanisms and factors that contribute to the robustness and reliability of the explanations.

Later on, some contributions have been made related to the concept of MAP independence and robustness of explanations (Renooij (2022), Valero-Leal et al. (2023)).

The actual distribution of types of explanations is shown in the Figure 2.1.

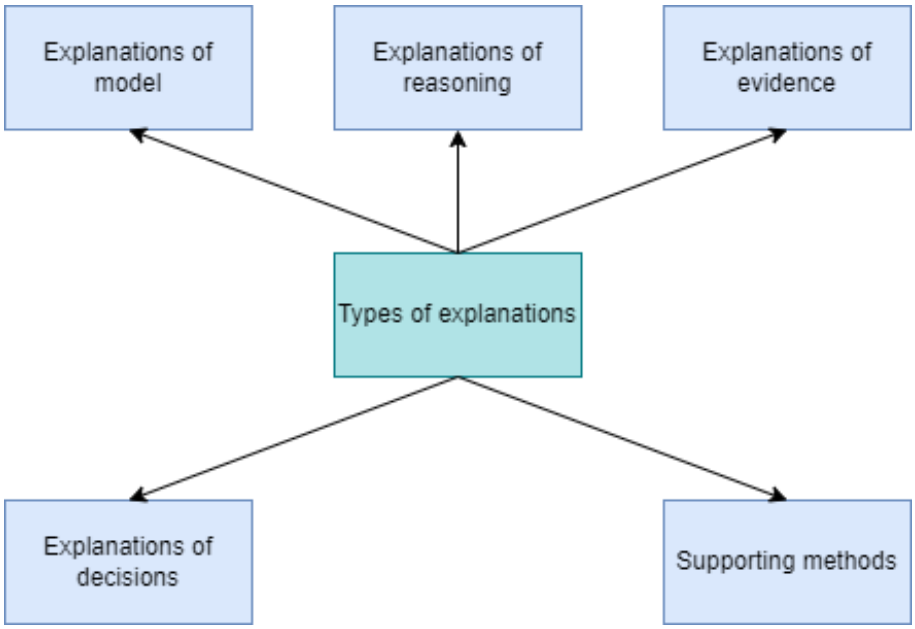

Figure 2.1: Explanation types distribution

2.2. Bayesian classifiers based on the Markov blanket selection

Bielza and Larrañaga (2014) divided into three different sections Bayesian classifiers, depending on how they made the structural learning: finding conditional independencies, score + search techniques and hybrid approaches.

Finding conditional independencies: Markov blanket search can be seen as a feature selection problem in which, with a backward greedy strategy, starting from all the features, each step a variable is eliminated until an approximate Markov blanket of the class variable MB_C is obtained. A predictor variable is eliminated if it provides little or no additional information about C beyond what is known from the other variables. Usually all these algorithms assumes that $D = (x^{(1)}, c^{(1)}), ..., (x^{(n)}, c^{(n)})$ (where x are values of the predictor variables and c values of the class variable) is a sample from a probability distribution P faithful to a DAG representing a Bayesian network. It is important to note that none of these algorithms consider the arcs within the Markov blanket that do not involve the class variable.

The first method presented by Koller and Sahami (1996), eliminates feature by feature trying to keep the conditional probability of C given the current estimation of the Markov blanket at step t, $p(C|MB_C^{(t)})$, close to $p(C|\mathbf{X})$. Metric used to define closeness is the expected Kullback-Leibler (KL) divergence. After iterate, the distance between the current estimation of the Markov blanket and the real Markov blanket, is as short as possible, so the algorithm converges to the true Markov blanket of the class variable.

The number of steps is defined as a parameter before the algorithm. Another predefined parameter is k, the number of variables selected in each step with the smallest KL divergence. Because of these predefined parameters, the algorithm is suboptimal since its effectiveness heavily relies on the accurate selection of both parameter values.

This algorithm is based on the observation that the class variable C, and a feature that is not present in the MB_C are conditionally independent under P given MB_C.

The grow-shrink (GS) Markov blanket algorithm (Margaritis and Thrun (2000)) starts inversely to the previous algorithm, that is, it starts with an empty Markov blanket. Variables are added as long as C and $\mathbf{X}$ are not conditionally independent given the MB at each instant, until there are no variables left. This first phase is called growing phase. The second phase, shrinking phase, identifies and removes the variables one by one that are independent of C given the other variables in the MB_C.

The GS Markov blanket algorithm is the first correct Markov blanket induction algorithm, meaning it returns the true Markov blanket of the class variable and not an approximated one, assuming faithfulness between P and the DAG. The algorithm is not considered data efficient as it requires a sample size that increases exponentially as the complexity of the problem grows to ensure reliable results from the conditional independence tests.

Tsamardinos and Aliferis (2003) introduced a variant of the GS algorithm called the incremental association Markov blanket (IAMB) algorithm. It consists of two phases: a forward phase and then a backward one. The first phase checks the same condition as the growing phase but variables are ordered by the conditional mutual information between the variable and the class variable, and the stopping condition is when this value is weak. For the backward phase, a variable is removed if it is independent of C given the other variables in the MB_C. This algorithm is also correct but data inefficient.

Variants of this algorithm have been developed such as InterIAMBnPC (Tsamardinos et al. (2003b)), which alternates both phases. It also substitutes the backward conditioning phase with the PC algorithm(Spirtes and Glymour (1991)). Fast-IAMB (Yaramakala and Margaritis (2005)) speeds up IAMB, minimizing the number of tests during the initialization phase.

The HITON algorithm by Aliferis et al. (2003), first identifies the parents and children of the class variable by calling algorithm HITON-PC. After that, it obtains the parents and children of the parent and children of the class variable. This obtained set is a superset of the Markov blanket of the class variable. False positives are eliminated using a statistical test inspired by the SGS algorithm (Spirtes et al. (2000)). In HITON-PC, variables from the current estimation of the parents and children set, are examined individually.

If it is found that any variable, when considering a particular subset, becomes conditionally independent of the class variable then that variable cannot belong to the parents and children set (PCs). Consequently it is remove and not considered more in the process. Even though they studied the correctness of the algorithm (Aliferis et al. (2002)), later on was proved it is not correct by Peña et al. (2007).

The max-min Markov blanket (MMMB) algorithm (Tsamardinos et al. (2003a)) is similar to HITON, which is why it has the same properties. However it changes the selection phase proposing a max-min greedy search strategy to identify the best feature.

The parents and children-based Markov boundary (PCMB) algorithm by Peña et al. (2007), takes a divide-and-conquer approach. First it gets parents and children of the class variable, and after that it identifies the rest of the parents. Using the property that if a variable is contained in the parent and children of the class variable the reverse implication must also be given, they find false positives in the Markov blanket, proving the correctness and data efficiency of the algorithm.

The property is relaxed in the Markov boundary search using the OR condition (MBOR) algorithm (Rodrigues de Morais and Aussem (2010)) using an OR condition when addressing neighbours: given two features X and Y, X and Y are neighbours if $X \in PCs(Y)$ "OR" if $Y \in PCs(X)$, instead of an AND operator, this process is known as symmetry check.

The discovery of the PCs set in the Markov blanket discovery algorithms is the most expensive steps in terms of computation, due to the exhaustive search and the implementation of the symmetry check, which increases the computational cost by $|PCs|$ times. In order to address this performance bottleneck, Gao and Ji (2016) proposed a method, called simultaneous Markov blanket (STMB), that aims to eliminate the need for the expensive symmetry check step while still effectively removing false positive PCs.

To achieve this, STMB initially follows the same procedure as PCMB. However, it goes a step further by identifying the spouse and simultaneously removing non Markov blanket descendants from the PCs set. Specifically, STMB examines the parent and children set (PCf) for nodes that are potential false positives. It looks for a node Y that unblocks a path from the class variable to a feature X not included in PCf, thereby creating a candidate spouse set. If such a node Y is found, STMB tests for false positive spouses X, such as the parents of the spouses, by conditioning on other nodes that are unblocked by Y. If X is found to be independent of the class variable, it is removed from the spouse candidate set. Additionally, STMB examines other non Markov blanket descendants X in the PC set that may have multiple paths to the target. If X is determined to be independent of the class variable, it is removed from the PCs set. By adopting this approach, STMB avoids the need for the costly symmetry check step while still effectively eliminating false positive PCs.

Wu et al. (2019) introduce cross-check and complement Markov blanket (CCMB) algorithm based on the concept of PCMasking to the Bayesian network, which denotes an error that can be risen from the conditional independence test that accepts in the Markov blanket some children that may be independent from the class variable given their parents and some parents that might be independent from the class variable given their children.

To detect this error, the cross-check and complement processes is used, in which the crosscheck process can effectively detect the PCMaskings and the complement process can repair the symmetry (between PC variables) broken by this phenomenon.

The balanced Markov blanket learning (BAMB) algorithm (Ling et al. (2019)) identifies the parent and children set and spouses at the same time, interleaving them. It finds the candidates for both of the sets and removes false positives from the candidate list at the same time. To achieve this, when a new variable is added to PCs set, it search for spouses of the class variable. After that, it removes false positives and update the spouses list. This efficient use of data makes the algorithm data and time efficient. Trying to tackle the problem of BAMB, Wang et al. (2020) proposed a new algorithm known as efficient and effective Markov blanket (EEMB).

Khan et al. (2023) presented a new algorithm called feature selection via mining Markov blanket (FSMB). It obtains the Markov blanket through using a forward approach that induces the true positive variables of the PCs set of the class variable. It removes false positives and discard them for the rest of the process. At the same time, it finds the spouses set doing an exhaustive search using the V-structure strategy, an strategy to unveil converging connections in the network, to detect the true positive ones and removing the rest.

Score + search techniques: The partial Bayesian network (PBN) for the Markov blanket around C (Madden (2002)) follows a process of three steps. In the first step, all predictor variables are classified as parent, children or unrelated to C. In the second step, the spouses of C are chosen from the set of parents and unrelated nodes. In the last step the dependencies between children are determined. These three steps are guided by the K2 score, which requires a node ordering. The inclusion of an arc is decided with the score in a forward greedy way. A similar process using an ordering starting with variable C, to apply the K2 algorithm is also explored in the work of dos Santos et al. (2011). By employing this ordering, the algorithm provokes that C has no parents, which makes an approximate Markov blanket for C.

Rather than using a filter score, the search process can be guided by a wrapper approach using classification accuracy as the score metric. An example of it is illustrated in Sierra and Larrañaga (1998), where the search is performed by means of a genetic algorithm. In this approach, each individual in the population represents a Markov blanket structure of the class variable.

Niinimäki and Parviainen (2012) presented score-based local learning (SLL). It first learns the PCs set by selecting a feature from the set of features and removing it from the set, after that it learns the structure of a Bayesian network composed of the set of class variable, PCs set of the class variable and the new feature selected with a score-based approach. From the DAG of the Bayesian Network, SLL obtains the new PCs.

SSL uses a score-based variant of symmetry checks for pruning the PCs set. For obtaining possible spouses given the current PCs set, SSL finds the union of PCs of features in the PCs of the class variable. After that, as in the first step, it obtains the DAG, but this time including spouses, and from it, it calculates the set of spouses.

Hybrid techniques: The tabu search-enhanced Markov blanked is an algorithm presented in Bai et al. (2008) with two phases. In the first phase an initial Markov Blanket is obtained based on conditional independence test by a breadth-first search heuristic. In the last phase, it makes a tabu search enhancement, allowing four possible movements: arc addition, arc deletion, arc switch and arc switch with node pruning. Each movement is guided by its classification accuracy.

Chapter 3

Structure learning algorithms

This chapter presents the different methods used and implemented. Moroever it also shows a new Markov Blanket selection method using estimation of distribution algorithms (EDAs). The programming language chosen is Python. Python offers an extensive range of libraries and frameworks specifically designed for machine learning and probabilistic modeling, making it a natural choice for working with Bayesian networks. The graphical user interface explained in the next chapter has also been developed in this language. The structure of the code developed[1] is shown in Figure 3.

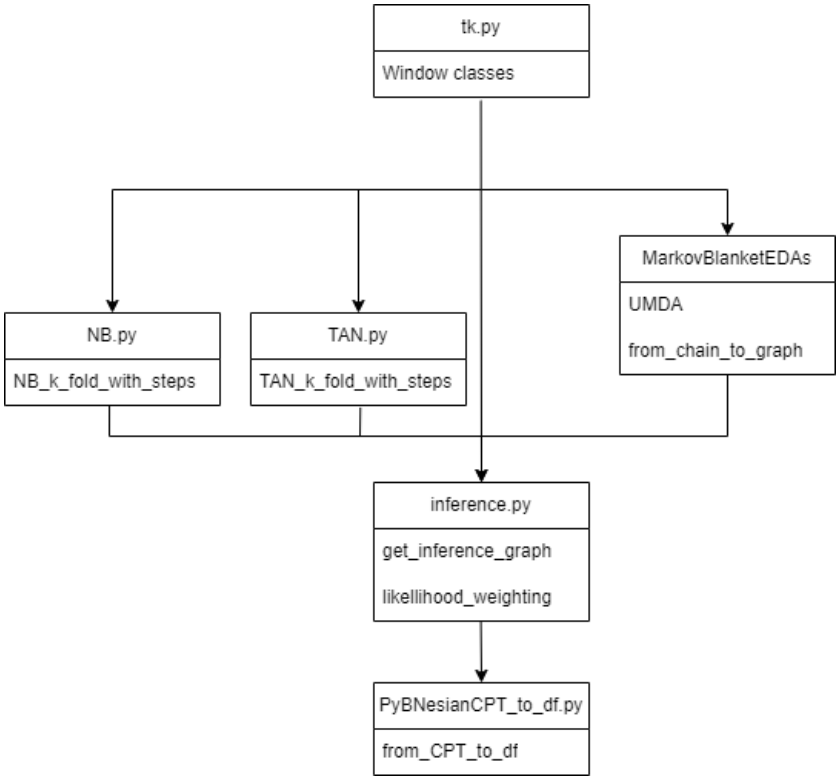

Figure 3.1: Structure of the python files

[1]The code developed is in the repository: https://github.com/IvanTelloLopez/ISL_BN

The functions that are not used in other python files and are used as auxiliary functions are not included in the graph to make it more readable.

The main file, which is **tk.py**, contains the code related to the graphical user interface that will be explained in the Chapter 4. To build each of the models, the main file calls the functions of the models (**NB.py**, **TAN.py** and **MarkovBlanketEDAs.py**) with initial parameters introduced by the user (i.e. dataframe, class variable). This functions return the results of the models and the graphic displays of the networks obtained. **Inference.py** does the same but receiving not the initial parameter but the network already built in the previous steps.

Lastly, **PyBNesian_CPT_to_df.py**, converts the string resulting from calling the method CPT of PyBNesian library (Atienza et al. (2022)) into a pandas (library made by McKinney et al. (2010)) dataframe.

3.1. Naive Bayes classifier and tree augmented naive Bayes (TAN)

The naive Bayes classifier (Maron and Kuhns (1960); Minsky (1961)) is the simplest Bayesian network classifier shown in the figure 3.1. The algorithm assumes that all predictor features are conditionally independent given the class variable. While this assumption may not hold in scenarios where features are correlated, it can still yield good results in practice.

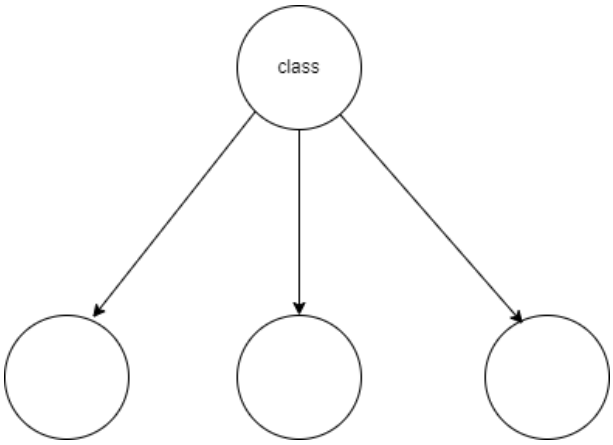

Figure 3.2: Example of naive Bayes classifier structure

However, it is important to note that due to this assumption, the classifier is not be able to accurately capture and interpret the relationships between features, which can potentially lead to false interpretations of the results. Despite this limitation, naive Bayes classifiers remain popular and effective in many real-world applications.

Tree-augmented naive Bayes (TAN) (Friedman et al. (1997)) is an extension of the naive Bayes classifier that incorporates a tree structure to model the dependencies between the feature variables given the class variable. In TAN, a tree structure is created where each predictor variable is connected to the class variable and to other predictor variables based on their conditional dependencies.

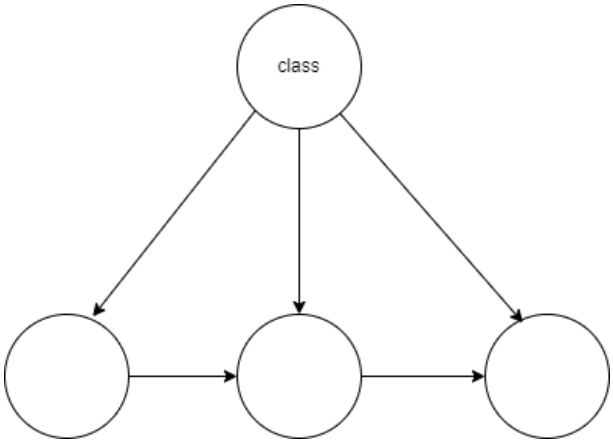

Figure 3.3: Example of tree-Augmented naive Bayes classifier structure

The advantage of TAN over the Naive Bayes classifier is that it takes into account the dependencies between the feature variables, which can improve the accuracy of the classification.

For the implementation of these methods, the library bnclassify (Mihaljevic et al. (2018)) was used. However, since bnclassify is designed for the R programming language, a way to integrate it into our python environment was needed. To accomplish it, rpy2[2] library was used, which allows interaction between python and R libraries from python. This enabled the use of bnclassify functionalities within our python-based implementation.

For our interactive structural learning of these algorithms, it was needed, not only the final result, as provided by the bnclassify library, but also the intermediate steps of the process if the user decided to see them, thus facilitating the understanding of the structural learning of the algorithms and also allowing users to choose an intermediate step as the final model.

This is why, by means of the function of obtaining the mutual information between each of the features and the class variables and the conditional mutual information between two features given the class variable (for the case of the TAN).

Thanks to this information, the order in which the edges and nodes entered the network is known, thus allowing the reconstruction of the network at each step.

But a new problem arose: a library was needed to allow the use of inference to make predictions and obtain the accuracy of the intermediate models. This was a complicated step throughout the development of the project, since it was not known how to make inference over a network with a previously fixed structure with most libraries. Finally, it was done with the pyAgrum library (Ducamp et al. (2020)).

[2]Oficial repository of rpy2: https://rpy2.github.io/doc.html

3.2. Markov blanket selection with EDAs

Bayesian classifiers based on identifying the Markov blanket of the class variable can be seen as a feature selection problem, where identifying the variables, from the original set of predictor variables, that are in the Markov blanket is the objective.

A variable is eliminated if it provides little or no additional information about C beyond what is known from the other variables. This means that variables outside the MB would have little or no effect on the prediction of the classifier.

For obtaining the optimal Markov blanket for each classification problem there are several methods such as those cited in the Chapter 2.

In this work the method presented is going to be based on estimation of distribution algorithms approach.

3.2.1. Estimation of Distribution Algorithms (EDA)

Estimation of distribution algorithms (EDAs) is a class of evolutionary algorithms that incorporates probabilistic models to guide the search for optimal solutions in optimization problems. Unlike traditional evolutionary algorithms that rely on mutation and crossover operators, EDAs focus on learning and modeling the underlying probability distribution of promising solutions.

The main idea behind EDAs is to iteratively update and refine a probability distribution based on a selected set of candidate solutions. This distribution represents the knowledge gained from the solutions and is used to generate new candidate solutions for subsequent iterations. By explicitly modeling the solution space using probability distributions, EDAs are able to capture important features and dependencies among variables, leading to more efficient exploration and exploitation of the search space.

3.2.2. Individual codification

In each EDAs problem a different codification for individuals is used depending on what is to be modeled for each individual. In some problems there are even variable length codifications such as length-adaptive genetic algorithm with Markov blanket (LAGAM) proposed by Zhou et al. (2022).

Choosing a good codification is of utmost importance in EDAs as it directly affects the effectiveness and efficiency of the process. A codification refers to the representation of solutions or individuals in the search space. It involves mapping the problem-specific individuals, in this case the MB networks, into a format suitable for the algorithm to operate with.

The initial approach involved developing a binary codification, to operate with algorithms that work with binary representations. In this codification, the first n (number of features) bits denoted the presence or absence of variables in the Markov blanket, followed by $(n+1)^n$ bits representing all possible arcs in the Bayesian network. As it can be seen in the figure:

The main problem with this initial codification was the redundancy and the complexity it generated in the validation process. The presence of cycles in DAG and the

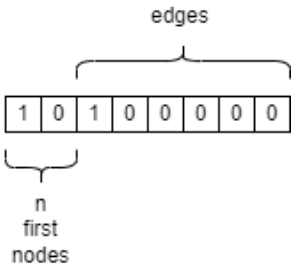

Figure 3.4: Codification of an individual in early stages

inclusion of arcs connected to nodes outside the Markov blanket violated the requirements of the model. Furthermore, the redundant duplicate information was evident in the arc values since if there was an arc from node A to node B, then there could not be an arc from node B to node A.

To tackle this problem a new codification was chosen, departing from the binary format. The first n values still represented the nodes; however, their interpretation and representation underwent a significant change, as follows for a generic node X_i:

$$X_i = \begin{cases} 0 & \text{if } X_i \text{ is not in the Markov blanket,} \\ 1 & \text{if } X_i \text{ is a parent of the class variable,} \\ 2 & \text{if } X_i \text{ is a son of the class variable,} \\ 3 & \text{if } X_i \text{ is an spouse of the class variable} \end{cases}$$

The representation of arcs has also undergone a change, resulting in a reduction in their number to $\frac{n(n-1)}{2}$. This reduction is possible because the arcs connecting to the class variable are already represented in the node values, in addition to the new encoding scheme. Let $x_i x_j$ be a position belonging to the arcs:

$$x_i x_j = \begin{cases} 0 & \text{if } X_i \to X_j \text{ and } X_j \to X_i \text{ are not present,} \\ 1 & \text{if } X_i \to X_j \text{ is in the Markov Blanket,} \\ 2 & \text{if } X_j \to X_i \text{ is in the Markov Blanket,} \end{cases}$$

Introducing the new codification successfully resolved the issue of redundancy, enhancing the validation process and improving the algorithm's overall performance. Cycles in the chain are solve during validation process that is explained in the next Subsection 3.2.3. However, the new codification also introduced certain challenges.

A way of establish the correspondence between the codification index in the chain and the arc it represent was needed. An initial approach for it was employing a dictionary, where the index served as the key and the corresponding arcs were stored as values. Nonetheless, using a dictionary for large Bayesian networks may incur severe memory costs, so a new approach may be needed for future works.

3.2.3. Algorithm process

The EDA used is the simplest but eficcient algorithm, univariate marginal distribution algorithm (UMDA) (Mühlenbein and Paass (1996)). The core idea of this algorithm lies in modeling the joint probability distribution of the variables based on the statistics of the selected candidates of the current population, by estimating the marginal distribution of each variable independently (we are assuming independency between variables). The estimated univariate marginal distributions are then used to generate new candidate solutions by sampling from them.

After sampling them, since in our current codification, some individuals may contain cycles or arcs connecting nodes that do not belong to the Markov blanket. To ensure that the generated chains represent a valid Markov blanket, it is necessary validate them from the probability distribution in each iteration to verify that they represent a feasible Markov blanket.

To perform this validation, the process starts by randomly selecting an initial position in the chain to avoid any biases that may favor the first nodes and arcs over the rest since the first positions would always be the first to be checked.

Then, iteratively we check if the value at the position is correct and forms no cycles, knowing the values of the previous checked positions. If any issues are identified, the chains are adjusted and corrected during the validation process. This ensures that we obtain a correct and valid chain by the end of the process.

Initially, the first approach was to discard any incorrectly generated chains. However, due to the vast number of potential incorrect chains in the search space, this approach significantly increased execution times. Therefore, we opted for the strategy of correcting and adjusting the chains during the validation process, reducing the need for discarding and improving overall execution efficiency.

Currently, candidate selection process is performed by choosing the best individuals according to the fitness function (elitism approach). This selection process ensures that promising solutions are used to estimate the distribution of the next generations.

The algorithm process of the Markov blanket selection is shown in the figure 3.5.

Through repeated iterations of sampling, evaluation, and selection, UMDA explores the search space and converges towards promising solutions.

While UMDA was originally designed for binary-encoded optimization problems, it has been adapted for handling discrete variables with non-binary encoding, as in this research. This adaptation involves modifying the encoding scheme and adjusting the probability distribution estimation techniques accordingly.

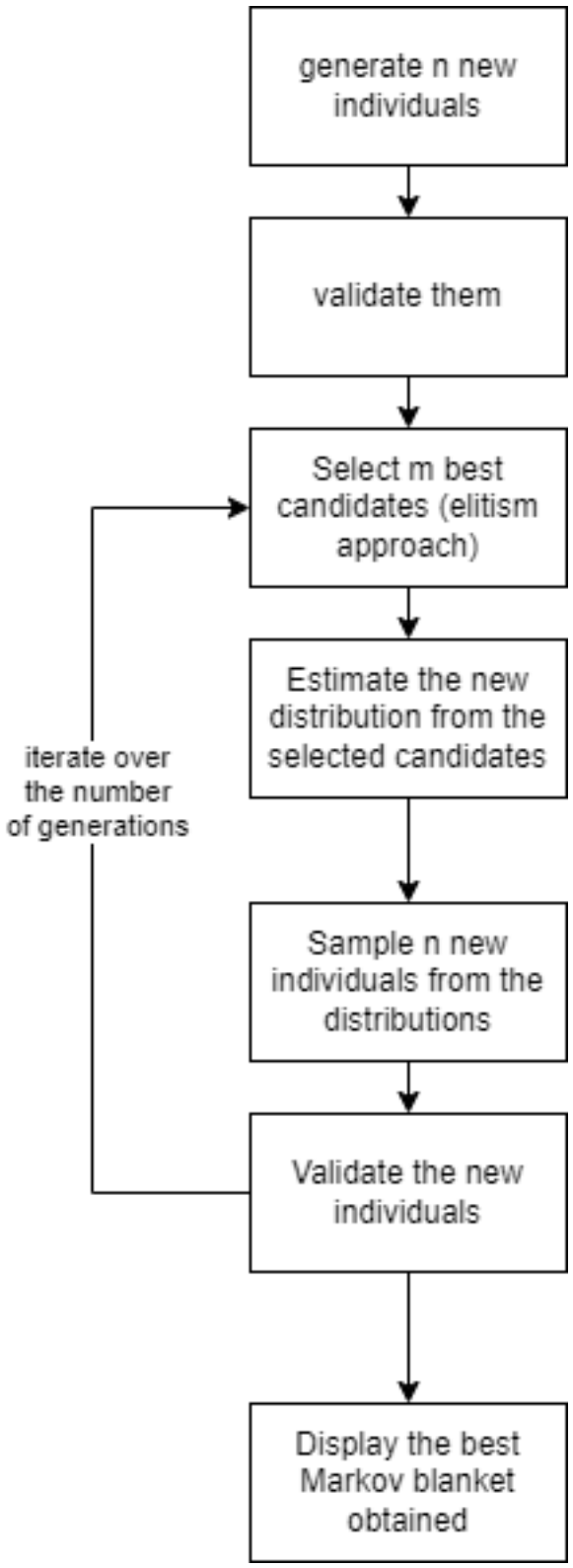

Figure 3.5: Algorithm scheme

3.2.4. Fitness metric

The fitness of each candidate solution is evaluated using a fitness function, in this case the performance of the classifier made by the Markov Blanket or the Bayesian information criterion (BIC) metric of the network associated with each individual can be chosen as the fitness function.

Initially, the fitness of each individual was calculated by creating the corresponding Bayesian network using the individual's codification using pyAgrum library.

However, as will be explained in Chapter 5, during experimentation with different datasets, it was observed that using Bayesian networks with an average size of 20 nodes and a dataset of 5000 instances, the library consumed too much RAM memory causing a performance failure in the process during the execution of some functions in the library.

As this problem was external to the implemented algorithm, alternative solutions were explored. After careful evaluation of different libraries based on their performance and resource consumption, the decision was made to switch to PyBNesian library. This library has proven to be efficient. The main drawback of using PyBNesian was the lack of implemented inference methods, so a new one had to be developed.

The chosen approach for approximate inference was **likelihood weighting**, explained in the Section 1.2.3.1. This method only needs to focus on sampling the class variable since, in a classification problem, the features are already evidenced.

With this implementation the memory cost problem was solved but the code was still underperforming in terms of execution times.

Chapter 4

Interactive Interface

The interactive interface has been built with tkinter a popular Python graphical user interfaces (GUI) toolkit. It provides a set of tools and widgets to build interactive and visual interfaces, this library was chosen over others because it offers several advantages such as:

- It is already included in the standard Python library, which means it is readily available without the need for additional installations. This makes it highly accessible and ensures compatibility across different Python distributions and versions.
- It is easy to understand and to use for beginners in GUI programming and has many tutorials about how to implement different software apps.
- It offers a wide range of built-in widgets, including buttons, labels, text entry fields, and more. These widgets can be customized and styled to match the desired look and feel of the application. It also supports the use of images allowing to show graphics and results of the networks.

As someone with limited GUI experience but a desire to implement a user interface for enhanced system interaction and explainability, Tkinter proved to be the optimal selection.

As explained in Chapter 3, the file **tk.py** contains the code for the tkinter interface. Each window is represented as a Python class, with an '**__init__**' method that initializes the window, with each window having its own set of functions.

To provide an overview of the user interface and have a better understanding of how the windows are displayed, a window navigation diagram was made (see Figure 4), also known as a window flow diagram or screen flow diagram.

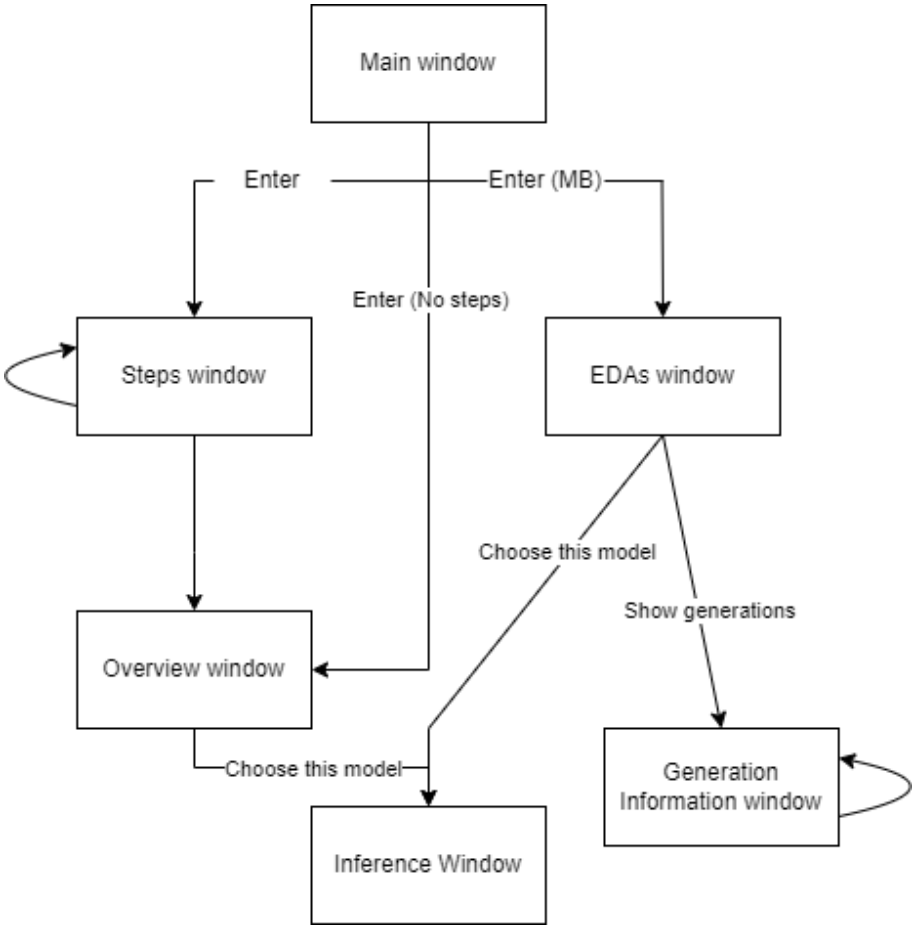

Figure 4.1: Navigation diagram

The interface is divided in 6 different windows: Main window, steps window, overview window, EDAs window, generation information window and inference window.

The **main window** (see Figure 4) is the first window, in which the parameters are selected. The first parameters to choose are the dataset and the model. Once the model has been chosen, the rest of the parameters are unlocked. These parameters change depending on the model:

- In the Markov blanket selection (MBS) by EDAs model, the following parameters can be selected: The number of generations, the number of individuals, the number of fixed candidates considered when calculating the distribution for the next generation, the class variable chosen from the dataset variables, and the fitness metric used to sort the individuals.
- For the rest of the models (naive Bayes and TAN) the parameters are: iterations between steps (number of steps not displayed between shown steps), selection parameter (order in which arcs are added to the Bayesian network), no steps (skips all steps), and the class variable chosen from the dataset variables.

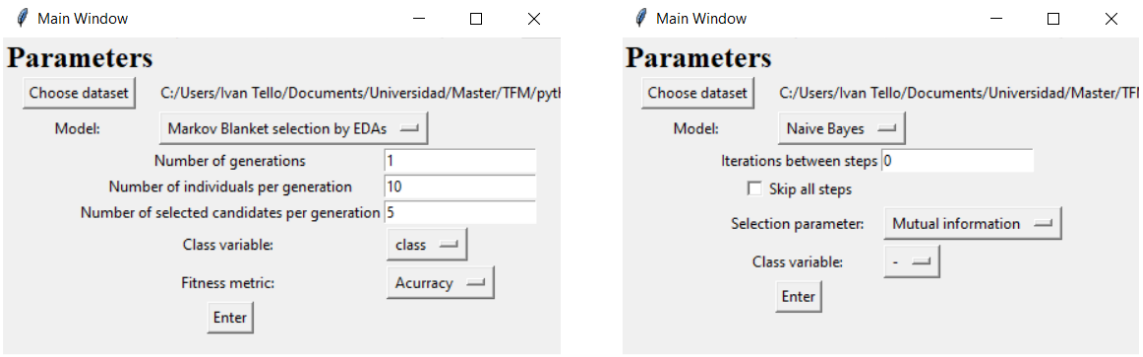

Figure 4.2: Main window: Markov blanket option (left) and the rest of the options (right).

From this window, three possible paths can be followed based on the user's choices.

If the user selects MBS by EDAs model, he/she will navigate to the **EDAs window**. In the EDAs window, the best solution obtained by the algorithm is displayed, showing its score and the graph of the Markov blanket obtained, see figure 4.

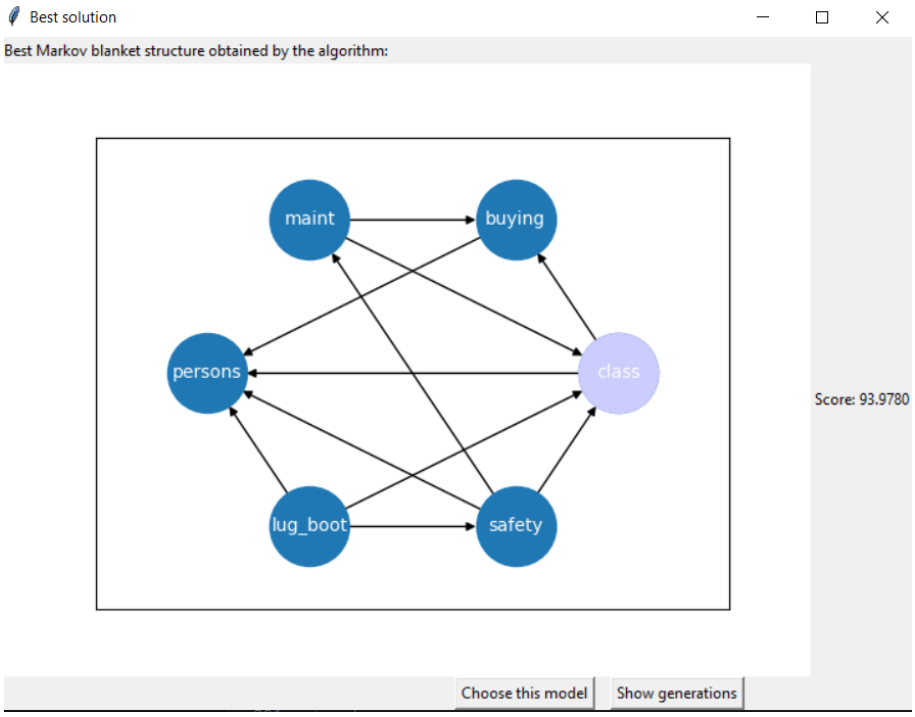

Figure 4.3: EDAs window

From the EDAs window (Figure 4), the user has two navigation options: If the user clicks on the *'Show generations'* button, navigates to the **Generation Info window** or either proceeds to the final **Inference window** clicking on *"Choose this model'* button.

The **Generation Info window** provides detailed information about each iteration displaying the best Markov blanket in each generation, and a graph displaying the difference with the last generation. User can navigate with the buttons of prev and next between generations.

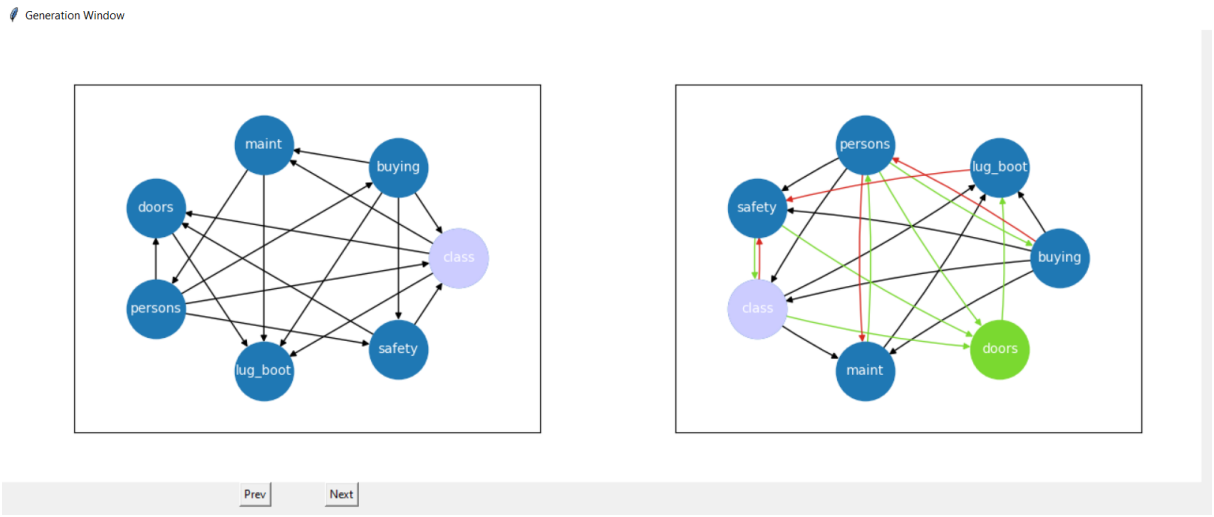

Figure 4.4: Generation info window

In this graph, arcs that were present in the best solution of the last generation but are not in the current one would be displayed in red. This color indicates the removal or absence of those arcs from the current solution.

Similarly, new nodes added in the current generation would be shown in green, along with the arcs connecting them. This color scheme highlights the introduction or appearance of these new nodes in the current solution.

Nodes and arcs that remain unchanged between the last and current generation would be represented in blue for nodes and black for arcs, indicating their continuity and persistence throughout the evolutionary process.

By utilizing this graph, users can easily identify the specific modifications that have occurred in the network structure.

The red, green, and blue color scheme provides a clear visual distinction, enabling a quick comprehension of how the nodes and arcs have changed or remained constant between the last and current generations of the Markov blanket of the class variable.

On the other hand, if the user selects a different model (from MBS by EDAs), it would navigate to two possible windows. If no additional steps are chosen, he/she will transition to the **overview window**. Alternatively, if the user selects additional steps, it would navigate through a series of **step windows**.

Each **Step window** (see Figure 4) displays a specific step or stage in the process, guiding the user through the required steps of the structural learning of the selected method, in each step the current Bayesian network classifier is displayed, showing in each arc how they are changing the score of the structure. Once the algorithm is finished, the user navigates to the Overview window of the model.

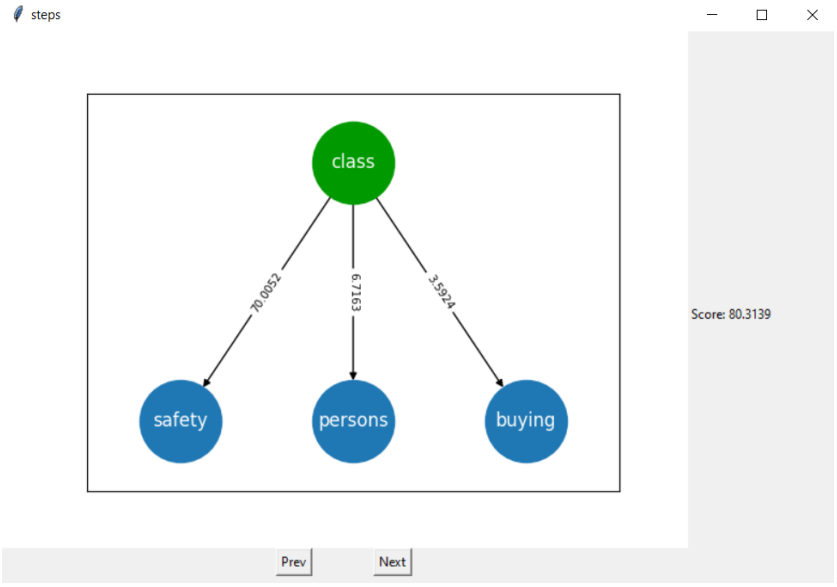

Figure 4.5: Steps window

In **overview window** (see Figure 4), an overview of the selected model is displayed showing all graphs of each step allowing choosing one (button over score) for the next last step: The inference. To navigate to this functionality the user must click on *'Choose this model'*.

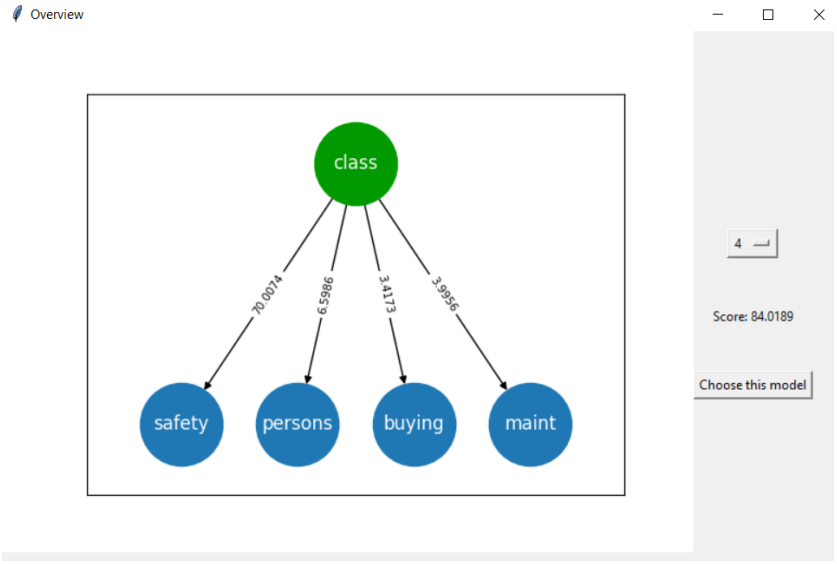

Figure 4.6: Overview window

Lastly, where all models would converge to a common window, in the **Inference window** (see Figure 4), the graph of the conditional probabilities is presented, allowing the user to interact and set fixed values (evidence) for specific variables. By fixing values for certain variables, the user can observe how these changes affect the probabilities of the non-fixed variables. This interactive feature provides a dynamic visualization of how the univariate marginal probability distributions within the Bayesian network are influenced by the user's input.

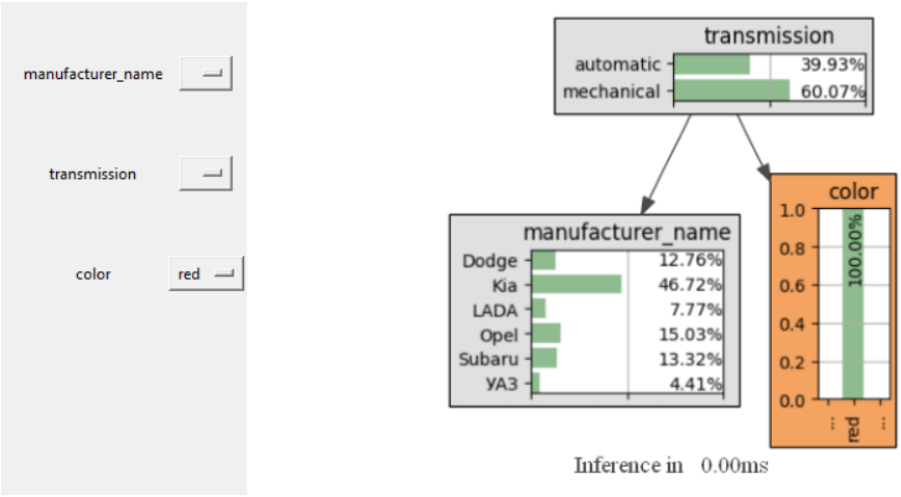

Figure 4.7: Inference window

Through this graph, users can gain insights into the dependencies and relationships between variables, as well as understand the impact of variable values on the overall probability distribution. By exploring different scenarios and adjusting variable values, users can analyze and interpret the behavior of the Bayesian network in response to specific conditions, thereby enhancing their understanding of the underlying probabilistic model.

Thus, in Figure 4, at left, the variables are shown with a button that displays all the values each variable can have, when fixing one of them, the graphical node changes its color to orange and all probabilities are updated. The user can evidence all the variables he/she wants to.

The explanations created in this work would be framed within explanation of model, displaying graphical model, showing the step by step processes and allowing the user an interaction with the system in all stages of construction of the model.

Chapter 5

Results

In this chapter we study the performance results of the implemented method MBS with EDAs. Other algorithms are not tested because they are from other libraries, as said in Chapter 3, and in this work we did not make any changes that may affect their performance.

A test with users to study the interpretability and user friendliness of the interface is something that would enhance the results of the interface because its objective its not improving state-of-the-art algorithm performance but to improve the algorithms in terms of explaining the models and build trust of users in AI models. However, the lack of time and people to make the tests did not let us carry them.

MBS with EDAs has been tested with different parameters and different datasets from UCI datasets[1]: `cars_evaluation`, `monk's problems` and `Tic-Tac-Toe endgame`. In the next tables we are going to display the results of the algorithm with the different parameters used.

The `cars_evaluation` dataset which its class variable is named as class and has 4 different values (unacc, acc, good, vgood). This dataset has 6 different features which are: buying (vhigh, high, med, low), maint (vhigh, high, med, low), doors (2, 3, 4, 5more), persons (2, 4, more), lug_boot (small, med, big) and safety (low, med, high). The number of instances is 1728.

The `monk's problems` dataset which its class variable is binary and named class. And 6 features: a1 (1, 2, 3), a2 (1, 2, 3), a3 (1, 2), a4 (1, 2, 3), a5 (1, 2 , 3, 4), a6 (1, 2). The number of instances is 432.

The `Tic-Tac-Toe endgame` dataset which its class variable is binary and named class. And 9 features (top-left-square, top-mid-square, top-right-square, mid-left-square, mid-mid-square, mid-right-square, bot-left-square, bot-mid-square, bot-right-square) with the same three posible values (x, o, b). The number of instances is 958.

The brfg column contains the result (% accuracy) of the best individual in the first generation of the EDA.

[1]UCI datasets webpage: https://archive.ics.uci.edu/datasets

individuals	selected candidates	generations	time (s)	brfg (%)	best result (%)
50	25	3	35.03	86.3426	95.3704
100	50	3	60.90	93.0556	95.6019
100	50	5	90.53	87.9630	95.3704
300	150	3	78.54	94.6759	96.0648
300	150	5	187.95	94.2130	96.7593
500	250	3	94.20	96.9907	96.2963
500	250	5	148.12	95.8333	97.4537
1000	500	5	283.12	96.5278	97.4325
2000	1000	5	518.04	96.2963	96.7593
2000	1000	10	936.10	96.5278	98.1481
2000	1500	5	558.87	95.3704	96.7593
5000	1000	10	1454.20	97.9167	98.3796

Table 5.1: `Cars_evaluation` dataset results in accuracy

As shown in Table 5.1 the best performance is 98.3796% and the network structure of the solution is shown in Figure 5.1.

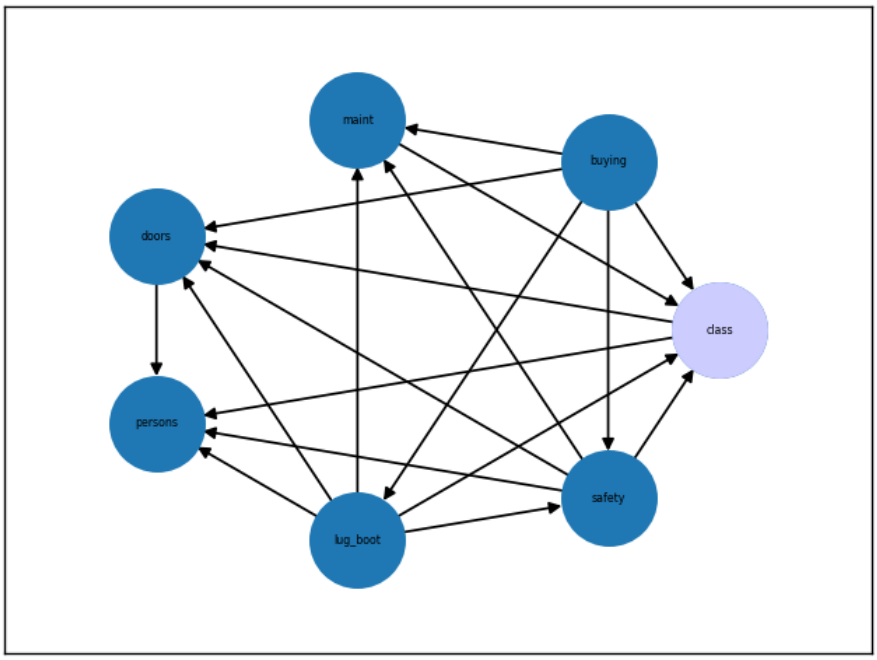

Figure 5.1: Best Markov blanket structure in `Cars_evaluation` dataset

Naive Bayes and TAN results for this dataset are around **85.5244**% and **93.7458**%. Which are worse than the obtained with our algorithm.

Table 5.2 Shows the results for the `monk's` dataset.

individuals	selected candidates	generations	time (s)	brfg (%)	best result (%)
300	150	5	90.95	67.6259	71.9424
500	250	3	125.40	65.4676	71.2230
500	250	5	186.12	70.2230	73.3230
1000	500	5	257.60	71.2230	75.4181
2000	1000	5	388.18	73.3813	76.9784
5000	2500	5	539.70	70.5036	74.6190
5000	2500	10	675.74	71.2230	80.4173
10000	2500	10	927.22	71.9424	83.4532

Table 5.2: `monk's` dataset results in accuracy

Note that the best performance is 83.4532 % and the network structure of the solution is shown in Figure 5.2.

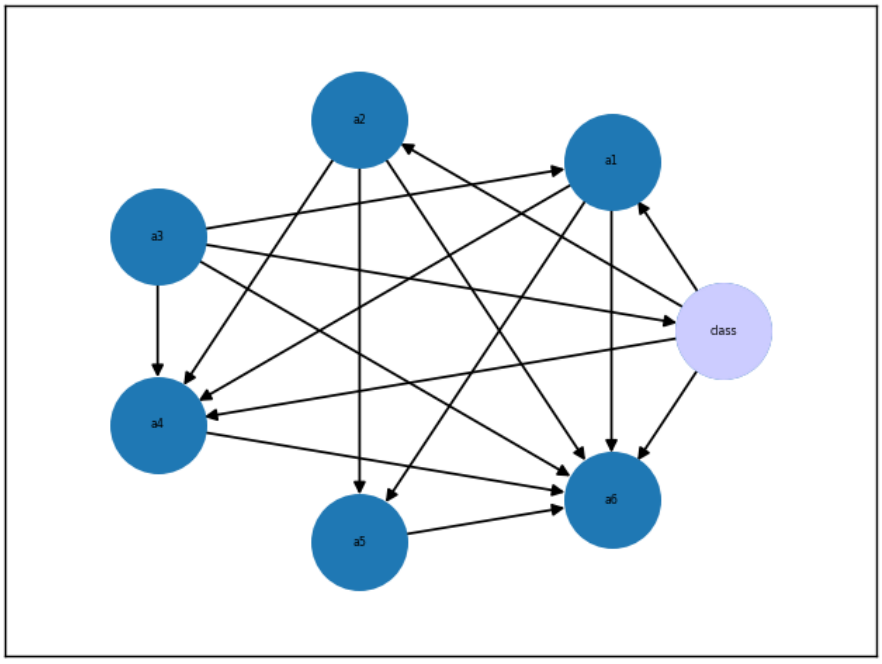

Figure 5.2: Best Markov blanket structure in `monk's` dataset

Thanks to the fact that in monk's dataset results from other machine learning models are included in the UCI webpage (see Table 5.3), we are going to compare our result with them: Our model surpasses the Logistic Regression (70.6020 %) and Neural Network Classification (78.704 %). Its near to Support Vector Machine (86.8060 %) but its outperformed by Xgboost and Random Forest Classification.

model	worst result (%)	best result (%)
Xgboost	95.6020	98.6110
Support Vector Machine	79.6300	86.8060
Random Forest	88.6570	93.9810
Neural Network	70.3700	78.7040
Logistic Regression	61.8060	70.6020

Table 5.3: monk's dataset results

Table 5.4 Shows the results for the Tic-Tac-Toe endgame dataset.

individuals	selected candidates	generations	time (s)	brfg (%)	best result (%)
50	25	3	40.60	41.6667	43.3333
100	50	3	78.90	40.4167	44.5833
100	50	5	95.53	42.5000	45.8333
300	150	3	96.54	41.6667	43.9167
300	150	5	151.95	42.5000	46.2500
500	250	3	87.20	42.0833	44.5833
500	250	5	192.12	43.7500	47.0833
2000	1000	5	287.66	45.0000	47.8333
5000	2500	3	178.20	44.1667	48.5833
5000	2500	5	180.57	42.5000	46.2500

Table 5.4: Tic-Tac-Toe endgame dataset results in accuracy

The best performance for the Tic-Tac-Toe endgame dataset is 48.5833 % and the network structure of the solution is shown in Figure 5.3.

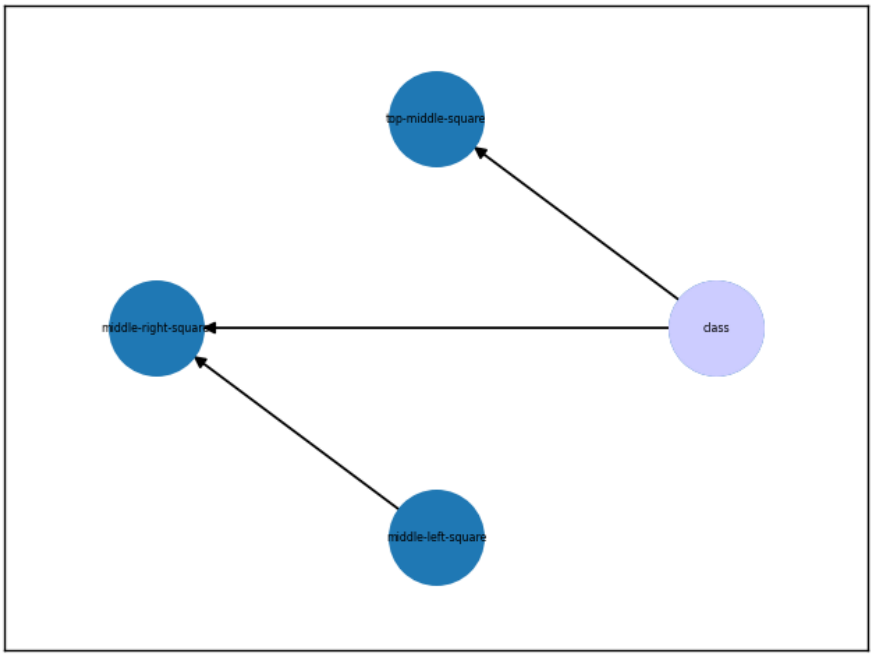

Figure 5.3: Structure of the best Markov blanket in `Tic-Tac-Toe endgame` dataset

Same as before, in the `Tic-Tac-Toe endgame` dataset, some results are included in the UCI dataset webpage (see Table 5.5), but this time our model performs as the Random Forest Classification with an average of 47.5% of accuracy which is the worst of the models displayed. The rest of the models outperforms ours.

model	worst result (%)	best result (%)
Xgboost	52.9170	65.4170
Support Vector Machine	82.9170	91.6670
Random Forest	41.250	53.7500
Neural Network	98.7500	100.0000
Logistic Regression	97.0830	100.0000

Table 5.5: `Tic-Tac-Toe endgame` dataset results

Chapter 6

Conclusions and future work

The development of a software application that enhances interpretability for Bayesian network classifiers through an interactive interface was the main objective of this work.

By providing an intuitive and interactive interface, the software application empowers users to explore and comprehend the underlying mechanisms of Bayesian network classifiers, allowing users to visualize the structure of the Bayesian network, understand the conditional probability tables, and analyze the impact of different variable values on the overall probabilities. This level of transparency and control enhances the interpretability of the model and enables users to gain insights into the decision-making process of Bayesian classifiers.

Interpretability plays a crucial role in AI systems as it enables users to understand and trust the decisions made by these models by providing clear explanations for the model's predictions and decisions. Furthermore, interpretability helps in identifying biases, potential errors, or limitations in the model, enabling improvements and mitigating potential risks.

The work developed focused on interpretability aligns with the growing demand for responsible and ethical AI practices. It empowers users to understand, validate, and explain the decisions made by Bayesian network classifiers, fostering transparency and accountability in AI systems.

In addition to the creation of the interface to enhance the interpretability of Bayesian networks, a new algorithm has been created to find the Markov blanket of the class variable. This algorithm is not exhaustive, so it does not need to traverse the entire search space, although it does not guarantee reaching the optimum.

Although many functionalities have been developed, this field is still in its early stages so there are many things still to be done.

This is why there are many future lines of research, especially in relation to the developed software application itself.

The first line of future work is the extension of the variable domain. Due to the libraries used such as bnclassify or pyAgrum that are designed for discrete variables, our development was also conditioned to be limited to these, which is why the extension to continuous variables is an interesting proposal to continue this work.

Another possible line of research is the inclusion of new types of models. The software application could be expanded to incorporate a broader range of Bayesian classifiers, providing users with a diverse set of modeling options (i.e. k-DB, Bayesian multinet, AODE). Furthermore, enabling users to compare different models would facilitate the exploration of model complexity and its impact on performance.

This comparative analysis would empower users to study the trade-offs between model complexity, interpretability, and predictive accuracy, deepening their understanding of the underlying relationships and aiding in informed decision-making.

Additionally, ongoing efforts can focus on improving the user experience and interface design. Usability studies and user feedback can provide valuable insights for refining the software application's interface, making it more intuitive, user-friendly, and accessible. This iterative process of user-centric design would ensure that the interface meets the specific needs and preferences of the target users, further enhancing their ability to interact with and interpret Bayesian Network classifiers effectively.

In addition to enhancing the user-friendliness of the application, conducting user testing with real users would be a valuable step to ensure that the techniques employed to enhance model interpretability are truly beneficial for end-users.

By involving real users in the evaluation process, valuable insights can be gained such as identify possible issues in terms of usability, ensure that the interface meets the specific needs and preferences of the target users and that the explanations that have been developed are of value to users.

Unfortunately, due to the time constraints of the project, conducting extensive user testing was not feasible.

Continuing along this line of work, it would be an interesting approach to explore additional techniques for generating explanations and improving the interpretability that our interface brings to Bayesian network models. While the developed software application already provides information about the model's behavior and conditional probabilities, there is still room for further innovation in this field.

A possible area of interest could be the development of explanation generation methods. Currently, the software application allows users to explore the impact of fixing evidenced features on the probabilities of non-evidenced values and also visual explanations of the model construction.

However, generating concise and meaningful explanations for model decisions or predictions could greatly enhance interpretability such as conterfactual or contrastive explanations.

Another crucial aspect for future work is the improvement of performance and execution time of the models, particularly when dealing with large Bayesian networks. As the complexity of the network increases, the computational demands grow exponentially, posing challenges for tractability and real-time analysis, making it currently unfeasible.

By addressing these performance challenges, the software application can empower users to tackle complex real-world problems that involve intricate and bigger Bayesian network structures, expanding their applicability and usability.

As explained throughout this chapter, the field of interpretability for AI presents many research lines for further exploration and advancement. It is a rapidly evolving area that has great potential for introducing machine learning techniques on a daily basis for end-users. I am very grateful for the opportunity given to work in this field, especially the invaluable support and guidance of my tutors throughout this work.

Undoubtedly, there are several aspects that are left unexplored due to the scope of the project. However, I am fulfilled with working in such important matter and have completed this work.

References

Aliferis, C., Tsamardinos, I., and Statnikov, A. (2002). Algorithms for large-scale local causal discovery in the presence of small sample or large causal neighborhoods. *Book Algorithms for Large-Scale Local Causal Discovery in the Presence of Small Sample or Large Causal Neighborhoods, Vanderbilt University*, pages 445–498.

Aliferis, C. F., Tsamardinos, I., and Statnikov, A. (2003). HITON: a novel Markov blanket algorithm for optimal variable selection. In *AMIA Annual Symposium Proceedings*, volume 2003, page 21.

Arrieta, A. B., Díaz-Rodríguez, N., del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Atienza, D., Bielza, C., and Larrañaga, P. (2022). PyBNesian: An extensible python package for Bayesian networks. *Neurocomputing*, 504:204–209.

Bai, X., Padman, R., Ramsey, J., and Spirtes, P. (2008). Tabu search-enhanced graphical models for classification in high dimensions. *INFORMS Journal on Computing*, 20(3):423–437.

Bielza, C. and Larrañaga, P. (2014). Discrete Bayesian network classifiers: A survey. *ACM Computing Surveys*, 47(1):1–43.

Bodlaender, H. L., van den Eijkhof, F., and van der Gaag, L. C. (2002). On the complexity of the MPA problem in probabilistic networks. In *European Conference on Artificial Intelligence*, pages 675–679.

Choi, A., Xue, Y., and Darwiche, A. (2012). Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning*, 53(9):1415–1428.

Derks, I. P. and De Waal, A. (2020). A taxonomy of explainable Bayesian networks. In *Artificial Intelligence Research: First Southern African Conference for AI Research, SACAIR 2020, Muldersdrift, Proceedings 1*, pages 220–235. Springer.

dos Santos, E. B., Hruschka Jr, E. R., Hruschka, E. R., and Ebecken, N. F. (2011). Bayesian network classifiers: Beyond classification accuracy. *Intelligent Data Analysis*, 15(3):279–298.

Druzdzel, M. J. (1996). Qualitiative verbal explanations in Bayesian belief networks. *AISB QUARTERLY*, pages 43–54.

Ducamp, G., Gonzales, C., and Wuillemin, P.-H. (2020). aGrUM/pyAgrum: a toolbox

to build models and algorithms for probabilistic graphical models in python. In *International Conference on Probabilistic Graphical Models*, pages 2640–3498. PMLR.

Echegoyen, C., Mendiburu, A., Santana, R., and Lozano, J. A. (2011). Toward understanding edas based on bayesian networks through a quantitative analysis. *IEEE Transactions on Evolutionary Computation*, 16(2):173–189.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29:131–163.

Fung, R. and Chang, K.-C. (1990). Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *Machine Intelligence and Pattern Recognition*, volume 10, pages 209–219. Elsevier.

Gao, T. and Ji, Q. (2016). Efficient Markov blanket discovery and its application. *IEEE Transactions on Cybernetics*, 47(5):1169–1179.

Henrion, M. and Druzdzel, M. J. (1990). Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In *Uncertainty in Artificial Intelligence*, volume 6, pages 17–32.

Khan, W., Kong, L., Noman, S. M., and Brekhna, B. (2023). A novel feature selection method via mining Markov blanket. *Applied Intelligence*, 53(7):8232–8255.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Koller, D. and Sahami, M. (1996). Toward optimal feature selection. In *International Conference on Machine Learning*, volume 96, page 292.

Koopman, T. (2020). Computing contrastive, counterfactual explanations for Bayesian networks. Master's thesis, Utrecht University.

Kwisthout, J. (2011). Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning*, 52(9):1452–1469.

Kwisthout, J. (2021). Explainable AI using MAP-independence. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 243–254. Springer International Publishing.

Lacave, C. and Díez, F. J. (2002). A review of explanation methods for Bayesian networks. *The Knowledge Engineering Review*, 17(2):107–127.

Ling, Z., Yu, K., Wang, H., Liu, L., Ding, W., and Wu, X. (2019). BAMB: A balanced Markov blanket discovery approach to feature selection. *ACM Transactions on Intelligent Systems and Technology*, 10(5):1–25.

Madden, M. G. (2002). A new Bayesian network structure for classification tasks. In *Artificial Intelligence and Cognitive Science: 13th Irish Conference*, pages 203–208. Springer.

Margaritis, D. and Thrun, S. (2000). Bayesian network induction via local neighborhoods. *Advances in Neural Information Processing Systems*, 12.

Maron, M. E. and Kuhns, J. L. (1960). On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Mihaljevic, B., Bielza, C., and Larrañaga, P. (2018). bnclassify: Learning Bayesian network classifiers. *The R journal*, 10(2):455–468.

Mihaljević, B., Bielza, C., and Larrañaga, P. (2021). Bayesian networks for interpretable machine learning and optimization. *Neurocomputing*, 456:648–665.

Miller, T. (2021). Contrastive explanation: A structural-model approach. *The Knowledge Engineering Review*, 36:e14.

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

Mühlenbein, H. and Paass, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters. In *International Conference on Parallel Problem Solving from Nature*, pages 178–187. Springer.

Niinimäki, T. and Parviainen, P. (2012). Local structure discovery in bayesian networks. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 634–643.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Peña, J. M., Nilsson, R., Björkegren, J., and Tegnér, J. (2007). Towards scalable and data efficient learning of Markov boundaries. *International Journal of Approximate Reasoning*, 45(2):211–232.

Renooij, S. (2022). Relevance for robust Bayesian network MAP-explanations. In *International Conference on Probabilistic Graphical Models*, pages 13–24. PMLR.

Rodrigues de Morais, S. and Aussem, A. (2010). A novel Markov boundary based feature subset selection algorithm. *Neurocomputing*, 73(4-6):578–584.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Shimony, S. E. (1994). Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410.

Sierra, B. and Larrañaga, P. (1998). Predicting survival in malignant skin melanoma using Bayesian networks automatically induced by genetic algorithms. An empirical comparison between different approaches. *Artificial Intelligence in Medicine*, 14(1-2):215–230.

Spirtes, P. and Glymour, C. (1991). An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–72.

Spirtes, P., Glymour, C. N., and Scheines, R. (2000). *Causation, Prediction, and Search*. The MIT Press.

Tsamardinos, I. and Aliferis, C. F. (2003). Towards principled feature selection: Relevancy, filters and wrappers. In *International Workshop on Artificial Intelligence and Statistics*, pages 300–307. PMLR.

Tsamardinos, I., Aliferis, C. F., and Statnikov, A. (2003a). Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 673–678.

Tsamardinos, I., Aliferis, C. F., Statnikov, A. R., and Statnikov, E. (2003b). Algorithms for large scale Markov blanket discovery. In *FLAIRS conference*, volume 2, pages 376–380.

Valero-Leal, E., Bielza, C., Larrañaga, P., and Renooij, S. (2023). Efficient search for relevance explanations using MAP-independence in Bayesian networks. *International Journal of Approximate Reasoning*.

Valero-Leal, E., Larrañaga, P., and Bielza, C. (2022). Extending MAP-independence for Bayesian network explainability. In *Workshop on Heterodox Methods for Interpretable and Efficient Artificial Intelligence*.

Verma, T. and Pearl, J. (1990). Causal networks: Semantics and expressiveness. In *Machine Intelligence and Pattern Recognition*, volume 9, pages 69–76. Elsevier.

Wang, H., Ling, Z., Yu, K., and Wu, X. (2020). Towards efficient and effective discovery of Markov blankets for feature selection. *Information Sciences*, 509:227–242.

Wu, X., Jiang, B., Yu, K., Chen, H., and Miao, C. (2019). Accurate Markov boundary discovery for causal feature selection. *IEEE Transactions on Cybernetics*, 50(12):4983–4996.

Yaramakala, S. and Margaritis, D. (2005). Speculative Markov blanket discovery for optimal feature selection. In *Fifth IEEE International Conference on Data Mining*, pages 4–pp. IEEE.

Zhou, J., Wu, Q., Zhou, M., Wen, J., Al-Turki, Y., and Abusorrah, A. (2022). LAGAM: A length-adaptive genetic algorithm with markov blanket for high-dimensional feature selection in classification. *IEEE Transactions on Cybernetics*, 1:1–12.

Appendix

The code developed is in the repository: https://github.com/IvanTelloLopez/ISL_BN

Installation guide for the Python libraries:

The Python libraries used in this project are:

```
pip install pyAgrum
pip install rpy2
pip install pybnessian
pip install pandas
pip install networkx
pip install matplotlib
```

and the R libraries used are: bnclassify and utils.

Before installing rpy2, make sure you have R installed on your system. You can download and install R from the official R Project website (https://www.r-project.org/).

But if you are running the scripts in the operative system Windows there are some extra steps you must follow:

In order to work with inference graphics, the library graphviz is needed. For its installation, the installation of the anaconda software is required. After that execute this command:

```
conda install -c anaconda graphviz
```

If an error about “dot.exe not found” still persists, install the library from this page: "https://www.graphviz.org/download/".

The last resource to fix this error, is to change line 1784 of the graphviz.py from pydot folder from:

```
self.prog = 'dot'
```

to:

```
self.prog = 'dot.exe'
```