Profiling ALS disease progression through data mining techniques

Tiago Miguel da Silva Leão

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Alexandra Sofia Martins de Carvalho
Prof. Sara Alexandra Cordeiro Madeira

Examination Committee

Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Alexandra Sofia Martins de Carvalho
Members of the Committee: Prof. Rui Miguel Carrasqueiro Henriques
Prof. Mamede Alves de Carvalho

December 2020

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Acknowledgments

I would like to thank my supervisors, Prof. Alexandra Carvalho and Prof. Sara Madeira, for their crucial guidance in the development of this Thesis. Although working in different faculties, they were always available to help, scheduling meetings in either of the faculties. Even during a global pandemic (COVID-19 pandemic) and a national lockdown, their support was always present. I also want to thank the team from Instituto de Medicina Molecular that maintains the ALS dataset, in particular Drs. Mamede de Carvalho and Marta Gromicho, for their important clinical insights regarding the study of the ALS data. I would like to acknowledge LASIGE, where I found a place to work on my Thesis. I also want to thank all fellow students that I worked with throughout the five years of the Integrated Master's in Electrical and Computer Engineering, for their help in the several projects and assignments. Finally, I want to thank my family and friends, for their support in all aspects of my life.

Resumo

A esclerose lateral amiotrófica (ELA) é uma doença neurodegenerativa cujos pacientes sofrem um rápido declínio funcional. Estudar a progressão da doença usando técnicas de mineração de dados é utilíssimo, porque ajuda os médicos a entender quando devem administrar procedimentos que evitem a insuficiência respiratória dos pacientes (a principal causa de morte). Esta Tese aborda a progressão de ELA usando redes de Bayes dinâmicas (RBDs), uma técnica de aprendizagem automática que representa graficamente a distribuição de probabilidade conjunta de variáveis aleatórias dinâmicas (temporalmente dependentes). Para incluir informação estática (temporalmente independente) nas RBDs, são propostas as sdtDBNs, que aprendem RBDs com variáveis estáticas e dinâmicas, tendo complexidade temporal polinomial no número de variáveis. O modelo proposto permite introduzir conhecimento prévio (restringindo as relações das redes) e fazer inferência nas sdtDBNs. Uma implementação em software das sdtDBNs e uma interface gráfica estão publicamente disponíveis. A progressão de ELA é abordada usando observações de 1214 pacientes, primeiro, considerando todos os pacientes, e, depois, dividindo-os em três grupos de progressão. Para cada um destes quatro conjuntos de pacientes, prevê-se o seu declínio funcional e obtêm-se graficamente as correlações entre os indicadores clínicos, usando sdtDBNs. As previsões oferecem resultados promissores, com exatidões geralmente acima de 75%. As correlações encontradas fornecem uma descrição intuitiva das interações entre as variáveis. A Tese termina respondendo a três questões clínicas usando sdtDBNs, fornecendo uma análise com impacto clínico. Todas as avaliações apresentadas mostram que as sdtDBNs caracterizam adequadamente a progressão de ELA, motivando o seu uso como uma ferramenta clínica.

Palavras-chave: esclerose lateral amiotrófica, mineração de dados, progressão da doença, redes de Bayes dinâmicas, algoritmo polinomial, variáveis temporalmente dependentes e independentes

Abstract

Amyotrophic lateral sclerosis (ALS) is a neurodegenerative disease that causes a fast functional decline of the patients. Studying the disease progression using data mining techniques is extremely useful, as it helps clinicians understand when they should apply procedures to avoid patients' respiratory failure (the main cause of death). This Thesis tackles ALS disease progression using dynamic Bayesian networks (DBNs), a machine learning model that graphically displays the joint probability distribution of dynamic (time-dependent) random variables. To include static (time-independent) information in DBNs, the sdtDBN framework is proposed, which learns optimal DBNs with static and dynamic variables, having polynomial-time complexity in the number of variables. The sdtDBN framework can also introduce prior knowledge (by restricting the networks' relations) and make inference in learned sdtDBNs. A software implementation of the sdtDBN framework and a graphical user interface are publicly available. The disease progression is assessed using observations of 1214 ALS patients, considering, first, all patients, and, then, their division into three progression groups. For each of these four sets of patients, their functional decline is predicted using sdtDBNs, and the correlations between the clinical indicators are determined with the graphical display of sdtDBNs. The predictions provide promising results, with accuracies generally above 75%. The correlations found present an intuitive overview of the interactions among variables. The Thesis ends by answering three clinical questions using sdtDBNs, providing an analysis with clinical impact. All assessments presented show that sdtDBNs can properly profile ALS disease progression, motivating the use of sdtDBNs as a clinical tool.

Keywords: amyotrophic lateral sclerosis, data mining, disease progression, dynamic Bayesian networks, polynomial-time algorithm, time-dependent and time-independent variables

Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations

ALS	Amyotrophic lateral sclerosis.
ALS-FRS	Amyotrophic lateral sclerosis functional rating scale.
ALS-FRS-R	Amyotrophic lateral sclerosis functional rating scale revised.
ALS-FRSb	Bulbar sub-score of ALS-FRS.
ALS-FRSsLL	Lower limbs sub-score of ALS-FRS.
ALS-FRSsUL	Upper limbs sub-score of ALS-FRS.
AUC	Area under the curve.
BD	Bayesian Dirichlet.
BDe	Likelihood-equivalence Bayesian Dirichlet.
BMI	Body mass index.
BN	Bayesian network.
CPT	Conditional probability table.
DAG	Directed acyclic graph.
DBN	Dynamic Bayesian network.
DFS	Depth-first search.
EM	Expectation-maximization.
ESS	Expected sufficient statistics.
FVC	Forced vital capacity.
GHC	Greedy hill-climbing.
GUI	Graphical user interface.
KDD	Knowledge discovery from data.
LL	Log-likelihood.
LOCF	Last observation carried forward.
MCMC	Markov chain Monte Carlo.
MDL	Minimum description length.
MEP	Maximal expiratory pressure.
MIP	Maximal inspiratory pressure.
MIT	Mutual information test.
MLE	Most likely explanation.
MMHC	Maximum-minimum hill-climbing.

MND	Motor neuron disease.
MRF	Markov random field.
NIV	Non-invasive ventilation.
OFE	Observed frequency estimates.
PGM	Probabilistic graphical model.
PhrenMeanAmpl	Phrenic nerve response amplitude.
R	Respiratory sub-score of ALS-FRS-R.
sdtDBN	Tree-augmented dynamic Bayesian network with static and dynamic attributes.
SEM	Structural expectation-maximization.
tDBN	Tree-augmented dynamic Bayesian network.
VC	Vital capacity.

Chapter 1

Introduction

1.1 Contextualization

This work is the final endeavor of the Integrated Master's in Electrical and Computer Engineering, at Instituto Superior Técnico, Lisbon, in order for its author to obtain the Master's degree in Electrical and Computer Engineering. The Thesis studies amyotrophic lateral sclerosis (ALS) disease progression using dynamic Bayesian networks (DBNs), also providing improvements on state-of-the-art methods of DBNs structure (and parameter) learning and inference. An overview of the most common data preprocessing techniques is also done, as these techniques are needed to prepare the ALS data for learning the DBNs.

1.2 Motivation

Amyotrophic lateral sclerosis (ALS), the most common motor neuron disease (MND), is a neurodegenerative disease that quickly affects the loss of motor neurons of the patients [1], although, in general, without major cognitive damages. ALS has a prevalence of 3–5 cases per 100000 people, with risk increasing in men and elder people [2]. ALS symptoms are usually associated to limb weakness and difficulties in activities such as speaking and breathing. The main cause of death among ALS patients is respiratory failure [3]. ALS is divided into two groups: familial ALS (around 10%) and sporadic ALS (the remaining 90%). Patients with familial ALS are the ones who have relatives with ALS, being ALS usually detected earlier in these situations [4], as multiple gene mutations are known in these cases. Sporadic ALS is more difficult to identify, causing disease onset to be only 58–63 years, when comparing to familial ALS, where it is 43–63 years [5].

One of the main areas of ALS research is the diagnosis of the disease, as usually there is a delay of 13–18 months from the moment a patient develops ALS until the moment of the diagnosis [6], which is due to ALS being a rare disease and many symptoms overlapping with other neurodegenerative diseases [7]. To properly diagnose ALS, the main goal is to find clinical features to use as diagnostic predictors [8]. More than 20 genes have already been associated with ALS, being C9orf72 currently the gene most associated with ALS [2]. The Mine project [9] started a large-scale study of the genome sequencing of ALS patients. There are also some non-genetic factors associated to ALS [1, 10]. One of the most used criteria to diagnose ALS is the *El Escorial criteria*, whose result

is a degree of probability for the patient to have ALS [1]. To perform the final diagnosis, several kinds of tests can be used [6].

Currently, there is no cure for ALS. Therefore, treatments focus on slowing the progression of the disease and increasing survival of patients. Riluzole is the only known effective drug, with a median survival increase of seven months [1]. The remaining treatments consist mostly in pain attenuation, being performed several tests to keep track of the disease progression. The functional decline of a patient is assessed with a standardized set of tests that composes the *ALS functional rating scale* (ALS-FRS), currently in its *revised* version (ALS-FRS-R) [11]. As the main causes of death among ALS patients are respiratory problems, most prognostic exams are respiratory tests, such as the vital and forced vital capacities (VC/FVC) and the maximal inspiratory and expiratory pressures (MIP/MEP) [12]. The goal of these tests is to predict when ventilatory support will be needed, preventing the need for aspiration [4]. The ventilatory support is given through non-invasive ventilation (NIV), which helps improve the survival and quality of life of patients [5, 13], preventing them from dying from hypoventilation, mainly caused by hypoxemia and hypercapnia, associated with respiratory infections [3]. When patients suffer from malnutrition, Neogastric feeding is also beneficial for their survival [1].

As explained, NIV is essential to prevent hypoventilation, being of most importance to start NIV as soon as patients need it. However, there are no standard criteria to define the proper moment to start NIV, being up to doctors to identify the appropriate moment, strongly relying on their experience. Therefore, providing statistical information regarding the progression of ALS patients is a fantastic help, guiding doctors to start NIV at the right time, and, consequently, improving the survival and quality of life of patients.

1.3 Scope of the Thesis and main contributions

Given the motivation presented in Section 1.2, the ultimate goal of the Thesis is to develop a probabilistic model that can answer the question ***knowing the progression of an ALS patient until a certain consultation, how will his progression in the following consultation be described?***

To answer the mentioned question, the probabilistic model is created from data of the Portuguese ALS dataset. As doctors need to understand the reasoning for the model's decisions [14], this Thesis focuses on graphical models, particularly on Bayesian networks (BNs), and their extension to the temporal domain, dynamic Bayesian networks (DBNs), as they describe data in an interpretable fashion. It is developed a DBN framework capable of studying the ALS dataset, by creating networks that simultaneously include static (time-independent) and dynamic (time-dependent) variables. It is also provided a graphical interface for the proposed DBN framework, for doctors to be able to use it themselves. After presenting the proposed DBN framework, the ALS dataset is assessed. First, some data preprocessing techniques are applied, then, there are learned DBNs from ALS data in several scenarios, providing multiple analyses, useful for profiling ALS disease progression.

Overall, the main contributions of this Thesis are the following:

1. A theoretical revision of some important data preprocessing techniques, BNs and DBNs.
2. A DBN framework that optimally learns DBNs with static and dynamic variables, allowing a user to make restrictions in the networks and providing a user with inference capabilities. A software implementation of the proposed framework and a webpage explaining its functioning are publicly available (see Section 5.4).

3. A graphical user interface for the proposed DBN framework. A software implementation of the graphical interface and a webpage explaining its functioning are publicly available (see Section 6.4).

4. Some programs created to preprocess time-series data and to obtain relevant statistical indicators for the results of the analyses performed on the ALS dataset. The developed programs are publicly available at `https://github.com/ttlion/preprocessAndStatsThesis`.

5. A comprehensive assessment of the Portuguese ALS dataset, done using the proposed DBN framework.

6. A paper with focus on the developed DBN framework and a case-study in ALS, submitted in the Journal of Biomedical Informatics.

7. A paper providing a study of ALS disease progression using the proposed DBN framework and stratifying patients into three progression groups. This paper is already finished and waiting for the supervisors' review, being planned its submission in the journal BMC Medical Informatics and Decision Making.

8. A paper emphasizing the clinical relevance of studying ALS disease progression with DBNs, which is being written by the clinical team that maintains the Portuguese ALS dataset, using the results of this Thesis.

1.4 Previous work

Dynamic Bayesian networks

Although there is much work on BNs, there is not such an extensive literature on how to do proper structure and parameter learning of DBNs, especially when learning from medical data, where the quantity of data is usually low.

Comprehensive books [15, 16] are essential to understand the principles of DBNs, but they cover several other topics, so they do not get into specific details and scientific novelties on DBNs learning and inference.

In DBNs literature, there is an important Thesis contribution that presents a detailed explanation on how to learn a DBN and make inference on it [17]. There are other works that focus on several aspects of the learning procedure of DBNs, for instance: learning the structure of a DBN, either from complete or incomplete data [18]; learning the structure of DBNs from time-series [19]; learning DBNs using the mutual information test score (MIT) [20]; learning dynamic Bayesian multinets (an interesting topic not covered in this work) from data [21].

The aforementioned works show how DBNs are learned, applying the methods mostly to stationary DBNs, for simplification (see Section 4.2.4 for details on this assumption). Some works show how learning methods can be applied to non-stationary DBNs, presenting approaches such as: computing the transition probabilities in matrix format, using the l_1-regularized least square linear regression method [22]; using a Markov chain Monte Carlo method (MCMC) to learn the structure of DBNs from time-series data [23]; adapting to the temporal domain (DBNs) the maximum-minimum hill-climbing algorithm (MMHC) used in BNs, which is an algorithm that combines a local discovery (maximum-minimum parents and children, MMPC) with a greedy search procedure [24].

A recent work proposes an algorithm that learns both inter-slice and intra-slice parameters of a DBN [25]. The authors name this algorithm tree-augmented DBN (tDBN), as it restricts the search-space, in each time-slice, to tree structures, also allowing the variables of each time-slice to be affected by a maximum fixed number of variables from the previous time-slices (hence the "augmented"). The algorithm learns both stationary and non-stationary

(with maximum lag fixed) tDBNs, providing polynomial-time complexity in the number of variables, the most relevant parameter. The tDBN algorithm is proposed by researchers from the same investigation group in which this Thesis is included and constitutes the basis for the model developed in this work (see Chapter 5). Extensions to the tDBN work have already been tried. An example is the extension of the tDBN approach to a search-space of forests, using an ordering of the optimal branching of the tDBNs consistent with a breadth-first search (BFS) [26].

Data mining and DBNs applied to the study of ALS

As already stated, data mining processes are of tremendous importance in the medical environment. Regarding ALS studies, usually, the patients are stratified into three progression groups (slow, neutral and fast), according to the velocity of progression of the disease. This stratification can be seen, for example, in a work where the authors develop a model to infer an adequate moment to start NIV on a patient, according to his progression group [27]. An alternative approach can be, for example, to use four categories, combining early or late stage with slow or fast progression [28].

In the same research group where this Thesis is included, several attempts have been done to model the proper time to apply NIV to ALS patients [29, 30]. The same research group also has some interesting Thesis contributions for the study of ALS [31–33].

Regarding the use of DBNs to study ALS disease progression, the literature is not very extensive, as this is a relatively new topic. A recent work applies DBNs to simulate ALS progression and to find which biomarkers are associated with each of the several patients' statuses analyzed, such as survival time and movement impairment [34]. The authors use the MMHC method to learn the DBN structure.

A popular way of applying DBNs to medical data is the use of DBN structure learning to model gene interactions [35, 36]. This is important for the study of ALS, because some of the interactions among genes discovered in these studies may allow a better understanding of the genetic relations in ALS patients. There are also, in the context of using medical data, some studies where DBNs are used to describe a disease and predict specific outcomes [37, 38]. This is useful, because these studies can be transposed into using DBNs to study ALS data.

1.5 Outline of this work

Besides this chapter, this work has seven more chapters, which are presented next. **Chapter 2** provides an overview of the dataset used, in order for the reader to keep the dataset contextualization while reading the remaining chapters. **Chapter 3** gives an overview on how the data mining process works. **Chapter 4** goes deeper into the data mining process, explaining with some detail the proper background studied and needed for the Thesis elaboration. **Chapter 5** presents the improvements done on DBNs state-of-the-art methods, in order to include static attributes in the DBN framework, to learn DBNs with restrictions in the networks, and to provide a user with inference capabilities on created DBNs. **Chapter 6** introduces the graphical interface developed, so that doctors and, in general, non-experts in data mining and machine learning techniques can use the developed DBN framework. **Chapter 7** displays the application of the data mining process to study ALS data, detailing both the preprocessing stage and the knowledge extracted from the process. **Chapter 8** ends the work with a conclusion on the usefulness of the Thesis, providing an overview of its main contributions and some ideas for future work.

Chapter 2

Portuguese ALS dataset

To meet the goals described in Chapter 1, the dataset used in this Thesis is the Portuguese ALS dataset from the Translational Clinic Physiology Unit at Hospital de Santa Maria, IMM, Lisbon. The dataset was created in 1995 and its latest update was in March 2020. The current version of the dataset stores medical observations of 1374 ALS patients.

The dataset has two kinds of features: **static** and **dynamic**. **Static features** do not change over time and can be divided into **four subgroups**: **(i) demographic information**, such as gender and body mass index in the first evaluation; **(ii) medical and family history**, with information about motor neuron diseases of ancestors; **(iii) onset evaluation**, with parameters such as the onset form and the diagnostic delay; **(iv) genetic biomarkers**, with information about specific genes that doctors think that may be associated with ALS. **Dynamic features** vary over time and are the result of measurements done by doctors in different consultations. Dynamic features are divided into **five subgroups**: **(i) functional scores**; **(ii) respiratory tests**; **(iii) respiratory status**; **(iv) neurophysiological tests**; **(v) other physical values**. The full dataset has around 50 static features and 100 dynamic features. An overview of the main features of the Portuguese ALS dataset is presented in Table 2.1.

Table 2.1: Main features of the Portuguese ALS dataset (orange: numerical; pink: categorical).

Main static features	**Demographic information**	Gender	Age at onset	Body mass index (BMI) at onset	
				Retired at diagnosis	
	Medical and family history	Family history of motor neuron disease (MND)			
	Onset evaluation	Onset form	UMN vs LMN	El Escorial reviewed criteria	
		Diagnostic delay		Cirurgical interventions before	
	Genetic biomarkers	Expression of C9orf72 mutations			
Main dynamic features	**Functional scores**	ALS-FRS	ALS-FRSsUL	ALS-FRSb	R
		ALS-FRS-R	ALS-FRSsLL	ALS-FRSr	
	Respiratory tests	Vital capacity (VC)	Maximal inspiratory pressure (MIP)		
		Forced VC (FVC)	Maximal expiratory pressure (MEP)		
		P0.1, PO2, PCO2 concentrations	Maximal sniff nasal inspiratory pressure (SNIP)		
			Peak expiratory flow (PEF)		
		Forced expiratory volume (FEV)			
		Depressions in O2 saturation		Mean O2 saturation	
	Respiratory status	Starting date of non-invasive ventilation (NIV)			
	Neurophysiological tests	Phrenic nerve response amplitude (PhrenMeanAmpl)			
		Phrenic nerve response latency (PhrenMeanLat)			
		Phrenic nerve response area (PhrenMeanArea)			
	Other physical values	Cervical extension		Cervical flexion	

The functional scores of Table 2.1 consist of evaluations specific to the ALS treatment. ALS-FRS [39] (*ALS functional rating scale*) is a standard evaluation. ALS-FRS-R [40] (*ALS-FRS-revised*) is an improvement on ALS-FRS with more respiratory evaluations, as respiratory failure is the biggest cause of death of ALS patients. The scales consist of questions that evaluate conditions such as the ability to swallow, walk and speak. These questions must be answered with an integer number from 0 to 4 (five options in total), where 0 is the worst condition and 4 is the best. ALS-FRS has 10 questions. ALS-FRS-R has the first nine questions of ALS-FRS, plus three extra ones about specific respiratory indicators. The other functional scores of Table 2.1 are sub-scores of ALS-FRS-R that give information about a specific component of ALS-FRS-R. ALS-FRSb (*ALS-FRS bulbar*) is the sum of questions 1, 2 and 3, related to the ability to speak, salivate and swallow. ALS-FRSsUL (*ALS-FRS upper limbs*) is the sum of questions 4, 5 and 6, related to the upper limbs abilities. ALS-FRSsLL (*ALS-FRS lower limbs*) is the sum of questions 7, 8 and 9, related to the lower limbs abilities. ALS-FRSr is the score of question 10, the only respiratory question of ALS-FRS. R is the sum of the three extra respiratory questions of ALS-FRS-R.

An important feature of the Portuguese ALS dataset is that it stores the date of the application of non-invasive ventilation (NIV) to the patients. This helps distinguish the consultations before NIV and after NIV, which allows studying, as two distinct progressions, the patients' progressions before and after applying NIV. Having the NIV dates is also extremely useful when using the model to estimate the proper moment to apply NIV to the patients, because the real NIV dates can be used as the true labels, when evaluating the performance of the model.

One property that makes the database reliable is the fact that each patient is evaluated by the same group of doctors using standardized approaches and performing all evaluations in the same way, which reduces the bias introduced by the fact that doctors are humans. However, as the data is obtained in a real-life environment, there are some limitations of the database that must be met with proper preprocessing. Three limitations are mentioned next.

The first limitation is that, although each set of examinations is related to a single consultation, some examinations that should be done in the same day sometimes take some days, or even weeks, to be done. The database stores the days of each examination, allowing users to do their own preprocessing.

The second limitation is that there are several missing values, as sometimes it is not possible to do some examinations and patients miss some consultations.

The third limitation is the number of consultations of each patient being very low (life-expectancy of ALS patients is 3–5 years). The number of consultations also varies greatly among different patients (after the diagnosis, some may die in less than a year, while others may survive more than 10 years). The number of evaluations per patient of the dataset goes from 1 to 27.

In this Thesis, although some indicators (such as a patient's progression group, see Section 7.3) are obtained using the complete ALS dataset, most data used for learning the DBNs is retrieved from a pretreated version of the latest update of the dataset (March 2020). This pretreated version was provided by a student from the same research group in which this Thesis is inserted, and has records of 1214 patients, with measures of 9 static features and 31 dynamic features. It has a minimum of 1 and a maximum of 27 records (consultations) per patient, with average and median values of, respectively, 5,52 and 4 records per patient. The average and median values of the number of static features measured per patient are, respectively, 8,04 and 8 static features. The average and median values of the number of dynamic features measured per consultation are respectively, 21,94 and 21 dynamic features (these two low values are biased by some particular dynamic features, which have several missing values).

Chapter 3

Data mining – full process overview

Data mining can be defined as the process of discovering useful and interesting patterns from large amounts of data [41], where the process must be at least semi-automatic [42], as there must exist some computerized aid in order to apply the process to large amounts of data.

Data mining is one step of the full process of knowledge discovery from data (KDD) [41], being one of the most important parts (some authors even use both terms as being equivalent). Fig. 3.1 presents an overview of the process of KDD.

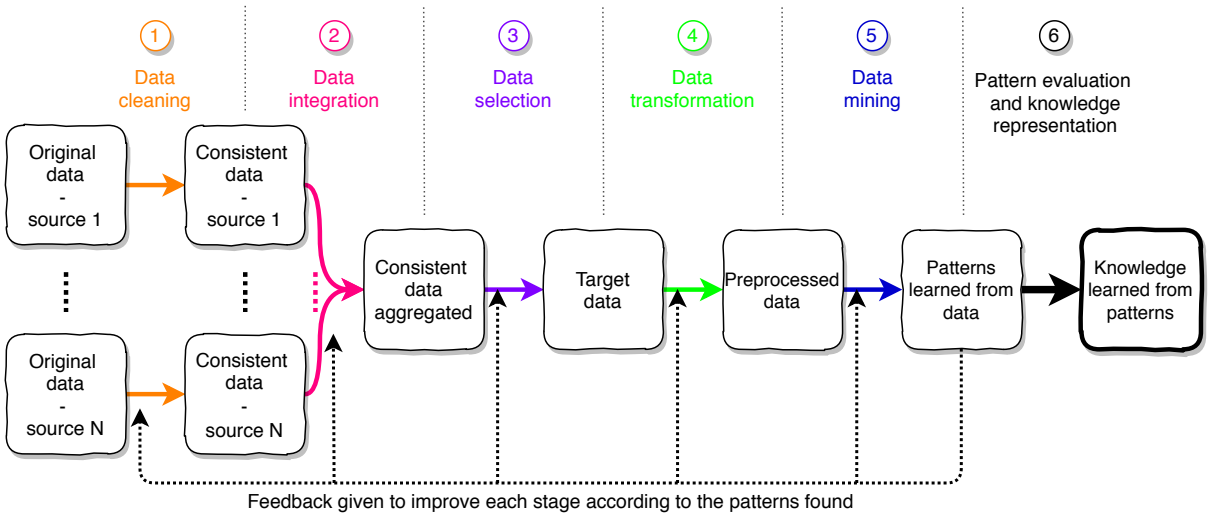


Figure 3.1: Overview of the process of knowledge discovery from data.

As Fig. 3.1 shows, the process of KDD can be divided into six main steps:

1. **Data cleaning:** this step allows getting, from the full dataset, only reliable and usable data by either changing, replacing, or deleting some data. For this step, domain knowledge and a good insight on the database itself are extremely important, in order to accurately identify the incomplete, inaccurate and inconsistent data and correctly change it.

2. **Data integration:** in this step, if different sources of data are being used, they are all combined, in order to have a unified database, which is essential to have all data in the same format.

3. **Data selection:** this step is where data not relevant for the problem is removed from the full dataset. For this operation, constant feedback is needed from the patterns learned, as the patterns found may indicate that some features kept in the dataset are not important, or that some features retrieved from the dataset may be crucial and should not have been removed.

4. **Data transformation:** in this step, data is put in appropriate formats for the mining operation. If possible, a process of data reduction may be applied, reducing the dimensionality of the data while keeping the same information. Another usual transformation is the discretization of data, if needed. Some features may even be excluded in this step, if the mining process cannot work with them.

5. **Data mining:** this is the step where knowledge is extracted from data. The goal of this step is to discover data patterns. The way these patterns are discovered is by means of learning procedures, usually associated with machine learning ideas and processes. This step also gives feedback to the previous steps, providing information on how the data could be differently processed, in order for this mining step to discover different patterns, or confirm already discovered ones.

6. **Pattern evaluation and knowledge representation:** in this step, there are defined which patterns discovered are interesting (by using some defined metric), and the knowledge given by those patterns is presented in a way a user can understand.

Given the previous explanation of the steps that compose the KDD process, it can be seen that steps 1 to 4 prepare the data, putting it in a way suitable for the learning task. Therefore, **steps 1 to 4** compose a big stage named ***data preprocessing stage***.

Chapter 4 provides an in-depth description of some important techniques used in the aforementioned steps of the KDD process.

Section 4.1 details some important aspects used in data preprocessing, being divided into three subsections. Section 4.1.1 describes one of the most important elements of step 1, which is the imputation of missing values. Section 4.1.2 describes the process of identifying outliers, which is very useful for step 3. Section 4.1.3 describes important procedures used in step 4. Step 2 is not covered in Chapter 4, because only one dataset is used in this Thesis, and the doctors that maintain it already assure the reliability and consistency of the data.

Section 4.2 describes how this work applies step 5 by resorting to graphical models, specifically BNs and DBNs. Using graphical models to perform the learning task, Section 4.2 also describes step 6, as it comes naturally from the graphical framework used by the models.

Chapter 4

Data mining – step by step

4.1 Data preprocessing

In the real world, data is often offered in big and confusing datasets, from which it is impossible to extract any reliable knowledge. As a result, in order to learn from data, proper preprocessing is needed, as shown in Chapter 3. In real-world scenarios, the preprocessing of data is a significant and possibly time-consuming activity, with some studies estimating that up to 60% of the effort in a data mining process is devoted to proper preprocessing [43]. This section approaches some of the most relevant procedures regarding data preprocessing.

4.1.1 Missing values

In most datasets, some observations either do not have values for all features analyzed in the generality of the observations, or do not have a label associated. In these cases, there is a missing value, which is a problem for the learning procedure, so it must be corrected.

The simplest option is to ignore/discard the observations with any missing value. However, by doing this, none of the values of those observations is used, when some can be useful. This is especially unfeasible in small datasets, where, with this procedure, one would get as a result an almost empty dataset.

As replacing the data manually is unfeasible and statistically unreliable, there is the need of having estimators, which are procedures that make logical predictions of the value of a feature (or a label) that is not in the original data [44]. An estimator should be unbiased, that is, the expected value of the features should be the same before and after the imputation of the values determined by an estimator.

Some of the most used estimators to fill a missing occurrence of a value of a feature are [41, 44]:

- **Using a constant** to fill each missing occurrence of a feature. This is usually not desired, as it does not take into account the context in which the missing data occurs.

- **Using an indicator of central tendency**, such as the mode, the mean or the median of all values of that feature. This method is fast and easy to implement, but may introduce some bias if not applied carefully. For example, in time-series data, imputing with the mean of all values of a feature is usually not correct, as each feature is expected to evolve over time.

- **Using an indicator of central tendency** as before, but **having the observations separated into different classes** and computing the indicator only over the values from observations belonging to the same class as the observation with a missing value. This method reduces some bias in cases where features behave very differently in distinct classes.

- **In time-series, using last observation carried forward (LOCF)**. This is a very popular method used in time-series. The method fills a missing value of a feature with the value of that feature in the previous observation of the respective time-series. The rationale behind this method is that, not knowing some value, the best "guess" is to assume that the value stays the same as before, which is often a reasonable assumption, especially in the medical context. However, it may introduce some bias in the observations [45].

- **Using the most probable value** to fill the missing value. This is the probabilistic way of imputing missing values. The concept is to choose as value to make the imputation the most probable value according to a probabilistic model, such as regression or a Bayesian method like the structural expectation-maximization (SEM) algorithm (see Sections 4.2.3 and 4.2.6 for details on this algorithm). Probabilistic models are often used, as they help avoid the introduction of bias.

4.1.2 Outlier detection

Even after satisfactorily filling the missing data and having consistent data, there are subsets of the dataset whose behavior is different from the common behavior of the observations of the dataset. These subsets are called outliers. Finding outliers is very important, as these observations are often originated by a different mechanism [46], so, they should be separated from the primary dataset, either for studying the outliers themselves or just for the study of the primary dataset not to be affected by the outliers.

Outliers can be divided into three groups/types [41]:

1. **Global outliers**. These outliers are the ones that significantly deviate from the whole dataset.

2. **Contextual/conditional outliers**. These observations are only outliers given a certain context of the observations. Regarding this kind of outliers, it is needed, for an observation, to specify its contextual attributes (the observation's context) and behavioral attributes (what defines the observation), being the relation between these two kinds of attributes that determines if an observation is an outlier or not.

3. **Collective outliers**. This kind of outliers consists of several observations that, together, are outliers, but individually may not be.

Regarding the approaches for finding outliers, they can be divided into two groups: according to the format of the data and according to the assumptions made about the differences between normal observations and outliers.

According to the format of the data, outlier detection methods can be divided into three groups [47]:

1. **Unsupervised methods**. In these methods, there is the goal of finding outliers without any prior knowledge of the data. When using these methods, there is the assumption that outliers can be completely separated from the primary dataset.

2. **Supervised methods**. In these methods, there is already training data pre-labeled as normal or outlier, so that an outlier detector can be properly trained. Outliers must be present in the training data in the most different ways possible, for the training to be effective.

3. **Semi-supervised methods**. In these methods, only the normal data is given labeled. Therefore, the outlier detection model to be generated must create some boundary for a new observation to be considered normal, classifying the observation as an outlier if it does not lie within that boundary.

According to the assumptions made regarding differences between normal observations and outliers, outlier detection methods can be divided into three groups [41]:

1. **Statistical/model-based methods**. These methods assume that normal observations are generated by a certain probabilistic model, from which outliers are not. In parametric methods, the probabilistic model is assumed *a priori*, while in non-parametric methods the probabilistic model is learned from data.

2. **Proximity-based methods**. These methods assume that an outlier is an observation whose proximity to its neighbors in the feature space is very different from the proximity of most observations to their neighbors. The two major types of this kind of methods differ on how "proximity" is determined: distance-based methods use the distance of an observation to its neighbors, while density-based methods use the density of an observation and its neighbors.

3. **Clustering-based methods**. These methods consider that big and dense clusters have normal data, while small and sparse clusters have outliers. Observations that are not in any cluster are also seen as outliers.

The aforementioned methods are common to the generality of outlier detection scenarios. When detecting outliers in temporal data, some additional considerations should be taken into account, so that the detection mechanisms are considerably influenced by the temporal continuity of the data: unexpected changes in the temporal evolution of data are a substantial indicator of outliers.

Regarding detecting outliers in time-series, there is the need of specifying which of the **two following goals** is desired: **(i)** given a dataset of several time-series, determine the time-series (or the sub-sequences) that deviate from the dataset; or **(ii)** given just a single time-series, determine the observations (or the sub-sequences) of that time-series that deviate from the time-series being analyzed.

To determine the outlier time-series in a certain dataset, often two approaches are considered [48]:

1. **Directly detecting the outlier time-series**. In this situation, a model is learned from the whole dataset. Then, assuming that most time-series of the dataset are not outliers, it is determined how different each time-series is from the model learned (usually, this is done using a score, which is higher the more similar a time-series is to the model), considering outliers the time-series that most deviate from the model.

2. **Detecting the outlier time-series by detecting outlier time-windows.** In these techniques, each time-series is seen as a succession of overlapping sub-sequences (time-windows) and it is evaluated, for each time-series, the similarity between each time-window and the respective time-window in all time-series of the dataset. The time-series with the most different time-windows are considered outliers (there are several ways of defining what it means for a time-series to have more different time-windows than other time-series).

To determine the outlier observations in a time-series, often three approaches are considered [48]:

1. **Using prediction models**. In this approach, a model is created for each observation of the time-series. Then, each observation is compared with the predicted value using the model created. The observations that most deviate from the model's predictions are considered outliers.

2. **Determining profile similarities.** The idea of this approach is to have a profile for normal observations and compare each observation with the profile, assessing an observation as an outlier if it significantly deviates from the profile of normal observations.

3. **Adopting a minimum description length (MDL) strategy.** The MDL is an information-theoretic scoring function which considers the trade-off between a model that accurately represents the data and a simple model (see Section 4.2.3 for more details). Applying the concept to outlier detection, the idea is to evaluate an observation as an outlier if the model that represents the time-series not considering the observation is substantially simpler than the model that represents the time-series when the observation is considered.

4.1.3 Data transformation

Data transformation, as explained in Chapter 3, is the last step of the preprocessing stage, and consists in converting data into formats adequate for patterns to be learned. In this section, there are covered three of the main possible data transformations: feature selection, normalization and discretization.

Feature selection

Feature selection consists in determining which features should contribute for learning patterns from data. This data transformation is very important, because, in practice, having irrelevant features may confuse the learning procedures [42].

There are two main approaches to select the best subset of features. The first approach is called filter method. The idea of the filter method is to filter the data according to the general properties of the dataset being used, producing a subset of data that, *a priori*, should be the most adequate. The second approach is called wrapper method. The idea of the wrapper method is to produce subsets from the whole dataset and evaluate them using the learning algorithm. The second approach is the most used, as it allows the learning procedure to give feedback about the proper features to be selected from the dataset.

Normalization

Normalization of data can be divided into the two following main categories [44]:

1. **Normalizing the variables' range**, that is, transforming the data so that all data lies within a specific range, which is usually small (typical ranges are [-1, 1] or [0, 1]). This kind of normalization is crucial, as it eliminates the dependency of the results on the unit of measure chosen for each feature. Three examples of possible normalizations are the min-max normalization, the Z-score normalization, and the decimal scaling normalization (each of these three given examples has a respective mathematical expression [41]).

2. **Normalizing the variables' distributions**, where the data is transformed to change the patterns obtained when grouping the several instances of the variables, so that these patterns conform to some desired ones. This kind of normalization is extremely important, as having the variables satisfying some known probabilistic distributions may help the development of simple and effective data mining processes. Two often used distributions are the normal and the uniform distributions.

Discretization

Discretization is the method used to transform the values of the features of a dataset into some fixed groups. These groups are usually identified by a number (unique number per group), and can represent either numeric intervals (for example, instead of representing all possible months in a dataset, one can divide months into two groups: months 1–6 and months 7–12) or conceptual categorical labels (in the same example, the division could be first and second semesters of a year). Discretization is an advantageous method of data reduction, which makes it an essential step to use simple learning models that only need to learn the specified groups, instead of the range of all possible values of the features. Three of the most used discretization techniques are the following [41]:

1. **Discretization by binning**, where the possible range of values is divided into K bins, being K *a priori* specified, and eventually tuned.

2. **Discretization by histogram analysis**, which consists in defining intervals according to histograms of the values of the data.

3. **Discretization by cluster analysis**. In this technique, there are generated clusters from data using a clustering algorithm, and then each cluster is seen as a group of the discretization to be done. As this procedure takes into account how close the values are (to create the clusters), this method usually gets good discretizations. However, clustering may be an expensive operation.

4.2 Learning with graphical models

Any learning model can be formulated as a set of complex mathematical expressions. However, it is beneficial that the mechanism behind the learning procedure is easily described, as that allows more people to understand the model and constructs models that are more easily interpretable [49].

One of the best ways to describe anything is through a graphical representation. Probabilistic graphical models (PGMs) represent the probability distribution of random variables in a graph-based way, providing several useful properties [50], such as giving insight into the model's properties just by graphical inspection and allowing complex mathematical expressions to become implicit by expressing them in terms of graphical manipulations.

Given that the main goal of the work is to provide doctors with a useful and easily understandable framework (see Section 1.3), graphical models are one of the best options to represent the data. The two most used PGMs are the Markov random fields (MRFs) and the Bayesian networks (BNs).

Markov random fields (MRFs)

Markov random fields [15, 51, 52] are undirected PGMs, which means that the edges do not mean by themselves any causality. In MRFs, the elements of interest are maximal cliques, which are cliques (subsets of nodes where each node is connected to every other node in the subset) that stop being cliques if any other node of the graph is added to the subset. In MRFs, the joint probability distribution of the random variables is given by

$$P(\boldsymbol{X}) = \frac{1}{Z} \prod_{C} \psi_C(\boldsymbol{X}_C), \tag{4.1}$$

where $\boldsymbol{X}$ are the model's random variables, C represents all possible maximal cliques of the graph, with nodes $\boldsymbol{X}_C$, Z is a normalization constant, and ψ_C are the potential functions, which may be any non-negative function and this is why there is the Z term, as the values may not be normalized. The main idea behind Eq. (4.1) is that if there is a function defined over a maximal clique C with nodes $\boldsymbol{X}_C$, defining any function over a subset of $\boldsymbol{X}_C$ would be redundant [50]. The MRF representation has a strong mathematical component, as the potential functions ψ_C do not have a probabilistic interpretation by themselves. This is why MRFs are not the option used in this Thesis. Besides, the calculation of Z has exponential complexity (although that exponential component can be minimized).

4.2.1 Bayesian networks

Bayesian networks are directed PGMs, where the edges have information of causality between nodes, and the whole directed graph represents the joint probability distribution of a set of variables. Definition 4.1 states rigorously the concept of Bayesian network.

Definition 4.1. *A Bayesian network B consists of a triple $\{\boldsymbol{X}, \boldsymbol{G}, \boldsymbol{\theta}\}$, where:*

- $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ *is a vector of n random variables.*
- $\boldsymbol{G} = \{\boldsymbol{X}, \boldsymbol{E}\}$ *is a directed acyclic graph (DAG) with nodes $\boldsymbol{X}$ and edges $\boldsymbol{E}$. Each node represents a random variable and each edge represents a conditional dependency relation between a pair of nodes. For simplicity of notation, each X_i represents both the random variable and the respective node in $\boldsymbol{G}$. The parents of X_i in $\boldsymbol{G}$ are represented by $\boldsymbol{pa}(X_i)$.*
- $\boldsymbol{\theta}$ *is the set of all conditional probabilities needed to encode the joint distribution of $\boldsymbol{X}$. If the variables are discrete with each variable X_i having at most r_i states, this allows the specification $\boldsymbol{\theta} = \{\theta_{ijk}\}$, where*

$$\theta_{ijk} = P_B(X_i = x_{ik} \mid \boldsymbol{pa}(X_i) = w_{ij}). \tag{4.2}$$

In Eq. (4.2)*, $i \in \{1, \ldots, n\}$, $k \in \{1, \ldots, r_i\}$ and $j \in \{1, \ldots, q_i\}$, with $q_i = \prod_{X_l \in \boldsymbol{pa}(X_i)} r_l$. The only probabilities that need to be given to define $\boldsymbol{\theta}$ are the probabilities for each node X_i to take each of the possible r_i values, given each of the possible parents' configurations w_{ij}. For each node X_i, the several θ_{ijk} are usually given in conditional probability tables (CPTs), where the rows are the possible w_{ij}, the columns are the possible x_{ik} and each entry is a θ_{ijk} according to Eq.* (4.2)*.*

The particularization done in the last bullet point of Definition 4.1 is only valid for discrete variables. As the dataset used in this work is discretized, this is not a big restriction. Although outside of the scope of this work, the same main principles apply to continuous-state random variables, where one main idea is to use mixtures of Gaussians [15, 50] instead of the CPTs used with discrete variables.

A BN always encodes the rule given in Definition 4.2.

Definition 4.2 (Conditional independence assumption in BNs)**.** *Given its parents, a node is conditionally independent of all other variables in the BN.*

Applying Definition 4.2 to Definition 4.1, a BN allows specifying the joint probability distribution over $\boldsymbol{X}$ in a simple way, given by

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \boldsymbol{pa}(X_i)). \tag{4.3}$$

Eq. (4.3) states that the joint probability distribution is obtained by multiplying all CPTs. As the CPTs are small comparing to the whole graph, the description of the joint probability distribution using BNs is significantly smaller than a naive description (exponential in the number of variables). This gain is due to the *implicit independence assumptions* [53] of the BN framework, stated in Definition 4.2. An also important aspect of Eq. (4.3) is that the joint probability distribution already comes normalized, unlike what happens in MRFs.

D-separation theorem

One of the main advantages of BNs is that the conditional independence properties can be expressed through graphical inspections. To do that, there is a theorem called d-separation. The theorem is based on the possible graphical constructions provided in Fig. 4.1.

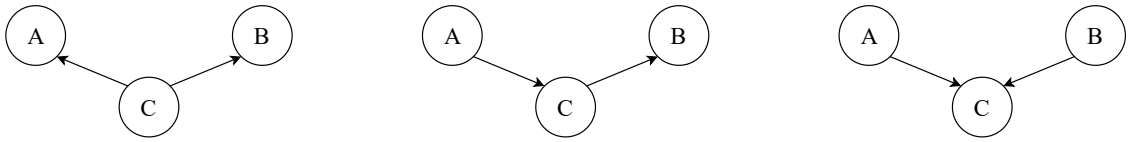


Figure 4.1: Possible configurations of three nodes in a BN: (a) diverging/tail-to-tail; (b) linear/head-to-tail; (c) converging/head-to-head/v-structure.

Given the structures of Fig. 4.1, the d-separation theorem is presented in Theorem 4.1 [50].

Theorem 4.1 (D-separation theorem)**.** *Given any non-intersecting sets of nodes A, B and C from a BN N, a path from a node in A to a node in B is blocked by C if it includes a node satisfying one of the following conditions:*

- *The arrows on the path meet either head-to-tail or tail-to-tail at that node and the node is in C.*
- *The arrows on the path meet head-to-head at that node and neither the node nor any of its descendants is in C (this phenomenon is known as explaining away).*

If all paths from all nodes in A to all nodes in B are blocked by C, then A is said to be d-separated from B by C and the joint probability distribution represented by the BN N satisfies $A \perp\!\!\!\perp B \mid C$.

Some authors define the meaning of a *d-connecting* path and state Theorem 4.1 using *d-connecting* paths [53].

Markov blanket

Given a BN B, the Markov blanket of a node X_i is the minimal set of nodes of B that separates X_i from the rest of the graph. In other words, given the values of the variables of the Markov blanket of X_i, no other variable of B influences the value taken by X_i. Therefore, knowing the Markov blanket of X_i allows knowing for which nodes to search to completely specify the distribution of X_i. Given Definition 4.2, one could think that the Markov blanket of X_i would be its parents and its children. However, the d-separation theorem (see Theorem 4.1) introduces the explaining away phenomenon in head-to-head configurations. As a result, the parents of the children of X_i also influence the value of X_i. This leads to conclude that **the Markov blanket of a node X_i in a BN is composed by the parents, children and parents of children (co-parents) of X_i.**

Dependency, independence and perfect maps

A graph is a dependency map (d-map) of a distribution if every conditional independence statement of the distribution is reflected in the graph. A graph is an independence map (i-map) of a distribution if every conditional independence property encoded in the graph is present in the distribution. A graph is said to be a perfect map of a distribution if it is simultaneously a d-map and an i-map of that distribution. The goal of a BN is to be a perfect map of the distribution it encodes.

I-equivalence

Two graphs are i-equivalent if they encode the same conditional independence properties. If two graphs are i-equivalent, there is no distribution for which one of them is an i-map and the other is not. If two graphs encoding a distribution $P(\boldsymbol{X})$ have the same skeleton (undirected graph obtained by removing the directions of all edges) and the same head-to-head structures, they are i-equivalent. If the graphs of two different BNs are i-equivalent, the two BNs encode the same joint probability distribution. G is an essential graph of a group Q of graphs if G is i-equivalent to all graphs in Q. Essential graphs may be partially DAGs (PDAGs) and are extremely useful when creating BNs to encode a certain probability distribution.

4.2.2 Inference in BNs

A probabilistic model should be able to, given the values of certain observed nodes, perform **two important queries**: **(i)** What are the values of certain unobserved nodes? **(ii)** What are the reasons for the observations? These queries are answered by making inference, where, given some **observed** nodes, there are found probability distributions over the values that some **hidden/latent** nodes can take.

Inference in BNs is based on Bayes' rule, applying it in the way described by

$$p(h \mid e) = \frac{p(e \mid h)p(h)}{p(e)}. \tag{4.4}$$

In Eq. (4.4), e denotes the *evidence*, the nodes whose values are observed, while h denotes the *hidden* nodes, whose values are unknown. Therefore, Eq. (4.4) allows computing the probability distribution over the values of the unknown nodes, given the information provided by the observed nodes.

Exact inference in BNs

In general, exact inference in BNs is NP-hard [54], forcing people to resort to approximate inference. However, when imposing restrictions in the graph structure, there are efficient exact inference algorithms. The most common restriction is to assume that the graph of a BN is a tree, where polynomial-time algorithms can be found.

The sum-product algorithm [50] is one of the best algorithms to make exact inference in polytrees (trees where each node may have more than one parent, as long as the graph stays singly connected).

To apply the sum-product algorithm, there is needed the concept of factor graph. Given a probability distribution over a graph, the joint distribution of $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ can be encoded in a product of factors, given by

$$P(\boldsymbol{X}) = \prod_s f_s(\boldsymbol{X}_s). \tag{4.5}$$

In Eq. (4.5), $\boldsymbol{X}_s$ denotes a subset of nodes. Comparing Eqs. (4.3) and (4.5), it can be seen that, in a BN, the factors f_s are given by the conditional probability expressions of Eq. (4.3).

A factor graph is a graph obtained by explicitly creating additional nodes for the f_s terms, called *factor nodes* and represented with squares, connecting them with undirected edges to the proper X_i on which each f_s depends, being these X_i called *variable nodes* and represented with circles. An example of a conversion of a graph of a BN to a factor graph is provided in Fig. 4.2.

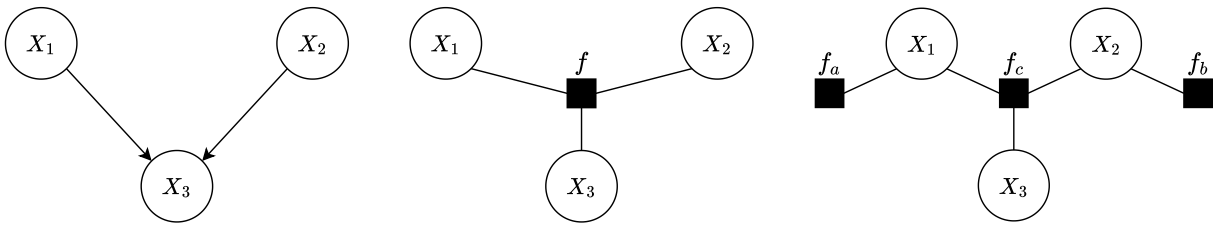


Figure 4.2: Directed graph and two equivalent factor graphs: (a) original directed graph; (b) factor graph with $f(X_1, X_2, X_3) = p(X_1)p(X_2)p(X_3 \mid X_1, X_2)$; (c) factor graph with $f_a(X_1) = p(X_1)$, $f_b(X_2) = p(X_2)$ and $f_c(X_1, X_2, X_3) = p(X_3 \mid X_1, X_2)$.

Fig. 4.2.c is the most interesting conversion, as it explicitly uses the conditional independence properties of BNs, presented in Eq. (4.3).

The sum-product algorithm uses propagation messages from factor nodes to variable nodes and vice-versa. These propagation messages are given by $\mu_{\text{sender}\to\text{receiver}}$. The messages sent from leaf factor nodes to variable nodes are given by

$$\mu_{f_{\text{Leaf}}\to X_i}(X_i) = f(X_i), \tag{4.6}$$

whereas the messages sent from non-leaf factor nodes to variable nodes are given by

$$\mu_{f_s\to X_i}(X_i) = \sum_{X_1} \cdots \sum_{X_M} f_s(X_i, X_1, \ldots, X_M) \prod_{m\in ne(f_s)\setminus X_i} \mu_{X_m\to f_s}(X_m). \tag{4.7}$$

The messages sent from leaf variable nodes to factor nodes are given by

$$\mu_{X_{\text{Leaf}}\to f_s}(X_{\text{Leaf}}) = 1, \tag{4.8}$$

whereas the messages sent from non-leaf variable nodes to factor nodes are given by

$$\mu_{X_m \to f_s}(X_m) = \prod_{k \in ne(X_m) \setminus f_s} \mu_{f_k \to X_m}(X_m). \tag{4.9}$$

In Eqs. (4.7) and (4.9), $ne(\cdot)$ denotes the neighbors of the specified node. In Eq. (4.7), the set $\{X_1, \ldots, X_M\}$ represents all neighbors of f_s, excluding X_i. The sum-product algorithm finds the probability distribution of every node X_i of a BN, being presented in Algorithm 4.1 [50].

Algorithm 4.1: Sum-product algorithm.

Input : A Bayesian network.
Output: Joint probability distribution over all variables.

1 Convert the BN graph into a factor graph.
2 Choose one X_i to be the root node and identify the according leaves.
3 Get the $\mu_{f_s \to X_i}(X_i)$ of all f_s connected to X_i, by starting propagations at the leaves according to Eqs. (4.6) and (4.8), and propagating messages until the root using Eqs. (4.7) and (4.9).
4 Compute $\widetilde{p}(X_i) = \prod_{f_s \in ne(X_i)} \mu_{f_s \to X_i}(X_i)$.
5 Normalize $\widetilde{p}(X_i)$ into $p(X_i)$ if needed (in BNs it is not needed, but in MRFs it would be).
6 Send $\mu_{X_i \to f_s}(X_i)$ to all f_s neighbors of X_i and propagate the messages from the root (X_i) to the leaves using Eqs. (4.7) and (4.9).
7 All nodes X_j of the graph of the BN have the $\mu_{f_s \to X_j}(X_j)$ of all f_s connected to them, so, every node X_j can compute its $\widetilde{p}(X_j)$ and its $p(X_j)$, by applying steps 4 and 5 to the desired X_j.

There is a famous algorithm called *belief propagation* [55], which is simply a special case of the presented sum-product algorithm.

If, instead of finding the joint probability distribution, one wants to determine the setting of variables with the highest probability and find that probability, there is the max-sum algorithm [50], which is an algorithm similar to Algorithm 4.1, but changing Eqs. (4.6) to (4.9) of the sum-product algorithm to reach the new desired goal.

Approximate inference in BNs

In situations where the structure of the graph of the BNs is not simplified, as inference is NP-hard, there must be used approximate methods. One of the most used algorithms is the expectation-maximization (EM) algorithm [15]. The idea of the EM algorithm is to perform a two-step iteration until convergence. In the *expectation* step, given the current parameters $\boldsymbol{\theta}$ and the observed values, there are computed *expected sufficient statistics* (ESS). In the *maximization* step, the ESS are treated as if they were actually observed and, using the maximum likelihood (see Section 4.2.3), there is derived a new set of parameters $\boldsymbol{\theta}$.

4.2.3 Structure and parameter learning in BNs

The problem of learning a BN B can be stated as, having a set of data S, finding the graph $\boldsymbol{G}$ and the parameters $\boldsymbol{\theta}$ that produce the BN that best describes S. One way of evaluating how well a certain BN B describes certain data S is using a scoring function $\phi(B, S)$ [56], being $\phi(B_1, S) > \phi(B_2, S)$ if B_1 describes the data S better than B_2. Therefore, given a search-space B_n with possible BNs, learning a BN B that best describes S can be put as the optimization problem given by

$$B_{\text{Best fit}} = argmax_{B_i \in B_n} \phi(B_i, S). \tag{4.10}$$

From the structure learning problem stated in Eq. (4.10), **two questions** immediately arise: **(i)** How is a proper scoring function ϕ defined? **(ii)** How should the search for BNs in Eq. (4.10) be done and what should the search-space B_n be? These two questions are answered in the next paragraphs.

Scoring functions

- **Decomposability and score-equivalence**

Before specifying several scoring functions, it is important to mention two properties, presented in Definitions 4.3 and 4.4. Decomposability is essential because it allows knowing that local changes in a variable X_i only affect a certain component of the scoring function, making sure there are no indirect effects. Score-equivalence is an interesting property that allows having simple algorithms.

Definition 4.3 (Decomposability)**.** *A scoring function of a BN B is decomposable iff*

$$\phi(B, S) = \sum_{i=1}^{n} \phi_i((X_i \mid \boldsymbol{pa}(X_i)), S). \tag{4.11}$$

Definition 4.4 (Score-equivalence)**.** *A scoring function of a BN B is score-equivalent if it assigns the same score to DAGs that are represented by the same essential graph (see Section 4.2.1).*

Scoring functions are usually divided into two groups, which are presented next.

1. **Bayesian scoring functions**

The rationale of the Bayesian scoring functions is that the best BN B is the one that maximizes the probability of, given the data S, B having been the BN that generated it. Therefore, the best BN B is the one that maximizes $P(B \mid S)$. As $P(S)$ is the same for all networks, maximizing $P(B \mid S)$ is the same as maximizing $P(B, S)$, which in turn is equal to maximizing $log(P(B, S))$. The latter option is normally preferred, as it is easier to maximize $log(P(B, S))$, because the values are usually small.

The following scoring functions are among the most used Bayesian scoring functions:

- *Bayesian Dirichlet* (BD) score [56]. This score makes certain assumptions on $P(B, S)$ and depends on multiple hyperparameters, which makes it unfeasible in practice.
- *K2 scoring function* [57], which is a specific case of the BD score where there are no hyperparameters, not using *a priori* information.
- *Likelihood-equivalence Bayesian Dirichlet* (BDe) score [56]. This score reduces the number of hyperparameters needed in the BD score, by imposing constraints on the model.
- *Likelihood-equivalence uniform joint distribution Bayesian Dirichlet* (BDeu) score [58], which is a specific case of the BDe score, where all distributions are assumed to be uniform *a priori*.

2. **Information-theoretic scoring functions**

This class of scoring functions is based on compression, which means that BNs that can represent the data S using simpler models are preferred.

- *Log-likelihood* (LL) score

The information content of S by B can be seen as the size of an optimal code induced by B when encoding S, being given by

$$L(S \mid B) = -logP(S \mid B) = -\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk} log(\theta_{ijk}), \tag{4.12}$$

where n is the number of nodes of the BN, r_i is the number of possible states of X_i and q_i is the number of possible configurations of the parents of X_i. N_{ijk} is the number of times in S where X_i takes its k-th value x_{ik} and the variables in $\boldsymbol{pa}(X_i)$ take their j-th configuration w_{ij} (see Eq. (4.2)).

The goal of BN structure learning can be seen as minimizing $L(S \mid B)$, as simpler models are preferred. Therefore, the simplest model that generates the data S is the one that directly maximizes $logP(S \mid B)$, as that is equivalent to minimizing $L(S \mid B)$, according to Eq. (4.12). By applying Gibbs' inequality, it can be shown that $\theta_{ijk} = \frac{N_{ijk}}{N_{ij}}$ minimizes $L(S \mid B)$, being N_{ijk} as stated in the previous paragraph and N_{ij} the number of times in S where $\boldsymbol{pa}(X_i)$ take their j-th configuration w_{ij}. This originates the LL score [56], which is given by

$$\phi_{LL}(B, S) = LL(S \mid B) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk} log\left(\frac{N_{ijk}}{N_{ij}}\right). \tag{4.13}$$

- Scores that directly penalize the complexity of the BNs

The two major problems of the LL score are that it overfits the data S and that it favours BNs with complete network structures. Therefore, there is the need to explicitly penalize the complexity of the BNs in the LL scoring function, which is usually done using scoring functions that take the structure of

$$\phi(B, S) = LL(S \mid B) - f(N) \times |B|, \tag{4.14}$$

where $f(N)$ is a penalty function, with N being the total number of observations in S, whereas $|B|$ represents the complexity of the proposed BN B [59], determined using the number of parameters of B, given by $|B| = \sum_{i=1}^{n}(r_i - 1)q_i$, being r_i and q_i as defined in Eq. (4.2). The *minimum description length* (MDL) [60] and the *Bayesian information criterion* (BIC) [61] scoring functions use $f(N) = \frac{1}{2}log(N)$. The *Akaike information criterion* (AIC) scoring function [62] uses $f(N) = 1$.

Search procedure for BNs

In Eq. (4.10), if the search-space B_n is composed by all possible BNs with n nodes, searching all BNs in B_n is NP-hard. There are two solutions: to reduce the search-space or to apply approximate algorithms.

- Search-space reduction

When reducing the search-space, a common approach is, as in inference, to reduce the search-space to trees, where the Chow & Liu algorithm [63], presented in Algorithm 4.2, can find a tree-BN that maximizes the LL score.

Algorithm 4.2: Chow & Liu algorithm.

Input : Input data S.
Output: A tree Bayesian network that maximizes the LL score.

1 Compute $I(X_i, X_j)$ between each pair (i, j), with $i \neq j$, being $I(X_i, X_j)$ the mutual information between X_i and X_j, which expresses how much information X_j gives about X_i and is given by $I(X_i, X_j) = \sum_{x_i \in X_i, x_j \in X_j} P(x_i, x_j) log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}$.

2 Build a complete undirected graph with vertices being the variables $\boldsymbol{X}$ and an edge between X_i and X_j having weight $I(X_i, X_j) = I(X_j, X_i)$.

3 Build an undirected maximum weighted spanning tree on the previously created undirected graph.

4 Transform the undirected spanning tree into a directed tree by choosing any node as root and giving directions to the arrows in the direction outwards the root, until reaching the leaves.

The Chow & Liu algorithm has polynomial-time complexity in the number of nodes of the BN and can be adapted to maximize any other decomposable scoring function ϕ (different from LL). If the scoring function is decomposable and score-equivalent, the adaptation consists in, at steps 1 and 2 of Algorithm 4.2, replacing $I(X_i, X_j)$ for the computation of $\phi_j(X_i, S) - \phi_j(\emptyset, S)$, which is the same as $\phi_i(X_j, S) - \phi_i(\emptyset, S)$. If the scoring function used is decomposable, but not score-equivalent, there must be created a directed graph having edges from X_i to X_j with weight $\phi_j(X_i, S) - \phi_j(\emptyset, S)$ and edges from X_j to X_i with weight $\phi_i(X_j, S) - \phi_i(\emptyset, S)$. Then, Edmonds' algorithm [64] can be used to compute a directed maximum spanning tree from the created directed graph.

- Approximate search algorithms

If the BN structure cannot be simplified, the only efficient option is resorting to heuristics. One of the simplest yet most effective techniques is the greedy hill-climbing (GHC). Starting at some initial BN and using a decomposable scoring function ϕ, GHC tries several local changes to the BN structure and applies the changes that increase the score of ϕ. The algorithm stops after a defined number of iterations, or when it reaches a local maximum (when each change it tries to do only worsens the score of the BN). There are several ways to, using GHC, be "smart" in the initial BN chosen and in the changes that are tried.

Another often used technique is the structural expectation-maximization (SEM) algorithm [17, 65]. As explained in Section 4.2.2, the EM algorithm is used to make inference in BNs. The SEM algorithm applies the same principles of the EM algorithm, but changes the maximization step, so that it uses expected counts according to the current BN structure. SEM allows finding BNs with high score, according to scores like the BDe score.

Parameter learning in BNs

Parameter learning in BNs is usually done assuming that the structure of the BN is already known/learned. With known structure, the simplest and most common way of learning the parameters of a BN is through the observed frequency estimates (OFE). Given a BN B, with known structure, and a dataset S, OFE determines each parameter θ_{ijk} of Eq. (4.2) by doing

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}, \tag{4.15}$$

where N_{ijk} is the number of times in S where the node X_i of B takes its k-th value x_{ik} and the nodes in $\boldsymbol{pa}(X_i)$ of B take their j-th configuration w_{ij}, while N_{ij} is the total number of times in the dataset S where the nodes

in $\boldsymbol{pa}(X_i)$ of B take their j-th configuration w_{ij} (see Eq. (4.2)). OFE is a particular case of the more general parameter learning method described by

$$\hat{\theta}_{ijk} = \frac{N'_{ijk} + N_{ijk}}{N'_{ij} + N_{ij}}, \tag{4.16}$$

where N'_{ijk} and N'_{ij} are Dirichlet parameters that express prior information about the probability distributions [66]. Making $N'_{ijk} = N'_{ij} = 0$ leads to Eq. (4.15), where no prior knowledge is assumed.

4.2.4 Dynamic Bayesian networks

Section 4.2.1 presents BNs as a framework to represent the joint probability distribution over random variables $\boldsymbol{X}$. However, it does not allow reasoning about the evolution of the state of the variables over time. In real-world applications, variables are often presented as time-series, where values are given in different temporal moments and there is the need to reason about how random variables $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ evolve over time.

Dynamic Bayesian networks (DBNs) offer a framework for working with variables expressed as time-series, by extending the concepts of Section 4.2.1 in order to represent dynamic systems.

To represent the time associated to the random variables, $X_i[t]$ is used to denote the random variable that, in timestep t, is associated to the feature X_i. To use this notation, there is the assumption that the time-series is discretized into intervals of time with the same length. It is also assumed that a model has T total timesteps/time-slices and that the first timestep is zero. Therefore, $X_i[0:T]$ denotes all random variables of the model that are associated to the feature X_i. In general, $X_i[a:b]$ denotes all random variables associated to X_i from timestep a to timestep b, where there is one random variable associated to X_i in each intermediate timestep.

In temporal models, there is the notion of *trajectory* of a random variable X_i, which is an assignment of values to X_i in each intermediate timestep from a certain initial to a certain final timestep. The joint probability distribution of all possible trajectories of a variable X_i from timestep a to timestep b is denoted as $P(X_i[a:b])$. Therefore, the joint probability distribution of all possible trajectories of all features of a DBN, from timestep a to timestep b, is denoted as $P(\boldsymbol{X}[a:b])$. DBNs allow having a probabilistic representation of the joint probability distribution of all possible trajectories from timestep 0 to timestep T, that is, $P(\boldsymbol{X}[0:T])$. Definition 4.5 presents the rigorous definition of DBN [18].

Definition 4.5 (Dynamic Bayesian network)**.** *A dynamic Bayesian network (DBN) is a graphical model given by a pair (B_0 , $\boldsymbol{B}_{\rightarrow}$), where:*

- *B_0 is a prior BN that, using the BN framework, defines a probability distribution over the variables in timestep 0, that is, $B_0 = P(\boldsymbol{X}[0])$.*

- *$\boldsymbol{B}_{\rightarrow}$ consists of the transition networks, where $B_{\rightarrow}[0:t]$ defines the distribution over the variables in timestep t, given all the trajectories the variables may take from timestep 0 until timestep $t-1$. Rigorously,*

$$B_{\rightarrow}[0:t] = P(\boldsymbol{X}[t] \mid \boldsymbol{X}[0:t-1]), \tag{4.17}$$

and a DBN defines $\boldsymbol{B}_{\rightarrow}$, which is the set of all $B_{\rightarrow}[0:t]$, for $t \in \{1, \ldots, T\}$.

- *Using B_0 and $\boldsymbol{B}_{\rightarrow}$ and applying the chain rule, a DBN defines the joint probability distribution of all possible trajectories of all features using the expression*

$$P(\boldsymbol{X}[0:T]) = B_0 \prod_{t=1}^{T} B_{\rightarrow}[0:t] = P(\boldsymbol{X}[0]) \prod_{t=1}^{T} P(\boldsymbol{X}[t] \mid \boldsymbol{X}[0:t-1]). \tag{4.18}$$

In a DBN, the relations between the nodes (random variables) are often distinguished into two types: the intra-slice connectivity, regarding dependencies between random variables from the same timestep, and the inter-slice connectivity, regarding the effects that random variables from previous timesteps have in future time-slices.

From Definition 4.5, it can be concluded that, given B_0 and $\boldsymbol{B}_{\rightarrow}$, a DBN can be represented as a BN, using for DBNs the same graphical representation of BNs. Fig. 4.3 presents an example of a DBN composed by three time-slices and unrolled to be shown graphically as a BN.

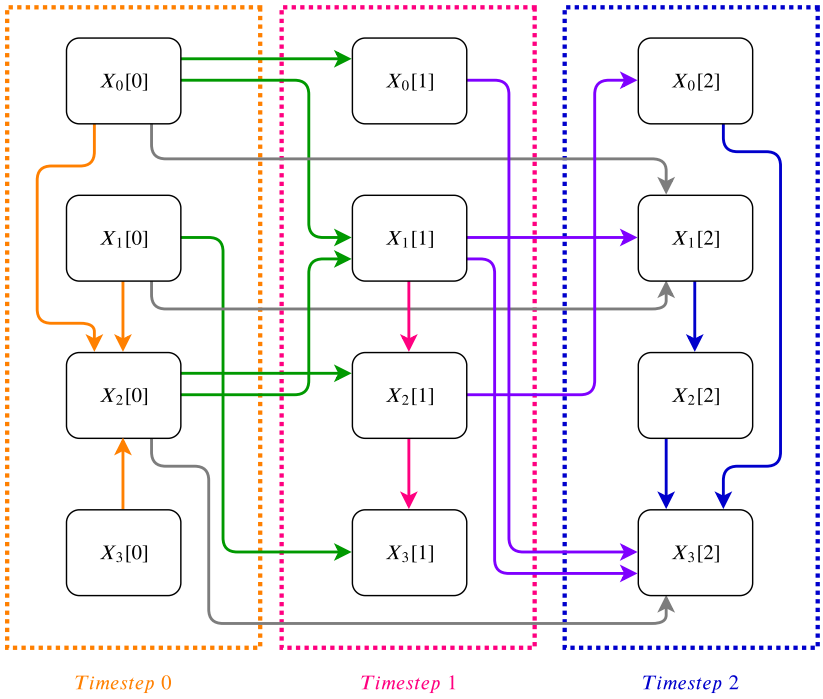


Figure 4.3: DBN with three timesteps unrolled as a BN.

In Fig. 4.3, the orange, pink and blue arrows show the intra-slice connectivity of timesteps 0, 1 and 2, respectively, while the green and purple arrows show the inter-slice connectivity between timesteps 0 and 1 and between timesteps 1 and 2, respectively. The grey arrows show the inter-slice connectivity between timesteps 0 and 2. In Fig. 4.3, if the proper CPTs were given, then: **(i)** the orange arrows would define B_0; **(ii)** the orange, green and pink arrows would define $B_{\rightarrow}[0:1]$; **(iii)** $B_{\rightarrow}[0:2]$ would be defined by all arrows of Fig. 4.3.

In long time-series with a high number of features, specifying all $B_{\rightarrow}[0:t]$ of $\boldsymbol{B}_{\rightarrow}$ from Definition 4.5 is unfeasible, as the number of possibilities is exponential in the number of variables and in the number of timesteps (one can see that for the simple example of Fig. 4.3 there are already several arrows). Therefore, often two assumptions, Markov and stationary, are imposed to DBNs, to make the representation a treatable problem.

Markov assumption

The m^{th}-order Markov assumption, presented in Definition 4.6, simplifies the problem of finding all $B_{\rightarrow}[0:t]$ by assuming that the distribution of $\boldsymbol{X}$ in a certain timestep t, instead of being dependent on the values of $\boldsymbol{X}$ from timestep 0 until timestep $t-1$, only depends on the values of $\boldsymbol{X}$ from m timesteps before timestep t.

Definition 4.6 (m^{th}-order Markov assumption)**.** *A dynamic system satisfies the m^{th}-order Markov assumption, where m is called the Markov lag, iff, for all $t \in \{1, \ldots, T\}$,*

$$\boldsymbol{X}[t] \perp\!\!\!\perp \boldsymbol{X}[0:t-1-m] \mid \boldsymbol{X}[t-m:t-1]. \tag{4.19}$$

The Markov assumption is very useful, as the conditional independence property of Eq. (4.19) allows simplifying the expression of each $B_{\rightarrow}[0:t]$ of Eq. (4.17), getting the simplified expression given by

$$B_{\rightarrow}[0:t] = P(\boldsymbol{X}[t] \mid \boldsymbol{X}[t-m:t-1]). \tag{4.20}$$

Eq. (4.20) allows simplifying Eq. (4.18), defining $P(\boldsymbol{X}[0:T])$ as

$$P(\boldsymbol{X}[0:T]) = B_0 \prod_{t=1}^{T} B_{\rightarrow}[0:t] = P(\boldsymbol{X}[0]) \prod_{t=1}^{T} P(\boldsymbol{X}[t] \mid \boldsymbol{X}[t-m:t-1]). \tag{4.21}$$

If, in Definition 4.6, $m = 1$, there is the very often used 1^{st}-order Markov assumption, that can be stated as *the future being conditionally independent of the past given the present* [15].

In terms of the graphical representation of DBNs, making an m^{th}-order Markov assumption affects the range that the grey arrows of Fig. 4.3 can take. For example, if there was a 1^{st}-order Markov assumption on the DBN of Fig. 4.3, the grey arrows could not exist.

Stationary assumption

While the Markov assumption simplifies the expressions of $B_{\rightarrow}[0:t]$ by assuming conditional independence properties, the stationary assumption, presented in Definition 4.7, allows simplification by assuming that the transition networks in a dynamic system satisfying an m^{th}-order Markov assumption are all the same.

Definition 4.7 (Stationary assumption)**.** *A dynamic system satisfying the m^{th}-order Markov assumption is said to be stationary if $P(\boldsymbol{X}[t] \mid \boldsymbol{X}[t-m:t-1])$ is the same for all t.*

Assuming that all transition probabilities are equal is a huge simplification that is often not theoretically appropriate. For example, in the medical domain, the transition probabilities are usually not the same for a patient in different stages of a certain disease. However, comparing to modeling stationary DBNs, modeling non-stationary DBNs is much more difficult as, even if making a Markov assumption, Eq. (4.21) results in the necessity of getting several different expressions for the transition probabilities.

In terms of the graphical representation of DBNs, making a stationary assumption implies restricting the possible configurations of the green and purple arrows of Fig. 4.3. For example, if there were made a stationary and

a 1^{st}-order Markov assumptions on the DBN of Fig. 4.3, besides the grey arrows not being allowed, the configuration of the green and purple arrows would have to be the same (in that situation, the parent-children relations between the nodes of two consecutive timesteps should always be the same).

4.2.5 Inference in DBNs

As in BNs, the goals of making inference in DBNs are also determining both the values of certain unobserved nodes and the reasons for certain observations. The concept from Section 4.2.2 regarding observed and hidden variables remains valid in the study of inference in DBNs, as does Eq. (4.4). However, in DBNs, there is the added dimension of having features in different timesteps. When having certain nodes observed from timestep 0 until a certain timestep t_0, being these observations denoted as $\boldsymbol{e}[0 : t_0]$, there are mainly four operations that may be done regarding inference in DBNs [16]:

1. **Filtering:** if inference is done to, given the evidences from timesteps $t \leq t_0$, determine the probability distributions of hidden nodes in timestep t_0, that is, $P(\boldsymbol{X}[t_0] \mid \boldsymbol{e}[0 : t_0])$.

2. **Prediction:** if inference is done to, given the evidences from timesteps $t \leq t_0$, determine the probability distributions of hidden nodes in a timestep $t > t_0$, that is, $P(\boldsymbol{X}[t_0 + k] \mid \boldsymbol{e}[0 : t_0])$, with $k > 0$.

3. **Smoothing:** if inference is done to, given the evidences from timesteps $t \leq t_0$, determine the probability distributions of hidden nodes in a timestep $t < t_0$, that is, $P(\boldsymbol{X}[t_0 - k] \mid \boldsymbol{e}[0 : t_0])$, with $k > 0$.

4. Compute the **most likely explanation (MLE):** if the goal is to, given the evidences from timesteps $t \leq t_0$, determine the most probable sequence of values, of the random variables $\boldsymbol{X}$ between timesteps 0 and t_0, to have generated the observed evidences, that is, the goal is to determine $argmax_{\boldsymbol{X}[0:t_0]}\{P(\boldsymbol{e}[0 : t_0])\}$. It is important to emphasize that computing the MLE is usually not the same as performing smoothing for each timestep separately, as just aggregating values that individually maximize the probabilities in each timestep may not result in the sequence that maximizes the overall probability.

Exact inference in DBNs

If exact inference in BNs is already an NP-hard problem, adding the temporal component of DBNs the problem becomes even more difficult. The main ideas for exact inference in DBNs consist in generalizing to the dynamic domain the concepts of exact inference in BNs, only being the algorithms efficient when imposing restrictions in the graph structure.

To perform filtering, prediction and smoothing, one of the most used algorithms is the forward-backward algorithm [15, 16]. The idea of the algorithm is to generalize the sum-product algorithm (see Algorithm 4.1) to the temporal domain. The forward pass performs filtering and predictions, followed by a backward pass that performs smoothings.

To compute the MLE, there is the Viterbi algorithm [16], which is identical to the forward pass of the forward-backward algorithm, but changing some summations for maximums, just as when changing the sum-product algorithm to get the max-sum algorithm (see Section 4.2.2).

Approximate inference in DBNs

To perform approximate inference in DBNs, one algorithm often used is the particle filtering [15–17]. The idea of the algorithm is to, starting from a population of γ samples randomly taken from the probability distribution of $\boldsymbol{X}[0]$, apply approximate filterings by repeating the following **three steps** when going from timestep t to timestep $t+1$: **(i) propagate** each sample from timestep t to timestep $t+1$ using the known transition probabilities; **(ii) weight** each sample by the probability of the evidence observed given the known model; **(iii) resample** the population of particles according to the weights previously produced, in order to obtain a new estimation of $P(\boldsymbol{X}[t+1] \mid \boldsymbol{e}[0:t+1])$. Repeating the three previous steps in each timestep transition and performing the procedure for a considerable number of particles, the algorithm is consistent (converges to the correct probabilities as $\gamma \to \infty$). It is usually also efficient, in the sense that, generally, there is not needed an extremely high number of particles to get relatively good estimates of the desired probability distributions.

4.2.6 Structure and parameter learning in DBNs

Regarding structure learning in DBNs, as happens in inference, the same principles of BNs apply to DBNs. The problem in DBNs can also be seen as an optimization problem described by Eq. (4.10), and the scoring functions presented in Section 4.2.3 still apply.

However, comparing to BNs, DBNs have, in the search-space of possible structures, the influence of the inter-slice connectivity, which BNs do not have. Regarding the intra-slice connectivity, the learning procedures used are essentially the same of BNs, as each time-slice can be seen as a BN. Regarding the inter-slice connectivity, the fact that only lower timesteps affect future ones, and not the other way around, avoids needing to confirm if the graph is acyclic, which can lead to simple and fast algorithms, especially in small datasets [67].

A commonly used set of assumptions resides in considering the 1^{st}-order Markov assumption and that the intra-slice connectivity is fixed (it is usual to only determine the intra-slice connectivity of timestep 0). This allows stating structure learning of DBNs as a feature-selection problem, where, for each $t \in \{1, \ldots, T\}$, the goal is to determine which nodes from timestep $t-1$ are the parents of each node in timestep t. If also making the stationary assumption, the complexity of the problem may reduce even more.

Most literature on structure learning of DBNs [17, 18] divides the learning problem into two groups, according to all random variables being observed in data or not. The two groups are presented next.

1. **Learning fully observed DBNs / learning from complete data**

These situations concern the cases where each X_i of each timestep of a DBN is directly observed in the dataset, with no hidden variables nor missing data. In these cases, the learning methods are essentially extensions of the methods used in BNs. There must be some care in trying to take advantage of the existing constraints of the DBNs when applying the methods showed in Section 4.2.3, instead of just unrolling a DBN into a large BN and then learning the unrolled BN. Due to the fact that DBNs usually have a high number of nodes, greedy search procedures (where there are techniques to fasten the processes, such as searching for B_0 independently from $\boldsymbol{B}_{\rightarrow}$, as they may not be correlated) or the MCMC algorithm are very often used. Another approach is to consider the intra-slice connectivity as trees and use the Chow & Liu algorithm for intra-slice structure learning, also adding parents from preceding time-slices to each node, creating a tree-augmented DBN (tDBN) [25].

2. **Learning partially observed DBNs / learning from incomplete data**

These situations are the most common ones, as they may arise, for example, from the model itself not being fully observable, from having missing data (that can be seen as *hidden* nodes) or even from the necessity of creating hidden nodes to properly make the assumption of having a stationary m^{th}-order Markov process, but according to those hidden nodes and not to the actual observations. In these circumstances, generalizing to DBNs the scoring functions used in BNs may not be straightforward and must be done carefully, according to the learning algorithms. One of the most common approaches in these situations is to use an extension of the structural expectation-maximization (SEM) algorithm (see Section 4.2.3) to dynamic systems [18, 68].

Parameter learning in DBNs

To perform parameter learning in DBNs, it is important to define how the parameters of a DBN are represented. The parameters θ_{ijk} from the BN framework (see Definition 4.1) can be extended to the parameters $\theta_{ijk}[t]$ of the DBN framework, by, given a DBN B_d with known structure, designating $\theta_{ijk}[t]$ as

$$\theta_{ijk}[t] = P_{B_d}(X_i[t] = x_{ik} \mid \boldsymbol{pa}(X_i[t]) = w_{ij}[t]). \tag{4.22}$$

In Eq. (4.22), $\boldsymbol{pa}(X_i[t])$ denotes the parents of $X_i[t]$ in B_d, which can be in timesteps $t' \leq t$. The terms x_{ik} and $w_{ij}[t]$ are the several values that $X_i[t]$ and the parents of $X_i[t]$ may take, respectively (see Eq. (4.2)). The x_{ik} terms are not indexed by t, because the values that a node $X_i[t]$ can take are independent of the timestep t. The $w_{ij}[t]$ terms are indexed by t, as the possible values that the parents of $X_i[t]$ can take vary according to the parents of $X_i[t]$, which depend on the timestep t. In particular, $j \in \{1, \ldots, q_i[t]\}$, with $q_i[t] = \prod_{X_l \in \boldsymbol{pa}(X_i[t])} r_l$, where a certain r_l is the maximum number of states of the respective node X_l (see Eq. (4.2)).

Parameter learning in DBNs is an extension of parameter learning in BNs to the temporal domain. As in BNs, the structure is normally assumed to be known, and parameter learning is usually done using the OFE.

In non-stationary DBNs, given a DBN B_d, with known structure, and a dataset D with observations of the nodes of B_d, the OFE method used in BNs can be applied to DBNs by extending Eq. (4.15) to use the $\theta_{ijk}[t]$ parameters defined in Eq. (4.22), obtaining the expression

$$\hat{\theta}_{ijk}[t] = \frac{N_{ijk}[t]}{N_{ij}[t]}, \tag{4.23}$$

where $N_{ijk}[t]$ is the number of times in D where the node $X_i[t]$ of B_d takes its k-th value x_{ik} and the nodes in $\boldsymbol{pa}(X_i[t])$ of B_d take their j-th configuration $w_{ij}[t]$, while $N_{ij}[t]$ is the total number of times in the dataset D where the nodes in $\boldsymbol{pa}(X_i[t])$ of B_d take their j-th configuration $w_{ij}[t]$ (see Eq. (4.22)).

In stationary DBNs, as the transition networks are the same for all timesteps, the $\theta_{ijk}[t]$ parameters should be the same for all t, representing the global OFE of a certain dataset. This can be done by changing Eq. (4.23) to

$$\hat{\theta}_{ijk} = \frac{\sum_t N_{ijk}[t]}{\sum_t N_{ij}[t]}, \tag{4.24}$$

where $N_{ijk}[t]$ and $N_{ij}[t]$ are as defined in Eq. (4.23), and $\hat{\theta}_{ijk}[t] = \hat{\theta}_{ijk}, \forall t$.

Chapter 5

Contributions on DBNs state-of-the-art methods

5.1 Learning DBNs with static attributes (sdtDBNs)

When starting the study of ALS disease progression from patients' data, it can be noticed that, although the ALS dataset has both static and dynamic features/attributes, the standard DBN framework only allows including dynamic attributes in the study of the disease progression. This issue is of enormous importance in the generality of situations where machine learning techniques are applied to study the temporal progression of medical indicators, as usually there are static attributes of patients that may not be taken into account if applying machine learning methods that only allow considering dynamic attributes. Therefore, the first main concern of the Thesis is to develop a mechanism to include static attributes in the analysis of the disease progression using DBNs.

5.1.1 sdtDBNs as extensions of tDBNs

As already explained, this Thesis uses as basis to the proposed DBN learning methodology the tDBN framework, developed by a previous student [25, 69] (see Section 1.4 for more details). The tDBN algorithm optimally learns the intra and inter time-slice relations of a DBN, restricting the search-space of the intra-slice connectivity to tree structures and considering, for each node of each timestep, a maximum number of parents from the preceding timesteps, being the preceding timesteps considered for each node defined by a certain Markov lag.

In this work, the proposal is to include static attributes in the tDBN framework by considering these static attributes as nodes in a "meta-timestep" whose nodes can always influence any dynamic node of any timestep, independently of the specified Markov lag. The proposed optimal DBN structure learning algorithm produces tDBNs with both static and dynamic attributes, being named sdtDBN ("sd" stands for static and dynamic). The learning procedure, described next, follows closely the learning procedure of tDBNs [25, 69].

Notation

To represent the static and dynamic nodes of the DBNs, $\boldsymbol{Y}$ denotes the static attributes and $\boldsymbol{X}[t]$ represents the dynamic attributes in timestep t. It is assumed that exist n_{static} static attributes and n dynamic attributes in each timestep, with a maximum of T timesteps. Therefore, Y_k, with $k \in \{0, \ldots, n_{static} - 1\}$, denotes each possible static attribute, while $X_i[t]$, with $i \in \{0, \ldots, n-1\}$ and $t \in \{0, \ldots, T-1\}$, designates each possible dynamic attribute X_i in each possible timestep t. All attributes are assumed to be discrete, with a finite number of states.

The DBNs to be created can be stationary or non-stationary and it is assumed that the DBNs have a certain Markov lag m. From the tDBN definition [25, 69], each node $X_i[t]$ has one parent from timestep t (except for the root of the intra-slice tree of each timestep). It is also assumed that each node $X_i[t]$ has a maximum of p dynamic parents from the m previous timesteps and a maximum of b static parents.

The dataset with observations of static attributes is denoted as S, whereas the dataset with observations of dynamic attributes is represented by D, with D_t^{t+j} being the observations of dynamic attributes between timesteps t and $t+j$ (including both t and $t+j$). The possible nodes' configurations of the DBNs are evaluated using a decomposable scoring function ϕ, with local terms given by ϕ_i, for each node $X_i[t]$ of a DBN.

Learning algorithm

As in the tDBN algorithm, the sdtDBN algorithm starts by determining the best possible set of parents for each node, excluding connections among nodes in the same timestep. To do that, $P_{\leq \gamma}(\boldsymbol{A})$ is used to represent all possible subsets, with cardinality at most γ, of a certain set $\boldsymbol{A}$. Therefore, $P_{\leq b}(\boldsymbol{Y})$ denotes all possible sets of static parents of each dynamic node, whereas $P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t])$ denotes, for a node $X_i[t+1]$, in timestep $t+1$, all possible sets of dynamic parents from the m preceding timesteps. Given the previous subsets, it is possible to rigorously obtain, for each node $X_i[t+1]$, the set of static and dynamic parents that achieves the best possible score when not considering any connections among nodes in timestep $t+1$, being that score given by

$$s_i = \max_{\boldsymbol{X}_{dp} \in P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t]), \boldsymbol{Y}_{sp} \in P_{\leq b}(\boldsymbol{Y})} \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp}, D_{t+1-m}^{t+1} \cup S), \tag{5.1}$$

where the $\boldsymbol{Y}_{sp}$ and $\boldsymbol{X}_{dp}$ that maximize Eq. (5.1) for each value of i are the optimal sets of static and dynamic parents of each node $X_i[t+1]$, when not considering any connections in timestep $t+1$.

When also considering the connections in timestep $t+1$, as each node $X_i[t+1]$ may have one parent from timestep $t+1$, it should be found the set of static and dynamic (from previous timesteps) parents that, for each node $X_i[t+1]$, achieves the best possible score, considering that each of the possible $X_j[t+1] \rightarrow X_i[t+1]$ connections in timestep $t+1$ is in the DBN structure, being the maximum score obtained for each pair ij given by

$$s_{ij} = \max_{\boldsymbol{X}_{dp} \in P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t]), \boldsymbol{Y}_{sp} \in P_{\leq b}(\boldsymbol{Y})} \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp} \cup X_j[t+1], D_{t+1-m}^{t+1} \cup S), \tag{5.2}$$

where the $\boldsymbol{Y}_{sp}$ and $\boldsymbol{X}_{dp}$ that maximize Eq. (5.2) are the optimal sets of static and dynamic parents of each node $X_i[t+1]$, considering that the connection $X_j[t+1] \rightarrow X_i[t+1]$ is in the DBN structure.

The general idea for optimally and globally choosing the best sdtDBN structure is to assess, for each pair of nodes $(X_i[t+1], X_j[t+1])$, the benefit e_{ij} of including $X_j[t+1]$ as a parent of $X_i[t+1]$ in the network structure,

instead of just having $X_i[t+1]$ with the optimal static parents from $\boldsymbol{Y}$ and the optimal dynamic parents from $\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t]$. The aforementioned benefit can be obtained from the relation between Eqs. (5.1) and (5.2), getting the benefit terms e_{ij} defined as

$$e_{ij} = s_{ij} - s_i. \tag{5.3}$$

After determining all e_{ij} terms with Eq. (5.3), which uses the s_i and s_{ij} terms from Eqs. (5.1) and (5.2), where the static attributes influence the optimal sets of parents of each node, the procedure to determine an optimal sdtDBN structure using the e_{ij} terms is the same procedure used in tDBNs structure learning [25, 69] (although in tDBNs the calculation of the e_{ij} terms does not have the influence of the static attributes).

In order to get the proper static and dynamic parents of the nodes in a timestep $t+1$, first, a complete directed graph with the nodes from $\boldsymbol{X}[t+1]$ should be computed, being the weight of each edge $X_j[t+1] \rightarrow X_i[t+1]$ the respective e_{ij} term. Then, a maximum spanning tree must be computed in the directed graph created. As explained in the tDBN work [25, 69], in general $e_{ij} \neq e_{ji}$, so, Edmonds' maximum branching algorithm [64] should be used to find the desired spanning tree.

After finding the mentioned spanning tree, its edges compose the intra-slice connectivity of timestep $t+1$. For each edge $X_j[t+1] \rightarrow X_i[t+1]$, the proper static parents of $X_i[t+1]$ and dynamic parents of $X_i[t+1]$ from the previous timesteps are the $\boldsymbol{Y}_{sp}$ and $\boldsymbol{X}_{dp}$ that maximize Eq. (5.2) for the respective pair ij. Regarding the root node of the spanning tree, denoted as $X_r[t+1]$, its static parents and dynamic parents from the previous timesteps are the $\boldsymbol{Y}_{sp}$ and $\boldsymbol{X}_{dp}$ that maximize Eq. (5.1) for $i = r$. By applying the described procedure to all timesteps, an optimal sdtDBN structure can be found. Algorithm 5.1 presents an overview of the sdtDBN learning methodology, using Algorithm 5.2 to determine the optimal sets of parents and the terms e_{ij}.

Algorithm 5.1: Structure learning of m^{th}-order Markov non-stationary sdtDBNs.

Input :

- $\boldsymbol{X}$: the n dynamic attributes of the DBN.
- $\boldsymbol{Y}$: the n_{static} static attributes of the DBN.
- T: the total number of timesteps of the DBN.
- D: dataset with observations for each dynamic node of the DBN.
- S: dataset with observations for each static node of the DBN.
- ϕ: a decomposable scoring function.

Output :

- A non-stationary tree-augmented DBN structure with both static and dynamic attributes (sdtDBN).

1 **for** *each timestep* $t+1$ *between* m *and* T **do**
2 Construct a complete directed graph with vertices $\boldsymbol{X}[t+1]$.
3 Determine the weights of all edges $X_j[t+1] \rightarrow X_i[t+1]$, $i \neq j$, using Algorithm 5.2, also extracting the optimal sets of static and dynamic parents of each node $X_i[t+1]$.
4 Apply a maximum branching algorithm in the graph of line 2 using the weights determined in line 3, in order to have a maximum spanning tree considering the weights of line 3.
5 Extract the static and dynamic parents of each node $X_i[t+1]$ from the spanning tree determined in line 4 and the sets of parents determined in line 3.
6 Join all transitions determined in the **for** loop, to get the complete description of the determined sdtDBN.

Algorithm 5.2: Edge weights and optimal parents for m^{th}-order Markov non-stationary sdtDBNs.

Input :

- $t+1$: the current timestep.
- m: the Markov lag of the DBN.
- $\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t+1]$: sets of nodes from the current timestep and the m previous timesteps, each timestep having n nodes.
- $\boldsymbol{Y}$: set of n_{static} static nodes.
- p: upper-bound on the number of dynamic parents from the m previous timesteps.
- b: upper-bound on the number of static parents.
- D_{t+1-m}^{t+1}: dataset with observations for each dynamic node, from timesteps $t+1-m$ to $t+1$.
- S: dataset with observations for each static node.
- ϕ_i(parentNodes, dataset): local terms of the decomposable scoring function ϕ, for each $X_i[t+1]$.

Output :

- $E_{[n \times n]}$: matrix with edge weights e_{ij}.
- *dynamicParentsPast*$_{[n]}$: for each node $X_i[t+1]$, the optimal set of dynamic parents from the m previous timesteps, when not considering connections in timestep $t+1$.
- *staticParentsPast*$_{[n]}$: for each node $X_i[t+1]$, the optimal set of static parents, when not considering connections in timestep $t+1$.
- *dynamicParents*$_{[n \times n]}$: for each node $X_i[t+1]$, the optimal set of dynamic parents from the m previous timesteps, when considering each possible connection $X_j[t+1] \rightarrow X_i[t+1]$ in the sdtDBN.
- *staticParents*$_{[n \times n]}$: for each node $X_i[t+1]$, the optimal set of static parents, when considering each possible connection $X_j[t+1] \rightarrow X_i[t+1]$ in the sdtDBN.

allDynamicParentSets $\leftarrow P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t])$
allStaticParentSets $\leftarrow P_{\leq b}(\boldsymbol{Y})$
for $X_i[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
  bestScore $\leftarrow -\infty$
  for $\boldsymbol{X}_{dp}$ **in** *allDynamicParentSets* **do**
    for $\boldsymbol{Y}_{sp}$ **in** *allStaticParentSets* **do**
      currentScore $\leftarrow \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp}, D_{t+1-m}^{t+1} \cup S)$
      if *currentScore* $>$ *bestScore* **then**
        bestScore $\leftarrow$ *currentScore*
        dynamicParentsPast$_i \leftarrow \boldsymbol{X}_{dp}$
        staticParentsPast$_i \leftarrow \boldsymbol{Y}_{sp}$
  for $X_j[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
    $E_{ij} \leftarrow -$*bestScore*
for $X_i[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
  for $X_j[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
    bestScore $\leftarrow -\infty$
    for $\boldsymbol{X}_{dp}$ **in** *allDynamicParentSets* **do**
      for $\boldsymbol{Y}_{sp}$ **in** *allStaticParentSets* **do**
        currentScore $\leftarrow \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp} \cup X_j[t+1], D_{t+1-m}^{t+1} \cup S)$
        if *currentScore* $>$ *bestScore* **then**
          bestScore $\leftarrow$ *currentScore*
          dynamicParents$_{ij} \leftarrow \boldsymbol{X}_{dp}$
          staticParents$_{ij} \leftarrow \boldsymbol{Y}_{sp}$
    $E_{ij} \leftarrow E_{ij}$ + *bestScore*

Stationary version of the learning algorithm

In order to learn stationary sdtDBNs, only one transition network should be learned, using all timesteps. All aspects mentioned in this Section, namely Eqs. (5.1), (5.2) and (5.3), are still valid, as the same notions of optimality apply

and the inclusion of static attributes in the tDBN framework can be done in the same way. Therefore, the stationary version of the sdtDBN learning mechanism is given by Algorithms 5.1 and 5.2, with some small changes.

In Algorithm 5.1, the for loop of line 1 should not exist (it should be just one iteration, instead of a loop), as only one transition network is being computed. There is not the need of having line 6 either.

In line 3 of Algorithm 5.1, where Algorithm 5.2 is used, instead of determining the score for each possible parents' configuration using only the sub-dataset D_{t+1-m}^{t+1}, relative to timestep $t+1$, all occurrences in D should be analyzed, to compute a solution that represents the maximum score considering all timesteps. Therefore, in lines 7 and 19 of Algorithm 5.2, D should be used instead of D_{t+1-m}^{t+1}.

5.1.2 Correctness of the sdtDBN learning algorithm

As explained in Section 5.1.1, the computation of the s_i, s_{ij} and e_{ij} terms of the sdtDBN framework generalizes the computation of the same terms in the tDBN framework, so that static attributes also influence the values and parents obtained. However, after determining s_i, s_{ij} and e_{ij}, both algorithms follow the same steps. Therefore, the optimality of the sdtDBN learning algorithm is proven by the optimality of the tDBN algorithm. Another rationale for this statement is that static nodes can be seen as "special" nodes that only exist in the initial timestep of a DBN and can influence any node in any timestep, which allows considering sdtDBNs as tDBNs having in the initial timestep these "special" nodes. Theorems 5.1 and 5.2 show the correctness of the sdtDBN learning algorithm.

Theorem 5.1 (sdtDBN optimality per timestep)**.** *At each timestep $t+1$, Algorithm 5.1 finds an optimal tDBN structure also with static features in the framework (sdtDBN).*

Proof of Theorem 5.1 (by contradiction). Considering B^* as an optimal sdtDBN at timestep $t+1$, with an intra-slice structure I^*, as in Eqs. (5.1) and (5.2) all possible combinations of static and dynamic parents are analyzed and maximized, it comes directly from Eqs. (5.1) and (5.2) that the total score of B^* is given by

$$s_{r^*} + \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^*} s_{ij}, \tag{5.4}$$

where X_{r^*} is the root node of the intra-slice tree I^*.

Considering B^s as the output of Algorithm 5.1 for timestep $t+1$, with an intra-slice structure I^s having a root node X_{r^s}, if B^s is considered sub-optimal, then

$$s_{r^s} + \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^s} s_{ij} < s_{r^*} + \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^*} s_{ij}, \tag{5.5}$$

which is equivalent to

$$s_{r^s} + \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^s} s_{ij} - \sum_{i\in n} s_i < s_{r^*} + \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^*} s_{ij} - \sum_{i\in n} s_i. \tag{5.6}$$

Applying Eq. (5.3) to Eq. (5.6), it follows that

$$\sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^s} e_{ij} < \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^*} e_{ij}. \tag{5.7}$$

However, from the correctness of Edmonds' maximum branching algorithm used in line 4 of Algorithm 5.1, the sum of the edges in I^s at the left side of Eq. (5.7) must be maximum. Therefore, Eq. (5.7) stands false, which makes false the supposition that the output B^s of Algorithm 5.1 for timestep $t+1$ is sub-optimal, which proves that the output B^s of Algorithm 5.1 for timestep $t+1$ is optimal. □

Theorem 5.2 (sdtDBN global optimality)**.** *Algorithm 5.1 finds a globally optimal tDBN structure also with static features in the framework (sdtDBN).*

Proof. In the structure of the created sdtDBNs, each node in a timestep t can only be a parent of nodes in a certain timestep $t+k$, $k \geq 0$. Regarding static nodes, only static nodes can be parents of dynamic nodes, and not the other way around. Therefore, for a certain iteration of the for loop of Algorithm 5.1 regarding a timestep $t+1$, it is assured that the posterior iterations of the loop do not affect the parents found for nodes in timestep $t+1$. Hence, as Theorem 5.1 proves optimality for each timestep, Algorithm 5.1 is globally optimal. □

5.1.3 Complexity analysis of the sdtDBN learning algorithm

As the sdtDBN algorithm is a generalization of the tDBN algorithm to include static attributes, the complexity over the components related to the dynamic attributes remains equal. However, there is extra computational complexity, for the new components related to the static attributes, namely the n_{static} static attributes, the b maximum possible static parents of each node and the usage of the dataset S, with static observations. Theorem 5.3 provides the computational complexity of the sdtDBN learning algorithm.

Theorem 5.3. *(Computational complexity of the sdtDBN learning algorithm) The worst-case complexity of Algorithm 5.1 is: polynomial in the number of dynamic attributes n and in the number of static attributes n_{static}; exponential in the number of dynamic parents p and in the number of static parents b; linear in the number of dynamic observations $N_{dynamic}$ and in the number of static observations N_{static}.*

Analyzing Algorithm 5.1, the bottleneck is step 3, where Algorithm 5.2 is used. Regarding Algorithm 5.2, the bottleneck is the for loop starting at line 14, whose complexity is analyzed next.

Regarding the for loops of lines 14 and 15, each of them has complexity $\mathcal{O}(n)$, as they iterate over the n dynamic attributes of a timestep.

Regarding the for loops of lines 17 and 18, it is first needed to find an upper-bound on the number of possible dynamic parents sets from the m previous timesteps, $P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t])$, and on the number of possible static parents sets, $P_{\leq b}(\boldsymbol{Y})$. The total number of possible dynamic parents sets from the m previous timesteps, with a maximum of p parents per set and a Markov lag m, is given by

$$|P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t])| = \sum_{i=0}^{p} \binom{nm}{i} < \sum_{i=0}^{p} (nm)^i \in \mathcal{O}((nm)^p). \tag{5.8}$$

The total number of possible static parents sets, with a maximum of b parents per set, is given by

$$|P_{\leq b}(\boldsymbol{Y})| = \sum_{i=0}^{b} \binom{n_{static}}{i} < \sum_{i=0}^{b} n_{static}^i \in \mathcal{O}(n_{static}^b). \tag{5.9}$$

Given Eqs. (5.8) and (5.9), the computational complexities of the for loops of lines 17 and 18 are, respectively, $\mathcal{O}((nm)^p)$ and $\mathcal{O}(n_{static}^b)$.

Regarding the complexity of determining the value of ϕ_i at line 19 in each iteration of the loop, in the worst-case scenario, each node has $b + p + 1$ parents (b static, p dynamic from previous timesteps and 1 dynamic from the same timestep). Assuming each attribute can take at most λ different values, there are, in each conditional probability distribution, $\lambda^{b+p+1} \times \lambda = \lambda^{b+p+2}$ different configurations. Therefore, the conditional distributions have $\mathcal{O}(\lambda^{b+p+2})$ complexity. Every different configuration must be analyzed in each possible observation of the datasets needed to evaluate ϕ_i, which are D_{t+1-m}^{t+1} and S. The space needed to store the dynamic dataset is $|D_{t+1-m}^{t+1}| \times (m+1) \times n$, while the space needed to store the static dataset is $|S| \times n_{static}$. Therefore, there are needed a total of $\mathcal{O}(|D_{t+1-m}^{t+1}| \times (m+1) \times n \times |S| \times n_{static})$ comparisons. As each comparison should be made for each configuration of parents and proper child node, line 19 of Algorithm 5.2 has a complexity of $\mathcal{O}(|D_{t+1-m}^{t+1}| \times (m+1) \times n \times |S| \times n_{static} \times \lambda^{b+p+2})$.

Putting together all the aforementioned complexities of each component of the for loop starting at line 14 of Algorithm 5.2, the total complexity of Algorithm 5.2 is given by

$$\mathcal{O}(n) \times \mathcal{O}(n) \times \mathcal{O}((nm)^p) \times \mathcal{O}(n_{static}^b) \times \mathcal{O}(|D_{t+1-m}^{t+1}| \times (m+1) \times n \times |S| \times n_{static} \times \lambda^{b+p+2}), \quad (5.10)$$

which can be put succinctly as

$$\mathcal{O}(n^{p+3} \times n_{static}^{b+1} \times m^{p+1} \times \lambda^{b+p+2} \times |D_{t+1-m}^{t+1}| \times |S|). \quad (5.11)$$

Regarding the total complexity of Algorithm 5.1, it is needed to account for the fact that Algorithm 5.2, with complexity described in Eq. (5.11), is done $\mathcal{O}(T)$ times in the for loop of Algorithm 5.1, where T is the total number of timesteps. Therefore, defining $N_{dynamic} = \sum_{t=0}^{T-1} |D_{t+1-m}^{t+1}|$ as the total dimension of the dynamic dataset and $N_{static} = |S|$ as the total dimension of the static dataset, the total complexity of Algorithm 5.1 can be obtained by applying $N_{dynamic}$ and N_{static} to Eq. (5.11), being the total complexity of Algorithm 5.1 given by

$$\mathcal{O}(n^{p+3} \times n_{static}^{b+1} \times m^{p+1} \times \lambda^{b+p+2} \times N_{dynamic} \times N_{static}). \quad (5.12)$$

5.2 Learning sdtDBNs with restrictions in the network structure

When learning a DBN that represents the relations between data (see Section 5.1), there may be some prior knowledge regarding the relations the model should learn. As DBNs are graphical models that intuitively represent the relations between variables, introducing prior knowledge in DBNs can be stated as forcing some relations between nodes either to exist or not to exist.

Incorporating prior knowledge in statistical models is of particular importance when working with medical data, as the developed models are presented to doctors, who validate the relations found between the several variables, because medical experts, having real-life experience, can intuitively expect the model to reflect some relations and not to reflect some other.

5.2.1 Learning algorithm with restrictions

The addition of restrictions to the sdtDBN framework is done by applying the restrictions to the learning algorithm presented in Section 5.1 (see Algorithms 5.1 and 5.2). Therefore, the notation used in this Section and in Sections 5.2.2 and 5.2.3 is the same notation introduced in Section 5.1.1.

Regarding sdtDBNs, there must be considered **three types of relations between nodes: (i)** relations between dynamic nodes in different timesteps; **(ii)** relations between dynamic nodes in the same timestep; **(iii)** relations between dynamic nodes and static nodes. For each of the previous types of relations, the restrictions may force certain relations either to or not to exist.

The inclusion of restrictions in relations between dynamic nodes in different timesteps and between dynamic and static nodes can be done directly in Algorithm 5.2, where, for each node, the best set of parents from previous timesteps and the best set of static parents are found by always evaluating all combinations of parents, choosing the combinations with the best score for each node. The restrictions can be inserted by considering, for each node, only the combinations of parents from previous timesteps and combinations of static parents that each node can actually have, according to the restrictions, instead of always testing all combinations of parents. By adjusting the search-spaces, Algorithm 5.2 can remain almost unchanged, while also satisfying the desired restrictions.

The restrictions in relations between dynamic nodes in the same timestep can be included by biasing the proper weights e_{ij}, as relations of this type are included in the DBN at step 4 of Algorithm 5.1 by applying a maximum branching algorithm using the weights e_{ij}. As each weight e_{ij} refers to the relation $X_j[t+1] \rightarrow X_i[t+1]$, being $t+1$ the timestep of the iteration of Algorithm 5.1 being analyzed, the maximum branching algorithm inserts $X_j[t+1] \rightarrow X_i[t+1]$ in the DBN if $e_{ij} \rightarrow +\infty$ and does not insert $X_j[t+1] \rightarrow X_i[t+1]$ in the DBN if $e_{ij} \rightarrow -\infty$. Therefore, if $X_j[t+1] \rightarrow X_i[t+1]$ is mandatory, e_{ij} should be biased with a big positive value (several orders of magnitude higher than the values of the weights e_{ij}), whereas, if $X_j[t+1] \rightarrow X_i[t+1]$ is forbidden, e_{ij} should be biased with a big negative value. The bias in each e_{ij} should only be introduced after Algorithm 5.2 has done all calculations involving the respective e_{ij}, so that Algorithm 5.2 remains practically unchanged. The bias should also be done by adding (or subtracting) a big number, but not actually $+\infty$ (or $-\infty$), so that the biased relations remain distinguishable among themselves. In the algorithm's implementation, the big number used is $+10^{10}$ (or -10^{10}), as 10^{10} is larger than any weight obtained in a real-world situation, and, using 10^{10}, the program can distinguish two equally biased weights whose values differ at least 10^{-5}, which is sufficient in any real-world scenario. With this approach, if not all restrictions regarding intra-slice relations can be met (which may happen, because the intra-slice connectivity must be a tree, see Section 5.1), the maximum branching algorithm selects the relations that maximize the global score, while respecting the largest possible number of restrictions.

Given the previously described changes to include restrictions in the network, the sdtDBN learning procedure including restrictions in the network consists in using Algorithm 5.1, but changing line 3 of Algorithm 5.1 so that Algorithm 5.3 is used, instead of Algorithm 5.2.

5.2.2 Correctness of the learning algorithm with restrictions

As the learning algorithm of sdtDBNs with restrictions is an extension of the learning algorithm of the sdtDBNs presented in Section 5.1, the proof of correctness of Algorithm 5.1 using Algorithm 5.3 instead of Algorithm 5.2

Algorithm 5.3: Edge weights and optimal parents for m^{th}-order Markov non-stationary sdtDBNs with restrictions in the structure of the network.

Input :

- $t+1$, m, $\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t+1]$, $\boldsymbol{Y}$, p, b, D_{t+1-m}^{t+1}, S, ϕ_i(parentNodes, dataset): as defined in Algorithm 5.2.
- $MPT_{[n]}$: for each $X_i[t+1]$, the set of mandatory dynamic parents from the m previous timesteps.
- $FPT_{[n]}$: for each $X_i[t+1]$, the set of forbidden dynamic parents from the m previous timesteps.
- $MST_{[n]}$: for each $X_i[t+1]$, the set of mandatory dynamic parents from the same timestep.
- $FST_{[n]}$: for each $X_i[t+1]$, the set of forbidden dynamic parents from the same timestep.
- $MS_{[n]}$: for each $X_i[t+1]$, the set of mandatory static parents.
- $FS_{[n]}$: for each $X_i[t+1]$, the set of forbidden static parents.

Output :

- $E_{[n\times n]}$, *dynamicParentsPast*$_{[n]}$, *staticParentsPast*$_{[n]}$, *dynamicParents*$_{[n\times n]}$, *staticParents*$_{[n\times n]}$: as defined in Algorithm 5.2.

for $X_i[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
  allDynamicParentSets$_i \leftarrow$
    $\{\Omega \subseteq P_{\leq p}(\boldsymbol{X}[t+1-m] \cup \cdots \cup \boldsymbol{X}[t]) \mid \Omega \cap MPT_i = MPT_i \wedge \Omega \cap FPT_i = \emptyset\}$
  allStaticParentSets$_i \leftarrow \{\Omega \subseteq P_{\leq b}(\boldsymbol{Y}) \mid \Omega \cap MS_i = MS_i \wedge \Omega \cap FS_i = \emptyset\}$
bigNumber $\leftarrow$ *Positive number with a much higher order of magnitude than the values of the scores*
for $X_i[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
  bestScore $\leftarrow -\infty$
  for $\boldsymbol{X}_{dp}$ **in** *allDynamicParentSets*$_i$ **do**
    for $\boldsymbol{Y}_{sp}$ **in** *allStaticParentSets*$_i$ **do**
      currentScore $\leftarrow \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp}, D_{t+1-m}^{t+1} \cup S)$
      if *currentScore* $>$ *bestScore* **then**
        bestScore $\leftarrow$ *currentScore*
        dynamicParentsPast$_i \leftarrow \boldsymbol{X}_{dp}$
        staticParentsPast$_i \leftarrow \boldsymbol{Y}_{sp}$
  for $X_j[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
    $E_{ij} \leftarrow -$*bestScore*
for $X_i[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
  for $X_j[t+1]$ **in** $\boldsymbol{X}[t+1]$ **do**
    bestScore $\leftarrow -\infty$
    for $\boldsymbol{X}_{dp}$ **in** *allDynamicParentSets*$_i$ **do**
      for $\boldsymbol{Y}_{sp}$ **in** *allStaticParentSets*$_i$ **do**
        currentScore $\leftarrow \phi_i(\boldsymbol{X}_{dp} \cup \boldsymbol{Y}_{sp} \cup X_j[t+1], D_{t+1-m}^{t+1} \cup S)$
        if *currentScore* $>$ *bestScore* **then**
          bestScore $\leftarrow$ *currentScore*
          dynamicParents$_{ij} \leftarrow \boldsymbol{X}_{dp}$
          staticParents$_{ij} \leftarrow \boldsymbol{Y}_{sp}$
    if $X_j[t+1] \in MST_i$ **then**
      $E_{ij} \leftarrow E_{ij}$ + *bestScore* + *bigNumber*
    else if $X_j[t+1] \in FST_i$ **then**
      $E_{ij} \leftarrow E_{ij}$ + *bestScore* - *bigNumber*
    else
      $E_{ij} \leftarrow E_{ij}$ + *bestScore*

closely follows Theorems 5.1 and 5.2. It is presented in Theorems 5.4 and 5.5 the correctness of the sdtDBN learning algorithm including restrictions in the network.

Theorem 5.4 (Optimality per timestep of sdtDBNs with restrictions)**.** *At each timestep $t+1$, Algorithm 5.1, using Algorithm 5.3 instead of Algorithm 5.2, finds an optimal sdtDBN, given all desired restrictions.*

Proof of Theorem 5.4. The proof of Theorem 5.4 must be divided into **two parts: (i)** proving that all restrictions are represented in the learned sdtDBNs; **(ii)** proving that, given the restrictions, the sdtDBNs are optimal.

A node in timestep $t+1$ cannot possibly have dynamic parents from previous timesteps not following the restrictions, nor static parents that do not follow the restrictions, because Algorithm 5.3 evaluates, for each dynamic node in timestep $t+1$, only sets of dynamic parents from previous timesteps and sets of static parents that respect the restrictions imposed in the network. Regarding relations between nodes in timestep $t+1$, when comparing to relations that are neither mandatory nor forbidden, all mandatory relations are given a much higher weight and all forbidden relations are given a much lower weight. Therefore, it comes from the soundness of Edmonds' maximum branching algorithm (used in line 4 of Algorithm 5.1) that, in the relations among nodes in timestep $t+1$, all mandatory relations are introduced and none of the forbidden relations are introduced. The previous explanations show that all restrictions must always be respected by the learned sdtDBNs.

To prove the optimality of the learned sdtDBNs, a proof by contradiction can be used. As all possible combinations of static and dynamic parents respecting the restrictions are analyzed and maximized in Algorithm 5.3, then, considering M as the set with the mandatory relations between nodes in timestep $t+1$, and B^* as an optimal sdtDBN at timestep $t+1$, with an intra-slice structure I^*, the total score of B^* is given by

$$s_{r^*} + \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^*\setminus M} s_{ij} + \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in M} s_{ij}, \tag{5.13}$$

where X_{r^*} is the root node of the intra-slice tree I^*, and $I^* \cap M = M$.

In the optimal intra-slice structure I^*, all restrictions must be respected. It was also already shown that, in each timestep $t+1$, all restrictions are satisfied by the output of Algorithm 5.1 (using Algorithm 5.3 instead of Algorithm 5.2). Therefore, being B^s the output of Algorithm 5.1 (using Algorithm 5.3 instead of Algorithm 5.2) in timestep $t+1$, with an intra-slice structure I^s having a root node X_{r^s}, if B^s is considered sub-optimal, then, applying the same reasoning as in Eqs. (5.5) and (5.6) of Theorem 5.1, together with Eq. (5.13), it can be concluded that

$$\sum_{\substack{ij\,:\\ X_j[t+1]\rightarrow X_i[t+1]\in I^s\setminus M}} e_{ij} + \sum_{\substack{ij\,:\\ X_j[t+1]\rightarrow X_i[t+1]\in M}} e_{ij} < \sum_{\substack{ij\,:\\ X_j[t+1]\rightarrow X_i[t+1]\in I^*\setminus M}} e_{ij} + \sum_{\substack{ij\,:\\ X_j[t+1]\rightarrow X_i[t+1]\in M}} e_{ij}, \tag{5.14}$$

which can be written as

$$\sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^s\setminus M} e_{ij} < \sum_{ij\,:\,X_j[t+1]\rightarrow X_i[t+1]\in I^*\setminus M} e_{ij}, \tag{5.15}$$

where $I^s \cap M = M$ and $I^* \cap M = M$.

However, as the same edges are removed from I^s and from I^* when doing $I^s \setminus M$ and $I^* \setminus M$, respectively, the correctness of Edmonds' maximum branching algorithm, used in line 4 of Algorithm 5.1, allows concluding that Eq. (5.15) stands incorrect, which shows that B^s cannot be sub-optimal, thus proving the optimality, in each timestep $t+1$, of the sdtDBN learning algorithm, given the desired restrictions. □

Theorem 5.5 (Global optimality of sdtDBNs with restrictions)**.** *Algorithm 5.1, using Algorithm 5.3 instead of Algorithm 5.2, finds a globally optimal sdtDBN including all desired restrictions.*

Proof. The proof of Theorem 5.5 comes from showing that Theorem 5.2 remains valid when including restrictions. Theorem 5.2 is proven by demonstrating that the parents found for nodes in a timestep $t+1$ are not changed by posterior iterations of Algorithm 5.1. Both Algorithms 5.2 and 5.3 provide, as output, the optimal sets of parents of nodes in a certain timestep $t+1$, being the only difference that Algorithm 5.3 includes restrictions in the network. Therefore, when applying Algorithm 5.1 using Algorithm 5.3 instead of Algorithm 5.2, the parents of nodes in a timestep $t+1$ remain unaffected by posterior iterations of Algorithm 5.1. Hence, as Theorem 5.4 already states the optimality per timestep, Theorem 5.2 remains valid in the learning procedure with restrictions, thus proving the global optimality of Algorithm 5.1 when using Algorithm 5.3 instead of Algorithm 5.2. □

5.2.3 Complexity analysis of the learning algorithm with restrictions

Regarding the complexity analysis of the learning algorithm with restrictions, Theorem 5.3 still holds, because the worst-case complexity scenario consists of a network without restrictions, in which case the learning algorithms with and without restrictions have the same computational complexity.

To assess the influence of the restrictions in the complexity of Algorithm 5.3, as Algorithm 5.3 evaluates all sets of parents of nodes in timestep $t+1$ that respect the restrictions, it can be defined Z_{prev} as the maximum number of possible (respecting the restrictions) sets of dynamic parents from previous timesteps of the nodes in timestep $t+1$, and Z_{static} as the maximum number of possible (respecting the restrictions) sets of static parents of the nodes in timestep $t+1$. Relating Z_{prev} and Z_{static} with the given pseudocode of Algorithm 5.3, $Z_{prev} = max_i\{|\textit{allDynamicParentSets}_i|\}$, whereas $Z_{static} = max_i\{|\textit{allStaticParentSets}_i|\}$.

The remaining elements of the computational complexity of Algorithm 5.3 remain equal to the ones described in Section 5.1.3. In particular, the restrictions in the intra-slice connectivity do not reduce the complexity, as those restrictions are introduced by biasing the weights e_{ij}, being all combinations of intra-slice relations analyzed in Algorithm 5.3 (as happens in Algorithm 5.2). Therefore, if, in the analysis of Section 5.1.3, the complexities of Eqs. (5.8) and (5.9) are replaced by, respectively, Z_{prev} and Z_{static}, Eq. (5.10) can be changed to

$$\mathcal{O}(n) \times \mathcal{O}(n) \times \mathcal{O}(Z_{prev}) \times \mathcal{O}(Z_{static}) \times \mathcal{O}(|D_{t+1-m}^{t+1}| \times (m+1) \times n \times |S| \times n_{static} \times \lambda^{b+p+2}), \quad (5.16)$$

which reflects the computational complexity of Algorithm 5.3 and can be succinctly written as

$$\mathcal{O}(n^3 \times n_{static} \times Z_{prev} \times Z_{static} \times m \times \lambda^{b+p+2} \times |D_{t+1-m}^{t+1}| \times |S|). \quad (5.17)$$

To get the computational complexity of Algorithm 5.1 using Algorithm 5.3 instead of Algorithm 5.2, the same reasoning explained in the transition from Eq. (5.11) to Eq. (5.12), in Section 5.1.3, can be applied. Defining $N_{dynamic}$ and N_{static} as in Section 5.1.3, and defining $\overline{Z}_{prev}$ and $\overline{Z}_{static}$ as the upper-bounds on the values that Z_{prev} and Z_{static} may take considering all timesteps of the for loop of Algorithm 5.1, the complexity of Algorithm 5.1, using Algorithm 5.3 instead of Algorithm 5.2, can be described by

$$\mathcal{O}(n^3 \times n_{static} \times \overline{Z}_{prev} \times \overline{Z}_{static} \times m \times \lambda^{b+p+2} \times N_{dynamic} \times N_{static}). \quad (5.18)$$

As stated in the beginning of this Section, the worst-case complexity of the learning algorithm with restrictions and the learning algorithm without restrictions is the same. If there are no restrictions in the network, the sets of parents evaluated in Algorithm 5.3 are the same as the ones tested in Algorithm 5.2. In this situation, $\mathcal{O}(\overline{Z}_{prev}) = \mathcal{O}((nm)^p)$ and $\mathcal{O}(\overline{Z}_{static}) = \mathcal{O}(n^b_{static})$, which makes Eq. (5.18) the same as Eq. (5.12).

5.3 Inference in learned sdtDBNs

After generalizing tDBNs to sdtDBNs by introducing static attributes in the tDBN framework (as explained in Section 5.1) and incorporating restrictions in the networks when learning the sdtDBNs (as presented in Section 5.2), another main concern of the Thesis is to allow a user to make inference in a learned sdtDBN, so that a user can employ the predictive mechanisms of the created sdtDBNs. Inference mechanisms are extremely useful in contexts, such as this Thesis, of studying a disease progression, as they allow a user to predict the values of certain medical indicators, based on a developed statistical model.

5.3.1 Inference algorithm

When making inference, the goal is to, given certain observations and the probability distributions of the learned sdtDBN, estimate the probability distribution of a certain attribute in a certain timestep. Naturally, in this scenario, the implicit independence assumptions of BNs and DBNs are used, in order for inference to be a feasible problem and for the sdtDBN structure to actually be useful.

Notation and assumptions

To perform inference, an sdtDBN must be learned. Therefore, it is assumed that an sdtDBN was previously learned from user data (see Sections 5.1 and 5.2). It is also assumed that there is a specific node, $X_i[t]$, whose distribution should be found (at the end of the Section, there is presented a generalization for when it is desired to learn the distributions of multiple nodes). Finally, to perform inference, there are needed observations to address the several CPTs. The static observations are denoted as *staticObs*, while the dynamic observations are represented as *dynObs*. It is assumed that only a single subject is analyzed, which means that *staticObs* and *dynObs* only have a maximum of one measurement per node. At the end of the Section, there is presented a generalization for situations where there are given observations for several subjects, each one having its observations for each node.

Algorithm

According to the conditional independence assumptions of the DBN framework, there is only needed to know the values of the parents of $X_i[t]$ in the sdtDBN to determine the distribution of $X_i[t]$. Therefore, the first step of the inference algorithm is to, from the sdtDBN structure, get the static and dynamic parents of $X_i[t]$. Given these parents, there are three possible scenarios regarding the provided observations:

1. **All static parents are given in *staticObs* and all dynamic parents are given in *dynObs*.** In this situation, all that is needed is to get the CPT of $X_i[t]$ and address it with the values of the parents in *staticObs* and *dynObs*, to get the proper distribution of $X_i[t]$.

2. **Some static parents are not given in *staticObs*.** In this situation, inference is not possible, because, in the sdtDBN framework, static nodes do not have parents, so, an sdtDBN cannot provide CPTs with conditional distributions of static nodes, thus not being possible to estimate the values of static nodes.

3. **All static parents are given in *staticObs*, but some dynamic parents are not given in *dynObs*.** In this situation, the values of the parents not given in *dynObs* should be estimated using the sdtDBN structure. If the estimation of at least one of the unknown parents is not possible, inference is not possible as in scenario 2. If all parents can be estimated, the estimated values should be written in *dynObs*, leading to scenario 1.

The estimation of the value of a certain node $X_j[t]$ using a provided sdtDBN and given observations is done using a depth-first search (DFS) approach. For a node $X_j[t]$, all values of its parents should be checked in *staticObs* and *dynObs*. If some dynamic parent $X_k[t]$ does not have a value in *dynObs*, the parents of $X_k[t]$ should be checked to estimate $X_k[t]$, recursively leading to a DFS approach. Therefore, when a certain node does not have an observation, its parents are recursively checked, until there is a node for which all parents have values in *staticObs* and *dynObs*, making it possible to estimate that node's value. If, at an iteration of this process, some attribute has static parents with unknown values, or the process would need to estimate the values of a node in one of the initial m timesteps of the sdtDBN (for which there are no CPTs, because the sdtDBN is assumed to be an m^{th}-order Markov process), the algorithm concludes that there is no possible way of estimating the value of $X_j[t]$, as at least one needed node cannot be estimated.

Algorithm 5.4 presents an overview of the inference algorithm described, using Algorithm 5.5 to estimate the values of the parents of node $X_i[t]$ whose values are not provided in the given observations. Algorithm 5.5 implements the DFS approach previously explained using a stack data structure to store the nodes whose values must be estimated. In a stack, *PUSH* means inserting a node at the top of the stack, *POP* means removing the node from the top of the stack and *PEEK* means only checking which node is at the top of the stack, without removing it.

Inference given static and dynamic observations of several subjects

The previously given explanations of the inference algorithm are provided considering that inference is desired, for a node $X_i[t]$, given static and dynamic observations of a single subject/entity. In a more general situation, it may be desired to make inference for several subjects, each having its own static and dynamic observations. In that circumstance, instead of having *staticObs* and *dynObs* with a single dimension (for each node of the sdtDBN, there is only given an observation of a single subject), *staticObs* and *dynObs* have M dimensions, being M the total number of subjects for which inference in node $X_i[t]$ should be made. The generalization of the inference algorithm consists in applying Algorithm 5.4 for a total of M times, creating a loop where, in each iteration, Algorithm 5.4 is applied using the *staticObs* and *dynObs* of a different subject, in order to cover all M subjects.

Inference for several nodes

The inference algorithm is presented assuming that it is only desired to make inference for a single node $X_i[t]$. In a more general context, it can be useful to make inference for each node among a certain set of nodes $\boldsymbol{X}_{inf}$. Assuming that $\boldsymbol{X}_{inf}$ has a cardinality of L nodes, the generalization of the inference algorithm is done by applying Algorithm 5.4 for a total of L times, creating a loop where, in each iteration, Algorithm 5.4 is applied using a different node $X_i[t] \in \boldsymbol{X}_{inf}$, in order to make inference for all nodes of $\boldsymbol{X}_{inf}$.

Algorithm 5.4: Distribution of node $X_i[t]$ according to an sdtDBN structure and given observations.

Input :
- sdtDBN: sdtDBN structure and parameters learned from data (see Sections 5.1 and 5.2).
- $X_i[t]$: sdtDBN node whose distribution should be estimated.
- *staticObs*: static observations of the subject in which inference is being made.
- *dynObs*: dynamic observations of the subject in which inference is being made.

Output :
- *distribution*: estimated distribution of node $X_i[t]$, according to the structure of the given sdtDBN and the observations from *staticObs* and *dynObs*.

```
cpt ← get CPT of node X_i[t] from sdtDBN structure
dynParents ← get dynamic parents of node X_i[t] from sdtDBN structure
staticParents ← get static parents of node X_i[t] from sdtDBN structure
for sParent in staticParents do
    if sParent not in staticObs then
        end inference as it is not possible
    else
        staticConfig(sParent) ← staticObs(sParent)
for dParent in dynParents do
    if dParent not in dynObs then
        use Algorithm 5.5 to try to determine the value of dynObs(dParent) according to sdtDBN structure
    if dParent not in dynObs then
        end inference as it is not possible
    else
        dynConfig(dParent) ← dynObs(dParent)
distribution ← cpt(staticConfig, dynConfig)
```

Prediction of the trajectory of all nodes until a certain timestep

The generalization of the algorithm to make inference for every node $X_i[t] \in \boldsymbol{X}_{inf}$ can be put together with the generalization to make inference given observations of M subjects. In that case, it is necessary a nested loop, where the outer loop evaluates each of the L nodes of $\boldsymbol{X}_{inf}$, while the inner loop evaluates each *staticObs* and *dynObs* of the several M subjects.

If the goal is to predict the trajectory of all nodes X_i until a certain timestep T', it is necessary to add another outer loop to the previously described nested loop, so that this added outer loop selects, in each iteration, $\boldsymbol{X}_{inf}$ as the subset of all nodes of a certain timestep whose value is not yet estimated, starting at timestep T' and ending at the timestep corresponding to the Markov lag, because only for timesteps bigger or equal than the Markov lag there are transition networks (and corresponding conditional distributions) determined by the sdtDBN framework. The procedure is illustrated in Algorithm 5.6.

5.3.2 Complexity analysis of the inference algorithm

Theorem 5.6 presents the computational complexity of the inference algorithm.

Theorem 5.6 (Computational complexity of inference in sdtDBNs)**.** *When making inference for a node $X_i[t]$ of an sdtDBN with a maximum of p dynamic parents from previous timesteps and b static parents per node, Algorithm 5.4 has a worst-case complexity which is linear in b, polynomial in p and exponential in the timestep t.*

Algorithm 5.5: Estimation of the value of $X_j[t]$ according to an sdtDBN structure and given observations.

Input :

- sdtDBN: sdtDBN structure and parameters learned from data (see Sections 5.1 and 5.2).
- $X_j[t]$: sdtDBN node whose value should be estimated.
- *staticObs*: static observations of the subject in which inference is being made.
- *dynObs*: dynamic observations of the subject in which inference is being made.

Output :

- Modified *dynObs*: if possible, the value of $X_j[t]$ is estimated and written in *dynObs* and every value needed to estimate $X_j[t]$ is also estimated and written in *dynObs*.

```
stack ← initialize an empty stack to store the sdtDBN nodes whose values must be estimated
stack.PUSH(X_j[t])
while stack is not empty do
    currNode ← stack.PEEK(top of stack)
    if currNode does not have parents in the sdtDBN then
        end algorithm because inference is not possible
    if currNode already has value determined in dynObs then
        stack.POP()
        continue
    currDynParents ← get dynamic parents of node currNode from sdtDBN structure
    currStaticParents ← get static parents of node currNode from sdtDBN structure
    for sParent in currStaticParents do
        if sParent not in staticObs then
            end algorithm because inference is not possible
    allParentsSpecified = true
    for dParent in currDynParents do
        if dParent not in dynObs then
            stack.PUSH(dParent)
            allParentsSpecified = false
    if allParentsSpecified == false then
        continue
    stack.POP()
    for sParent in currStaticParents do
        staticConfig(sParent) ← staticObs(sParent)
    for dParent in currDynParents do
        dynConfig(dParent) ← dynObs(dParent)
    cpt ← get CPT of node currNode from sdtDBN structure
    distribution ← cpt(staticConfig, dynConfig)
    dynObs(currNode) ← randomly estimate a value for currNode using the probabilities of distribution
```

Proof of Theorem 5.6. In the worst-case scenario, every node of the sdtDBN has p dynamic parents from the timestep immediately before, b static parents and one parent from the same timestep (except for the root node of each timestep), and only the dynamic nodes in the first m timesteps have observations, being m the Markov lag of the sdtDBN (these nodes must have observations for inference to be assured). All static attributes must also have the proper observations, for inference to always be possible.

Starting with node $X_i[t]$ in Algorithm 5.4, it has p dynamic parents from the previous timestep that need to be estimated. When calling Algorithm 5.5 for a parent of $X_i[t]$, that parent will have p dynamic parents from the previous timestep to be estimated, each one also having p parents from the previous timestep to be estimated. This situation is repeated until reaching timestep m, where all nodes have all their parents with observations. Therefore,

Algorithm 5.6: Prediction of the trajectory of all nodes until a certain timestep of an sdtDBN.

Input :

- sdtDBN: sdtDBN structure and parameters learned from data (see Sections 5.1 and 5.2), with Markov lag m.
- T': the timestep until which all nodes' values should be estimated.
- *staticObs*$_{[M]}$: static observations of the M subjects in which inference is being made.
- *dynObs*$_{[M]}$: dynamic observations of the M subjects in which inference is being made.

Output :

- Modified *dynObs*: all nodes' values are estimated and the estimations are written in *dynObs*.

```
currTimestep ← T'
while currTimestep ≥ m do
    X_inf ← all X_i[currTimestep] that do not have a value specified in at least one subject of dynObs
    for currNode in X_inf do
        for dObs in dynObs do
            if dObs(currNode) does not have a value specified then
                sObs ← observations of staticObs corresponding to the same subject of dObs
                distribution ← call Algorithm 5.4 with (sdtDBN, currNode, sObs, dObs)
                dObs(currNode) ← estimate dObs(currNode) using the probabilities of distribution
    currTimestep ← currTimestep −1;
```

a maximum of $\sum_{k=0}^{t-1-m} p^k$ nodes can be put in the stack of Algorithm 5.5. In this counting, the fact that nodes have one parent from the same timestep can be discarded from the analysis, considering that, for every node, its parents from previous timesteps are given in a topological ordering, as, in that case, the parent of each node from the same timestep is certainly already determined, according to Algorithms 5.4 and 5.5.

As Algorithm 5.5 is called p times by Algorithm 5.4, the number of parents put in all stacks created by calling Algorithm 5.5 a total of p times has complexity

$$\mathcal{O}(p) \times \mathcal{O}\left(\sum_{k=0}^{t-1-m} p^k\right) = \mathcal{O}(p) \times \mathcal{O}\left(\frac{1-p^{t-m}}{1-p}\right) \approx \mathcal{O}(p^{t-m}). \tag{5.19}$$

For each node analyzed in Eq. (5.19), all p dynamic parents, b static parents and the dynamic parent from the same timestep should be checked to address the proper CPT, leading the overall complexity of Algorithm 5.4 to be

$$\mathcal{O}(p^{t-m}) \times \mathcal{O}(p+b+1) \approx \mathcal{O}(p^{t-m+1} + bp^{t-m}). \tag{5.20}$$

□

Although it is only presented Theorem 5.6 for the computational complexity of Algorithm 5.4, the complexity expressed by Eq. (5.20) can be extended to the generalizations provided in Section 5.3.1. These extensions are straightforward, as the generalizations consist in applying Algorithm 5.4 in loops. Therefore, the complexity of the generalization of Algorithm 5.4 to the situation where there are given observations of M subjects is

$$\mathcal{O}(M) \times \mathcal{O}(p^{t-m+1} + bp^{t-m}) = \mathcal{O}(Mp^{t-m+1} + Mbp^{t-m}), \tag{5.21}$$

while the complexity of the generalization where there are L nodes in which inference should be made is

$$\mathcal{O}(L) \times \mathcal{O}(p^{t-m+1} + bp^{t-m}) = \mathcal{O}(Lp^{t-m+1} + Lbp^{t-m}). \tag{5.22}$$

Regarding the prediction of the trajectory of all nodes until a certain timestep T', the computational complexity can be determined from Algorithm 5.6 using the complexity of Algorithm 5.4, given by Eq. (5.20), leading to an overall complexity represented as

$$\mathcal{O}(T' - m + 1) \times \mathcal{O}(n) \times \mathcal{O}(M) \times \mathcal{O}(p^{T'-m+1} + bp^{T'-m}), \tag{5.23}$$

which can also be written as

$$\mathcal{O}((T' - m + 1)nMp^{T'-m+1} + (T' - m + 1)nMbp^{T'-m}), \tag{5.24}$$

where n is the number of attributes per timestep of the sdtDBN and the remaining parameters are as already previously defined in this Section.

5.4 Available implementations

The sdtDBN framework, presented in Section 5.1.1, the inclusion of restrictions in the networks, explained in Section 5.2.1 and the inference capabilities, detailed in Section 5.3.1, are implemented in Java, being the source code available at `https://github.com/ttlion/sdtDBN_code`. The provided implementation is an extension of the tDBN framework, with several classes and methods redefined and new classes and methods created to introduce static attributes in the sdtDBNs. It allows learning DBNs with static and dynamic attributes (sdtDBNs) or only with dynamic attributes (tDBNs), inserting restrictions when learning the networks if desired, and performing inference for several nodes, given observations of multiple subjects.

The sdtDBN program is released under the Apache License 2.0. It is possible to obtain the latest executable version of the program (provided as a JAR file) at `https://ttlion.github.io/sdtDBN`, where there are also given examples detailing how to insert the inputs of the program and interpret the several outputs.

Chapter 6

Intuitive graphical interface

6.1 Importance of a graphical interface in this Thesis' context

As explained in Chapter 1, this Thesis concerns the employment of data mining techniques to study medical data (in particular, ALS data, see Chapters 1 and 2). Therefore, after developing the sdtDBNs (see Chapter 5), there is the need of making the usage of the sdtDBNs available, in an intuitive manner, to general users that, like most clinicians, may not be computer science or data mining experts.

Providing doctors with a tool for them to use the sdtDBNs themselves is of extreme importance, as it gives doctors the possibility of having a statistical indicator to help them make decisions regarding current patients, instead of only performing posterior analyses with the help of data mining experts.

One of the most intuitive ways of providing a user with the ability to manage a program is through a graphical interface, so that the end user does not need to learn any programming language, nor to work with the command line, to use the created program. This Chapter details the developed graphical user interface (GUI), for a user to be able to exploit all capabilities of sdtDBNs only through graphical interactions.

6.2 Overview of the developed GUI

The developed GUI is composed by seven tabs, which give the user the ability to employ the several potentialities of the sdtDBN framework, presented in Chapter 5. Fig. 6.1 shows the graphical display of the first tab of the GUI.

Throughout the explanations of this Chapter, the tabs of the GUI are identified by their position in the selection menu at the top of the GUI, being this selection menu always displayed as presented at the top of Fig. 6.1. The tab **Learn DBN from data** is denoted as the **first tab**, the tab **Predictions for many IDs** is identified as the **seventh tab** and the intermediate tabs are represented accordingly.

All tabs of the GUI have a display similar to Fig. 6.1, having each tab its specific options, to perform the proper actions. The general workflow of the GUI is as follows:

1. The user must employ either the first tab or the second tab to learn an sdtDBN. The first tab learns an sdtDBN from observations, whereas the second tab retrieves an already learned sdtDBN, stored in a file. The sdtDBN learned in one of these tabs is the one used in the remaining tabs of the GUI.

Figure 6.1: Screenshot of the first tab of the GUI.

2. The third tab allows a user to get the graphical representation of a learned sdtDBN.

3. In the fourth tab, the user must introduce the observations that should be used for making inference, which can then be made in the fifth tab, the sixth tab or the seventh tab, in the following way:

   (a) The fifth tab allows a user to estimate the probability distribution of a specific attribute in a particular timestep, given the data of a certain subject.

   (b) In the sixth tab, a user can predict the progression of either all attributes or a particular attribute until a defined timestep, given the data of a certain subject.

   (c) Predictions for several subjects can be made using the seventh tab.

Section 6.3 presents a detailed explanation of each tab's functioning, relating each tab with the proper Section of Chapter 5.

6.3 Details on each tab of the GUI

In this Section it is explained the functioning of each tab of the GUI. As the general display of the GUI is already given in Fig. 6.1, there is only provided a textual overview regarding each tab's functioning. The reader may check `https://ttlion.github.io/sdtDBNsGUI` for examples that show the display and usage of the several tabs of the GUI, together with the format of the multiple input files.

First tab: learning an sdtDBN from observations

The first tab (see Fig. 6.1) learns an sdtDBN from input observations, using the learning algorithm of the sdtDBNs, explained in Section 5.1.1, and also providing the user with the capability of inserting restrictions in the learned network, as presented in Section 5.2.1. Therefore, this tab uses Algorithm 5.1, which also employs either Algorithm 5.2 (if no restrictions are given) or Algorithm 5.3 (if there are given restrictions). Table 6.1 illustrates how each input of Algorithms 5.1, 5.2 and 5.3 is extracted from the user's input in the first tab.

Table 6.1: Relation between the inputs of Algorithms 5.1, 5.2 and 5.3 and the inputs of the first tab of the GUI.

Input of Algorithms 5.1, 5.2 or 5.3	**User's input in the first tab of the GUI**
D	File inserted by the user in the GUI, having the dataset with dynamic observations
$\boldsymbol{X}$	Attributes retrieved from the input dataset with dynamic observations
T	Maximum timestep of the input dataset with dynamic observations
S	File inserted by the user in the GUI, having the dataset with static observations
$\boldsymbol{Y}$	Attributes retrieved from the input dataset with static observations
$MPT, FPT, MST, FST, MS, FS$	Each type of restriction has its respective input file in the GUI, for the user to select to which nodes of the sdtDBN each type of restriction should be applied
ϕ	Directly selected by the user in the GUI
m	
p	
b	

If the user chooses to only introduce dynamic observations, the GUI will learn a tDBN [25, 69], only with dynamic attributes. If the user does not introduce any restriction in the network, Algorithm 5.2 is used in each iteration of Algorithm 5.1, otherwise, it is Algorithm 5.3 the one used in each iteration of Algorithm 5.1 (see Sections 5.1 and 5.2 for more details). The first tab also allows the user to select whether to learn a stationary or a non-stationary sdtDBN (see the stationary version of the learning algorithm, presented in Section 5.1.1). The learned sdtDBN is stored in a file for future use and is provided to the remaining tabs of the GUI.

Second tab: retrieving an sdtDBN stored in a file

As stated in the first tab's explanation, a learned sdtDBN is stored in a file. In this second tab, the user can select a file with a stored sdtDBN, so that it can be used in the remaining tabs of the GUI.

Third tab: getting the graphical representation of an sdtDBN

The third tab simply provides the graphical representation of the sdtDBN learned in the first tab or the second tab.

Fourth tab: inserting the observations that will be used for making inference in an sdtDBN

In the fourth tab, the user should introduce **two input files: (i)** the first file should be a dataset with dynamic observations, for several subjects; **(ii)** the second file should be a dataset with static observations, for the same subjects for which there are dynamic observations in the first file.

The dynamic and static observations should be given for the attributes of the sdtDBN learned in either the first tab or the second tab. The introduced datasets will be used for making inference in the fifth tab, the sixth tab and the seventh tab, in particular to get the *staticObs* and *dynObs* of Algorithms 5.4, 5.5 and 5.6, from Section 5.3.1.

Fifth tab: estimating the distribution of a selected attribute in a certain timestep, for a defined subject

The fifth tab uses the ability presented in Algorithms 5.4 and 5.5 of Section 5.3.1, which allows estimating the distribution of an attribute of an sdtDBN in a certain timestep, given the observations of a subject. Table 6.2 illustrates how each input of Algorithms 5.4 and 5.5 is extracted from the user's input in the fifth tab.

Table 6.2: Relation between the inputs of Algorithms 5.4 and 5.5 and the inputs of the fifth tab of the GUI.

Input of Algorithms 5.4 or 5.5	**User's input in the fifth tab of the GUI**
sdtDBN	File with the sdtDBN already learned in either the first tab or the second tab
$X_i[t]$	X_i and t are both directly selected by the user in the GUI
staticObs	User selects the desired subject in the GUI. The program automatically gets the proper *staticObs* and *dynObs*, from the observations inserted in the fourth tab
dynObs	

The fifth tab only allows a user to select subjects whose observations are in the files inserted in the fourth tab. To change some subject's observations or add observations for a new subject, the user should submit new observation files in the fourth tab. This approach allows the fifth tab to be intuitive, presenting to the user all subjects in which inference can be made, instead of forcing the user to remember exactly how each subject is identified.

For every subject a user can choose in the fifth tab, there may be estimated the distributions of several attributes in different timesteps, by using multiple times the fifth tab. As the learned sdtDBN is stored in a file, the computational complexity in this situation is the complexity of applying Algorithm 5.4 several times, as the sdtDBN is only learned once. To change the sdtDBN being used, the user must return to either the first tab or the second tab.

Sixth tab: predicting the progression of one or all attributes, for a defined subject

The sixth tab uses the generalization of Algorithm 5.4 for several nodes, which consists in applying Algorithm 5.4 multiple times, for each desired node (see Section 5.3.1). According to the user's input, there are selected the proper nodes needed to estimate the desired progression, given a certain subject's observations. Regarding the possible subjects and the sdtDBN used in this tab, it is valid the same reasoning presented in the explanation of the fifth tab.

The user should specify a maximum timestep T and a dynamic attribute X_j of the sdtDBN. Then, Algorithm 5.4 is applied for all timesteps until T. The inputs of Algorithms 5.4 and 5.5 are obtained from the GUI as described in Table 6.2, being the only difference in the $X_i[t]$ input, which in the sixth tab is always the X_j attribute previously mentioned, with the timestep t varying from 0 to T (in the sixth tab, the user selects the X_j and T, instead of selecting the X_i and t presented in Table 6.2). The user can also choose to determine the progression of all attributes until a specified timestep T. In that case, the previous process is done for all dynamic attributes of the sdtDBN, instead of just for a particular attribute X_j.

In both situations previously described, the user can select if the GUI should provide an estimation of the probability distribution or an estimation of the value of each node. When estimating the values of the several nodes, the user can choose whether to always present the most probable value given a node's distribution or to randomly determine each node's value using its probability distribution.

Seventh tab: making predictions for several subjects

The seventh tab allows a user to apply the generalizations, for multiple subjects, of the inference algorithm presented in Section 5.3.1. The seventh tab is composed by the *attribute inference* and the *progression until timestep* modes.

In the *attribute inference* mode, the user can exploit the generalization of Algorithm 5.4 for several nodes and several subjects (presented in Section 5.3.1). This is done by using a similar reasoning for the tab's inputs as the one expressed in Table 6.2. The **two differences** in the seventh tab are the following: **(i)** instead of specifying a node $X_i[t]$ as in Table 6.2, the user should give, in this seventh tab, a file specifying the multiple nodes in which inference must be made; **(ii)** regarding the *staticObs* and *dynObs*, the user does not select a specific subject as presented in Table 6.2, being inference made for all subjects in the observation files inserted in the fourth tab.

In the *progression until timestep* mode, the user can employ the generalization given by Algorithm 5.6 regarding the inference procedure (see Section 5.3.1). Table 6.3 illustrates how each input of Algorithm 5.6 is extracted from the user's input in the seventh tab.

Table 6.3: Relation between the inputs of Algorithm 5.6 and the inputs of the seventh tab of the GUI.

Input of Algorithm 5.6	**User's input in the seventh tab of the GUI**
sdtDBN	File with the sdtDBN already learned in either the first tab or the second tab
T'	Directly selected by the user in the GUI
$staticObs_{[M]}$	Files with dynamic and static observations introduced in the fourth tab. The value of M is inferred by the number of subjects in these files
$dynObs_{[M]}$	

6.4 Available implementations

The developed GUI, presented in Sections 6.2 and 6.3, is implemented in Python, being the source code available at `https://github.com/ttlion/sdtDBNsGUI_code`. The implementation of the GUI is done using Python's `tkinter` package to generate the graphical display.

The latest version of the sdtDBNs GUI is available at `https://ttlion.github.io/sdtDBNsGUI`, where there are provided examples explaining the inputs and outputs of each tab, together with the proper display of the GUI in each situation. There are also available, in the aforementioned website, executable standalone versions of the GUI, for Windows and for Linux, so that a user does not need to install Python to use the GUI (most users may not be computer science experts). The standalone versions of the GUI were created using `PyInstaller`.

As the GUI uses the sdtDBN framework presented in Chapter 5, the executable version of the sdtDBNs (see Section 5.4) is always provided with all distributions of the GUI, for the GUI to work properly.

Chapter 7

Assessment of Portuguese ALS Dataset

7.1 Data preprocessing

As already stated in Chapter 2, the sdtDBNs created to study ALS disease progression are obtained using data from a pretreated version of the March 2020 update of the Portuguese ALS dataset. For simplicity, this pretreated version is referred as the "ALS dataset" throughout the whole Chapter 7. Although already pretreated, the dataset used still needed some preprocessing, to be in the proper format for learning the sdtDBNs. The preprocessing done to the dataset is explained in this Section. The static and dynamic features of the ALS dataset must be preprocessed in a distinct way, because the dynamic data has the additional temporal component and should be separated into several sub-datasets, to learn sdtDBNs in interesting contexts.

The preprocessing of the static variables of the ALS dataset consists in selecting the proper features and discretizing them. The selection is done by eliminating the features with high percentage of missing values. The discretization is done by grouping the values of each feature into several classes, proposed by clinical experts. The static features selected from the ALS dataset and their discretization are presented in Table 7.1.

Table 7.1: Selection and discretization of the static features of the dataset. The missing values are measured over the 1214 patients of the dataset.

Feature	**Missing values (%)**	**Selected?**	**Discretization (displayed as *label: respective elements*)**
Gender	0,00	Yes	1: male; 2: female
BMI	23,06	Yes	1: [0,20[; 2: [20,25[; 3: [25,30[; 4: [30,+∞[
Familiar history MND	7,91	Yes	1: yes; 2: no; 3: unknown
Age at onset (years)	0,25	Yes	1: [0,30[; 2: [30,50[; 3: [50,70[; 4: [70,+∞[
Disease duration (months)	0,99	Yes	1: [0,6]; 2:]6,12]; 3:]12,18]; 4:]18,36]; 5:]36,+∞[
El Escorial reviewed criteria	11,45	Yes	1: definitive; 2: probable; 3: possible; 4: progressive muscular atrophy
UMN vs LMN	52,55	No	Variable not selected due to the high number of missing values
Onset form	0,08	Yes	1: spinal; 2: bulbar; 3: respiratory/axial; 4: mixed; 5: frontotemporal degeneration
C9orf72	0,00	Yes	1: yes; 2: no; 3: unknown

The preprocessing of the dynamic variables of the ALS dataset has multiple steps, to generate several sub-datasets, useful for learning sdtDBNs in different scenarios. An overview of the preprocessing steps applied to the dynamic variables is presented in Fig. 7.1, being each step described in the following paragraphs.

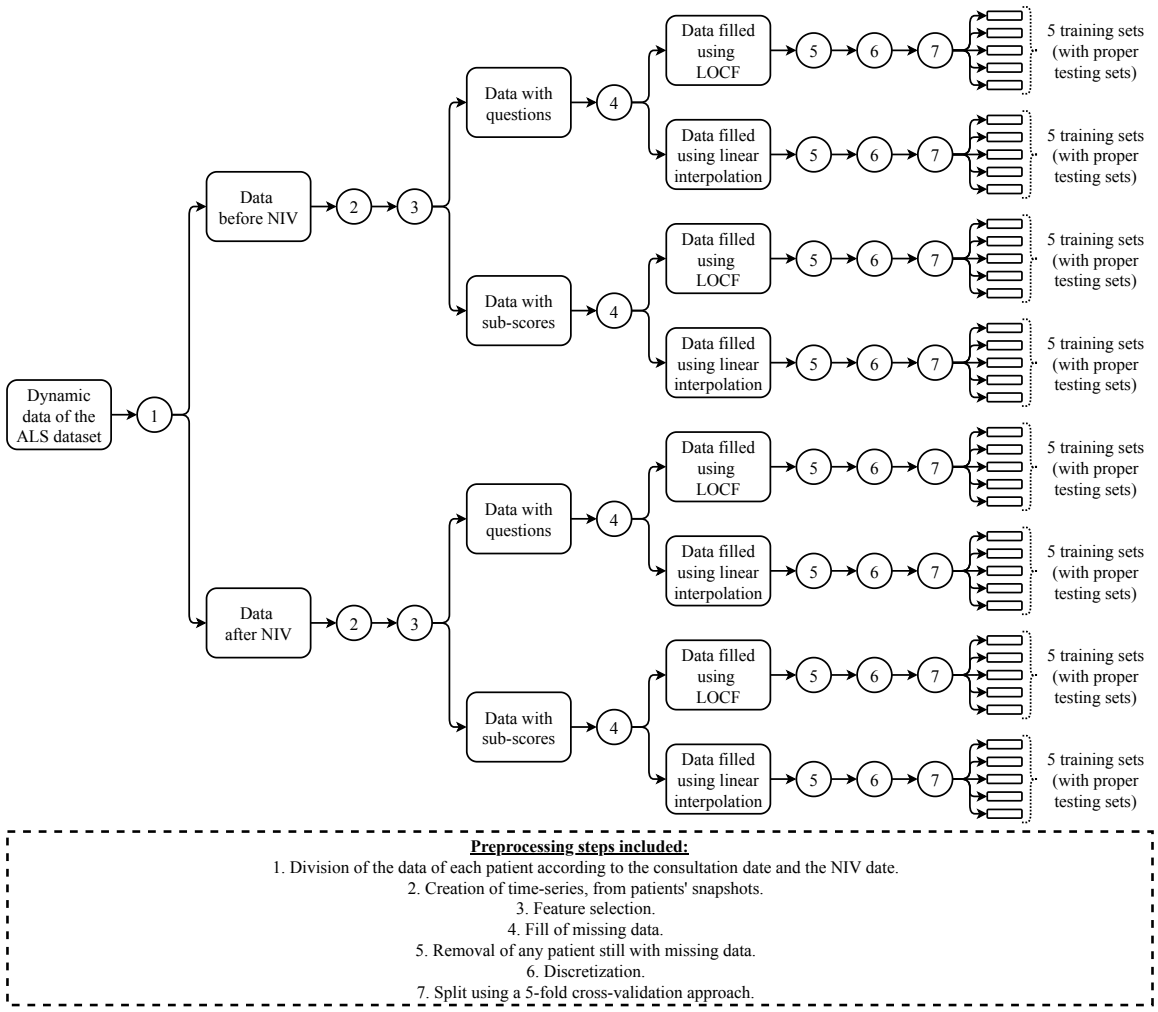


Figure 7.1: Overview of the preprocessing of the dynamic features of the ALS dataset.

Step 1: dataset division into data before and after NIV

As the application of non-invasive ventilation (NIV) to a certain patient has an important influence in his evolution, clinicians completely distinguish the progression of a patient before and after NIV being applied. Therefore, the dynamic data must be divided, in order to obtain the datasets before NIV and after NIV.

Splitting the data into before and after NIV is simple, because the dataset stores the date of the consultation where each measurement was done, and also stores the date when NIV was applied to each patient. Patients to which NIV has not been applied (without NIV date in the dataset) have all their data in the dataset before NIV.

Step 2: from patients' snapshots to time-series

The ALS dataset contains patients' snapshots, which provide the measurements of each consultation of every patient (the consultations' dates are also given). These patients' snapshots must be converted to time-series, because the interval between two consecutive consultations of a certain patient is not always the same, but the sdtDBN

framework describes data provided as time-series, with a constant interval between two consecutive timesteps.

The snapshots of each patient are converted to time-series by assuming that the interval between any two consecutive consultations is always three months (this value was obtained with the help of ALS doctors). The conversion is done as follows. For a certain patient, timestep 0 has the data of his first consultation, timestep 1 has the data of his consultation three months after the first consultation, timestep 2 has the data of his consultation six months after the first consultation, and the remaining timesteps of that patient are determined with the same reasoning, until the data of the last consultation of the patient is used. If the patient does not have a consultation in the month corresponding to a certain timestep, the data from the nearest consultation is used in that timestep. The previous procedure is done for all patients, to convert the whole dataset from patients' snapshots to time-series.

Step 3: feature selection

The ALS dataset provides the questions of the ALS-FRS-R scale and the sub-scores of this scale, which are sums of specific questions (see Chapter 2). If all these variables were used for learning a unique DBN, the DBN would learn the relations between a certain sub-score and the questions that compose it, which is not useful. To avoid this phenomenon, the dynamic features are separated into a sub-dataset containing only the sub-scores and a sub-dataset containing only the questions. The remaining dynamic features (see Table 2.1) are used in both sub-datasets.

Some features must be removed from the dataset, due to having an excessive number of missing values, or not being useful for a specific reason. This selection was done with the help of clinical experts. Table 7.2 presents the feature selection performed, also showing the discretization of each dynamic feature (the discretization is only done in step 6, for the reason mentioned in the explanation of step 6).

Steps 4 and 5: fill of missing data and posterior removal of patients

Having each dataset with the proper features, the missing values must be filled. To improve the reliability of the results, there are generated two datasets from each of the four datasets obtained until step 4 (see Fig. 7.1), one filling data using LOCF (see Section 4.1.1) and the other using linear interpolation.

When using LOCF, two iterations are performed. The first iteration is the usual LOCF, filling an unknown value with the respective value from the previous timestep of the proper patient. The second is a backwards iteration, where a value still unknown is filled with the respective value from the immediately posterior timestep. This backwards iteration follows the same temporal idea as the usual LOCF (see Section 4.1.1), and guarantees that the data of a feature is also filled in situations where the initial timesteps are unknown, but the remaining are known.

After filling a dataset, either with LOCF or with interpolation, a patient only has a variable with missing values if there is not any measurement for that variable in all timesteps of the time-series of that patient. As the sdtDBN framework cannot properly learn an sdtDBN using a time-series where a variable does not have any observation, the patients still with any missing value in any variable after filling a certain dataset are removed from that dataset.

The removal of patients can be significant, because the number of missing values of some variables is relatively high (see Table 7.2). The number of patients in each of the first nine timesteps of the datasets obtained after step 3 and after step 5 is provided in Table 7.3, which only presents counts for the datasets before and after NIV, as the division into sub-scores and questions (done in step 3) does not change the number of patients of a dataset, and the number of patients eliminated from a dataset is the same when filling with LOCF or interpolation.

Table 7.2: Selection and discretization of the dynamic features of the dataset. The missing values are measured over the 5919 and the 3353 patients' consultations (converted to time-series) of, respectively, the datasets before and after NIV. In the "Selected?" column, A and B denote the datasets before NIV using, respectively, sub-scores and questions, while C and D denote the datasets after NIV using, respectively, sub-scores and questions.

Feature	**Missing values (%)**		**Selected?**	**Discretization (displayed as *label: respective elements*)**
	Before NIV	**After NIV**		
ALS-FRS	9,34	4,06	Yes, A and C	1: {0,1,...,11}; 2: {12,13,...,23}; 3: {24,25,...,35}; 4: {36,37,...,40}
ALS-FRS-R	12,72	5,58	No	Variable not selected because it is the sum of all sub-scores (unnecessary)
ALS-FRSb	9,43	4,15	Yes, A and C	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
ALS-FRSsUL	10,66	4,21	Yes, A and C	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
ALS-FRSsLL	10,66	4,41	Yes, A and C	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
ALS-FRSr	10,64	4,35	No	Variable not selected because P10 has the same information
R	11,67	5,85	Yes, A and C	1: {0,1,2,3}; 2: {4,5,6,7}; 3: {8,9,10,11}; 4: {12}
Each question of ALS-FRS (P1, ..., P10)	$\sim$ 10,50	$\sim$ 4,00	Yes, B and D	1: {0}; 2: {1}; 3: {2}; 4: {3}; 5:{4}
Each question of R (R1, R2, R3)	$\sim$ 13,00	$\sim$ 6,00	Yes, B and D	1: {0}; 2: {1}; 3: {2}; 4: {3}; 5:{4}
VC	56,39	86,04	No	Variable not selected because FVC has similar information
FVC	54,57	85,71	Yes, A and B	1: [0,40[; 2: [40,60[; 3: [60,80[; 4: [80,100]
MIP	59,35	86,67	Yes, A and B	1: [0,40[; 2: [40,60[; 3: [60,100]
MEP	59,50	86,76	Yes, A and B	1: [0,40[; 2: [40,60[; 3: [60,80[; 4: [80,100]
P0.1	73,81	88,52	No	Variable not selected due to the high number of missing values
SNIP	87,62	98,99	No	Variable not selected due to the high number of missing values
PhrenMeanLat	61,19	91,56	No	Variable not selected because PhrenMeanAmpl has similar information
PhrenMeanAmpl	61,14	91,56	Yes, A and B	1: [0; 0,4[; 2: [0,4;+∞[
CervicalFlex	89,44	98,90	No	Variable not selected due to the high number of missing values
CervicalExt	89,46	98,90	No	Variable not selected due to the high number of missing values

Table 7.3: Number of patients per timestep, for each dataset obtained after steps 3 and 5 of Fig. 7.1.

Dataset		**Timestep**								
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Before NIV	After step 3 of Fig. 7.1	1177	794	608	471	392	310	241	203	164
	After step 5 of Fig. 7.1	685	581	463	374	313	250	194	162	128
After NIV	After step 3 of Fig. 7.1	598	442	368	305	239	191	160	135	111
	After step 5 of Fig. 7.1	567	428	359	299	234	187	156	131	110

Step 6: discretization

After filling all variables (step 4) and removing the proper patients from each dataset (step 5), the dynamic features should be discretized. The discretization is only done after filling the missing data, because, in the datasets where the data is linearly interpolated, the interpolation should be done using the real data, instead of the labels of the discretization. The several dynamic features of the ALS dataset are discretized as already presented in Table 7.2.

Step 7: generation of training and testing sets using stratified 5-fold cross-validation

The last preprocessing step consists in generating training and testing sets. For every dataset obtained until step 7, five training sets (and proper testing sets) are obtained, using 5-fold cross-validation. This cross-validation is stratified for each dataset, using as stratification class the length of the several time-series of the respective dataset. Therefore, the training sets obtained from each dataset have a similar number of time-series of each length.

The stratification class is the length of the time-series because, in the datasets before NIV, it indicates the kind of progression of a patient until needing NIV, whereas, in the datasets after NIV, it is an indicator of a patient's survival time after NIV being applied. As these indicators can distinguish the several kinds of patients, using the length of the time-series as stratification class allows learning each sdtDBN using different kinds of patients.

After generating the training and testing sets with dynamic data as previously explained, the respective training and testing sets with static data must be obtained. For a certain training/testing set D with dynamic data, the proper set with static data is obtained by selecting, from the whole static dataset (with the features selected and discretized as shown in Table 7.1), only the static data of the patients whose dynamic data is in the set D.

7.2 Study of the whole ALS dataset

After preprocessing the dataset and obtaining the proper training sets (see Section 7.1), there are learned sdtDBNs to study the progression of ALS patients. As the sdtDBNs are probabilistic models, the first analysis, presented in Section 7.2.1, consists in learning sdtDBNs with the training data and evaluating the performance of the model when predicting the values of certain variables, using the learned sdtDBNs and the testing data.

However, a major advantage of sdtDBNs is their graphical representation of the conditional dependencies among variables, which is not considered when only making predictions of the values of variables. To exploit this graphical representation of sdtDBNs, the second analysis, presented in Section 7.2.2, graphically determines the influence each variable has in each timestep, whereas the third analysis, presented in Section 7.2.3, graphically assesses the correlations among variables throughout the disease progression.

Two general considerations must be presented regarding Sections 7.2.1, 7.2.2 and 7.2.3.

The first consideration is that, in all assessments, there are learned sdtDBNs in the four scenarios presented in Fig. 7.1, being the learned sdtDBNs denoted as: **(i)** sdtDBNs before NIV with questions; **(ii)** sdtDBNs before NIV with sub-scores; **(iii)** sdtDBNs after NIV with questions; **(iv)** sdtDBNs after NIV with sub-scores. As shown in Fig. 7.1, each scenario is composed by 10 training sets (and respective testing sets), five filled using LOCF and five filled using linear interpolation. As Section 7.1 already addresses the creation of the training and testing sets, the remaining of Chapter 7 focuses on the four mentioned scenarios, only mentioning if, in each situation, all 10 training and testing sets are used, or only a subset of these training and testing sets is used.

The second consideration is that all learned sdtDBNs have nine timesteps, from timestep 0 to 8. As the interval between consecutive timesteps is three months (see Section 7.1), the sdtDBNs before NIV allow studying the progression of ALS patients during two years before NIV being applied, whereas the sdtDBNs after NIV allow studying the progression of ALS patients during two years after NIV being applied.

7.2.1 Prediction of the variables' values

To assess the predictive capabilities of sdtDBNs, there are learned sdtDBNs using training data in the four scenarios presented in Section 7.2, being the models' performance evaluated with the respective testing data. Each scenario has 10 training sets and the respective testing sets. In the following explanations, a training set with dynamic data (see Fig. 7.1) is denoted as D_{train}, the respective testing set with dynamic data is denoted as D_{test}, the static data of the patients in D_{train} is denoted as S_{train}, and the static data of the patients in D_{test} is denoted as S_{test}.

As the goal is to make predictions, there are learned stationary sdtDBNs, to get reliable parameters, considering that the quantity of training data is relatively low (see Table 7.3). As stated in Section 7.2, the maximum timestep of the sdtDBNs is 8. To avoid overfitting the training data and for the learning procedure to be done in reasonable time, the sdtDBNs have, using the notation of Chapter 5, $m = 1$, $p = 2$ and $b = 1$. The previous combination of parameters is used to learn sdtDBNs that maximize the LL score or the MDL score. Therefore, each $\{D_{train}, S_{train}\}$ is used to learn two sdtDBNs, being all learned sdtDBNs presented in Table 7.4.

Table 7.4: Number of sdtDBNs learned to make predictions in each scenario. Each set of 10 sdtDBNs concerns the 10 training sets of each scenario (five filled using LOCF and five filled using linear interpolation).

Structure of the learned stationary sdtDBNs	**Scenario**			
	Before NIV		**After NIV**	
	Sub-scores	**Questions**	**Sub-scores**	**Questions**
$\{m = 1, p = 2, b = 1, T = 8\}$ + LL	10	10	10	10
$\{m = 1, p = 2, b = 1, T = 8\}$ + MDL	10	10	10	10

To assess the performance of an sdtDBN learned using a certain $\{D_{train}, S_{train}\}$, there are predicted values of variables using the learned sdtDBN and the respective $\{D_{test}, S_{test}\}$, being these predictions done for every variable in every timestep. As an important goal in ALS is to study the functional decline of patients, in the scenarios with sub-scores the predicted variables are the sub-scores, while in the scenarios with questions the predicted variables are the questions. To evaluate the predictions made using the learned sdtDBNs, the three used metrics are the accuracy, the sensitivity and the area under the curve (AUC), which are calculated for the predictions of every variable in every timestep, using the sdtDBNs determined in each scenario (see Table 7.4).

Considering all sdtDBNs learned in a certain scenario W, and considering some variable A in a timestep t, the accuracy of the predictions of $A[t]$ done by all sdtDBNs learned in scenario W is determined as follows. For a certain sdtDBN learned in scenario W from training data $\{D_{train}, S_{train}\}$, the value of $A[t]$ is estimated for each patient of the respective $\{D_{test}, S_{test}\}$, being the estimation either correct or incorrect. This procedure is done for every sdtDBN learned in scenario W, being the accuracy of the predictions of $A[t]$ using all sdtDBNs learned in scenario W given by

$$\textit{accuracy} = \frac{\text{Number of correct predictions of } A[t] \text{ using all sdtDBNs learned in scenario } W}{\text{Total number of predictions of } A[t] \text{ using all sdtDBNs learned in scenario } W}. \tag{7.1}$$

Considering again all sdtDBNs learned in a scenario W (see Table 7.4), and a certain variable A in a timestep t, the sensitivity and the AUC of the predictions of $A[t]$ done by all sdtDBNs learned in scenario W are determined as follows. First, Q_1 is determined, which is the first quartile of the values of $A[t]$, considering the values of all patients in all testing sets in scenario W. Then, for a certain sdtDBN learned in scenario W from training data $\{D_{train}, S_{train}\}$, the value of $A[t]$ is estimated for each patient of the respective $\{D_{test}, S_{test}\}$, being each estimation of $A[t]$ classified according to Table 7.5.

Table 7.5: Classification of an estimation, done using an sdtDBN learned from training data $\{D_{train}, S_{train}\}$ in a scenario W, of the value of a variable $A[t]$, for a certain patient in the corresponding testing data $\{D_{test}, S_{test}\}$. Q_1 is the first quartile of the values of $A[t]$, considering the values of all patients in all testing sets of scenario W.

	Estimation of $A[t] \leq Q_1$	Estimation of $A[t] > Q_1$
Correct value of $A[t]$ in $D_{test} \leq Q_1$	True positive (TP)	False negative (FN)
Correct value of $A[t]$ in $D_{test} > Q_1$	False positive (FP)	True negative (TN)

The previous procedure is done for every sdtDBN learned in scenario W, classifying each estimation of $A[t]$ according to Table 7.5. Having the total number of TP, FN, FP and TN for predictions of $A[t]$ done with sdtDBNs learned in scenario W, the sensitivity of the predictions of $A[t]$ using all sdtDBNs learned in scenario W is given by

$$\textit{sensitivity} = \frac{\text{Total number of TP in the predictions of } A[t] \text{ using all sdtDBNs learned in scenario } W}{\text{Total number of TP+FN in the predictions of } A[t] \text{ using all sdtDBNs learned in scenario } W}. \quad (7.2)$$

With the total number of TP, FN, FP and TN determined, it is also possible to obtain the AUC, which is the area under the receiver operating characteristic (ROC) curve. The ROC curve is a plot which takes into account the true positive rate and the false positive rate.

The motivation for using the sensitivity and the AUC should be addressed. As can be seen in the discretization presented in Table 7.2, the labels of the sub-scores and the labels of the questions are ordered by functional capability: for every sub-score, the lowest label contains the lowest functional capabilities, the highest label contains the highest functional capabilities and the intermediate labels are ordered accordingly. The same is valid for the ALS-FRS-R questions. The reasoning for using the sensitivity and the AUC is based on the clinical goal of correctly determining when the functional capabilities of a patient will get bellow a certain threshold. Using the ordering of the labels, Q_1 (which is determined as already explained) can be used to represent this threshold (adapting the threshold to each variable in each timestep), and the mentioned clinical goal can be achieved by assessing the sensitivity using Eq. (7.2), with the classes defined as explained in Table 7.5. The AUC is also determined, as it gives an overview of the performance of the predictions regarding the classes defined in Table 7.5.

Tables 7.6 and 7.7 present the results of predicting the values of each sub-score in each timestep, using, respectively, the sdtDBNs before and after NIV with sub-scores. The results of predicting the questions (with the proper sdtDBNs before and after NIV) are given in Appendix A.1.1. The provided results are obtained with sdtDBNs learned using the MDL score, as these sdtDBNs have a better performance than the sdtDBNs using the LL score.

In Tables 7.6 and 7.7, the number of evaluated patients may not be the same for all variables in the same timestep, because the variables may have distinct static parents in the sdtDBNs. As the static datasets may have missing values, if two variables have distinct static parents, the number of patients for which it is possible to predict the value of each variable may vary, according to the missing data.

Table 7.6: Results of the predictions, for every sub-score in every timestep, of the sdtDBNs before NIV with sub-scores, using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ALS-FRS	Accuracy (%)	81,20	76,51	77,54	83,10	74,67	80,51	80,64	83,63
	Sensitivity (%)	86,91	85,07	90,42	75,92	64,24	72,68	73,87	72,82
	AUC (%)	89,81	87,84	92,03	86,55	81,31	84,86	86,42	83,72
	Evaluated patients	1090	864	690	568	446	354	284	226
ALS-FRSb	Accuracy (%)	88,20	81,46	79,88	81,70	78,80	79,48	87,68	87,73
	Sensitivity (%)	91,41	86,11	83,14	86,65	82,18	87,87	89,47	88,51
	AUC (%)	95,03	91,57	89,71	91,51	89,62	90,47	94,04	91,25
	Evaluated patients	1042	836	666	552	434	346	276	220
ALS-FRSsUL	Accuracy (%)	81,06	73,99	78,85	78,10	74,37	80,53	81,13	84,17
	Sensitivity (%)	71,87	68,72	79,10	77,60	76,66	90,91	83,70	86,56
	AUC (%)	84,95	83,24	88,00	86,79	85,92	93,50	89,98	93,28
	Evaluated patients	1098	892	714	598	476	380	302	240
ALS-FRSsLL	Accuracy (%)	82,55	76,30	74,12	79,97	78,08	78,10	79,03	80,08
	Sensitivity (%)	80,68	82,43	77,98	84,96	91,09	88,06	94,41	88,97
	AUC (%)	89,40	88,31	85,34	89,09	92,94	89,09	92,63	91,51
	Evaluated patients	1146	920	738	614	488	388	310	246
R	Accuracy (%)	86,18	80,86	83,33	82,07	81,11	84,97	83,70	85,45
	Sensitivity (%)	70,31	59,13	62,44	61,05	58,61	63,83	57,14	58,03
	AUC (%)	82,23	75,03	77,91	76,71	75,15	79,74	75,66	77,75
	Evaluated patients	1042	836	666	552	434	346	276	220

Table 7.7: Results of the predictions, for every sub-score in every timestep, of the sdtDBNs after NIV with sub-scores, using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ALS-FRS	Accuracy (%)	81,19	77,18	79,03	85,45	82,04	82,01	79,49	82,83
	Sensitivity (%)	83,52	82,63	86,77	84,76	71,43	69,77	73,43	78,95
	AUC (%)	90,26	89,71	92,75	92,38	85,31	84,88	84,78	87,83
	Evaluated patients	760	644	534	426	334	278	234	198
ALS-FRSb	Accuracy (%)	84,74	83,85	79,03	79,11	83,83	80,22	79,49	87,38
	Sensitivity (%)	87,14	81,95	82,11	90,53	93,94	90,91	90,33	95,12
	AUC (%)	92,95	90,18	90,71	93,99	96,47	95,45	93,75	97,56
	Evaluated patients	760	644	534	426	334	278	234	198
ALS-FRSsUL	Accuracy (%)	82,08	75,64	78,87	79,22	81,50	84,44	84,53	85,72
	Sensitivity (%)	85,26	77,17	82,11	83,49	83,33	87,36	85,81	92,09
	AUC (%)	92,11	87,68	89,87	89,55	89,61	92,12	87,35	94,64
	Evaluated patients	826	698	582	462	362	302	252	210
ALS-FRSsLL	Accuracy (%)	83,31	77,17	78,38	79,29	80,89	83,10	82,94	91,72
	Sensitivity (%)	83,14	77,29	79,38	80,26	82,80	84,36	87,10	91,99
	AUC (%)	91,10	88,19	88,18	89,48	89,59	90,06	91,40	95,99
	Evaluated patients	809	684	569	454	356	296	246	205
R	Accuracy (%)	82,90	80,13	79,96	80,05	80,24	83,81	82,48	79,80
	Sensitivity (%)	72,13	65,95	72,73	74,59	74,69	80,00	83,61	83,33
	AUC (%)	83,16	80,43	83,34	83,82	82,70	86,09	86,45	83,82
	Evaluated patients	760	644	534	426	334	278	234	198

The results provided in Tables 7.6 and 7.7 demonstrate that the sdtDBNs can accurately predict the progression of ALS patients, both before and after NIV being applied, as the accuracy and the AUC have extremely high values in most timesteps of all sub-scores before and after NIV. Some results are above 90% and almost all results are above 75%, which are extremely positive indicators.

The sensitivity does not have such high values, in particular in the R sub-score before NIV (see Table 7.6). The low sensitivity results in the R score before NIV happen because, when the respiratory functionalities (assessed with the R score) decline, the patient gets NIV, thus moving from the series before NIV to the series after NIV. Therefore, the training data before NIV in which R does not have the highest possible value is in extremely low quantity, which causes the sdtDBNs before NIV not to learn properly the transition of the R score from the highest value to lower values. This phenomenon happens in sdtDBNs (and DBNs in general), because they find the conditional dependencies among variables, but if some value of a parent is not often observed, the corresponding distribution when addressing the CPT is not properly determined. This also happens in some timesteps of other variables in Tables 7.6 and 7.7, but the R score before NIV is the most noticeable case, due to the explained reason.

The results presented in Tables 7.6 and 7.7 are competitive with state-of-the-art works applying machine learning techniques to study the progression of ALS patients [27, 29, 70, 71]. Although some of these works provide a slightly better performance, most works focus on a specific goal, for example, NIV prediction [27, 29], instead of predicting the values of all variables in all timesteps/consultations, as shown in Tables 7.6 and 7.7. This crucial difference must be taken into account when comparing Tables 7.6 and 7.7 with most results in literature.

Several results are slightly better in the last two timesteps of Tables 7.6 and 7.7 than in the initial timesteps. As in the initial timesteps there are patients with all kinds of progressions, but in the last timesteps there only patients with a slow progression, this slightly better performance in the last timesteps may indicate that a proper division of patients according to their kind of progression and a study of each progression group would be beneficial, thus providing a motivation for Section 7.3.

7.2.2 Influence of each variable in every timestep

The prediction of variables (see Section 7.2.1) does not exploit the graphical display provided by sdtDBNs, which is one of the main characteristics that distinguishes sdtDBNs from other machine learning models. In this Section, the graphical display of sdtDBNs is used to assess the influence each variable has in the variables of every timestep.

As in Section 7.2.1, the sdtDBNs used in this Section are learned in the four scenarios presented in Section 7.2, being a certain training data used to learn an sdtDBN denoted as $\{D_{train}, S_{train}\}$, as defined in Section 7.2.1. The training data is obtained as explained in Section 7.1, being only used the data filled with LOCF, as the results when applying LOCF and interpolation are identical. This Section does not use the testing data, because the goal is to assess the graphical display of the learned sdtDBNs (the models' performance was already tackled in Section 7.2.1).

To assess the relations among variables in each timestep, there are learned non-stationary sdtDBNs. As stated in Section 7.2, the maximum timestep of the sdtDBNs is 8. To get multiple relations between variables, the sdtDBNs are learned using the LL score. The remaining parameters are tuned to get several relations among variables, while learning the sdtDBNs in reasonable time. Using the notation of Chapter 5, the sdtDBNs with sub-scores are learned with $\{m = 1, p = 2, b = 1\}$ and $\{m = 1, p = 2, b = 2\}$, while the sdtDBNs with questions are only learned with $\{m = 1, p = 2, b = 1\}$, because the combination of parameters using $b = 2$ does not allow learning sdtDBNs with

questions in reasonable time, due to the high number of questions.

All previously specified sdtDBNs are learned with the restriction that each variable can never be a parent of itself. Therefore, all edges $A[t] \rightarrow A[t+1]$, where A is any variable of an sdtDBN and t is any timestep, are defined as forbidden (see Section 5.2), when learning each sdtDBN. If the sdtDBNs were not restricted, each dynamic variable in a certain timestep would almost always be a parent of itself in the following timestep (in general, the better predictor of a variable in timestep $t+1$ is the same variable in timestep t), which would harm the results of this Section, because the goal is to find the influence each variable has in other variables (the influence a variable has in itself is already intuitively known). All learned sdtDBNs are presented in Table 7.8.

Table 7.8: Number of sdtDBNs learned to assess the influence each variable has in the variables of every timestep. Each set of five sdtDBNs concerns the five training sets of each scenario filled using LOCF (see Fig. 7.1). As notation, t represents any timestep of a learned sdtDBN.

Structure of the learned non-stationary sdtDBNs, where each variable in a timestep t cannot be a parent of itself in timestep t + 1	**Scenario**			
	Before NIV		**After NIV**	
	Sub-scores	**Questions**	**Sub-scores**	**Questions**
$\{m = 1, p = 2, b = 1, T = 8\}$ + LL	5	5	5	5
$\{m = 1, p = 2, b = 2, T = 8\}$ + LL	5	Not learned	5	Not learned

The influence each variable has in every timestep is determined using the following graphical reasoning. As the edges of an sdtDBN denote the conditional dependencies among variables, given any two variables X and Y of an sdtDBN, if X is a parent of more variables in timestep t than Y, it can be concluded that X has more influence than Y in the variables of timestep t of the mentioned sdtDBN. Given this reasoning, if creating a 2D table where each row designates a timestep, each column represents a variable, and each element contains the number of children a certain variable has in a certain timestep, each row of this table provides an ordering of the influence that the variables have in the respective timestep. Timestep 0 is never considered in this analysis, because variables in timestep 0 of sdtDBNs do not have parents. Providing an example, for an sdtDBN with three timesteps (1, 2 and 3) having three variables (X, Y and Z), the explained table has the format presented in Table 7.9.

Table 7.9: Example of the table used to assess the influence each variable of an sdtDBN has in each timestep. It is used an sdtDBN with three timesteps and with variables X, Y and Z. As notation, # means "number of".

		Variable		
		X	***Y***	***Z***
Timestep	**1**	#(children of X in $t = 1$)	#(children of Y in $t = 1$)	#(children of Z in $t = 1$)
	2	#(children of X in $t = 2$)	#(children of Y in $t = 2$)	#(children of Z in $t = 2$)
	3	#(children of X in $t = 3$)	#(children of Y in $t = 3$)	#(children of Z in $t = 3$)

Table 7.9 provides an ordering of the influence of the variables in each timestep, according to the values associated to the counts of each row. To assess the influence of the variables of the ALS dataset in every timestep of the disease progression, it is determined, for each scenario analyzed (see Section 7.2 and Table 7.8), a table using the same reasoning of Table 7.9. In each of these tables, the timesteps go from 1 to 8 and the variables are either the ALS sub-scores (in the scenarios with sub-scores) or the ALS questions (in the scenarios with questions).

For a certain scenario W, the corresponding table similar to Table 7.9 is obtained as follows. First, there are obtained several tables similar to Table 7.9, using the sdtDBNs learned from every possible training data $\{D_{train}, S_{train}\}$ in scenario W (see Fig. 7.1 and Table 7.8). Then, it is done an element-wise sum of all these tables, obtaining one table similar to Table 7.9 that includes the counts of all sdtDBNs created in scenario W.

Having the mentioned tables learned for all four scenarios, it is possible to uniformize all results by normalizing each row of each table, so that all elements are between 0 and 1 (the normalization is done by dividing each element by the sum of the elements of its respective row). To provide an intuitive overview of the obtained results, these normalized tables are presented in stacked bar charts, where the sizes of the stacked bars in a certain timestep t allow comparing the influence that the several variables have in the variables of that timestep t (larger bars mean higher influence in the corresponding timestep t).

The influence each variable has in the variables of each timestep of the sdtDBNs with questions before and after NIV is presented, respectively, in Figs. 7.2 and 7.3. The influence each variable has in each timestep of the sdtDBNs with sub-scores (before and after NIV) is presented in Appendix A.1.2.

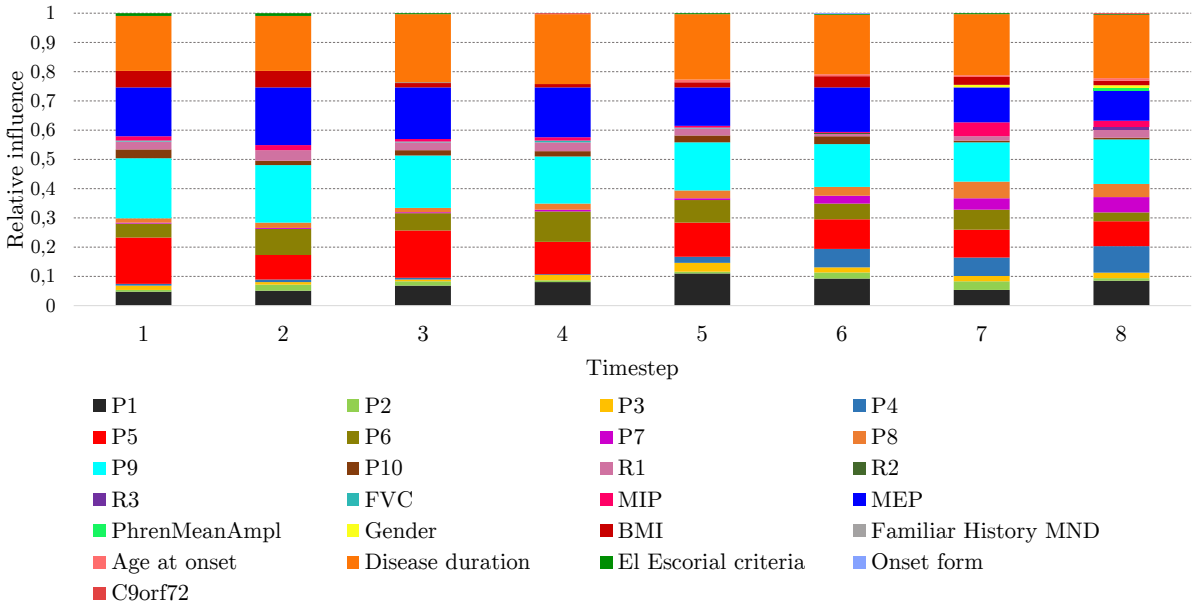


Figure 7.2: Influence of each variable in every timestep of the sdtDBNs before NIV with questions. The learned sdtDBNs are detailed in Table 7.8.

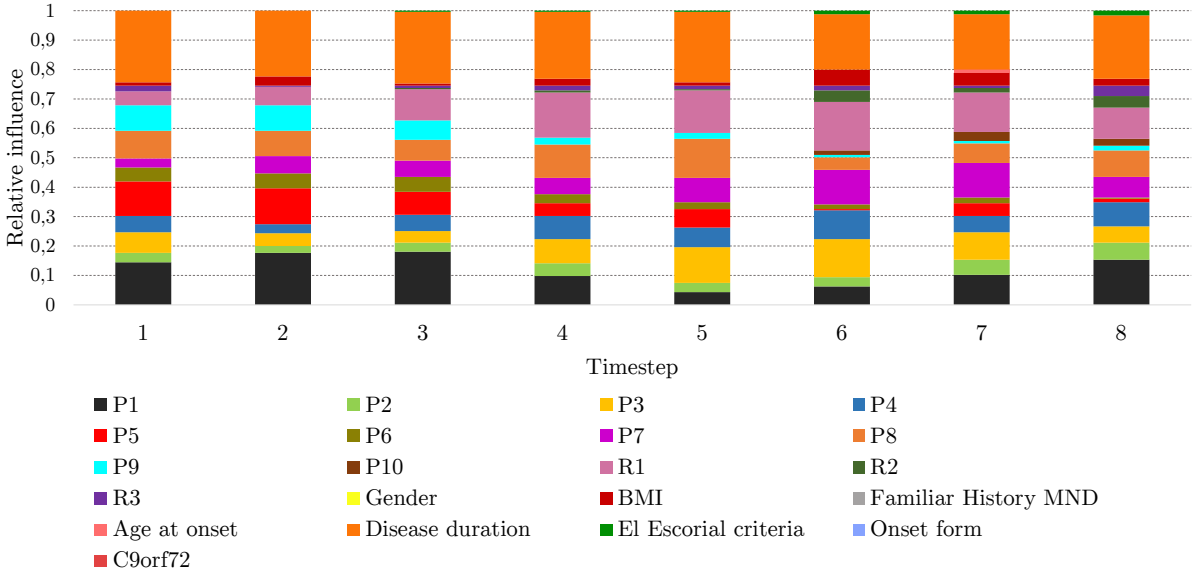


Figure 7.3: Influence of each variable in every timestep of the sdtDBNs after NIV with questions. The learned sdtDBNs are detailed in Table 7.8.

To keep the following analysis intuitive, it is presented a qualitative assessment of Figs. 7.2 and 7.3. However, a quantitative analysis could be done, by checking the values of the tables that originate Figs. 7.2 and 7.3.

Regarding the ALS-FRS-R questions, Fig. 7.2 shows that P5 and P9 are the most important ALS-FRS-R questions in the disease progression before NIV, while Fig. 7.3 presents P1 and R1 as the most influential questions in the progression after NIV. In the results after NIV (Fig. 7.3), it can also be noticed that P5 has a large influence in the initial timesteps, P8 has a large influence in the intermediate timesteps, and P3 and P7 have a large influence in the final timesteps. The questions with the highest influence before and after NIV belong to different sub-scores of the ALS-FRS-R scale, which validates the division of the ALS-FRS-R scale into sub-scores, as all sub-scores are represented when analyzing the datasets with questions. On the other hand, the differences found between the relevant questions before and after NIV may suggest that the questions of the ALS-FRS-R scale can be adapted to a patient's clinical situation, by focusing on the relevant questions according to NIV having been applied or not.

The respiratory questions of ALS-FRS-R (R1, R2 and R3) do not have a large influence in the disease progression before NIV, which can be counter-intuitive, as respiratory problems are the main cause of death among ALS patients. To explain this phenomenon, the influence of the respiratory tests (FVC, MIP, MEP and PhrenMeanAmpl) must be tackled, which can only be done in Fig. 7.2, as the respiratory tests are only used in the sdtDBNs before NIV (see Table 7.2). Fig. 7.2 shows that MEP is the most influential respiratory test before NIV, which suggests that MEP is a better prognostic indicator than the remaining respiratory tests. The influence of MEP is also significant when comparing all variables of Fig. 7.2. In particular, MEP has more influence in the disease progression than all respiratory questions of ALS-FRS-R before NIV, which suggests that MEP is a better prognostic indicator than the respiratory questions of ALS-FRS-R. After NIV (see Fig. 7.3), the influence of the R1 question in the disease progression is larger than before NIV, which can be justified by the fact that the respiratory tests are not used in the sdtDBNs after NIV.

The influence of the static variables in the disease progression can also be assessed using Figs. 7.2 and 7.3, which show that the disease duration is the most influential static variable, both before and after NIV. This finding is extremely relevant, as it supports the importance of an early diagnosis (the diagnosis of the disease is a major area of research in ALS).

7.2.3 Correlations among variables throughout the disease progression

In Section 7.2.2, it is used the graphical display of sdtDBNs to determine the influence of each variable in every timestep of the sdtDBNs learned in each of the four scenarios presented in Fig. 7.1 (see Section 7.2 for an explanation of these scenarios). Although the analysis of Section 7.2.2 allows checking the most important variables in the disease progression, it does not allow evaluating which variables are the most correlated in the learned sdtDBNs. This Section assesses the correlations among variables, by graphically inspecting the learned sdtDBNs.

As the goal is to inspect the graphical display of the sdtDBNs, the reasoning for learning sdtDBNs in this Section is the same as in Section 7.2.2. Therefore, the sdtDBNs learned in this Section are the ones presented in Table 7.8, and all considerations done in Section 7.2.2 to obtain Table 7.8 are still valid in the current Section. In particular, there are also learned sdtDBNs in the four scenarios of Table 7.8, it is also used the notation $\{D_{train}, S_{train}\}$ to represent a certain training data used to learn an sdtDBN, and the sdtDBNs are also learned by making the restriction that each variable can never be a parent of itself (see Section 7.2.2).

The correlations between variables throughout all timesteps of an sdtDBN are determined using the following graphical reasoning. As an edge of an sdtDBN represents a conditional dependency relation among two variables, given any three variables X, Y and Z of an sdtDBN, if, throughout all timesteps of the sdtDBN, there are more edges between X and Y than between X and Z, it can be concluded that X and Y are more correlated than X and Z. Given this reasoning, a 2D table can be created, where each dimension has all variables of an sdtDBN, and a certain element of the table contains the number of edges between the two respective variables, considering all timesteps of the sdtDBN. The values of the elements of this table provide an ordering of the correlations among the variables of the sdtDBN, because the higher the value of an element of the table, the higher the correlation between the respective variables, according to the graphical reasoning previously explained.

The aforementioned table is always symmetric, because, when counting the number of edges between two variables, it is not distinguished which variable is the parent and which is the child (in general, given any two variables A and B, both $A \rightarrow B$ and $B \rightarrow A$ indicate a correlation between A and B). As the table is always created using sdtDBNs where a variable can never be a parent of itself, the diagonal of the explained table is never considered, as it always has zeros, due to the imposed restriction. Providing an example, for an sdtDBN having three variables (X, Y and Z), the explained table has the format presented in Table 7.10.

Table 7.10: Example of the table used to assess the correlations between variables, considering all timesteps of an sdtDBN. It is used an sdtDBN with variables X, Y and Z. As notation, # means "number of". The counts of the numbers of edges are done considering all timesteps of the sdtDBN. The diagonal of the table is not considered, because the sdtDBNs learned in this context are restricted so that each variable can never be a parent of itself.

	$\boldsymbol{X}$	$\boldsymbol{Y}$	$\boldsymbol{Z}$
$\boldsymbol{X}$	—	#(edges from X to Y) + #(edges from Y to X)	#(edges from X to Z) + #(edges from Z to X)
$\boldsymbol{Y}$	#(edges from X to Y) + #(edges from Y to X)	—	#(edges from Y to Z) + #(edges from Z to Y)
$\boldsymbol{Z}$	#(edges from X to Z) + #(edges from Z to X)	#(edges from Y to Z) + #(edges from Z to Y)	—

Table 7.10 provides an ordering of the correlations among variables in all timesteps of an sdtDBN, according to the values associated to the counts of each element of the table. To evaluate the correlations among variables of the ALS dataset, it is determined, for each scenario analyzed (see Section 7.2 and Table 7.8), a table with the same reasoning of Table 7.10, using as variables either the ALS sub-scores (in the scenarios with sub-scores) or the ALS questions (in the scenarios with questions).

For a certain scenario W, the corresponding table similar to Table 7.10 is obtained as follows. First, there are obtained several tables similar to Table 7.10, using the sdtDBNs learned from every possible training data $\{D_{train}, S_{train}\}$ in scenario W (see Fig. 7.1 and Table 7.8). Then, it is done an element-wise sum of all these tables, obtaining one table similar to Table 7.10 that includes the counts of all sdtDBNs created in scenario W.

With the four tables similar to Table 7.10 determined (one for each scenario, using the proper sdtDBNs, presented in Table 7.8), the results can, as in Section 7.2.2, be uniformized, by normalizing each table. The normalization is done per column, dividing each value of each table by the sum of all values in the same column. For each of these normalized tables, a certain column, associated to some variable X, provides an overview of the correlations between that variable X and every variable of the dataset, considering all relations in which X is involved (in the sdtDBNs of the respective scenario).

The correlations among the variables of the sdtDBNs with sub-scores before and after NIV are presented, respectively, in Tables 7.11 and 7.12, using the explained tables normalized per column. The correlations between the variables of the sdtDBNs with questions before and after NIV are presented in Appendix A.1.3. To ease the interpretation of the mentioned tables, a color scheme is used, which evolves from red (the lowest values) to green (the highest values).

Table 7.11: Correlations between variables of the sdtDBNs before NIV with sub-scores. The table is normalized per column. The learned sdtDBNs are detailed in Table 7.8.

	ALS-FRS	**ALS-FRSb**	**ALS-FRSsUL**	**ALS-FRSsLL**	**R**	**FVC**	**MIP**	**MEP**	**PhrenMeanAmpl**
ALS-FRS	—	7,24%	16,36%	14,78%	5,23%	0,63%	1,28%	7,08%	5,30%
ALS-FRSb	10,98%	—	15,64%	12,93%	16,00%	15,31%	12,98%	13,42%	15,56%
ALS-FRSsUL	32,22%	20,31%	—	19,40%	16,62%	19,06%	16,27%	16,13%	11,26%
ALS-FRSsLL	30,55%	17,64%	20,36%	—	20,00%	19,69%	18,46%	16,44%	23,51%
R	4,06%	8,19%	6,55%	7,51%	—	6,25%	6,40%	7,80%	2,32%
FVC	0,48%	7,72%	7,39%	7,27%	6,15%	—	3,29%	10,30%	2,65%
MIP	1,67%	11,18%	10,79%	11,66%	10,77%	5,63%	—	19,98%	11,26%
MEP	16,23%	20,31%	18,79%	18,24%	23,08%	30,94%	35,10%	—	28,15%
PhrenMeanAmpl	3,82%	7,40%	4,12%	8,20%	2,15%	2,50%	6,22%	8,84%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 7.12: Correlations between variables of the sdtDBNs after NIV with sub-scores. The table is normalized per column. The learned sdtDBNs are detailed in Table 7.8.

	ALS-FRS	**ALS-FRSb**	**ALS-FRSsUL**	**ALS-FRSsLL**	**R**
ALS-FRS	—	27,21%	23,88%	21,08%	7,38%
ALS-FRSb	39,38%	—	30,18%	28,83%	37,81%
ALS-FRSsUL	29,69%	25,93%	—	27,93%	27,29%
ALS-FRSsLL	24,12%	22,79%	25,70%	—	27,52%
R	6,80%	24,07%	20,23%	22,16%	—
Sum per column:	100%	100%	100%	100%	100%

The ALS-FRS columns of Tables 7.11 and 7.12 show that ALS-FRSsUL and ALS-FRSsLL are the sub-scores most correlated with ALS-FRS before NIV, while ALS-FRSb is the sub-score most correlated with ALS-FRS after NIV. This difference suggests that ALS-FRSsUL and ALS-FRSsLL are the sub-scores most correlated with the several variables before NIV and ALS-FRSb is the sub-score most correlated with the several variables after NIV, which is confirmed by checking the ALS-FRSb, ALS-FRSsUL and ALS-FRSsLL rows of Tables 7.11 and 7.12. These conclusions match the results of Section 7.2.2, where P5 and P9 (from ALS-FRSsUL and ALS-FRSsLL) have more influence before NIV, whereas P1 (from ALS-FRSb) has more influence after NIV.

The R columns of Tables 7.11 and 7.12 show that R is similarly correlated with all sub-scores before NIV (ALS-FRSb, ALS-FRSsUL and ALS-FRSsLL), while, after NIV, the respiratory indicator R is clearly more correlated with ALS-FRSb than with the other sub-scores. It should also be noticed that ALS-FRSsUL and ALS-FRSsLL are correlated with each other before and after NIV (as shown in the respective columns of Tables 7.11 and 7.12).

It must be highlighted the low correlation that R has with the remaining variables, which can be seen in the R rows of Tables 7.11 and 7.12. The low values in the R row of Table 7.11 can be justified by the presence of MEP, which also provides information regarding the respiratory capabilities of a patient, as explained in Section 7.2.2.

The ALS-FRS rows of Tables 7.11 and 7.12 show that the correlation of ALS-FRS with other variables is, in general, low. This can be explained by ALS-FRS being a global score (sum of 10 questions), while the remaining scores are sub-scores (sums of three questions). Therefore, the values of ALS-FRS are not as specific as the values of the sub-scores, which makes the values of ALS-FRS less useful than the values of the sub-scores, when indexing the conditional probability distributions.

Regarding the respiratory tests (FVC, MIP, MEP and PhrenMeanAmpl), they are only included in Table 7.11, because only the sdtDBNs before NIV have the respiratory tests (see Table 7.2). The MEP row of Table 7.11 shows that MEP is the respiratory test most correlated with all variables before NIV, which matches the results of Section 7.2.2, where MEP is the most influential respiratory test before NIV. In the MEP column of Table 7.11, it is shown that MEP is highly correlated with MIP, which is logical. Finally, it is interesting to notice, in the PhrenMeanAmpl column of Table 7.11, that PhrenMeanAmpl is highly correlated with ALS-FRSsLL.

7.3 Division into progression groups

The preprocessing and results presented, respectively, in Sections 7.1 and 7.2 use all patients of the ALS dataset (the pretreated version, as explained in Section 7.1). However, as distinct patients may have different functional declines, it should be made an analysis considering the diversity of ALS patients, which can be done by splitting patients into progression groups and analyzing the data of the patients of each group.

In this work, patients are divided into three progression groups, slow, average and fast, according to an estimation of their *progression rate*. For a certain patient, his *progression rate* is determined by

$$\textit{progression rate} = \frac{48 - \text{ALS-FRS-R}_{\text{First consultation}}}{\text{Number of months between the initial symptoms and the first consultation}}, \tag{7.3}$$

where 48 is used because it is the maximum value of ALS-FRS-R (it is the value assumed at the month in which the initial symptoms appear), $\text{ALS-FRS-R}_{\text{First consultation}}$ is the value of ALS-FRS-R in the first consultation, and the number of months between the initial symptoms and the first consultation is determined by checking the respective dates of the corresponding patient, for which the complete ALS dataset is checked, because the date of the initial symptoms of a certain patient is not in the pretreated version used throughout Chapter 7 (see Section 7.1).

After determining the *progression rate* of every patient, the patients with the lowest 25% *progression rates* compose the slow progression group, while the patients with the highest 25% *progression rates* compose the fast progression group. The patients with the remaining 50% *progression rates* compose the average progression group.

After splitting patients into progression groups as explained, each group can be treated as an individual dataset. Therefore, the preprocessing of each group is done by applying Section 7.1 to each progression group. The selection and discretization of static features remain the ones presented in Table 7.1. The dynamic features are preprocessed by applying the methods described in Fig. 7.1 to the data of each progression group, with the selection and discretization of dynamic features remaining the ones presented in Table 7.2. In Table 7.13, there are presented the counts of Table 7.3 for each progression group.

After preprocessing each progression group, the analysis performed (for the whole dataset) in Section 7.2 must be done for each group. Sections 7.3.1, 7.3.2 and 7.3.3 present the application of, respectively, Sections 7.2.1, 7.2.2 and 7.2.3 to each progression group. The considerations done in the beginning of Section 7.2 remain valid in Sections 7.3.1, 7.3.2 and 7.3.3. In particular, for each progression group, there are considered the four scenarios of Fig. 7.1 (before NIV with questions, before NIV with sub-scores, after NIV with questions and after NIV with sub-scores), each with its proper training (and testing) sets, in the corresponding progression group, and the sdtDBNs learned in Sections 7.3.1, 7.3.2 and 7.3.3 always have nine timesteps, from 0 to 8.

Table 7.13: Number of patients per timestep, for each dataset obtained after steps 3 and 5 of Fig. 7.1, in each progression group (slow, average and fast). This table is the application of Table 7.3 to each progression group.

Group	Dataset		Timestep								
			0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Slow	Before NIV	After step 3 of Fig. 7.1	269	204	183	163	151	135	116	104	89
		After step 5 of Fig. 7.1	171	157	142	132	121	108	94	84	70
	After NIV	After step 3 of Fig. 7.1	109	94	88	73	63	52	42	37	34
		After step 5 of Fig. 7.1	107	92	86	71	61	51	41	36	33
Average	Before NIV	After step 3 of Fig. 7.1	540	376	287	223	178	127	87	67	52
		After step 5 of Fig. 7.1	355	299	233	187	149	109	73	56	42
	After NIV	After step 3 of Fig. 7.1	326	242	204	177	137	106	94	75	60
		After step 5 of Fig. 7.1	320	241	203	176	136	105	93	74	60
Fast	Before NIV	After step 3 of Fig. 7.1	257	144	89	51	37	26	19	16	10
		After step 5 of Fig. 7.1	138	104	71	43	32	23	17	14	8
	After NIV	After step 3 of Fig. 7.1	115	71	53	40	27	21	13	13	11
		After step 5 of Fig. 7.1	113	71	53	40	27	21	13	13	11

7.3.1 Prediction of the variables' values

The detailed results of applying Section 7.2.1 to every progression group are provided in Appendix A.2.1. An overview of the results of the predictions of all sdtDBNs learned before and after NIV with sub-scores, for the whole dataset and for every progression group, is presented in Table 7.14, where each value is the average of the values of the respective indicator in all timesteps.

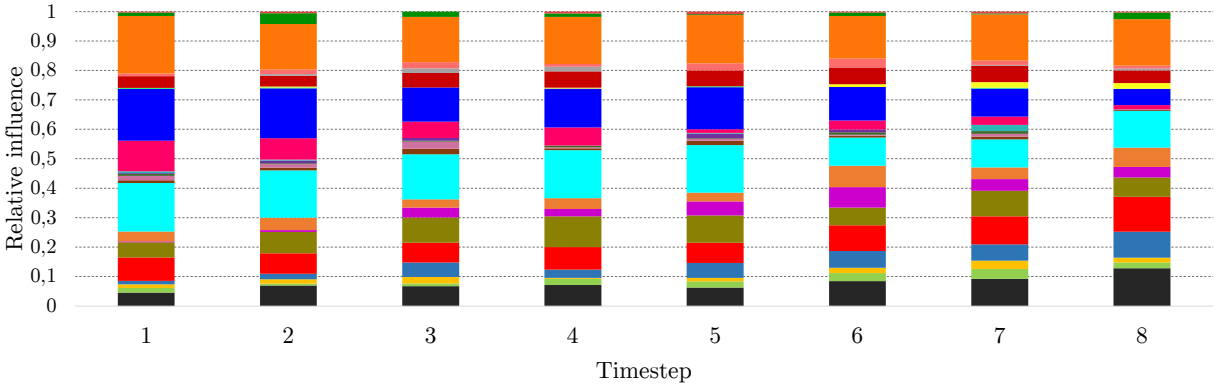
Table 7.14: Overview of the results of the predictions of the sdtDBNs before and after NIV with sub-scores, for the whole dataset and for each progression group. All sdtDBNs that contribute for the results are learned using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score (see Section 7.2.1). Each presented value is the average of the values of all timesteps, for the respective variable, in the respective scenario, in the respective progression group.

Metric		Accuracy (%)				Sensitivity (%)				AUC (%)			
Group		**None**	**Slow**	**Average**	**Fast**	**None**	**Slow**	**Average**	**Fast**	**None**	**Slow**	**Average**	**Fast**
Before NIV	ALS-FRS	79,72	82,40	77,43	76,61	77,74	84,50	71,04	70,10	86,57	89,16	82,82	84,87
	ALS-FRSb	83,12	88,15	79,59	78,33	86,92	83,39	83,23	74,07	91,65	90,09	90,75	86,54
	ALS-FRSsUL	79,02	80,04	76,54	71,87	79,39	78,70	71,54	70,58	88,21	87,39	84,51	83,46
	ALS-FRSsLL	78,53	80,89	73,99	68,33	86,07	78,60	71,12	64,14	89,79	87,55	84,57	80,47
	R	83,46	87,29	79,30	78,46	61,32	x	53,49	70,82	77,52	x	72,55	79,17
After NIV	ALS-FRS	81,15	78,34	79,50	82,82	78,91	79,70	75,42	74,91	88,49	87,31	86,82	86,89
	ALS-FRSb	82,20	81,90	82,28	81,14	89,00	86,53	81,24	88,22	93,88	92,24	89,87	93,89
	ALS-FRSsUL	81,50	81,22	80,30	81,00	84,58	80,73	80,20	85,59	90,36	88,52	88,91	90,19
	ALS-FRSsLL	82,10	82,52	81,32	75,58	83,29	82,03	81,49	80,99	90,50	90,76	89,80	89,38
	R	81,17	80,10	82,03	78,03	75,88	72,26	76,80	79,96	83,72	81,96	84,03	86,11

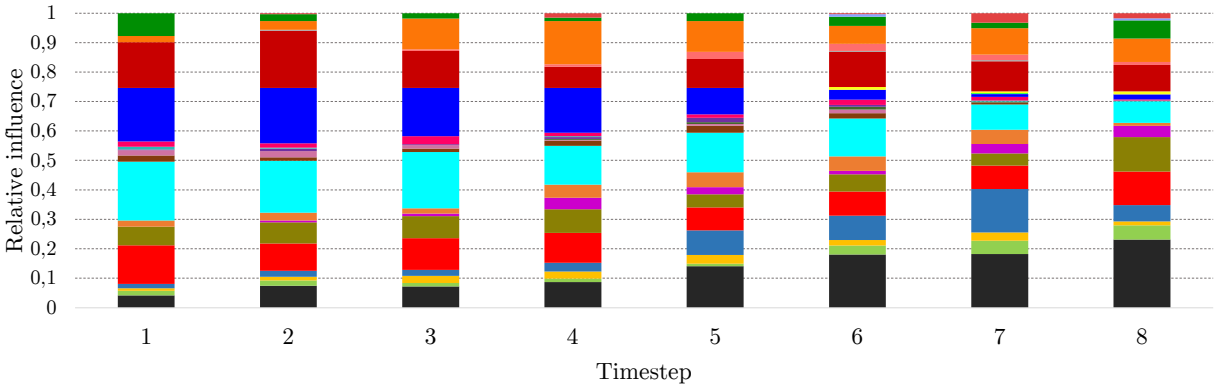
The sensitivity and the AUC of Table 7.14 are not presented when the first quartile of the respective values is the highest possible value (in that case, Table 7.5 cannot be used to define the labels). Table 7.14 shows that the results do not improve significantly when splitting the patients into progression groups (comparing to the results using the whole dataset). This is mainly due to the low quantity of observations of each progression group (see Table 7.13), which makes the sdtDBNs overfit the training data. For the slow progression group, the accuracies of ALS-FRSb and R before NIV are much higher than after NIV, while the other results are similar before and after NIV. For the fast progression group, the best results are obtained after NIV, which can be justified by NIV being applied early to these patients, so, after NIV, they still have data useful for learning the sdtDBNs.

7.3.2 Influence of each variable in every timestep

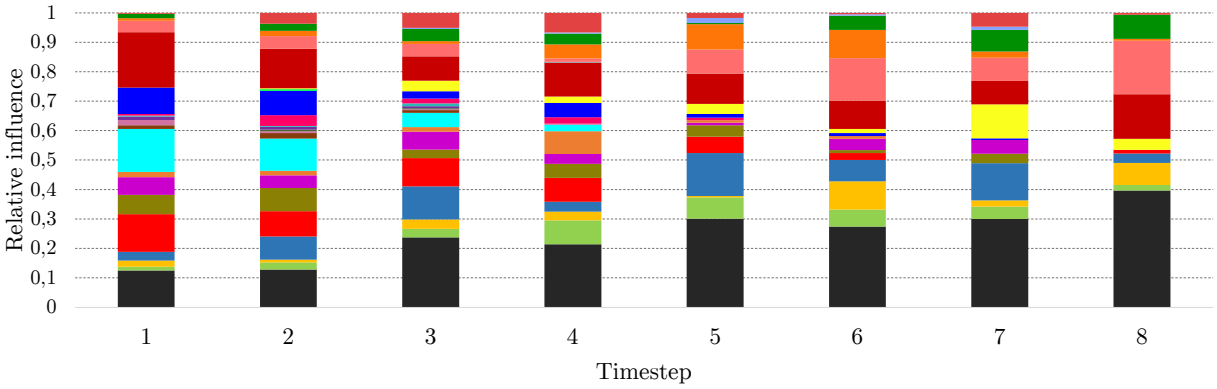
The influence of each variable in every timestep of the learned sdtDBNs is obtained by applying Section 7.2.2 to each progression group, being the results of the sdtDBNs before NIV with questions presented in Fig. 7.4.



(a) Slow progression group.



(b) Average progression group.



(c) Fast progression group.

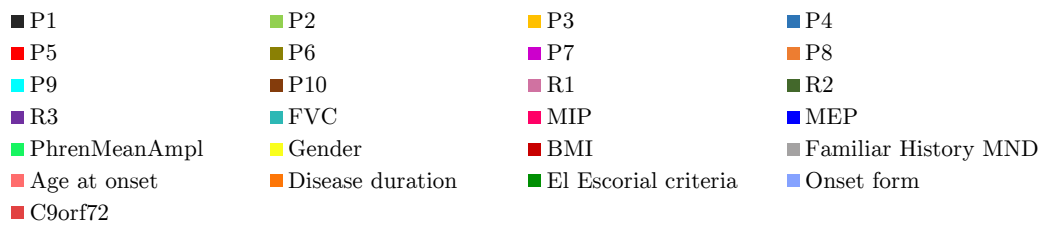


Figure 7.4: Influence of each variable in every timestep of the sdtDBNs before NIV with questions, for each progression group. The sdtDBNs detailed in Table 7.8 are learned for every progression group.

Fig. 7.4 presents the influence of each variable in every timestep of the learned sdtDBNs before NIV with questions, for each progression group. The results of the remaining scenarios (see Section 7.3), for each progression group, are provided in Appendix A.2.2.

The results of Fig. 7.4 confirm the relevance of dividing the ALS patients into progression groups, as the most influential variables throughout the disease progression vary according to the progression group of a patient. P9 has high influence in slow progressors and P1 has high influence in fast progressors, while in average progressors both P1 and P9 have significant influence. Regarding the respiratory tests, both MIP and MEP influence the evolution of slow progressors and MEP has high influence in average progressors, whereas, in fast progressors, no respiratory test has a substantial influence (only MEP has a small influence in the initial timesteps). Concerning the static variables, the slow progressors are mostly influenced by the disease duration, the average progressors by the BMI and the disease duration, while the fast progressors are mainly influenced by the BMI and the age at onset.

7.3.3 Correlations among variables throughout the disease progression

The correlations among variables throughout all timesteps of the learned sdtDBNs are obtained by applying Section 7.2.3 to each progression group. The results obtained using the sdtDBNs with sub-scores before and after NIV, for every progression group, are presented, respectively, in Tables 7.15 and 7.16. The results for the sdtDBNs using questions (before and after NIV), for each progression group, are provided in Appendix A.2.3.

Table 7.15: Correlations between variables of the sdtDBNs before NIV with sub-scores, for each progression group. Each table is normalized per column. The sdtDBNs of Table 7.8 are learned for every progression group.

(a) Slow progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R	FVC	MIP	MEP	PhrenMeanAmpl
ALS-FRS	—	9,75%	18,32%	18,34%	11,78%	9,20%	4,87%	9,50%	5,72%
ALS-FRSb	9,19%	—	11,32%	12,84%	8,46%	6,44%	9,06%	12,74%	17,51%
ALS-FRSsUL	26,47%	17,35%	—	20,90%	22,05%	17,18%	17,95%	14,53%	5,39%
ALS-FRSsLL	27,57%	20,47%	21,76%	—	16,01%	18,10%	15,77%	16,98%	11,45%
R	7,17%	5,46%	9,29%	6,48%	—	10,43%	7,89%	6,15%	0,67%
FVC	5,51%	4,09%	7,12%	7,21%	10,27%	—	2,18%	11,28%	4,04%
MIP	5,33%	10,53%	13,61%	11,49%	14,20%	3,99%	—	19,33%	26,60%
MEP	15,63%	22,22%	16,54%	18,58%	16,62%	30,98%	29,03%	—	28,62%
PhrenMeanAmpl	3,13%	10,14%	2,04%	4,16%	0,60%	3,68%	13,26%	9,50%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%

(b) Average progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R	FVC	MIP	MEP	PhrenMeanAmpl
ALS-FRS	—	11,13%	17,44%	15,56%	9,63%	7,33%	8,26%	7,87%	13,74%
ALS-FRSb	15,09%	—	16,64%	16,60%	23,26%	17,00%	20,54%	15,62%	11,45%
ALS-FRSsUL	25,34%	17,83%	—	17,25%	14,29%	18,00%	16,07%	19,07%	14,50%
ALS-FRSsLL	23,21%	18,26%	17,71%	—	16,61%	18,67%	14,96%	19,56%	22,14%
R	5,61%	9,99%	5,73%	6,49%	—	2,67%	6,25%	7,01%	6,11%
FVC	4,26%	7,28%	7,19%	7,26%	2,66%	—	3,13%	9,35%	7,25%
MIP	7,16%	13,12%	9,59%	8,69%	9,30%	4,67%	—	15,25%	5,34%
MEP	12,38%	18,12%	20,64%	20,62%	18,94%	25,33%	27,68%	—	19,47%
PhrenMeanAmpl	6,96%	4,28%	5,06%	7,52%	5,32%	6,33%	3,13%	6,27%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%

(c) Fast progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R	FVC	MIP	MEP	PhrenMeanAmpl
ALS-FRS	—	31,76%	33,05%	32,82%	37,31%	32,13%	26,09%	18,42%	40,94%
ALS-FRSb	20,56%	—	15,80%	21,78%	17,10%	16,06%	12,04%	20,10%	8,19%
ALS-FRSsUL	15,25%	11,27%	—	15,34%	6,74%	9,24%	9,03%	12,92%	6,43%
ALS-FRSsLL	14,19%	14,55%	14,37%	—	9,33%	3,21%	11,71%	5,50%	8,19%
R	9,55%	6,76%	3,74%	5,52%	—	5,22%	4,68%	6,22%	2,34%
FVC	10,61%	8,20%	6,61%	2,45%	6,74%	—	9,70%	10,29%	7,60%
MIP	10,34%	7,38%	7,76%	10,74%	7,25%	11,65%	—	17,46%	4,09%
MEP	10,21%	17,21%	15,52%	7,06%	13,47%	17,27%	24,41%	—	22,22%
PhrenMeanAmpl	9,28%	2,87%	3,16%	4,29%	2,07%	5,22%	2,34%	9,09%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table 7.16: Correlations between variables of the sdtDBNs after NIV with sub-scores, for each progression group. Each table is normalized per column. The sdtDBNs of Table 7.8 are learned for every progression group.

(a) Slow progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R
ALS-FRS	—	27,96%	25,27%	25,00%	17,70%
ALS-FRSb	36,88%	—	34,34%	36,18%	33,43%
ALS-FRSsUL	24,38%	25,12%	—	20,73%	23,88%
ALS-FRSsLL	25,63%	28,12%	22,03%	—	25,00%
R	13,13%	18,80%	18,36%	18,09%	—
Sum per column:	100%	100%	100%	100%	100%

(b) Average progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R
ALS-FRS	—	28,26%	26,90%	14,96%	12,27%
ALS-FRSb	39,79%	—	28,71%	29,92%	37,96%
ALS-FRSsUL	33,61%	25,48%	—	32,87%	23,61%
ALS-FRSsLL	15,67%	22,25%	27,56%	—	26,16%
R	10,93%	24,01%	16,83%	22,24%	—
Sum per column:	100%	100%	100%	100%	100%

(c) Fast progression group.

	ALS-FRS	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	R
ALS-FRS	—	40,90%	48,73%	43,77%	25,91%
ALS-FRSb	38,74%	—	27,97%	27,76%	53,44%
ALS-FRSsUL	23,33%	14,13%	—	14,95%	5,26%
ALS-FRSsLL	24,95%	16,70%	17,80%	—	15,38%
R	12,98%	28,27%	5,51%	13,52%	—
Sum per column:	100%	100%	100%	100%	100%

Tables 7.15 and 7.16 demonstrate that the variables are differently correlated in the several progression groups. Before NIV, the R columns of Table 7.15 show that the R score is mostly correlated with ALS-FRSsUL in slow progressors, with ALS-FRSb in average progressors and with ALS-FRS in fast progressors. This diversity of correlations of the respiratory sub-score can help distinguish the several kinds of patients. Also before NIV, the MEP rows of Table 7.15 display that the correlation of the several variables with MEP is higher in slow and average progressors than in fast progressors, which matches the conclusions of Section 7.3.2. After NIV, the ALS-FRSb rows of Table 7.16 show that ALS-FRSb has great correlation with all variables in every progression group. In particular, ALS-FRSb is the variable most correlated with the R score after NIV in all progression groups.

7.4 Analysis of some clinically relevant questions

The results obtained in Sections 7.2 and 7.3 (with the preprocessing from Section 7.1) are extremely interesting from a statistical perspective. Since the results can be easily understood by non-experts in data mining, they were presented to ALS doctors, who confirmed that several relations found by the sdtDBNs were expected from a clinical standpoint. The ALS doctors were also interested in knowing the rationale behind some other relations found by the sdtDBNs, which they did not expect the sdtDBNs to learn.

However, for the results to be useful from a clinical perspective, they must answer some well-defined clinical questions. Therefore, a study using sdtDBNs was done, with the goal of answering three clinical questions, proposed by ALS experts. The questions and their respective answers, obtained using sdtDBNs, are presented in the next paragraphs. In the answers of all questions, patients are divided into three progression groups (slow, average and fast), as explained in Section 7.3.

Question 1: which variables are the most associated to a patient needing NIV?

To answer the first question proposed by doctors, there are learned sdtDBNs to find which variables from the ALS dataset are the most correlated with non-invasive ventilation (NIV). As notation, the moment in which a patient gets NIV is associated to a certain year of that patient's progression, since the first consultation. For example, if the first consultation of a patient is in January 2000, that patient has NIV in the first year if NIV is given in the year 2000, has NIV in the second year if NIV is given in the year 2001, and other moments are represented accordingly.

Using the previous notation, patients are divided into the following six subsets (discussed with the ALS doctors): **(i)** slow progressors with NIV in any moment; **(ii)** average progressors with NIV in any moment; **(iii)** average progressors with NIV in the second year; **(iv)** average progressors with NIV after the second year; **(v)** fast progressors with NIV in the first year; **(vi)** fast progressors with NIV in the second year.

To determine how the variables of the ALS dataset are associated to NIV, it is added to the dataset a dynamic variable named NIV. In each consultation of a certain patient, this variable is 1 if the patient already has NIV, and is 0 otherwise. The dataset is then split into six sub-datasets, each one with data of one of the six aforementioned subsets of patients. For each sub-dataset, it is done a preprocessing similar to the one from Section 7.1, with two changes. First, only the data before NIV is used, as the goal is to study the patients' progression until NIV being applied. Second, it is only used the filling with LOCF, as the results with LOCF and with interpolation are similar.

For each subset of patients, the learned sdtDBNs are the ones from Table 7.8, with two differences. First, only the scenarios before NIV exist. Second, the sdtDBNs are learned using $T = 10$, as some subsets have patients with NIV only after the second year, and consecutive timesteps are separated by three months (see Section 7.1).

Having the sdtDBNs learned, an ordering of the influence of the variables in a patient's need of getting NIV can be obtained, by counting the number of edges that each variable has with the NIV variable (see the graphical reasoning used in Section 7.2.3). These counts are only done in the timesteps where the patients of the proper subsets may have NIV (for example, if a subset only has patients with NIV in the second year, only the timesteps in the second year contribute to the counts). The top 5 variables associated to NIV in the datasets with sub-scores are presented in Table 7.17, while the results of the datasets with questions are provided in Appendix A.3.

Table 7.17: Top 5 variables associated to NIV, in the datasets with sub-scores, for each subset of patients.

Progression group		**Slow**	**Average**			**Fast**	
NIV application		**Any time**	**Any time**	**2nd year**	**After 2nd year**	**1st year**	**2nd year**
Timesteps considered for correlations with NIV variable		{0,...,10}	{0,...,10}	{4,...,8}	{8,9,10}	{0,...,4}	{4,...,8}
Top 5 variables correlated with NIV variable	1st	MEP	BMI	ALS-FRS	ALS-FRS	ALS-FRS	ALS-FRS
	2nd	Disease duration	ALS-FRSb	ALS-FRSb	Disease duration	BMI	Age at onset
	3rd	BMI	MEP	BMI	ALS-FRSsUL	MIP	BMI
	4th	ALS-FRS	ALS-FRSsLL	MEP ALS-FRSsUL Disease duration	Gender ALS-FRSb BMI	MEP Age at onset	ALS-FRSsUL Gender
	5th	ALS-FRSb	Disease duration				

Table 7.17 shows that ALS-FRS and BMI influence NIV in most considered subsets of patients (in Table 7.17, ALS-FRS only does not appear in average progressors with NIV any time). The influence of ALS-FRS was expected, as it is the global functional score. In slow and average progressors, NIV is also strongly influenced by ALS-FRSb, MEP and disease duration (except in average progressors with NIV only after the second year, where NIV is not strongly influenced by MEP). In fast progressors, NIV is also greatly influenced by the age at onset.

Question 2: which variables are the most important in each year of the patients' progression?

The years of a certain patient's progression start being counted in the first consultation of that patient. Using the progression groups explained in Section 7.3, doctors are interested in determining the most influential variables in the first and second years of the progression of slow, average and fast progressors, and the most influential variables after the second year of the progression of slow and average progressors.

To answer the second clinical question, Sections 7.2.2 and 7.3.2 are employed, with some changes. For every progression group, the preprocessing is done as in Section 7.1. However, only the data before NIV is used (the goal is to study the patients' progression until NIV being applied), and data is only filled with LOCF (as in Sections 7.2.2 and 7.3.2). Then, the sdtDBNs from Table 7.8 are learned, with two differences. First, only the scenarios before NIV exist. Second, the sdtDBNs are learned using $T = 10$, as some results regard the influence of the variables after the second year of progression, and consecutive timesteps are separated by three months (see Section 7.1).

With the sdtDBNs learned, the influence of each variable in each year is determined by extending the graphical reasoning explained in Section 7.2.2 in the following way. Given any two variables X and Y of an sdtDBN, if X is a parent of more variables in timesteps of year t' than Y, it can be concluded that X has more influence than Y in year t' of the progression of the ALS patients that the sdtDBN describes. Given this reasoning, Table 7.9 is changed so that the rows have years instead of timesteps, which is done by summing element-wise the rows of the timesteps respective to each year. As consecutive timesteps are separated by three months, timesteps 1 to 4 refer to the first year, timesteps 5 to 8 refer to the second year, and timesteps 9 and 10 refer to years after the second year.

The influence of the several variables in each year is obtained, for each progression group, as detailed in Section 7.2.2, but creating tables similar to the changed table previously explained, instead of similar to Table 7.9. The top 10 most important variables in each year of each progression group using the sdtDBNs with sub-scores are presented in Table 7.18, while the results of the sdtDBNs with questions are provided in Appendix A.3.

Table 7.18: Top 10 most important variables in each year of the patients' progression, using the sdtDBNs with sub-scores. Patients are divided into slow, average and fast progressors (see Section 7.3).

Year		**1st year**			**2nd year**			**After 2nd year**	
Progression group		**Slow**	**Average**	**Fast**	**Slow**	**Average**	**Fast**	**Slow**	**Average**
Top 10 influential variables	1st	Disease duration	MEP	BMI	Disease duration	Disease duration	ALS-FRS	Disease duration	ALS-FRS
	2nd	MEP	ALS-FRSsLL BMI	MEP	MEP	ALS-FRSsUL	BMI	ALS-FRSsUL	Disease duration
	3rd	ALS-FRSsUL		ALS-FRS	ALS-FRSsLL	ALS-FRSb	Gender	ALS-FRSsLL	ALS-FRSb
	4th	ALS-FRSsLL	ALS-FRSsUL	ALS-FRSb	ALS-FRSsUL	ALS-FRSsLL	ALS-FRSb	ALS-FRS	ALS-FRSsUL
	5th	MIP	Disease duration	ALS-FRSsUL	ALS-FRSb	MEP	Disease duration	ALS-FRSb MEP	El Escorial criteria
	6th	BMI	ALS-FRSb	Age at onset	BMI	BMI	Age at onset		ALS-FRSsLL
	7th	ALS-FRS	El Escorial criteria	ALS-FRSsLL	MIP	ALS-FRS	El Escorial criteria	BMI	MEP
	8th	El Escorial criteria	MIP	MIP	ALS-FRS	MIP	MEP	MIP	BMI
	9th	ALS-FRSb	ALS-FRS	FVC	El Escorial criteria	El Escorial criteria	ALS-FRSsUL	El Escorial criteria	MIP
	10th	Age at onset	FVC	El Escorial criteria	Age at onset	C9orf72	Onset form	Age at onset	C9orf72

Table 7.18 allows extracting the major differences between the most influential variables in each progression group. In slow and average progressors, the ALS-FRSsUL and ALS-FRSsLL sub-scores have great influence, while in fast progressors it is ALS-FRSb the sub-score with the highest influence. MEP is the most important respiratory test in the several progression groups. Regarding the static variables, the disease duration has great influence in slow and average progressors, while in fast progressors the age at onset is a relevant indicator.

Question 3: considering the sub-score of ALS-FRS with the lowest value in the first consultation of a patient, how are its relations with the remaining variables, throughout that patient's progression?

The ALS-FRS sub-score with the lowest value in the first consultation of a patient reveals the major functional limitations of that patient at the start of the progression, being doctors interested in learning how those limitations affect the patient's progression. To achieve this goal, nine sub-datasets are created, by dividing patients into three progression groups (see Section 7.3) and, in each group, diving patients into three sub-groups, according to the sub-score with the lowest value in the first consultation being ALS-FRSb, ALS-FRSsUL or ALS-FRSsLL.

After getting all nine aforementioned sub-datasets, each one is preprocessed as explained in Section 7.1. However, only the datasets before NIV with sub-scores are obtained, because the goal is to study the relations of certain sub-scores in the patients' progression before NIV being applied. The data is also filled only using LOCF, as the results using LOCF and using interpolation are similar. Finally, the ALS-FRS variable is removed from the datasets, because correlations with the global score are not interesting in the context of this clinical question.

With the preprocessing completed, Section 7.2.3 is applied to every sub-dataset, only considering the scenario before NIV with sub-scores. From the analysis of a certain sub-dataset, a table similar to Table 7.11 is obtained, from which it is extracted only the column of the sub-score with the lowest value in the first consultation (this sub-score varies according to the sub-dataset). Considering all sub-datasets, nine columns are obtained (one per sub-dataset), which answer the third clinical question. Fig. 7.5 graphically presents (in a stacked bar chart) the nine mentioned columns, grouping the columns by progression group. In Appendix A.3, the same nine columns are presented, but the columns are grouped according to each sub-score with the lowest value in the first consultation.

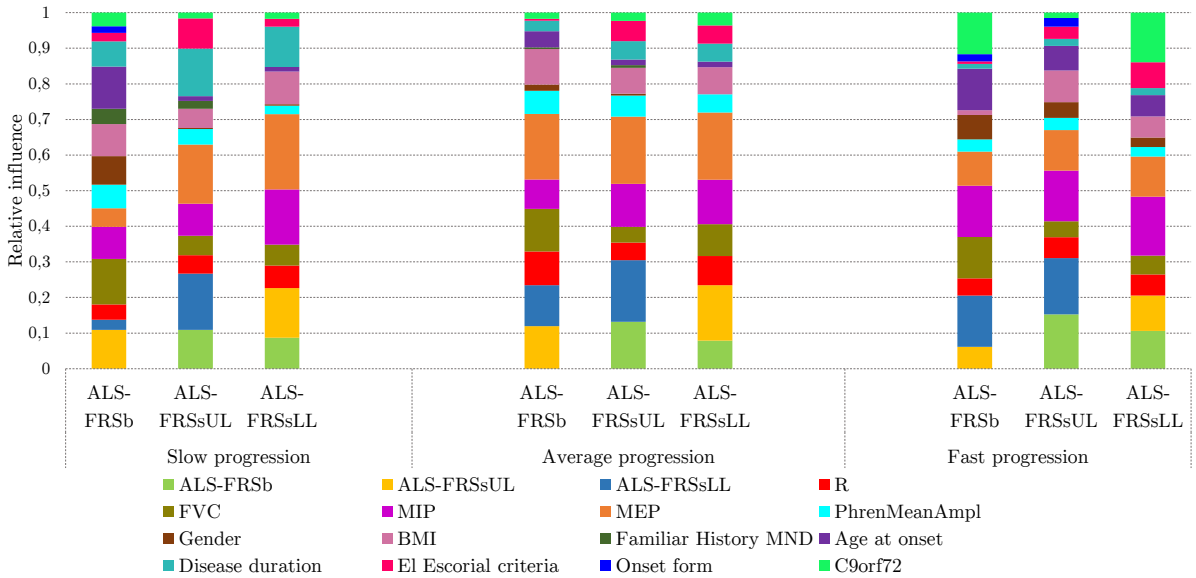


Figure 7.5: Correlations of the sub-score with the lowest value in the first consultation with the remaining variables throughout the patients' progression, grouping the columns by progression group.

Some relevant observations regarding Fig. 7.5 are given next. ALS-FRSb has high correlation with FVC, in all progression groups, and with the age at onset, in slow and fast progressors. ALS-FRSsUL is highly correlated with the El Escorial criteria, in slow progressors. ALS-FRSsLL has high correlation with MIP, in all groups. ALS-FRSb and ALS-FRSsLL are highly correlated with C9orf72, in fast progressors. ALS-FRSsUL and ALS-FRSsLL have high correlation with MEP, in slow and average progressors, and with the disease duration, in slow progressors.

Chapter 8

Conclusions and future work

8.1 Achievements and conclusions

This Thesis tackles ALS disease progression using data mining techniques. For this purpose, it is essential to have a machine learning model that can include static variables and the temporal evolution of dynamic variables.

The sdtDBN framework is proposed (see Chapter 5), as it optimally learns the structure and parameters of DBNs with dynamic and static variables. The learning algorithm has polynomial-time complexity in the number of dynamic and static variables. The sdtDBNs can also include prior knowledge in the relations among variables and estimate the distributions of unobserved variables. All capabilities of the sdtDBNs are implemented and publicly available. The inclusion of static variables in the sdtDBNs is a huge improvement on standard DBNs, which only have dynamic variables. This is particularly relevant when studying medical data, as most medical datasets have static information (such as the gender of the patients). The inclusion of prior knowledge is also essential when using sdtDBNs in the medical field, to properly incorporate clinicians' domain knowledge in the models.

A graphical interface for the sdtDBN framework is also presented (see Chapter 6), being implemented and publicly available. Having a graphical interface is extremely important, as it narrows the gap between machine learning models and clinical practice, by making the sdtDBNs easily available to any non-expert in machine learning.

A study of the Portuguese ALS dataset is done using sdtDBNs (see Chapter 7). The whole ALS dataset is tackled (see Section 7.2) with three approaches. The first approach predicts the values of the ALS-FRS-R sub-scores and questions, being the results competitive with state-of-the-art works. The second and third approaches use the graphical display of sdtDBNs to get the importance and correlations of the variables in the disease progression, providing relevant statistical insights regarding the ALS data. These graphical approaches are particularly relevant in the medical field, where presenting statistical information in a way doctors can understand (and use as a tool when treating current and future patients) is often more useful than creating a model with an extremely complex background (which doctors cannot understand), even if that model can accurately predict multiple outcomes.

After studying the whole dataset, the ALS patients are divided into three progression groups and each group is analyzed with the same three approaches used for the whole dataset (see Section 7.3). This study determines the main characteristics of each progression group, in particular with the approaches that use the graphical display of sdtDBNs. However, the reduced quantity of patients may lead the sdtDBNs to overfit the data in certain situations.

The last assessment of the ALS dataset using sdtDBNs consists of answering three specific clinical questions using sdtDBNs (see Section 7.4). This analysis allows the study using sdtDBNs to have real clinical impact.

The work of this Thesis resulted in three papers. The first paper, submitted in the Journal of Biomedical Informatics, focuses on the sdtDBN learning methodology, providing a case-study in ALS (in which some results of Section 7.2 are used). The second paper, finished and waiting for the supervisors' review in order to be submitted in the journal BMC Medical Informatics and Decision Making, emphasizes the study of the several progression groups of ALS patients using sdtDBNs (presenting the results of Section 7.3, with more detailed conclusions). The third paper is being written by the medical team that maintains the Portuguese ALS dataset, and focuses on the clinical relevance of the study done with sdtDBNs (addressing the three questions answered in Section 7.4).

8.2 Future work

Although the sdtDBN framework restricts the intra-slice connectivity to tree structures, this restriction may not fit some datasets. Other restrictions could be tried, creating a global framework that would test several intra-slice restrictions and choose the best. The maximization of scores different from LL and MDL could also be considered. Regarding the implementation of sdtDBNs, most loops of the learning procedure (see Algorithms 5.1, 5.2 and 5.3) could be parallelized. This is also suggested in the tDBN work [69], which the sdtDBN framework generalizes.

The sdtDBN framework cannot be employed to estimate the values of static variables, as they never have parents in learned sdtDBNs. It would be extremely beneficial to extend the sdtDBN framework, for sdtDBNs to learn the optimal parents of static variables. However, this is a difficult task, as considering all possible combinations of dynamic nodes from all timesteps is unfeasible, and keeping the acyclicity of the graph may not be easy.

The graphical interface of sdtDBNs is a program that runs locally in the user's machine. Other approaches are possible, such as developing a web-based interface that learns the sdtDBNs in a remote machine.

In the analysis of the ALS dataset, data is filled using LOCF and linear interpolation. It would be interesting to use sdtDBNs to fill missing data, by learning sdtDBNs from data before the filling process, and estimating missing values according to the respective (estimated) probability distributions, obtained using the mentioned sdtDBNs.

The division of patients into progression groups is done using their progression rates, determined with Eq. (7.3). An alternative approach would be to consider the progression group of a patient as a static variable, which could be predicted using sdtDBNs (if implementing the aforementioned extension to sdtDBNs, for static variables to have parents). These sdtDBNs could be learned from patients whose time-series are already ended, determining their progression groups, for example, by defining thresholds regarding the lengths of the time-series.

To get the correlations between the variables of the ALS dataset, the sdtDBNs are restricted, so that a variable cannot be a parent of itself. As, in the obtained results, an ALS-FRS-R question often has high correlation with questions from the same sub-score, it would be interesting to also prevent a question from being a parent of questions from the same sub-score, to assess the correlations among questions from different sub-scores.

Finally, there are interesting outcomes not covered in this work. For example, the learned sdtDBNs could be restricted, forcing the C9orf72 variable to always be a parent of every variable, in order to check how helpful the C9orf72 hexanucleotide repeat expansion can be to predict the disease progression. The sdtDBNs could also be used to predict a patient's necessity of getting NIV, and to predict a patient's survival time after NIV being applied.

Bibliography

[1] M. A. Van Es, O. Hardiman, A. Chiò, A. Al-Chalabi, R. J. Pasterkamp, J. H. Veldink, and L. H. Van den Berg. Amyotrophic lateral sclerosis. *The Lancet*, 390(10107):2084–2098, 2017.

[2] S. Martin, A. Al Khleifat, and A. Al-Chalabi. What causes amyotrophic lateral sclerosis? *F1000Research*, 6, 2017.

[3] C. Heffernan, C. Jenkinson, T. Holmes, H. Macleod, W. Kinnear, D. Oliver, N. Leigh, and M. Ampong. Management of respiration in MND/ALS patients: An evidence based review. *Amyotrophic Lateral Sclerosis*, 7(1):5–15, 2006.

[4] R. H. Brown and A. Al-Chalabi. Amyotrophic lateral sclerosis. *New England Journal of Medicine*, 377(2):162–172, 2017.

[5] P. Andersen, S. Abrahams, G. Borasio, M. de Carvalho, A. Chiò, P. Van Damme, O. Hardiman, K. Kollewe, K. E. Morrison, S. Petri, P.-F. Pradat, V. Silani, B. Tomik, M. Wasner, and M. Weber. EFNS guidelines on the clinical management of amyotrophic lateral sclerosis (MALS)–revised report of an EFNS task force. *European Journal of Neurology*, 19(3):360–375, 2012.

[6] S. Zarei, K. Carr, L. Reiley, K. Diaz, O. Guerra, P. F. Altamirano, W. Pagani, D. Lodin, G. Orozco, and A. Chinea. A comprehensive review of amyotrophic lateral sclerosis. *Surgical Neurology International*, 6, 2015.

[7] O. Hardiman, L. H. Van den Berg, and M. Kiernan. Clinical diagnosis and management of amyotrophic lateral sclerosis. *Nature Reviews Neurology*, 7(11):639–649, 2011.

[8] H. Creemers, H. Grupstra, F. Nollet, L. H. Van den Berg, and A. Beelen. Prognostic factors for the course of functional status of patients with ALS: A systematic review. *Journal of Neurology*, 262(6):1407–1423, 2015.

[9] W. Van Rheenen, S. L. Pulit, A. M. Dekker, A. Al Khleifat, W. J. Brands, A. Iacoangeli, K. P. Kenna, E. Kavak, M. Kooyman, R. L. McLaughlin, B. Middelkoop, M. Moisse, R. D. Schellevis, A. Shatunov, W. Sproviero, G. H. P. Tazelaar, R. A. A. Van der Spek, P. T. C. Van Doormaal, K. R. Van Eijk, J. Van Vugt, A. N. Basak, I. P. Blair, J. D. Glass, O. Hardiman, W. Hide, J. E. Landers, J. S. Mora, K. E. Morrison, S. Newhouse, W. Robberecht, C. E. Shaw, P. J. Shaw, P. Van Damme, M. A. Van Es, N. R. Wray, A. Al-Chalabi, L. H. Van den Berg, J. H. Veldink, and Project MinE ALS Sequencing Consortium. Project MinE: Study design and pilot analyses of a large-scale whole-genome sequencing study in amyotrophic lateral sclerosis. *European Journal of Human Genetics*, 26(10):1537–1546, 2018.

[10] Z. Zhangyu, C.-Y. Liu, C.-H. Che, and H.-P. Huang. Toward precision medicine in amyotrophic lateral sclerosis. *Annals of Translational Medicine*, 4(2), 2016.

[11] K. Kollewe, U. Mauss, K. Krampfl, S. Petri, R. Dengler, and B. Mohammadi. ALSFRS-R score and its ratio: A useful predictor for ALS progression. *Journal of the Neurological Sciences*, 275(1–2):69–73, 2008.

[12] F. Baumann, R. Henderson, S. Morrison, M. Brown, N. Hutchinson, J. Douglas, P. Robinson, and P. Mccombe. Use of respiratory function tests to predict survival in amyotrophic lateral sclerosis. *Amyotrophic Lateral Sclerosis*, 11(1–2):194–202, 2010.

[13] S. Bourke, M. Tomlinson, T. Williams, R. Bullock, P. J. Shaw, and G. Gibson. Effects of non-invasive ventilation on survival and quality of life in patients with amyotrophic lateral sclerosis: A randomised controlled trial. *The Lancet Neurology*, 5(2):140–147, 2006.

[14] K. J. Cios and G. William Moore. Uniqueness of medical data mining. *Artificial Intelligence in Medicine*, 26(1–2):1–24, 2002.

[15] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. ISBN 9780262013192.

[16] S. Russell, P. Norvig, and E. Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010. ISBN 9780136042594.

[17] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, California, USA, 2002.

[18] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 139–147. Morgan Kaufmann Publishers Inc., 1998.

[19] H. Lähdesmäki and I. Shmulevich. Learning the structure of dynamic Bayesian networks from time series and steady state measurements. *Machine Learning*, 71(2–3):185–217, 2008.

[20] X. V. Nguyen, M. Chetty, R. L. Coppel, and P. P. Wangikar. Polynomial time algorithm for learning globally optimal dynamic Bayesian network. In *Neural Information Processing - 18th International Conference*, pages 719–729. Springer, Berlin, Heidelberg, 2011.

[21] J. A. Bilmes. Dynamic Bayesian multinets. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pages 38–45. Morgan Kaufmann Publishers Inc., 2000.

[22] L. Song, M. Kolar, and E. P. Xing. Time-varying dynamic Bayesian networks. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, pages 1732–1740. Curran Associates Inc., 2009.

[23] J. W. Robinson and A. J. Hartemink. Learning non-stationary dynamic Bayesian networks. *Journal of Machine Learning Research*, 11: 3647–3680, 2010.

[24] G. Trabelsi, P. Leray, M. Ben Ayed, and A. Alimi. Dynamic MMHC: A local search algorithm for dynamic Bayesian network structure learning. In *Advances in Intelligent Data Analysis XII*, pages 392–403. Springer, Berlin, Heidelberg, 2013.

[25] J. L. Monteiro, S. Vinga, and A. M. Carvalho. Polynomial-time algorithm for learning optimal tree-augmented dynamic Bayesian networks. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI'15, pages 622–631. AUAI Press, 2015.

[26] M. Sousa and A. M. Carvalho. Polynomial-time algorithm for learning optimal BFS-consistent dynamic Bayesian networks. *Entropy*, 20(4), 2018.

[27] S. Pires, M. Gromicho, S. Pinto, M. de Carvalho, and S. C. Madeira. Predicting non-invasive ventilation in ALS patients using stratified disease progression groups. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 748–757. IEEE, 2018.

[28] R. Küffner, N. Zach, M. Bronfeld, R. Norel, N. Atassi, V. Balagurusamy, B. Di Camillo, A. Chiò, M. Cudkowicz, D. Dillenberger, J. García-García, O. Hardiman, B. Hoff, J. Knight, M. Leitner, G. Li, L. Mangravite, T. Norman, L. Wang, and A. Polewko-Klim. Stratification of amyotrophic lateral sclerosis patients: A crowdsourcing approach. *Scientific Reports*, 9, 2019.

[29] A. V. Carreiro, P. M. Amaral, S. Pinto, P. Tomás, M. de Carvalho, and S. C. Madeira. Prognostic models based on patient snapshots and time windows: Predicting disease progression to assisted ventilation in amyotrophic lateral sclerosis. *Journal of Biomedical Informatics*, 58:133–144, 2015.

[30] T. Pereira, S. Pires, M. Gromicho, S. Pinto, M. de Carvalho, and S. C. Madeira. Predicting assisted ventilation in amyotrophic lateral sclerosis using a mixture of experts and conformal predictors. In *Workshop on Applied Data Science for Healthcare, KDD, 2019*, 2019.

[31] S. Pires. A supervised learning approach for prognostic prediction in ALS using disease progression groups and patient profiles. Master's thesis, Universidade de Lisboa, Faculdade de Ciências, Lisbon, Portugal, 2018.

[32] A. V. Carreiro. *An integrative mining approach for prognostic prediction in neurodegenerative diseases*. PhD thesis, Universidade de Lisboa, Instituto Superior Técnico, Lisbon, Portugal, 2016.

[33] J. Matos. Biclustering electronic health records to unravel disease presentation patterns. Master's thesis, Universidade de Lisboa, Faculdade de Ciências, Lisbon, Portugal, 2019.

[34] A. Zandonà, R. Vasta, A. Chiò, and B. Di Camillo. A dynamic Bayesian network model for the simulation of amyotrophic lateral sclerosis progression. *BMC Bioinformatics*, 20(S4), 2019.

[35] K. Murphy and S. Mian. Modelling gene expression data using dynamic Bayesian networks. Technical report, University of California, Berkeley, California, USA, 1999.

[36] M. Zou and S. D. Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79, 2005.

[37] M. Van Gerven, B. Taal, and P. J. Lucas. Dynamic Bayesian networks as prognostic models for clinical patient management. *Journal of Biomedical Informatics*, 41(4):515–529, 2008.

[38] T. Charitos, L. C. Gaag, S. Visscher, K. Schurink, and P. J. Lucas. A dynamic Bayesian network for diagnosing ventilator-associated pneumonia in ICU patients. *Expert Systems with Applications: An International Journal*, 36(2):1249–1258, 2009.

[39] M. Proudfoot, A. Jones, K. Talbot, A. Al-Chalabi, and M. Turner. The ALSFRS as an outcome measure in therapeutic trials and its relationship to symptom onset. *Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration*, 17(5-6):414–425, 2016.

[40] N. Simon, M. Turner, S. Vucic, A. Al-Chalabi, J. Shefner, C. Lomen-Hoerth, and M. Kiernan. Quantifying disease progression in amyotrophic lateral sclerosis. *Annals of Neurology*, 76(5):643—657, 2014.

[41] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 3rd edition, 2011. ISBN 9780123814791.

[42] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 4th edition, 2016. ISBN 9780128042915.

[43] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi. *Discovering Data Mining: From Concept to Implementation*. Prentice Hall, 1998. ISBN 0137439806.

[44] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann Publishers Inc., 1999. ISBN 1558605290.

[45] R. Cook, L. Zeng, and G. Yi. Marginal analysis of incomplete longitudinal binary data: A cautionary note on LOCF imputation. *Biometrics*, 60(3):820–828, 2004.

[46] D. M. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980. ISBN 9780412219009.

[47] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, 2004.

[48] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

[49] Z. C. Lipton. The mythos of model interpretability. *ACM Queue*, 16(3):31—57, 2018.

[50] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 9780387310732.

[51] K. Murphy. An introduction to graphical models. Technical report, University of California, Berkeley, California, USA, 2001.

[52] D. Koller, N. Friedman, L. Getoor, and B. Taskar. Graphical models in a nutshell. In *Introduction to Statistical Relational Learning*, chapter 2, pages 13–55. MIT Press, 2007.

[53] E. Charniak. Bayesian networks without tears: Making Bayesian networks more accessible to the probabilistically unsophisticated. *AI Magazine*, 12(4):50–63, 1991.

[54] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks (research note). *Artificial Intelligence*, 42(2–3):393–405, 1990.

[55] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988. ISBN 0934613737.

[56] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[57] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4): 309–347, 1992.

[58] W. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'91, pages 52–60. Morgan Kaufmann Publishers Inc., 1991.

[59] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14(3):1080–1100, 1986.

[60] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10 (3):269–293, 1994.

[61] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[62] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[63] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[64] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B(4):233–240, 1967.

[65] N. Friedman. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 129–138. Morgan Kaufmann Publishers Inc., 1998.

[66] D. Heckerman. A tutorial on learning with Bayesian networks. In *Learning in Graphical Models*, pages 301—354. MIT Press, 1999.

[67] N. Dojer. Learning Bayesian networks does not have to be NP-hard. In *Proceedings of the 31st International Conference on Mathematical Foundations of Computer Science*, MFCS'06, pages 305–314. Springer, Berlin, Heidelberg, 2006.

[68] X. Boyen, N. Friedman, and D. Koller. Discovering the hidden structure of complex dynamic systems. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 91–100. Morgan Kaufmann Publishers Inc., 1999.

[69] J. L. Monteiro. Learning from short multivariate time series. Master's thesis, Universidade de Lisboa, Instituto Superior Técnico, Lisbon, Portugal, 2014.

[70] A. Taylor, C. Fournier, M. Polak, L. Wang, N. Zach, M. Keymer, J. D. Glass, and D. Ennist. Predicting disease progression in amyotrophic lateral sclerosis. *Annals of Clinical and Translational Neurology*, 3(11):866–875, 2016.

[71] T. Hothorn and H. H. Jung. Randomforest4life: A random forest for predicting ALS disease progression. *Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration*, 15(5–6):444–452, 2014.

Appendix A

Some more results of the analysis of the ALS dataset using sdtDBNs

A.1 Whole dataset

A.1.1 Prediction of the variables' values

Table A.1: Results of the predictions, for every question in every timestep, of the sdtDBNs before NIV with questions (see Section 7.2 in Chapter 7), using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	87,98	80,32	77,76	85,76	80,13	84,49	88,51	88,03
	Sensitivity (%)	93,16	89,90	80,91	87,92	76,23	91,49	78,95	90,36
	AUC (%)	95,89	94,74	89,61	93,96	87,83	95,74	89,47	94,52
	Evaluated patients	1090	884	706	590	468	374	296	234
P2	Accuracy (%)	83,78	78,47	80,93	81,34	78,11	75,43	81,88	84,10
	Sensitivity (%)	80,80	79,24	74,78	79,81	77,16	80,65	83,33	76,27
	AUC (%)	88,96	87,15	85,69	88,76	87,11	85,82	91,67	84,72
	Evaluated patients	1042	836	666	552	434	346	276	220
P3	Accuracy (%)	87,52	84,28	82,44	84,58	83,12	81,28	86,49	93,16
	Sensitivity (%)	90,40	83,83	81,63	85,96	75,60	90,61	88,35	85,91
	AUC (%)	93,72	90,84	89,00	91,01	87,41	92,64	92,62	92,96
	Evaluated patients	1090	884	706	590	468	374	296	234
P4	Accuracy (%)	82,87	76,32	76,72	78,72	76,46	78,27	76,65	82,92
	Sensitivity (%)	86,07	85,24	87,61	75,48	70,55	81,00	78,41	73,68
	AUC (%)	90,95	89,79	90,49	86,64	84,38	89,73	87,35	86,23
	Evaluated patients	1138	912	730	606	480	382	304	240
P5	Accuracy (%)	78,03	71,60	74,16	71,96	70,25	76,14	73,39	80,15
	Sensitivity (%)	69,58	71,20	71,27	70,47	72,68	82,18	71,74	82,58
	AUC (%)	83,78	84,40	84,65	84,37	85,50	90,03	84,23	89,87
	Evaluated patients	1066	860	686	571	451	360	286	227
P6	Accuracy (%)	76,15	68,44	67,29	70,68	65,81	67,65	74,32	75,64
	Sensitivity (%)	76,24	71,44	73,02	69,21	74,29	71,64	75,47	74,42
	AUC (%)	86,41	83,04	83,83	83,44	86,53	84,99	86,16	85,18
	Evaluated patients	1090	884	706	590	468	374	296	234
P7	Accuracy (%)	77,74	68,18	67,57	71,38	66,36	72,83	70,65	65,91
	Sensitivity (%)	83,39	61,15	69,86	70,65	68,16	80,13	80,30	80,39
	AUC (%)	89,54	79,64	83,59	84,19	82,32	89,54	89,46	87,65
	Evaluated patients	1042	836	666	552	434	346	276	220
P8	Accuracy (%)	82,99	82,94	77,65	80,78	77,46	76,81	76,45	71,55
	Sensitivity (%)	86,44	89,39	89,08	91,75	94,05	93,12	97,11	94,33
	AUC (%)	90,53	91,71	90,66	92,34	91,94	92,44	93,09	92,41
	Evaluated patients	1146	920	738	614	488	388	310	246
P9	Accuracy (%)	74,09	66,87	65,32	72,47	67,97	70,23	71,38	75,00
	Sensitivity (%)	62,17	63,81	64,46	73,40	77,83	61,22	75,00	79,39
	AUC (%)	80,82	80,44	80,70	85,50	88,12	80,61	86,48	89,69
	Evaluated patients	1042	836	666	552	434	346	276	220
P10	Accuracy (%)	92,23	92,38	93,22	90,87	89,94	89,41	91,91	95,92
	Sensitivity (%)	93,60	91,38	x	x	x	x	x	x
	AUC (%)	94,15	93,95	x	x	x	x	x	x
	Evaluated patients	1145	919	737	613	487	387	309	245
R1	Accuracy (%)	85,11	79,64	81,51	81,49	82,45	83,48	84,00	87,67
	Sensitivity (%)	71,60	55,79	59,52	x	x	x	x	x
	AUC (%)	83,14	73,95	76,75	x	x	x	x	x
	Evaluated patients	1041	835	665	551	433	345	275	219
R2	Accuracy (%)	92,13	92,35	92,04	92,57	91,71	92,49	94,20	94,55
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	1042	836	666	552	434	346	276	220
R3	Accuracy (%)	97,45	97,04	96,85	94,06	94,79	95,29	94,41	97,09
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	1138	912	730	606	480	382	304	240

Table A.2: Results of the predictions, for every question in every timestep, of the sdtDBNs after NIV with questions (see Section 7.2 in Chapter 7), using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	81,06	79,40	77,70	82,06	83,43	82,42	82,09	82,00
	Sensitivity (%)	83,81	78,85	76,39	81,54	83,67	84,35	75,72	79,31
	AUC (%)	91,56	89,21	87,47	90,45	91,04	90,24	85,51	86,84
	Evaluated patients	792	670	556	446	350	290	240	200
P2	Accuracy (%)	77,63	70,66	70,41	75,36	75,15	71,22	71,79	77,78
	Sensitivity (%)	82,72	75,00	77,55	83,12	89,58	73,81	78,95	79,41
	AUC (%)	88,84	84,38	85,82	88,25	92,69	84,84	87,57	87,40
	Evaluated patients	760	644	534	426	334	278	234	198
P3	Accuracy (%)	82,33	78,37	78,87	74,03	79,84	81,46	83,34	83,81
	Sensitivity (%)	84,69	74,26	82,47	81,65	84,25	90,98	86,56	90,66
	AUC (%)	90,42	85,86	89,96	87,96	89,33	94,31	92,53	95,33
	Evaluated patients	826	698	582	462	362	302	252	210
P4	Accuracy (%)	78,52	72,26	68,99	74,34	71,19	75,34	65,07	85,17
	Sensitivity (%)	83,54	77,32	75,60	84,17	80,00	83,18	78,15	88,57
	AUC (%)	91,60	88,10	87,55	91,45	88,29	90,00	88,31	92,37
	Evaluated patients	810	685	571	452	354	296	249	209
P5	Accuracy (%)	76,45	65,38	71,54	71,13	70,96	78,78	78,63	83,34
	Sensitivity (%)	83,00	60,49	75,32	74,29	77,05	80,77	80,85	78,26
	AUC (%)	90,39	78,90	87,14	85,74	87,11	89,24	88,28	89,13
	Evaluated patients	760	644	534	426	334	278	234	198
P6	Accuracy (%)	78,29	68,94	70,04	69,95	74,25	79,50	76,07	84,35
	Sensitivity (%)	77,34	69,61	73,08	75,26	81,71	81,58	84,62	85,25
	AUC (%)	88,32	84,01	85,62	86,77	89,68	90,79	89,42	92,62
	Evaluated patients	760	644	534	426	334	278	234	198
P7	Accuracy (%)	76,71	70,97	69,10	67,85	66,17	72,30	76,92	81,82
	Sensitivity (%)	80,85	76,89	76,05	58,57	72,41	67,80	82,35	85,11
	AUC (%)	89,80	87,81	86,67	79,29	83,91	82,02	89,66	91,59
	Evaluated patients	760	644	534	426	334	278	234	198
P8	Accuracy (%)	80,87	75,65	77,49	69,48	72,38	76,49	74,21	78,10
	Sensitivity (%)	80,89	71,98	73,53	70,97	71,43	75,00	70,37	86,96
	AUC (%)	90,11	85,59	86,50	84,89	84,91	86,45	81,02	90,94
	Evaluated patients	826	698	582	462	362	302	252	210
P9	Accuracy (%)	74,61	70,32	72,11	69,96	76,03	79,23	80,19	89,95
	Sensitivity (%)	77,14	72,98	75,51	77,30	78,91	84,24	87,28	95,42
	AUC (%)	87,97	85,73	86,09	87,69	88,06	91,12	91,14	96,24
	Evaluated patients	776	657	545	436	342	284	237	199
P10	Accuracy (%)	82,49	80,71	79,16	80,88	81,77	84,50	72,48	84,74
	Sensitivity (%)	84,77	87,15	89,47	86,82	90,54	94,34	86,28	90,66
	AUC (%)	87,13	84,67	79,63	83,73	82,77	84,67	78,53	89,67
	Evaluated patients	777	658	547	434	340	284	240	203
R1	Accuracy (%)	73,95	71,90	71,72	64,09	67,37	75,54	68,38	69,70
	Sensitivity (%)	78,65	75,54	78,14	76,07	82,20	73,68	61,76	87,71
	AUC (%)	84,46	82,25	84,17	82,57	83,76	85,85	78,47	80,04
	Evaluated patients	760	644	534	426	334	278	234	198
R2	Accuracy (%)	77,02	73,73	73,04	70,14	64,74	80,00	72,77	79,81
	Sensitivity (%)	82,59	80,75	82,80	76,76	78,63	75,00	81,87	88,97
	AUC (%)	85,85	84,46	82,46	83,95	78,60	84,17	81,60	86,55
	Evaluated patients	792	670	560	442	346	290	246	208
R3	Accuracy (%)	81,36	76,64	80,90	80,09	80,92	86,56	84,96	87,02
	Sensitivity (%)	83,84	79,36	88,61	84,11	89,52	92,96	92,70	90,33
	AUC (%)	87,76	83,27	87,31	86,34	87,62	91,29	89,00	89,50
	Evaluated patients	794	672	560	442	346	290	246	208

A.1.2 Influence of each variable in every timestep

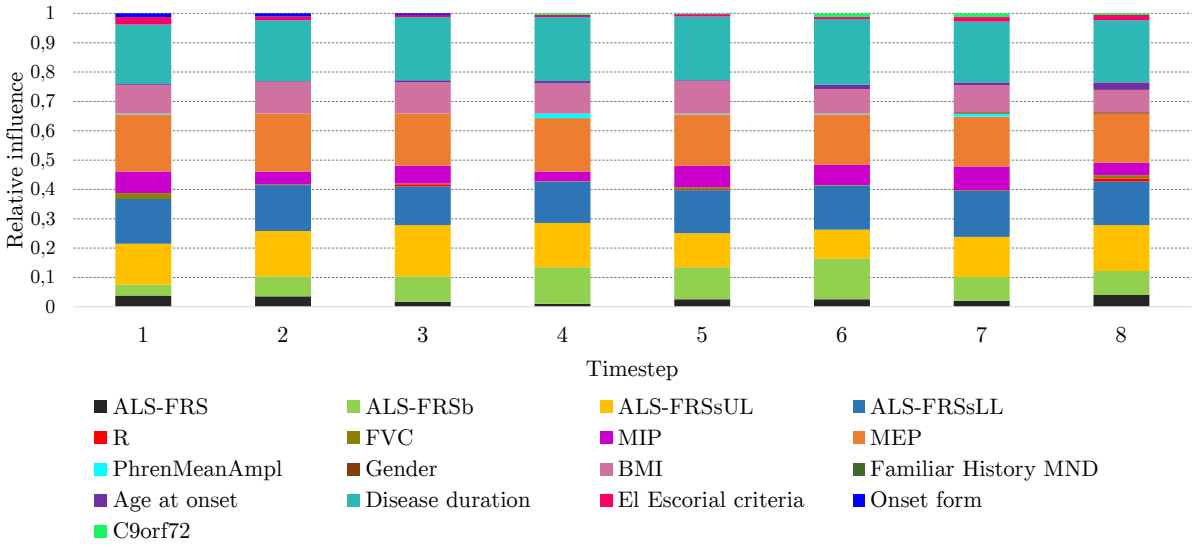


Figure A.1: Influence of each variable in every timestep of the sdtDBNs before NIV with sub-scores. The learned sdtDBNs are detailed in Table 7.8 of Chapter 7.

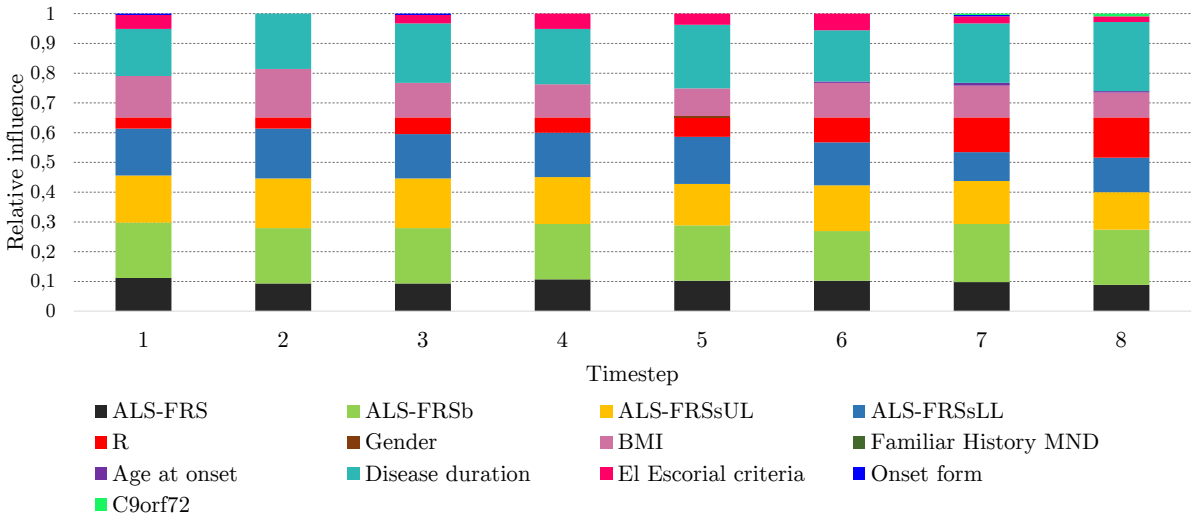


Figure A.2: Influence of each variable in every timestep of the sdtDBNs after NIV with sub-scores. The learned sdtDBNs are detailed in Table 7.8 of Chapter 7.

A.1.3 Correlations among variables throughout the disease progression

Table A.3: Correlations between variables of the sdtDBNs before NIV with questions. The table is normalized per column. The learned sdtDBNs are detailed in Table 7.8 of Chapter 7.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3	FVC	MIP	MEP	PhrenMeanAmpl
P1	—	34,85%	45,71%	5,73%	4,57%	4,36%	0,90%	7,81%	4,78%	5,69%	3,33%	13,73%	10,49%	7,69%	6,45%	5,39%	10,53%
P2	17,65%	—	5,14%	2,87%	2,67%	3,81%	0,90%	3,35%	3,59%	1,42%	0,83%	6,54%	2,10%	4,49%	0,92%	2,94%	2,63%
P3	20,46%	4,55%	—	1,79%	4,57%	2,18%	2,69%	0,37%	1,20%	2,37%	0,83%	3,27%	2,80%	1,28%	0,46%	2,12%	1,32%
P4	4,09%	4,04%	2,86%	—	15,05%	1,36%	8,07%	5,20%	6,58%	3,79%	1,25%	7,84%	4,90%	4,49%	5,53%	6,05%	2,63%
P5	6,14%	7,07%	13,71%	28,32%	—	21,53%	5,38%	7,81%	9,42%	6,64%	10,00%	15,69%	6,99%	19,23%	6,45%	11,11%	16,45%
P6	4,09%	7,07%	4,57%	1,79%	15,05%	—	20,18%	7,06%	10,16%	4,74%	2,08%	5,23%	4,20%	6,41%	7,37%	8,33%	4,61%
P7	0,51%	1,01%	3,43%	6,45%	2,29%	12,26%	—	3,72%	6,73%	3,79%	3,75%	1,96%	3,50%	2,56%	3,23%	7,19%	1,97%
P8	5,37%	4,55%	0,57%	5,02%	4,00%	5,18%	4,48%	—	14,80%	1,42%	3,33%	4,58%	3,50%	4,49%	5,07%	5,07%	1,97%
P9	8,18%	12,12%	4,57%	15,77%	12,00%	18,53%	20,18%	36,80%	—	10,90%	14,17%	16,99%	5,59%	18,59%	18,89%	14,38%	24,34%
P10	3,07%	1,52%	2,86%	2,87%	2,67%	2,72%	3,59%	1,12%	3,44%	—	26,67%	3,27%	20,98%	1,28%	0,46%	3,59%	0,66%
R1	2,05%	1,01%	1,14%	1,08%	4,57%	1,36%	4,04%	2,97%	5,08%	30,33%	—	5,88%	22,38%	1,28%	1,38%	5,07%	2,63%
R2	5,37%	5,05%	2,86%	4,30%	4,57%	2,18%	1,35%	2,60%	3,89%	2,37%	3,75%	—	2,10%	0,00%	0,46%	3,10%	0,00%
R3	3,84%	1,52%	2,29%	2,51%	1,90%	1,63%	2,24%	1,86%	1,20%	14,22%	13,33%	1,96%	—	1,28%	2,30%	1,14%	0,66%
FVC	3,07%	3,54%	1,14%	2,51%	5,71%	2,72%	1,79%	2,60%	4,33%	0,95%	0,83%	0,00%	1,40%	—	0,46%	6,54%	0,66%
MIP	3,58%	1,01%	0,57%	4,30%	2,67%	4,36%	3,14%	4,09%	6,13%	0,47%	1,25%	0,65%	3,50%	0,64%	—	12,58%	7,24%
MEP	8,44%	9,09%	7,43%	13,26%	12,95%	13,90%	19,73%	11,52%	13,15%	10,43%	12,92%	12,42%	4,90%	25,64%	35,48%	—	21,71%
PhrenMeanAmpl	4,09%	2,02%	1,14%	1,43%	4,76%	1,91%	1,35%	1,12%	5,53%	0,47%	1,67%	0,00%	0,70%	0,64%	5,07%	5,39%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table A.4: Correlations between variables of the sdtDBNs after NIV with questions. The table is normalized per column. The learned sdtDBNs are detailed in Table 7.8 of Chapter 7.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3
P1	—	17,51%	28,32%	13,35%	10,64%	10,48%	12,84%	11,78%	10,04%	5,88%	10,60%	10,95%	13,09%
P2	10,04%	—	10,03%	6,21%	3,55%	3,81%	2,75%	9,04%	5,24%	6,95%	8,76%	5,97%	12,04%
P3	21,43%	13,23%	—	5,90%	6,74%	4,76%	7,03%	5,48%	3,93%	9,09%	12,21%	13,93%	5,76%
P4	9,60%	7,78%	5,60%	—	24,11%	10,95%	13,76%	6,58%	9,17%	6,42%	5,76%	6,97%	4,19%
P5	6,70%	3,89%	5,60%	21,12%	—	20,95%	6,42%	4,93%	6,11%	5,35%	7,60%	5,97%	1,57%
P6	4,91%	3,11%	2,95%	7,14%	15,60%	—	13,15%	2,19%	11,79%	1,60%	3,92%	1,99%	0,52%
P7	9,38%	3,50%	6,78%	13,98%	7,45%	20,48%	—	13,70%	10,48%	6,95%	5,99%	7,46%	8,38%
P8	9,60%	12,84%	5,90%	7,45%	6,38%	3,81%	15,29%	—	31,44%	5,88%	11,75%	8,96%	8,90%
P9	5,13%	4,67%	2,65%	6,52%	4,96%	12,86%	7,34%	19,73%	—	3,21%	3,92%	1,49%	0,52%
P10	2,46%	5,06%	5,01%	3,73%	3,55%	1,43%	3,98%	3,01%	2,62%	—	8,06%	3,48%	25,65%
R1	10,27%	14,79%	15,63%	7,76%	11,70%	8,10%	7,95%	13,97%	7,42%	18,72%	—	30,35%	16,75%
R2	4,91%	4,67%	8,26%	4,35%	4,26%	1,90%	4,59%	4,93%	1,31%	3,74%	14,06%	—	2,62%
R3	5,58%	8,95%	3,24%	2,48%	1,06%	0,48%	4,89%	4,66%	0,44%	26,20%	7,37%	2,49%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

A.2 Progression groups

A.2.1 Prediction of the variables' values

Table A.5: Results of the predictions of the sdtDBNs before NIV with sub-scores, for the several progression groups (see Section 7.3.1 in Chapter 7), using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

(a) Slow progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ALS-FRS	Accuracy (%)	84,23	75,55	81,60	86,87	74,07	82,26	82,72	91,92
	Sensitivity (%)	72,28	67,11	77,99	88,70	88,16	89,86	91,94	100,00
	AUC (%)	83,76	79,46	86,80	94,35	83,14	94,93	93,34	97,50
	Evaluated patients	298	274	250	236	216	186	162	136
ALS-FRSb	Accuracy (%)	91,62	89,42	87,20	90,26	82,41	82,80	88,89	92,65
	Sensitivity (%)	83,33	84,52	74,65	81,95	68,29	83,78	90,62	100,00
	AUC (%)	90,03	89,80	86,21	90,37	81,91	90,11	95,31	97,00
	Evaluated patients	298	274	250	236	216	186	162	136
ALS-FRSsUL	Accuracy (%)	80,88	77,37	76,40	83,90	71,76	82,80	79,01	88,24
	Sensitivity (%)	82,58	85,56	87,30	91,54	54,81	82,76	64,29	80,77
	AUC (%)	86,71	89,59	92,20	93,64	76,01	90,60	81,20	89,19
	Evaluated patients	298	274	250	236	216	186	162	136
ALS-FRSsLL	Accuracy (%)	88,28	82,76	74,30	80,20	80,22	79,22	78,79	83,33
	Sensitivity (%)	92,13	90,11	42,27	76,00	82,86	78,79	86,67	80,00
	AUC (%)	95,43	91,05	70,43	85,64	90,54	87,12	91,94	88,28
	Evaluated patients	256	232	214	202	182	154	132	108
R	Accuracy (%)	88,59	83,58	90,00	86,02	87,04	88,18	85,19	89,71
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	298	274	250	236	216	186	162	136

(b) Average progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ALS-FRS	Accuracy (%)	81,50	75,45	78,09	81,16	76,77	76,76	76,00	73,68
	Sensitivity (%)	89,34	86,33	68,37	55,85	65,80	66,13	75,00	61,54
	AUC (%)	86,63	86,04	82,21	77,92	82,90	80,57	87,50	78,77
	Evaluated patients	562	444	356	276	198	142	100	76
ALS-FRSb	Accuracy (%)	87,19	78,38	77,53	75,00	71,72	75,36	86,00	85,53
	Sensitivity (%)	92,83	86,87	74,47	80,00	62,86	90,91	93,33	84,61
	AUC (%)	95,63	92,90	86,85	88,47	81,43	94,43	96,67	89,60
	Evaluated patients	562	444	356	276	198	142	100	76
ALS-FRSsUL	Accuracy (%)	80,79	71,17	79,78	72,10	77,78	76,06	81,00	73,68
	Sensitivity (%)	59,72	62,35	79,77	73,61	86,95	52,38	84,24	73,32
	AUC (%)	79,14	80,11	89,89	85,29	92,27	76,19	89,79	83,43
	Evaluated patients	562	444	356	276	198	142	100	76
ALS-FRSsLL	Accuracy (%)	79,00	71,17	75,00	73,55	68,69	72,54	77,00	75,00
	Sensitivity (%)	68,28	65,65	79,36	76,70	81,41	59,09	68,75	69,70
	AUC (%)	83,88	81,62	87,94	86,58	89,26	79,55	82,90	84,85
	Evaluated patients	562	444	356	276	198	142	100	76
R	Accuracy (%)	85,45	76,56	80,61	75,99	71,67	80,47	82,22	81,43
	Sensitivity (%)	71,88	54,90	58,77	57,18	48,00	47,06	x	36,67
	AUC (%)	82,74	71,31	76,15	72,70	66,31	70,34	x	68,34
	Evaluated patients	536	418	330	254	180	128	90	70

(c) Fast progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ALS-FRS	Accuracy (%)	78,37	69,45	67,39	81,82	72,73	73,53	82,14	87,50
	Sensitivity (%)	45,71	54,81	70,00	72,22	75,00	76,37	83,34	83,33
	AUC (%)	72,86	76,00	85,00	86,11	87,50	88,18	91,67	91,67
	Evaluated patients	208	144	92	66	44	34	28	16
ALS-FRSb	Accuracy (%)	83,18	75,00	71,74	68,19	86,36	82,35	78,57	81,25
	Sensitivity (%)	61,33	83,33	71,04	76,67	81,82	60,00	75,00	83,34
	AUC (%)	80,67	87,75	85,53	88,34	90,91	80,00	87,50	91,67
	Evaluated patients	208	144	92	66	44	34	28	16
ALS-FRSsUL	Accuracy (%)	66,35	61,81	68,48	68,19	81,82	67,65	85,71	75,00
	Sensitivity (%)	73,68	40,74	57,14	91,67	73,89	65,00	87,50	75,00
	AUC (%)	84,71	70,37	78,57	95,83	86,95	82,50	93,75	75,00
	Evaluated patients	208	144	92	66	44	34	28	16
ALS-FRSsLL	Accuracy (%)	59,62	58,34	41,31	72,73	72,73	79,41	75,00	87,50
	Sensitivity (%)	64,61	29,29	30,79	90,00	55,56	71,43	71,43	100,00
	AUC (%)	76,68	64,65	65,40	95,00	77,78	85,71	78,57	100,00
	Evaluated patients	208	144	92	66	44	34	28	16
R	Accuracy (%)	78,85	83,33	76,09	78,79	81,82	82,36	71,43	75,00
	Sensitivity (%)	65,91	70,97	65,00	62,50	75,53	80,00	66,67	80,00
	AUC (%)	77,12	81,83	76,73	78,31	82,71	90,00	73,33	73,33
	Evaluated patients	208	144	92	66	44	34	28	16

Table A.6: Results of the predictions of the sdtDBNs after NIV with sub-scores, for the several progression groups (see Section 7.3.1 in Chapter 7), using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

(a) Slow progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
ALS-FRS	Accuracy (%)	87,08	80,49	78,26	85,25	74,00	78,05	68,57	75,00
	Sensitivity (%)	88,05	83,33	87,80	82,93	81,08	86,67	50,00	77,78
	AUC (%)	94,03	90,20	92,12	91,46	86,69	84,24	73,00	86,71
	Evaluated patients	178	164	138	122	100	82	70	64
ALS-FRSb	Accuracy (%)	87,08	84,76	76,81	85,25	74,00	82,93	80,00	84,38
	Sensitivity (%)	88,00	77,78	77,78	91,67	94,44	93,33	84,62	84,62
	AUC (%)	93,22	87,98	88,89	93,13	95,66	96,67	90,03	92,31
	Evaluated patients	178	164	138	122	100	82	70	64
ALS-FRSsUL	Accuracy (%)	89,33	82,32	82,61	80,33	78,00	82,93	71,43	82,82
	Sensitivity (%)	89,92	73,08	86,96	86,36	76,19	77,78	70,59	84,93
	AUC (%)	93,51	86,54	93,48	93,18	86,37	88,89	76,96	89,23
	Evaluated patients	178	164	138	122	100	82	70	64
ALS-FRSsLL	Accuracy (%)	85,39	82,93	84,06	74,59	88,00	79,27	80,00	85,94
	Sensitivity (%)	85,19	82,14	82,61	72,00	81,82	88,89	81,25	82,35
	AUC (%)	92,59	90,15	90,22	86,00	90,91	94,44	90,62	91,18
	Evaluated patients	178	164	138	122	100	82	70	64
R	Accuracy (%)	84,83	78,05	85,51	77,05	78,00	87,80	71,43	78,12
	Sensitivity (%)	70,83	60,71	73,08	70,83	68,18	84,21	73,33	76,92
	AUC (%)	83,88	78,51	85,38	77,31	78,73	92,11	79,17	80,57
	Evaluated patients	178	164	138	122	100	82	70	64

(b) Average progression group.

Variable	Metric	Timestep							
		1	2	3	4	5	6	7	8
ALS-FRS	Accuracy (%)	82,51	74,61	76,36	82,18	78,35	80,23	81,43	80,36
	Sensitivity (%)	84,11	78,97	85,59	83,18	64,00	61,29	73,50	72,73
	AUC (%)	90,46	87,62	92,80	91,59	82,00	80,65	84,56	84,89
	Evaluated patients	446	386	330	258	194	172	140	112
ALS-FRSb	Accuracy (%)	86,16	83,68	77,91	74,81	87,82	80,57	81,56	85,71
	Sensitivity (%)	90,69	81,17	80,60	67,90	89,99	86,93	70,32	82,35
	AUC (%)	94,64	90,14	89,71	83,95	95,00	91,14	83,24	91,18
	Evaluated patients	455	392	335	262	197	175	141	112
ALS-FRSsUL	Accuracy (%)	79,49	75,71	77,80	72,91	81,38	86,24	83,74	85,17
	Sensitivity (%)	79,98	77,78	71,85	79,29	76,46	86,27	83,17	86,84
	AUC (%)	89,67	87,50	84,63	87,50	88,23	93,14	87,23	93,42
	Evaluated patients	434	375	320	251	188	167	135	108
ALS-FRSsLL	Accuracy (%)	84,53	74,10	73,33	79,07	78,35	80,82	85,71	94,64
	Sensitivity (%)	80,13	72,92	76,71	77,27	80,00	80,01	87,37	97,50
	AUC (%)	89,73	86,05	87,27	88,64	90,00	86,49	91,46	98,75
	Evaluated patients	446	386	330	258	194	172	140	112
R	Accuracy (%)	80,72	78,50	77,58	81,01	82,47	84,88	87,14	83,93
	Sensitivity (%)	68,42	67,38	72,46	71,93	79,59	80,00	87,50	87,10
	AUC (%)	81,15	79,40	81,81	83,19	84,59	85,83	88,75	87,55
	Evaluated patients	446	386	330	258	194	172	140	112

(c) Fast progression group.

Variable	Metric	Timestep							
		1	2	3	4	5	6	7	8
ALS-FRS	Accuracy (%)	65,67	82,00	82,50	81,48	85,71	91,67	91,67	81,82
	Sensitivity (%)	78,57	57,14	66,67	78,57	80,00	80,00	83,33	75,00
	AUC (%)	84,74	78,57	83,33	89,29	90,00	90,00	91,67	87,50
	Evaluated patients	134	100	80	54	42	24	24	22
ALS-FRSb	Accuracy (%)	82,35	80,39	82,50	74,07	80,95	79,17	83,33	86,37
	Sensitivity (%)	83,87	91,30	88,89	75,00	100,00	66,67	100,00	100,00
	AUC (%)	91,94	93,87	94,44	87,50	100,00	83,33	100,00	100,00
	Evaluated patients	136	102	80	54	42	24	24	22
ALS-FRSsUL	Accuracy (%)	75,00	78,43	75,00	74,07	66,67	91,67	91,67	95,46
	Sensitivity (%)	85,19	74,07	76,92	76,19	82,35	100,00	90,00	100,00
	AUC (%)	92,59	87,04	88,46	79,76	78,68	100,00	95,00	100,00
	Evaluated patients	136	102	80	54	42	24	24	22
ALS-FRSsLL	Accuracy (%)	71,64	74,00	72,50	74,07	80,95	70,84	83,33	77,28
	Sensitivity (%)	79,31	76,19	71,43	88,24	78,57	70,84	100,00	83,34
	AUC (%)	88,34	88,10	83,08	89,12	89,29	85,42	100,00	91,67
	Evaluated patients	134	100	80	54	42	24	24	22
R	Accuracy (%)	82,35	83,33	75,00	77,78	76,19	75,00	100,00	54,55
	Sensitivity (%)	84,00	73,22	65,00	80,00	75,00	87,50	100,00	75,00
	AUC (%)	87,35	84,97	80,00	90,00	81,94	93,75	100,00	70,83
	Evaluated patients	136	102	80	54	42	24	24	22

Table A.7: Results of the predictions of the sdtDBNs before NIV with questions, for the several progression groups (see Section 7.3.1 in Chapter 7), using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

(a) Slow progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	93,75	91,30	84,92	92,37	81,48	88,17	88,89	95,59
	Sensitivity (%)	92,11	85,71	74,29	82,86	73,68	87,88	96,15	x
	AUC (%)	95,18	92,86	85,49	91,43	84,70	92,27	98,08	x
	Evaluated patients	304	276	252	236	216	186	162	136
P2	Accuracy (%)	91,45	87,32	91,27	91,53	87,04	82,80	85,19	87,50
	Sensitivity (%)	x	x	x	x	82,14	84,62	86,96	x
	AUC (%)	x	x	x	x	90,45	90,07	93,48	x
	Evaluated patients	304	276	252	236	216	186	162	136
P3	Accuracy (%)	92,11	93,48	85,71	90,26	83,80	89,25	90,74	92,65
	Sensitivity (%)	x	x	x	x	62,13	92,59	x	x
	AUC (%)	x	x	x	x	80,40	95,54	x	x
	Evaluated patients	304	276	252	236	216	186	162	136
P4	Accuracy (%)	88,59	81,39	83,20	81,36	80,09	80,65	80,25	82,35
	Sensitivity (%)	76,42	76,27	83,64	81,90	86,84	86,00	85,11	58,82
	AUC (%)	87,17	85,57	88,25	87,20	91,95	91,84	91,08	78,43
	Evaluated patients	298	274	250	236	216	186	162	136
P5	Accuracy (%)	85,91	77,01	78,80	78,39	69,91	79,57	77,78	81,62
	Sensitivity (%)	84,92	79,41	87,50	71,67	67,57	84,38	72,41	68,42
	AUC (%)	91,88	87,53	90,47	85,27	83,08	91,37	85,25	83,19
	Evaluated patients	298	274	250	236	216	186	162	136
P6	Accuracy (%)	79,20	71,53	76,80	71,61	62,50	78,49	78,40	80,88
	Sensitivity (%)	78,56	82,93	62,50	69,44	58,54	87,50	70,97	93,66
	AUC (%)	84,66	85,10	80,17	81,67	75,54	92,11	83,48	95,71
	Evaluated patients	298	274	250	236	216	186	162	136
P7	Accuracy (%)	84,90	74,09	72,40	76,27	67,60	75,27	71,61	75,00
	Sensitivity (%)	79,78	70,34	74,60	82,22	51,72	64,29	65,52	73,91
	AUC (%)	87,06	81,01	84,08	87,65	74,60	80,99	81,80	84,74
	Evaluated patients	298	274	250	236	216	186	162	136
P8	Accuracy (%)	84,90	84,31	82,80	89,83	76,39	79,57	74,07	81,62
	Sensitivity (%)	76,91	82,06	65,31	89,36	81,23	84,84	86,84	88,71
	AUC (%)	86,64	90,01	82,00	93,98	87,57	90,55	93,42	93,01
	Evaluated patients	298	274	250	236	216	186	162	136
P9	Accuracy (%)	82,55	74,09	73,20	76,70	71,76	76,34	66,67	83,82
	Sensitivity (%)	81,18	79,44	72,65	71,88	82,76	79,31	70,00	86,96
	AUC (%)	89,07	87,93	84,07	85,36	90,75	89,66	82,06	92,37
	Evaluated patients	298	274	250	236	216	186	162	136
P10	Accuracy (%)	95,07	94,57	94,84	94,49	93,06	91,40	93,21	95,59
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	304	276	252	236	216	186	162	136
R1	Accuracy (%)	88,93	85,40	90,80	84,33	87,50	89,25	85,19	91,18
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	298	274	250	236	216	186	162	136
R2	Accuracy (%)	97,70	93,84	94,05	95,76	93,52	95,70	98,77	95,59
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	304	276	252	236	216	186	162	136
R3	Accuracy (%)	98,62	99,24	99,16	98,20	95,05	95,98	97,33	100,00
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	290	262	238	222	202	174	150	124

(b) Average progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	86,52	77,18	77,33	82,04	76,24	79,45	88,24	73,68
	Sensitivity (%)	92,33	70,00	74,14	91,49	75,00	76,19	100,00	70,00
	AUC (%)	95,80	84,71	87,07	95,74	87,50	87,13	100,00	81,43
	Evaluated patients	586	460	366	284	202	146	102	76
P2	Accuracy (%)	81,57	75,22	76,78	77,82	71,79	69,86	74,51	84,21
	Sensitivity (%)	82,31	76,92	79,17	75,00	62,96	78,95	92,31	85,71
	AUC (%)	88,73	87,14	86,66	86,08	79,45	85,77	96,15	88,69
	Evaluated patients	586	460	366	284	202	146	102	76
P3	Accuracy (%)	86,18	81,30	83,06	82,39	78,22	73,29	76,47	97,37
	Sensitivity (%)	91,20	86,51	86,47	71,05	68,75	83,33	81,25	100,00
	AUC (%)	93,85	91,42	91,20	85,05	83,65	90,65	90,62	100,00
	Evaluated patients	586	460	366	284	202	146	102	76
P4	Accuracy (%)	84,35	74,10	73,60	74,28	73,24	71,83	66,00	80,27
	Sensitivity (%)	87,25	80,74	69,60	69,86	69,86	57,89	76,47	69,23
	AUC (%)	92,11	89,22	84,42	82,86	83,33	78,95	86,72	84,62
	Evaluated patients	562	444	356	276	198	142	100	76
P5	Accuracy (%)	76,34	70,95	71,91	62,68	70,21	75,36	60,00	75,00
	Sensitivity (%)	63,32	68,80	68,28	60,00	70,34	89,06	65,22	81,25
	AUC (%)	80,39	83,66	83,74	78,86	83,30	94,53	78,90	88,35
	Evaluated patients	562	444	356	276	198	142	100	76
P6	Accuracy (%)	74,56	70,27	63,77	65,95	62,63	59,86	66,00	69,74
	Sensitivity (%)	69,51	72,38	67,35	65,16	63,51	44,44	70,83	76,47
	AUC (%)	82,89	84,29	82,12	81,00	81,76	72,22	83,49	88,24
	Evaluated patients	562	444	356	276	198	142	100	76
P7	Accuracy (%)	78,11	67,35	70,51	63,77	62,63	71,13	68,00	53,95
	Sensitivity (%)	79,34	56,47	66,41	63,63	67,60	60,00	53,33	60,00
	AUC (%)	88,52	77,77	81,89	81,18	81,58	80,00	76,67	80,00
	Evaluated patients	562	444	356	276	198	142	100	76
P8	Accuracy (%)	80,30	75,41	75,89	67,84	76,01	76,40	70,30	64,48
	Sensitivity (%)	73,71	72,26	84,05	75,33	89,26	87,68	61,25	78,18
	AUC (%)	85,97	84,07	89,49	86,07	94,63	93,84	79,20	89,09
	Evaluated patients	574	452	361	280	200	144	101	76
P9	Accuracy (%)	73,49	64,19	68,54	67,03	60,11	62,68	81,00	67,11
	Sensitivity (%)	75,48	59,86	70,93	68,00	56,63	57,69	83,33	69,05
	AUC (%)	86,74	78,94	83,60	82,02	77,62	78,85	91,67	84,52
	Evaluated patients	562	444	356	276	198	142	100	76
P10	Accuracy (%)	91,82	84,81	86,91	84,20	84,73	85,55	91,16	90,68
	Sensitivity (%)	84,46	73,21	77,98	68,50	76,92	x	x	x
	AUC (%)	90,56	83,53	86,32	82,02	87,82	x	x	x
	Evaluated patients	588	462	368	286	204	146	102	76
R1	Accuracy (%)	87,12	82,10	81,80	80,84	79,13	84,70	90,08	93,42
	Sensitivity (%)	81,69	70,64	70,78	70,95	x	x	x	x
	AUC (%)	89,08	82,41	83,84	82,75	x	x	x	x
	Evaluated patients	576	454	363	282	202	144	101	76
R2	Accuracy (%)	90,67	92,34	91,52	90,55	87,78	89,06	91,11	94,29
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	536	418	330	254	180	128	90	70
R3	Accuracy (%)	97,64	98,07	97,04	92,76	94,71	96,63	94,23	100,00
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	592	466	372	290	208	148	104	78

(c) Fast progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	77,89	70,14	66,31	77,28	86,37	85,30	82,14	81,25
	Sensitivity (%)	75,38	72,22	75,00	88,89	85,71	80,00	75,00	100,00
	AUC (%)	87,69	83,80	86,72	94,44	92,86	90,00	87,50	95,00
	Evaluated patients	208	144	92	66	44	34	28	16
P2	Accuracy (%)	78,85	74,31	64,13	60,61	70,46	70,59	71,43	62,50
	Sensitivity (%)	74,83	86,84	54,10	44,44	71,43	80,00	60,00	100,00
	AUC (%)	85,16	90,60	77,05	70,14	85,71	90,00	80,00	100,00
	Evaluated patients	208	144	92	66	44	34	28	16
P3	Accuracy (%)	78,37	77,09	73,91	72,73	90,91	67,65	75,00	87,50
	Sensitivity (%)	55,56	69,23	77,78	75,00	100,00	70,84	81,25	100,00
	AUC (%)	76,48	83,53	87,10	87,50	100,00	85,42	90,63	100,00
	Evaluated patients	208	144	92	66	44	34	28	16
P4	Accuracy (%)	64,91	59,03	76,09	72,73	68,19	61,77	75,00	75,00
	Sensitivity (%)	61,85	27,78	84,62	77,78	87,50	83,33	100,00	100,00
	AUC (%)	79,41	59,26	90,79	84,72	93,75	91,67	94,44	90,00
	Evaluated patients	208	144	92	66	44	34	28	16
P5	Accuracy (%)	67,31	56,25	68,48	78,79	65,91	76,47	78,57	87,50
	Sensitivity (%)	64,29	27,78	51,47	88,89	55,56	85,71	80,00	66,67
	AUC (%)	81,34	63,89	75,74	94,44	77,78	92,86	90,00	83,33
	Evaluated patients	208	144	92	66	44	34	28	16
P6	Accuracy (%)	64,42	56,25	67,39	69,70	68,19	67,65	78,57	87,50
	Sensitivity (%)	62,86	37,50	50,00	90,00	50,00	75,00	83,33	75,00
	AUC (%)	79,98	68,75	75,00	92,83	75,00	87,50	91,67	87,50
	Evaluated patients	208	144	92	66	44	34	28	16
P7	Accuracy (%)	63,94	56,95	51,09	80,31	75,00	61,77	75,00	62,50
	Sensitivity (%)	67,82	44,83	22,26	92,86	66,67	73,22	80,00	40,00
	AUC (%)	83,91	71,25	61,13	96,43	80,21	86,61	81,67	70,00
	Evaluated patients	208	144	92	66	44	34	28	16
P8	Accuracy (%)	66,35	57,64	47,83	66,67	61,37	44,12	35,71	25,00
	Sensitivity (%)	70,09	73,49	53,33	66,67	75,00	66,67	83,33	50,00
	AUC (%)	82,08	81,83	73,44	81,25	87,50	78,79	85,42	66,67
	Evaluated patients	208	144	92	66	44	34	28	16
P9	Accuracy (%)	62,02	59,72	38,05	84,85	70,46	76,47	71,43	87,50
	Sensitivity (%)	60,82	41,06	31,58	100,00	62,50	71,43	71,43	100,00
	AUC (%)	80,41	70,53	65,79	100,00	81,25	85,71	78,57	100,00
	Evaluated patients	208	144	92	66	44	34	28	16
P10	Accuracy (%)	77,40	79,87	78,26	75,76	84,09	76,47	64,29	75,00
	Sensitivity (%)	57,89	64,29	68,42	61,54	80,36	56,25	50,00	100,00
	AUC (%)	75,16	78,73	80,51	75,77	86,73	78,13	66,67	90,00
	Evaluated patients	208	144	92	66	44	34	28	16
R1	Accuracy (%)	75,00	80,56	80,43	81,82	93,18	73,53	64,29	75,00
	Sensitivity (%)	82,14	100,00	94,44	84,62	x	87,50	0,00	50,00
	AUC (%)	90,27	100,00	97,22	92,31	x	93,75	45,00	75,00
	Evaluated patients	208	144	92	66	44	34	28	16
R2	Accuracy (%)	83,65	84,03	86,96	90,91	90,91	88,24	85,71	87,50
	Sensitivity (%)	x	46,19	x	x	71,43	83,33	83,33	66,67
	AUC (%)	x	72,14	x	x	85,71	91,67	91,67	83,33
	Evaluated patients	208	144	92	66	44	34	28	16
R3	Accuracy (%)	97,12	97,22	97,83	93,94	90,91	88,24	92,86	100,00
	Sensitivity (%)	x	x	x	x	x	x	x	x
	AUC (%)	x	x	x	x	x	x	x	x
	Evaluated patients	208	144	92	66	44	34	28	16

Table A.8: Results of the predictions of the sdtDBNs after NIV with questions, for the several progression groups (see Section 7.3.1 in Chapter 7), using $\{m = 1, p = 2, b = 1, T = 8\}$ and the MDL score.

(a) Slow progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	86,52	75,61	76,81	81,97	80,00	85,37	85,71	84,38
	Sensitivity (%)	79,31	76,67	73,68	78,95	61,54	92,31	100,00	87,50
	AUC (%)	89,66	85,45	86,84	88,28	80,77	96,15	98,08	93,75
	Evaluated patients	178	164	138	122	100	82	70	64
P2	Accuracy (%)	79,78	78,05	65,22	81,15	69,00	78,05	71,43	71,88
	Sensitivity (%)	86,84	71,43	71,43	88,89	72,92	90,00	78,95	44,44
	AUC (%)	91,46	84,07	85,71	92,12	82,61	90,24	86,35	70,05
	Evaluated patients	178	164	138	122	100	82	70	64
P3	Accuracy (%)	89,89	79,27	78,26	68,85	67,00	82,93	74,29	84,38
	Sensitivity (%)	86,96	63,33	72,41	76,00	72,08	93,94	87,09	81,80
	AUC (%)	93,48	80,71	83,71	81,06	79,92	94,93	90,98	90,90
	Evaluated patients	178	164	138	122	100	82	70	64
P4	Accuracy (%)	83,15	76,83	73,19	81,15	75,00	75,61	62,86	82,82
	Sensitivity (%)	88,00	75,86	68,42	93,75	85,71	83,33	81,82	83,33
	AUC (%)	94,00	87,93	84,21	96,88	91,47	91,67	88,83	91,67
	Evaluated patients	178	164	138	122	100	82	70	64
P5	Accuracy (%)	82,03	71,95	77,54	72,13	68,00	80,49	77,14	89,06
	Sensitivity (%)	90,91	83,72	84,62	86,11	78,57	89,29	70,00	80,00
	AUC (%)	94,34	86,73	92,31	89,06	89,29	90,80	83,00	90,00
	Evaluated patients	178	164	138	122	100	82	70	64
P6	Accuracy (%)	84,83	71,95	76,09	66,39	73,00	82,93	65,71	70,32
	Sensitivity (%)	97,14	61,90	71,43	78,26	88,89	84,62	75,00	71,43
	AUC (%)	97,65	80,95	84,67	89,13	92,88	92,31	85,33	85,71
	Evaluated patients	178	164	138	122	100	82	70	64
P7	Accuracy (%)	83,71	80,49	69,57	68,03	66,00	76,83	74,29	81,25
	Sensitivity (%)	85,19	85,71	80,00	83,33	66,67	84,21	81,25	87,50
	AUC (%)	92,59	92,86	90,00	91,67	80,48	92,11	85,36	93,75
	Evaluated patients	178	164	138	122	100	82	70	64
P8	Accuracy (%)	87,08	76,83	79,71	73,77	72,00	85,37	74,29	75,00
	Sensitivity (%)	80,89	80,44	72,73	77,27	69,23	94,12	72,73	100,00
	AUC (%)	90,44	88,53	86,36	87,35	84,62	97,06	84,28	97,83
	Evaluated patients	178	164	138	122	100	82	70	64
P9	Accuracy (%)	80,34	79,27	81,16	63,12	79,00	80,49	68,57	81,25
	Sensitivity (%)	84,38	81,25	95,65	73,08	76,00	90,48	78,95	94,44
	AUC (%)	90,43	89,62	96,74	86,54	88,00	95,24	86,35	97,22
	Evaluated patients	178	164	138	122	100	82	70	64
P10	Accuracy (%)	79,22	81,71	82,61	82,79	78,00	82,93	81,43	90,62
	Sensitivity (%)	80,53	83,05	93,69	91,01	88,57	92,86	91,30	86,96
	AUC (%)	84,89	88,27	84,07	86,41	79,29	88,74	83,15	93,48
	Evaluated patients	178	164	138	122	100	82	70	64
R1	Accuracy (%)	78,65	70,73	76,09	62,30	62,00	80,49	77,14	79,69
	Sensitivity (%)	84,62	80,56	77,42	72,73	71,43	83,33	85,00	82,11
	AUC (%)	86,31	84,84	86,08	81,01	72,08	82,84	85,83	83,04
	Evaluated patients	178	164	138	122	100	82	70	64
R2	Accuracy (%)	82,02	78,05	75,36	73,77	66,00	81,71	71,43	84,38
	Sensitivity (%)	85,00	74,42	80,00	82,35	75,86	89,84	75,50	91,11
	AUC (%)	89,44	84,65	82,65	83,77	78,41	94,92	82,98	90,28
	Evaluated patients	178	164	138	122	100	82	70	64
R3	Accuracy (%)	82,02	75,61	75,36	73,77	76,00	85,37	85,71	89,06
	Sensitivity (%)	78,22	79,80	86,59	78,05	82,35	89,66	88,00	93,38
	AUC (%)	87,81	82,21	82,58	79,02	75,55	90,66	89,00	86,14
	Evaluated patients	178	164	138	122	100	82	70	64

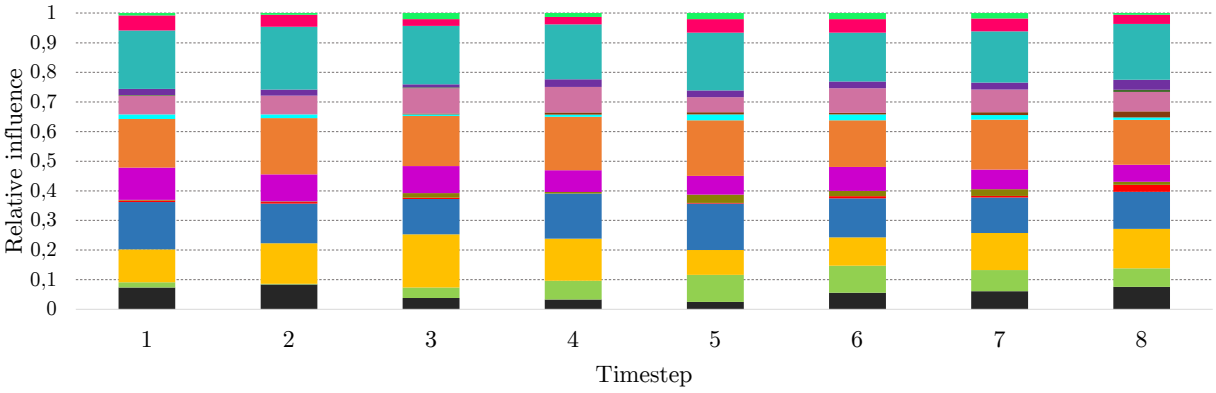
(b) Average progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	82,98	79,70	77,46	86,76	85,44	84,24	85,62	82,76
	Sensitivity (%)	90,14	81,43	72,00	81,82	84,38	82,56	81,25	84,21
	AUC (%)	95,07	90,71	85,59	90,91	91,48	90,46	88,59	90,82
	Evaluated patients	470	404	346	272	206	184	146	116
P2	Accuracy (%)	78,70	70,21	68,48	68,99	78,35	70,93	74,29	76,79
	Sensitivity (%)	86,21	73,58	73,44	79,25	91,18	73,53	92,31	85,71
	AUC (%)	90,12	83,54	82,76	85,68	94,79	84,84	92,74	91,43
	Evaluated patients	446	386	330	258	194	172	140	112
P3	Accuracy (%)	75,22	76,87	79,94	74,07	83,33	79,12	82,64	82,46
	Sensitivity (%)	85,40	76,22	86,15	86,15	87,23	74,37	83,33	75,00
	AUC (%)	89,52	87,19	91,95	90,93	92,71	86,45	91,67	87,50
	Evaluated patients	468	402	344	270	204	182	144	114
P4	Accuracy (%)	78,02	71,36	66,77	68,42	72,00	74,72	61,27	85,71
	Sensitivity (%)	78,86	77,42	71,88	76,32	74,19	86,21	73,53	89,29
	AUC (%)	89,14	88,16	84,52	87,11	84,92	91,44	86,76	92,86
	Evaluated patients	464	398	340	266	200	178	142	112
P5	Accuracy (%)	76,68	65,29	69,40	67,06	71,65	77,33	74,29	78,57
	Sensitivity (%)	79,82	77,89	71,43	65,00	78,12	72,73	75,86	74,07
	AUC (%)	88,81	84,73	84,90	80,81	87,52	85,42	85,49	87,04
	Evaluated patients	446	386	330	258	194	172	140	112
P6	Accuracy (%)	75,35	69,83	68,74	67,74	72,87	78,43	80,00	88,88
	Sensitivity (%)	75,87	71,35	69,32	73,40	71,47	79,03	82,97	89,32
	AUC (%)	87,32	84,34	83,91	85,29	85,73	88,74	87,23	94,66
	Evaluated patients	434	375	320	251	188	167	135	108
P7	Accuracy (%)	77,13	71,25	66,97	64,34	64,44	68,60	77,14	78,57
	Sensitivity (%)	76,61	79,47	75,00	53,49	76,67	60,00	81,82	83,33
	AUC (%)	87,58	88,67	86,35	76,74	86,09	78,91	89,56	89,74
	Evaluated patients	446	386	330	258	194	172	140	112
P8	Accuracy (%)	80,66	68,62	70,15	67,18	76,15	75,99	77,32	78,57
	Sensitivity (%)	81,35	70,28	71,18	64,55	81,99	76,81	73,04	80,00
	AUC (%)	90,68	84,79	84,67	82,01	90,26	87,46	82,67	90,00
	Evaluated patients	455	392	335	262	197	175	141	112
P9	Accuracy (%)	74,07	66,57	67,77	71,38	74,60	80,01	84,41	91,07
	Sensitivity (%)	76,92	68,39	75,32	76,39	81,20	82,80	88,90	97,44
	AUC (%)	88,46	83,35	86,00	88,20	89,35	89,51	92,07	95,78
	Evaluated patients	455	392	335	262	197	175	141	112
P10	Accuracy (%)	84,05	80,65	76,47	81,20	84,00	87,64	69,01	80,36
	Sensitivity (%)	84,85	86,90	86,72	84,26	93,67	95,65	86,79	93,02
	AUC (%)	89,44	83,26	79,07	84,13	84,93	85,33	73,95	88,82
	Evaluated patients	464	398	340	266	200	178	142	112
R1	Accuracy (%)	71,80	72,81	69,04	61,48	67,03	75,31	69,23	69,23
	Sensitivity (%)	73,66	74,47	80,63	74,63	83,18	70,37	66,67	70,59
	AUC (%)	82,22	80,70	83,98	80,95	84,26	84,26	81,06	82,44
	Evaluated patients	422	364	310	244	182	162	130	104
R2	Accuracy (%)	74,65	71,16	76,45	68,85	68,13	80,87	70,00	76,92
	Sensitivity (%)	80,43	80,95	86,36	74,39	52,17	70,83	76,47	80,00
	AUC (%)	84,87	84,31	84,97	83,45	72,41	82,79	84,07	88,65
	Evaluated patients	422	364	310	244	182	162	130	104
R3	Accuracy (%)	81,90	75,88	80,89	83,46	83,00	82,59	82,40	85,71
	Sensitivity (%)	84,68	76,80	87,18	89,13	94,37	90,38	95,19	90,91
	AUC (%)	87,71	82,32	86,99	89,69	92,01	88,21	87,07	91,29
	Evaluated patients	464	398	340	266	200	178	142	112

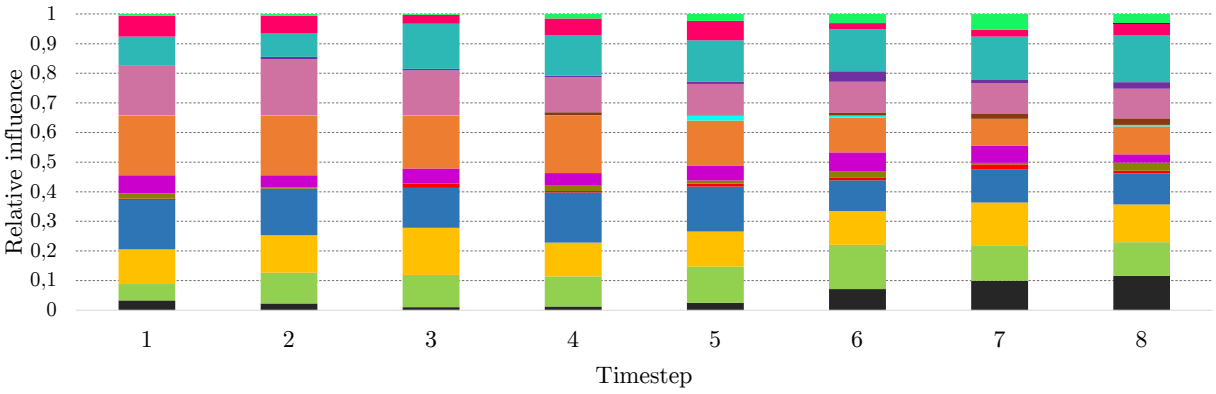
(c) Fast progression group.

Variable	Metric	Timestep							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P1	Accuracy (%)	77,94	88,24	85,00	70,37	80,95	91,67	83,33	81,82
	Sensitivity (%)	71,43	80,00	100,00	90,00	100,00	100,00	80,00	100,00
	AUC (%)	85,71	88,61	100,00	92,06	96,43	100,00	90,00	92,86
	Evaluated patients	136	102	80	54	42	24	24	22
P2	Accuracy (%)	72,06	68,63	77,50	77,78	52,38	66,67	75,00	90,91
	Sensitivity (%)	87,50	64,29	66,67	70,00	85,71	60,00	100,00	100,00
	AUC (%)	89,20	80,79	83,33	82,06	85,71	80,00	100,00	92,86
	Evaluated patients	136	102	80	54	42	24	24	22
P3	Accuracy (%)	85,29	70,59	77,50	77,78	80,95	75,00	91,67	63,64
	Sensitivity (%)	86,21	75,00	78,95	71,43	90,91	66,67	100,00	100,00
	AUC (%)	91,82	83,80	89,47	85,71	90,45	83,33	100,00	100,00
	Evaluated patients	136	102	80	54	42	24	24	22
P4	Accuracy (%)	64,71	68,63	65,00	66,67	66,67	75,00	100,00	90,91
	Sensitivity (%)	71,43	80,00	85,00	92,86	100,00	77,78	100,00	88,89
	AUC (%)	81,46	86,77	90,00	84,89	100,00	88,89	100,00	94,44
	Evaluated patients	136	102	80	54	42	24	24	22
P5	Accuracy (%)	69,12	54,90	80,00	66,67	71,43	75,00	91,67	90,91
	Sensitivity (%)	69,57	60,87	80,00	72,22	78,57	87,50	88,89	88,89
	AUC (%)	81,45	71,51	87,50	80,56	82,14	81,25	94,44	94,44
	Evaluated patients	136	102	80	54	42	24	24	22
P6	Accuracy (%)	72,06	68,63	65,00	77,78	71,43	83,33	91,67	90,91
	Sensitivity (%)	72,41	84,00	83,33	80,95	100,00	100,00	100,00	88,89
	AUC (%)	81,08	86,23	82,29	82,14	83,33	83,33	100,00	94,44
	Evaluated patients	136	102	80	54	42	24	24	22
P7	Accuracy (%)	63,24	68,63	75,00	59,26	66,67	66,67	83,33	100,00
	Sensitivity (%)	85,00	88,24	87,50	85,71	83,33	85,71	100,00	100,00
	AUC (%)	82,08	91,18	85,42	85,16	80,56	82,86	87,50	100,00
	Evaluated patients	136	102	80	54	42	24	24	22
P8	Accuracy (%)	66,18	69,61	82,50	70,37	66,67	66,67	75,00	72,73
	Sensitivity (%)	81,82	69,23	90,91	80,00	60,00	71,43	75,00	100,00
	AUC (%)	88,74	84,62	95,45	90,00	80,00	85,71	75,00	83,33
	Evaluated patients	136	102	80	54	42	24	24	22
P9	Accuracy (%)	70,59	64,71	65,00	77,78	66,67	83,33	91,67	81,82
	Sensitivity (%)	86,67	78,26	69,23	88,89	78,57	88,89	100,00	88,89
	AUC (%)	90,70	81,99	63,19	83,33	60,71	94,44	83,33	69,44
	Evaluated patients	136	102	80	54	42	24	24	22
P10	Accuracy (%)	77,21	79,41	71,25	74,07	83,33	66,67	87,50	68,19
	Sensitivity (%)	90,91	95,35	93,55	93,18	97,37	88,89	100,00	100,00
	AUC (%)	68,54	63,30	49,55	61,59	48,69	61,11	75,00	68,75
	Evaluated patients	136	102	80	54	42	24	24	22
R1	Accuracy (%)	69,12	74,51	67,50	70,37	71,43	75,00	75,00	40,91
	Sensitivity (%)	75,68	70,97	69,23	70,00	83,33	75,00	50,00	66,67
	AUC (%)	83,00	82,98	84,62	79,12	85,00	87,50	68,75	77,08
	Evaluated patients	136	102	80	54	42	24	24	22
R2	Accuracy (%)	70,15	73,00	60,00	55,56	42,86	66,67	91,67	63,64
	Sensitivity (%)	80,95	84,38	45,45	50,00	62,50	100,00	100,00	66,67
	AUC (%)	78,48	81,08	69,28	66,18	73,56	93,75	94,44	77,08
	Evaluated patients	134	100	80	54	42	24	24	22
R3	Accuracy (%)	79,41	82,35	85,00	66,67	61,90	91,67	91,67	90,91
	Sensitivity (%)	86,84	84,62	90,48	64,71	78,57	100,00	87,50	87,50
	AUC (%)	88,42	86,31	92,61	82,35	75,00	100,00	93,75	93,75
	Evaluated patients	136	102	80	54	42	24	24	22

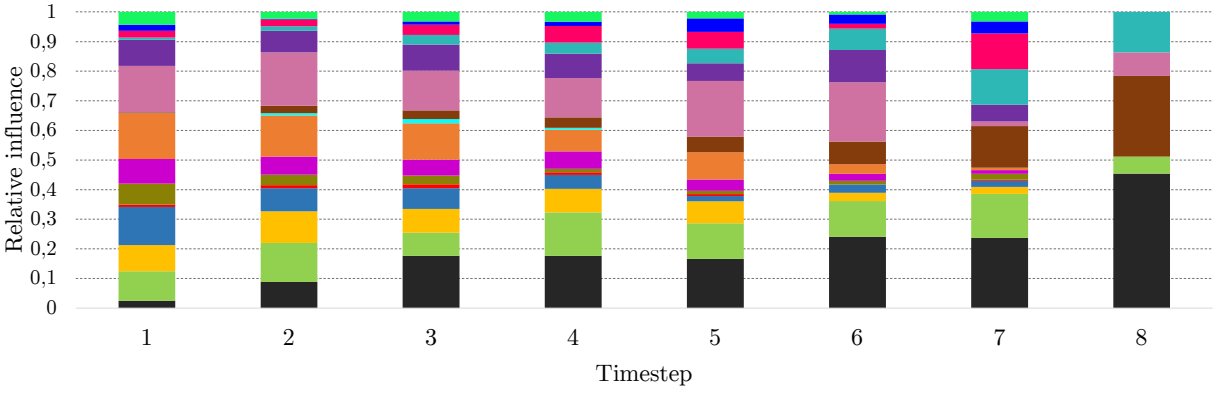
A.2.2 Influence of each variable in every timestep



(a) Slow progression group.



(b) Average progression group.



(c) Fast progression group.

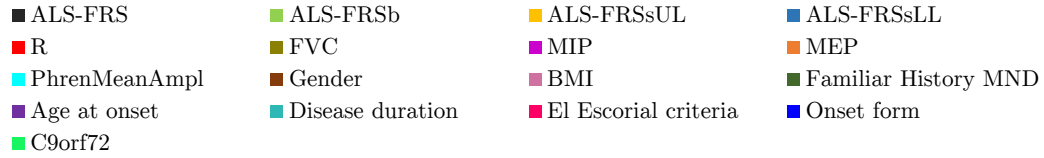
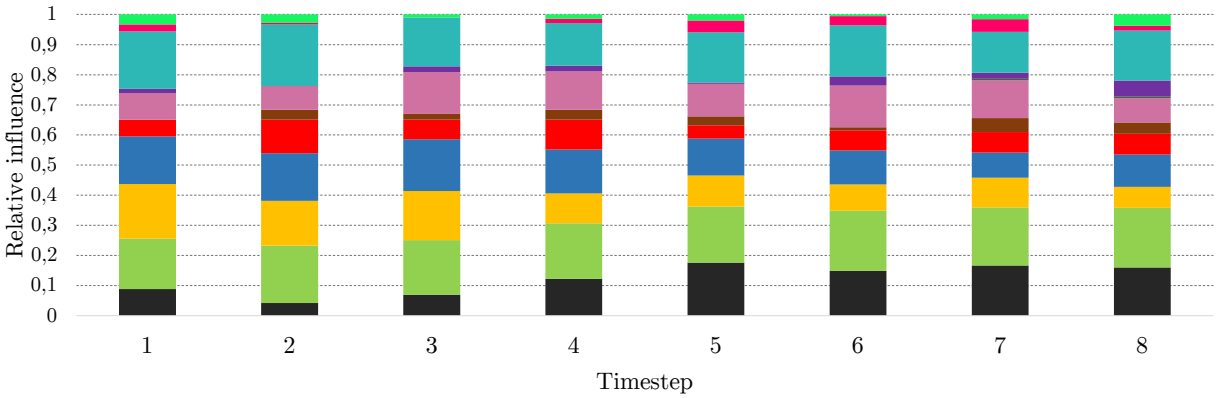
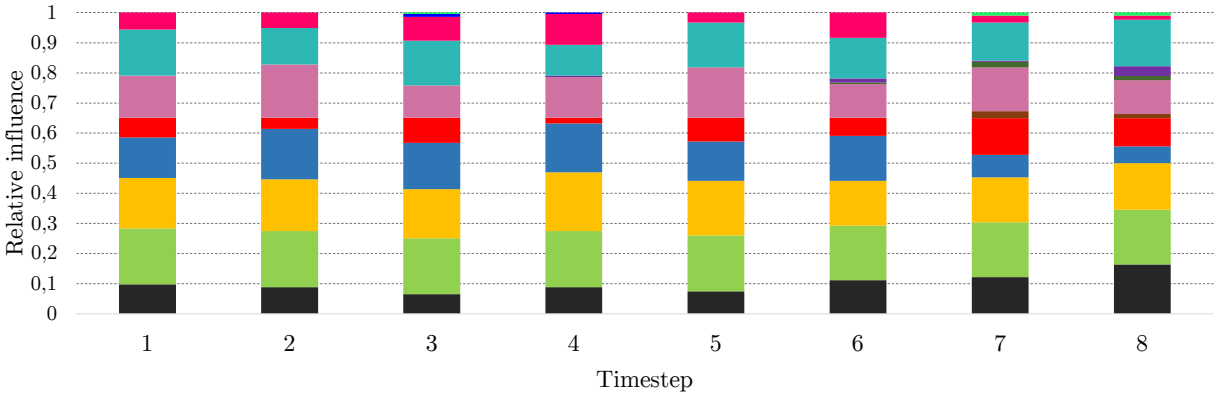


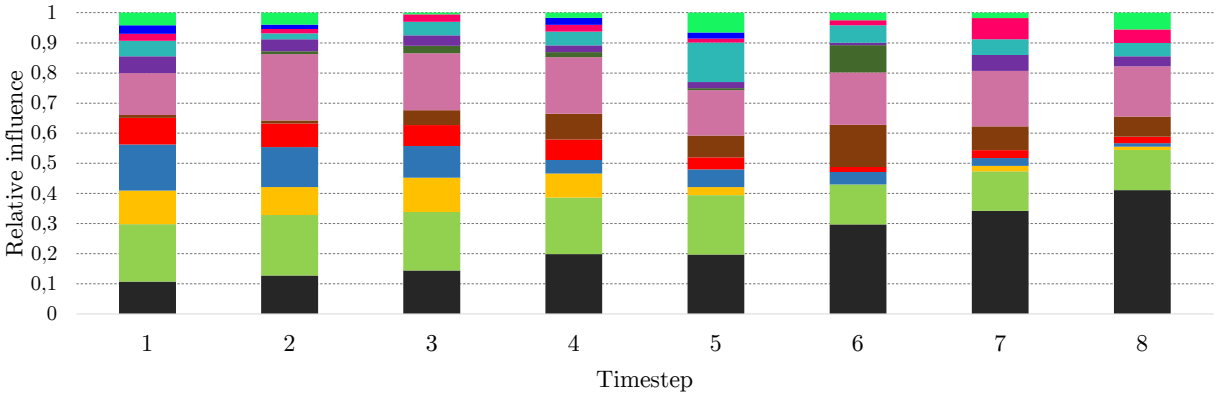
Figure A.3: Influence of each variable in every timestep of the sdtDBNs before NIV with sub-scores, for each progression group. The sdtDBNs detailed in Table 7.8 (see Chapter 7) are learned for every progression group.



(a) Slow progression group.



(b) Average progression group.



(c) Fast progression group.

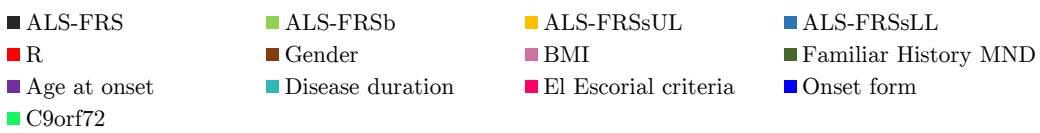
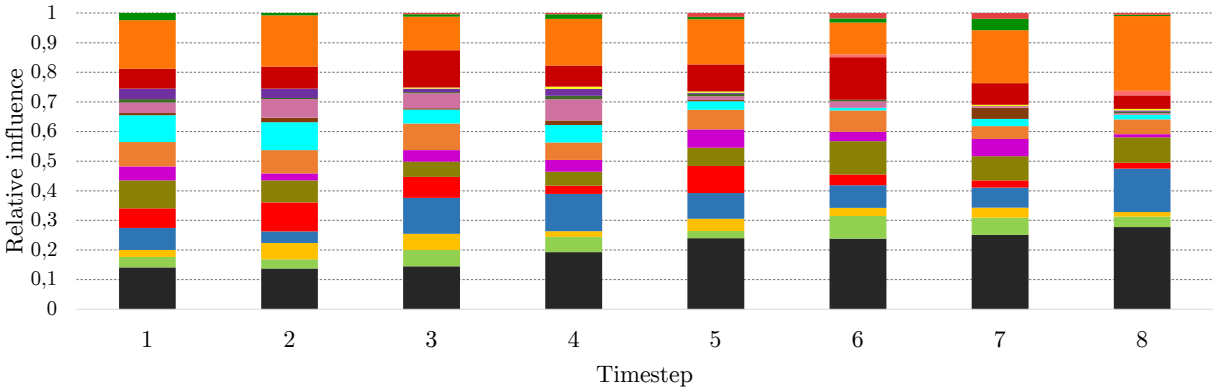
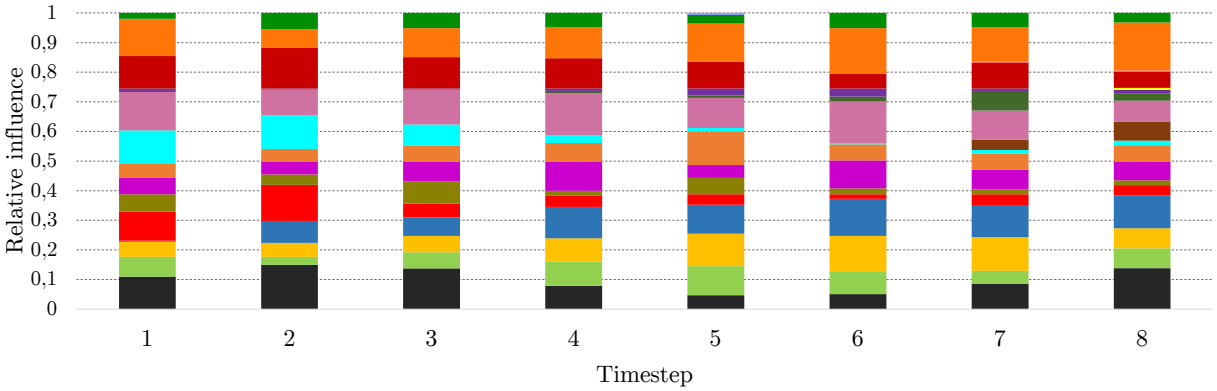


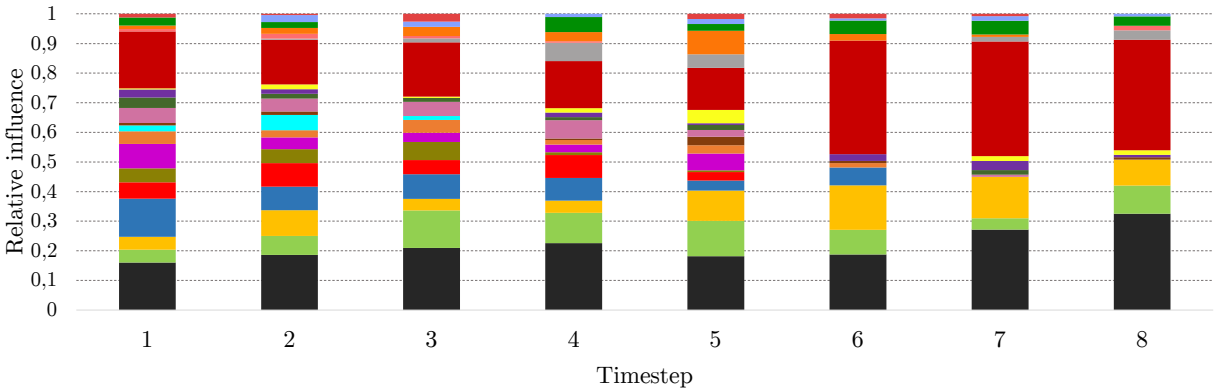
Figure A.4: Influence of each variable in every timestep of the sdtDBNs after NIV with sub-scores, for each progression group. The sdtDBNs detailed in Table 7.8 (see Chapter 7) are learned for every progression group.



(a) Slow progression group.



(b) Average progression group.



(c) Fast progression group.

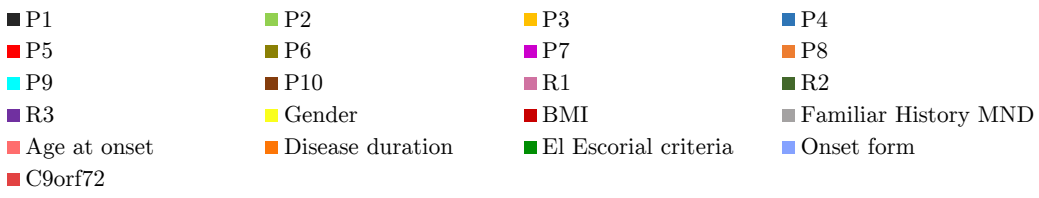


Figure A.5: Influence of each variable in every timestep of the sdtDBNs after NIV with questions, for each progression group. The sdtDBNs detailed in Table 7.8 (see Chapter 7) are learned for every progression group.

A.2.3 Correlations among variables throughout the disease progression

Table A.9: Correlations between variables of the sdtDBNs before NIV with questions, for each progression group. Each table is normalized per column. The sdtDBNs of Table 7.8 (see Chapter 7) are learned for every progression group.

(a) Slow progression group.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3	FVC	MIP	MEP	PhrenMeanAmpl
P1	—	32,00%	36,41%	3,66%	7,77%	3,51%	2,99%	4,97%	3,70%	14,04%	10,44%	23,88%	37,50%	7,98%	7,24%	3,63%	11,19%
P2	15,65%	—	13,59%	2,74%	1,70%	2,01%	0,37%	3,64%	1,94%	2,34%	0,55%	6,72%	4,55%	3,07%	2,07%	6,25%	2,80%
P3	16,38%	12,50%	—	1,22%	1,46%	1,75%	2,61%	1,32%	3,70%	2,34%	2,75%	2,99%	3,41%	1,23%	1,38%	3,63%	2,10%
P4	2,93%	4,50%	2,17%	—	19,90%	7,52%	10,07%	10,26%	3,53%	9,94%	6,59%	12,69%	18,18%	9,20%	3,10%	4,84%	2,10%
P5	7,82%	3,50%	3,26%	25,00%	—	15,04%	5,22%	5,30%	10,05%	8,19%	11,54%	7,46%	7,95%	9,82%	4,48%	8,87%	9,09%
P6	3,42%	4,00%	3,80%	9,15%	14,56%	—	22,76%	7,95%	11,11%	4,68%	5,49%	8,96%	4,55%	11,04%	9,66%	9,27%	4,20%
P7	1,96%	0,50%	3,80%	8,23%	3,40%	15,29%	—	4,30%	8,47%	5,85%	6,59%	4,48%	1,14%	1,23%	7,24%	6,45%	3,50%
P8	3,67%	5,50%	2,17%	9,45%	3,88%	6,02%	4,85%	—	17,28%	1,17%	6,59%	5,97%	6,82%	6,13%	3,79%	6,65%	5,59%
P9	5,13%	5,50%	11,41%	6,10%	13,83%	15,79%	17,91%	32,45%	—	7,60%	12,09%	8,21%	0,00%	18,40%	18,97%	15,12%	15,38%
P10	5,87%	2,00%	2,17%	5,18%	3,40%	2,01%	3,73%	0,66%	2,29%	—	19,23%	3,73%	6,82%	3,68%	2,41%	2,82%	1,40%
R1	4,65%	0,50%	2,72%	3,66%	5,10%	2,51%	4,48%	3,97%	3,88%	20,47%	—	3,73%	1,14%	4,29%	2,41%	2,62%	0,00%
R2	7,82%	4,50%	2,17%	5,18%	2,43%	3,01%	2,24%	2,65%	1,94%	2,92%	2,75%	—	3,41%	0,00%	1,03%	1,81%	0,00%
R3	8,07%	2,00%	1,63%	4,88%	1,70%	1,00%	0,37%	1,99%	0,00%	3,51%	0,55%	2,24%	—	0,00%	1,03%	0,00%	0,70%
FVC	3,18%	2,50%	1,09%	4,57%	3,88%	4,51%	0,75%	3,31%	5,29%	3,51%	3,85%	0,00%	0,00%	—	0,34%	7,46%	0,70%
MIP	5,13%	3,00%	2,17%	2,74%	3,16%	7,02%	7,84%	3,64%	9,70%	4,09%	3,85%	2,24%	3,41%	0,61%	—	14,52%	20,28%
MEP	4,40%	15,50%	9,78%	7,32%	10,68%	11,53%	11,94%	10,93%	13,23%	8,19%	7,14%	6,72%	0,00%	22,70%	24,83%	—	20,98%
PhrenMeanAmpl	3,91%	2,00%	1,63%	0,91%	3,16%	1,50%	1,87%	2,65%	3,88%	1,17%	0,00%	0,00%	1,14%	0,61%	10,00%	6,05%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

(b) Average progression group.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3	FVC	MIP	MEP	PhrenMeanAmpl
P1	—	30,70%	38,80%	10,68%	6,86%	6,63%	8,76%	12,55%	8,91%	15,15%	17,11%	20,77%	24,30%	18,75%	14,44%	10,80%	22,56%
P2	11,74%	—	12,57%	4,66%	2,52%	2,21%	2,30%	2,35%	4,82%	1,82%	1,32%	6,15%	5,61%	4,86%	3,33%	3,76%	3,76%
P3	12,63%	10,70%	—	3,01%	2,97%	2,49%	2,30%	1,18%	0,93%	4,24%	1,97%	3,08%	8,41%	1,39%	0,56%	3,76%	0,75%
P4	6,94%	7,91%	6,01%	—	18,54%	8,01%	6,91%	7,06%	7,24%	7,27%	7,24%	7,69%	13,08%	11,11%	7,78%	5,40%	12,03%
P5	5,34%	5,12%	7,10%	22,19%	—	17,40%	12,44%	8,63%	8,16%	5,45%	7,89%	6,92%	3,74%	11,11%	15,56%	11,74%	13,53%
P6	4,27%	3,72%	4,92%	7,95%	14,42%	—	11,06%	9,02%	11,32%	6,67%	9,21%	8,46%	1,87%	8,33%	12,22%	9,15%	7,52%
P7	3,38%	2,33%	2,73%	4,11%	6,18%	6,63%	—	3,92%	7,79%	1,82%	1,97%	7,69%	5,61%	4,86%	0,56%	7,28%	6,77%
P8	5,69%	2,79%	1,64%	4,93%	5,03%	6,35%	4,61%	—	14,66%	3,64%	3,29%	3,08%	5,61%	2,08%	4,44%	6,10%	3,01%
P9	8,54%	12,09%	2,73%	10,68%	10,07%	16,85%	19,35%	30,98%	—	12,12%	13,16%	14,62%	2,80%	17,36%	13,89%	13,38%	19,55%
P10	4,45%	1,40%	3,83%	3,29%	2,06%	3,04%	1,38%	2,35%	3,71%	—	21,71%	0,77%	15,89%	0,69%	0,00%	3,52%	1,50%
R1	4,63%	0,93%	1,64%	3,01%	2,75%	3,87%	1,38%	1,96%	3,71%	20,00%	—	5,38%	0,00%	0,00%	1,67%	3,05%	0,00%
R2	4,80%	3,72%	2,19%	2,74%	2,06%	3,04%	4,61%	1,57%	3,53%	0,61%	4,61%	—	3,74%	0,69%	1,67%	2,82%	0,00%
R3	4,63%	2,79%	4,92%	3,84%	0,92%	0,55%	2,76%	2,35%	0,56%	10,30%	0,00%	3,08%	—	2,78%	0,00%	1,41%	0,00%
FVC	4,80%	3,26%	1,09%	4,38%	3,66%	3,31%	3,23%	1,18%	4,64%	0,61%	0,00%	0,77%	3,74%	—	0,56%	5,16%	0,00%
MIP	4,63%	2,79%	0,55%	3,84%	6,41%	6,08%	0,46%	3,14%	4,64%	0,00%	1,97%	2,31%	0,00%	0,69%	—	9,86%	0,00%
MEP	8,19%	7,44%	8,74%	6,30%	11,44%	10,77%	14,29%	10,20%	10,58%	9,09%	8,55%	9,23%	5,61%	15,28%	23,33%	—	9,02%
PhrenMeanAmpl	5,34%	2,33%	0,55%	4,38%	4,12%	2,76%	4,15%	1,57%	4,82%	1,21%	0,00%	0,00%	0,00%	0,00%	0,00%	2,82%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

(c) Fast progression group.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3	FVC	MIP	MEP	PhrenMeanAmpl
P1	—	40,43%	40,51%	15,69%	17,99%	25,70%	25,82%	35,25%	21,61%	38,24%	32,32%	50,00%	56,92%	36,89%	26,89%	29,03%	42,17%
P2	10,58%	—	5,70%	5,88%	5,02%	2,79%	7,14%	2,16%	4,02%	3,92%	5,05%	6,98%	1,54%	8,74%	5,04%	6,99%	3,61%
P3	8,91%	4,79%	—	4,71%	6,28%	3,35%	3,85%	2,16%	4,02%	6,86%	4,04%	2,33%	4,62%	3,88%	3,36%	4,84%	1,20%
P4	5,57%	7,98%	7,59%	—	17,57%	12,29%	10,99%	9,35%	9,05%	6,86%	7,07%	8,14%	4,62%	7,77%	10,92%	10,22%	10,84%
P5	5,99%	6,38%	9,49%	16,47%	—	13,41%	7,14%	7,19%	7,04%	11,76%	5,05%	1,16%	9,23%	8,74%	11,76%	7,53%	6,02%
P6	6,41%	2,66%	3,80%	8,63%	10,04%	—	8,79%	4,32%	5,53%	1,96%	4,04%	5,81%	3,08%	6,80%	5,88%	7,53%	2,41%
P7	6,55%	6,91%	4,43%	7,84%	5,44%	8,94%	—	6,47%	11,06%	3,92%	11,11%	4,65%	1,54%	2,91%	1,68%	3,23%	4,82%
P8	6,82%	1,60%	1,90%	5,10%	4,18%	3,35%	4,95%	—	10,55%	2,94%	6,06%	0,00%	1,54%	2,91%	5,88%	1,61%	2,41%
P9	5,99%	4,26%	5,06%	7,06%	5,86%	6,15%	12,09%	15,11%	—	0,98%	3,03%	6,98%	0,00%	7,77%	13,45%	6,99%	8,43%
P10	5,43%	2,13%	4,43%	2,75%	5,02%	1,12%	2,20%	2,16%	0,50%	—	10,10%	3,49%	9,23%	1,94%	0,84%	0,54%	0,00%
R1	4,46%	2,66%	2,53%	2,75%	2,09%	2,23%	6,04%	4,32%	1,51%	9,80%	—	1,16%	1,54%	1,94%	0,84%	3,76%	0,00%
R2	5,99%	3,19%	1,27%	2,75%	0,42%	2,79%	2,20%	0,00%	3,02%	2,94%	1,01%	—	4,62%	0,00%	0,00%	2,15%	1,20%
R3	5,15%	0,53%	1,90%	1,18%	2,51%	1,12%	0,55%	0,72%	0,00%	5,88%	1,01%	3,49%	—	0,00%	0,00%	0,54%	0,00%
FVC	5,29%	4,79%	2,53%	3,14%	3,77%	3,91%	1,65%	2,16%	4,02%	1,96%	2,02%	0,00%	0,00%	—	2,52%	3,76%	0,00%
MIP	4,46%	3,19%	2,53%	5,10%	5,86%	3,91%	1,10%	5,04%	8,04%	0,98%	1,01%	0,00%	0,00%	2,91%	—	5,38%	3,61%
MEP	7,52%	6,91%	5,70%	7,45%	5,86%	7,82%	3,30%	2,16%	6,53%	0,98%	7,07%	4,65%	1,54%	6,80%	8,40%	—	13,25%
PhrenMeanAmpl	4,87%	1,60%	0,63%	3,53%	2,09%	1,12%	2,20%	1,44%	3,52%	0,00%	0,00%	1,16%	0,00%	0,00%	2,52%	5,91%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table A.10: Correlations between variables of the sdtDBNs after NIV with questions, for each progression group. Each table is normalized per column. The sdtDBNs of Table 7.8 (see Chapter 7) are learned for every progression group.

(a) Slow progression group.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3
P1	—	35,56%	38,16%	12,74%	20,34%	14,75%	20,10%	22,52%	18,31%	25,74%	21,95%	29,50%	29,14%
P2	13,36%	—	7,73%	5,42%	2,12%	6,47%	10,29%	4,58%	6,10%	8,82%	6,34%	3,60%	6,62%
P3	13,19%	7,11%	—	8,94%	4,66%	3,96%	6,37%	2,29%	1,88%	5,88%	6,83%	5,04%	3,31%
P4	7,85%	8,89%	15,94%	—	20,34%	19,06%	10,78%	10,69%	13,15%	13,24%	8,29%	21,58%	16,56%
P5	8,01%	2,22%	5,31%	13,01%	—	10,07%	9,31%	5,34%	8,92%	5,88%	9,27%	5,04%	6,62%
P6	6,84%	8,00%	5,31%	14,36%	11,86%	—	12,25%	12,21%	10,80%	4,41%	12,68%	4,32%	5,96%
P7	6,84%	9,33%	6,28%	5,96%	8,05%	8,99%	—	10,31%	3,29%	4,41%	4,39%	5,76%	3,97%
P8	9,85%	5,33%	2,90%	7,59%	5,93%	11,51%	13,24%	—	20,19%	6,62%	6,83%	7,19%	5,30%
P9	6,51%	5,78%	1,93%	7,59%	8,05%	8,27%	3,43%	16,41%	—	6,62%	7,80%	5,04%	3,31%
P10	5,84%	5,33%	3,86%	4,88%	3,39%	2,16%	2,94%	3,44%	4,23%	—	3,41%	2,16%	9,93%
R1	7,51%	5,78%	6,76%	4,61%	8,05%	9,35%	4,41%	5,34%	7,51%	5,15%	—	9,35%	7,95%
R2	6,84%	2,22%	3,38%	8,13%	2,97%	2,16%	3,92%	3,82%	3,29%	2,21%	6,34%	—	1,32%
R3	7,35%	4,44%	2,42%	6,78%	4,24%	3,24%	2,94%	3,05%	2,35%	11,03%	5,85%	1,44%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

(b) Average progression group.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3
P1	—	16,94%	22,26%	10,25%	14,50%	11,90%	8,71%	11,53%	10,59%	10,05%	10,46%	9,90%	16,57%
P2	12,41%	—	14,24%	7,48%	6,11%	5,24%	7,32%	9,49%	5,08%	12,17%	8,16%	11,46%	8,57%
P3	17,90%	15,64%	—	7,48%	7,25%	7,14%	8,36%	8,47%	4,66%	10,05%	9,18%	12,50%	8,00%
P4	8,83%	8,79%	8,01%	—	21,76%	14,76%	15,68%	9,15%	8,05%	7,94%	11,22%	9,90%	7,43%
P5	9,07%	5,21%	5,64%	15,79%	—	14,76%	6,62%	5,08%	8,47%	5,29%	5,87%	4,69%	2,86%
P6	5,97%	3,58%	4,45%	8,59%	11,83%	—	11,50%	2,71%	11,44%	2,65%	3,32%	3,65%	2,29%
P7	5,97%	6,84%	7,12%	12,47%	7,25%	15,71%	—	14,24%	12,71%	6,88%	4,34%	3,13%	6,86%
P8	8,11%	9,12%	7,42%	7,48%	5,73%	3,81%	14,63%	—	23,73%	0,53%	9,44%	9,38%	2,29%
P9	5,97%	3,91%	3,26%	5,26%	7,63%	12,86%	10,45%	18,98%	—	1,59%	5,61%	4,17%	1,71%
P10	4,53%	7,49%	5,64%	4,16%	3,82%	2,38%	4,53%	0,34%	1,27%	—	9,44%	4,69%	20,00%
R1	9,79%	10,42%	10,68%	12,19%	8,78%	6,19%	5,92%	12,54%	9,32%	19,58%	—	26,04%	22,86%
R2	4,53%	7,17%	7,12%	5,26%	3,44%	3,33%	2,09%	6,10%	3,39%	4,76%	12,76%	—	0,57%
R3	6,92%	4,89%	4,15%	3,60%	1,91%	1,90%	4,18%	1,36%	1,27%	18,52%	10,20%	0,52%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

(c) Fast progression group.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	R1	R2	R3
P1	—	29,18%	32,39%	25,27%	31,51%	32,20%	27,94%	31,82%	43,56%	32,63%	21,09%	20,51%	25,00%
P2	13,44%	—	15,49%	9,14%	5,48%	5,08%	3,68%	13,64%	3,96%	15,79%	12,24%	18,80%	16,96%
P3	13,64%	14,16%	—	4,30%	7,53%	4,24%	9,56%	7,58%	5,94%	8,42%	12,24%	11,97%	16,07%
P4	9,29%	7,30%	3,76%	—	17,81%	14,41%	10,29%	12,88%	3,96%	8,42%	7,48%	10,26%	4,46%
P5	9,09%	3,43%	5,16%	13,98%	—	12,71%	8,09%	4,55%	7,92%	3,16%	3,40%	2,56%	3,57%
P6	7,51%	2,58%	2,35%	9,14%	10,27%	—	6,62%	1,52%	13,86%	3,16%	2,04%	4,27%	0,89%
P7	7,51%	2,15%	6,10%	7,53%	7,53%	7,63%	—	11,36%	7,92%	3,16%	6,80%	2,56%	6,25%
P8	8,30%	7,73%	4,69%	9,14%	4,11%	1,69%	11,03%	—	1,98%	4,21%	4,08%	5,98%	2,68%
P9	8,70%	1,72%	2,82%	2,15%	5,48%	11,86%	5,88%	1,52%	—	0,00%	5,44%	0,85%	1,79%
P10	6,13%	6,44%	3,76%	4,30%	2,05%	2,54%	2,21%	3,03%	0,00%	—	7,48%	5,13%	2,68%
R1	6,13%	7,73%	8,45%	5,91%	3,42%	2,54%	7,35%	4,55%	7,92%	11,58%	—	10,26%	12,50%
R2	4,74%	9,44%	6,57%	6,45%	2,05%	4,24%	2,21%	5,30%	0,99%	6,32%	8,16%	—	7,14%
R3	5,53%	8,15%	8,45%	2,69%	2,74%	0,85%	5,15%	2,27%	1,98%	3,16%	9,52%	6,84%	—
Sum per column:	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

A.3 Clinically relevant outcomes

Table A.11: Top 5 variables associated to NIV, in the datasets with questions, for each subset of patients.

Progression group		**Slow**	**Average**			**Fast**	
NIV application		**Any time**	**Any time**	**2nd year**	**After 2nd year**	**1st year**	**2nd year**
Timesteps considered for correlations with NIV variable		{0,...,10}	{0,...,10}	{4,...,8}	{8,9,10}	{0,...,4}	{4,...,8}
Top 5 variables correlated with NIV variable	1st	P1	BMI	P1	P1	P1	P1
	2nd	P6	P9 MEP	BMI	BMI	BMI	Familiar History MND BMI El Escorial criteria
	3rd	BMI Disease duration		P4	Disease duration	Age at onset	
	4th		P1	Disease duration	El Escorial criteria	P4 P9	
	5th	P4	P6	P5 El Escorial criteria	P4		P2 Gender

Table A.12: Top 10 most important variables in each year of the patients' progression, using the sdtDBNs with questions. Patients are divided into slow, average and fast progressors (see Section 7.3 of Chapter 7).

Year		**1st year**			**2nd year**			**After 2nd year**	
Progression group		**Slow**	**Average**	**Fast**	**Slow**	**Average**	**Fast**	**Slow**	**Average**
Top 10 influential variables	1st	Disease duration	P9	P1	Disease duration	P1	P1	Disease duration	P1
	2nd	P9	MEP	BMI	P9	P9	Age at onset	P1	P4
	3rd	MEP	BMI	P5	MEP	BMI	BMI	P4	Disease duration
	4th	P6	P5	P9	P5	P4	P4	P9	P6
	5th	P5 MIP	Disease duration	P4	P1	P5	Disease duration	P5 P6	P5
	6th		P6	MEP	P6	Disease duration	P2		BMI Age at onset
	7th	P1	P1	P6	P4	P6	Gender	P8	
	8th	BMI	El Escorial criteria	P7	BMI	P8	El Escorial criteria	BMI	P8 P9
	9th	P8	P8	C9orf72	P8	MEP	P3	P7	
	10th	P4	P4	P2 Age at onset	P7	El Escorial criteria	P5	P2	P2

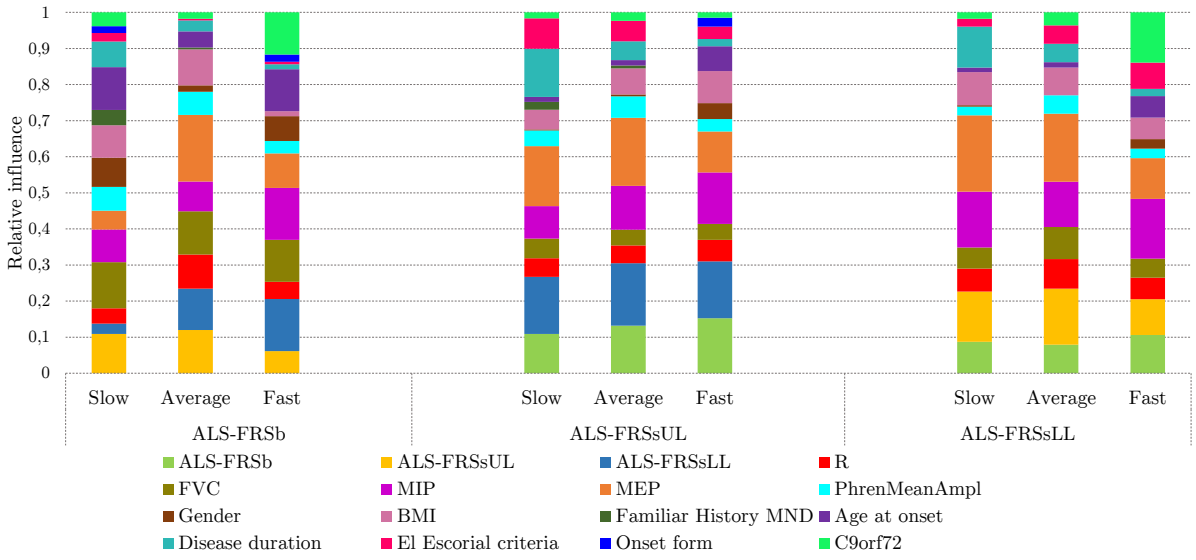


Figure A.6: Correlations of the sub-score with the lowest value in the first consultation with the remaining variables throughout the patients' progression, grouping the columns according to each sub-score with the lowest value in the first consultation.