A new class of generative classifiers based on staged tree models

Federico Carli[a], Manuele Leonelli[b], Gherardo Varando[c]

[a]*Dipartimento di Matematica, Universita degli Studi di Genova, Genova, Italy*
[b]*School of Science and Technology, IE University, Madrid, Spain*
[c]*Image Processing Laboratory, Universitat de Valencia, Valencia, Spain*

Abstract

Generative models for classification use the joint probability distribution of the class variable and the features to construct a decision rule. Among generative models, Bayesian networks and naive Bayes classifiers are the most commonly used and provide a clear graphical representation of the relationship among all variables. However, these have the disadvantage of highly restricting the type of relationships that could exist, by not allowing for context-specific independence. Here we introduce a new class of generative classifiers, called staged tree classifiers, which formally account for context-specific independence. They are constructed by a partitioning of the vertices of an event tree from which conditional independence can be formally read. The naive staged tree classifier is also defined, which extends the classic naive Bayes classifier whilst retaining the same complexity. An extensive simulation study shows that the classification accuracy of staged tree classifiers is competitive with that of state-of-the-art classifiers and an example showcases their use in practice.

Keywords: Bayesian networks, Model selection, Staged trees, Statistical classification

1. Introduction

Statistical classification aims to assign labels to instances described by a vector of feature variables. The classification task is guided by a statistical model learned using data containing labeled instances. The array of models designed to perform classification is constantly increasing and includes,

among others, random forests (Ho, 1995), recursive partitioning (Breiman et al., 1984), and probabilistic neural networks (Specht, 1990).

Bayesian network classifiers (BNCs) (Bielza and Larrañaga, 2014; Friedman et al., 1997) are special types of Bayesian networks (BNs) designed for classification problems. These have been applied to a wide array of real-world applications with competitive classification performance against state-of-the-art classifiers (Flores et al., 2012). There are numerous advantages associated with BNCs. First, they provide an explicit and intuitive representation of the relationship among features represented by a graph. Second, they are a fully coherent probabilistic model thus giving uncertainty measures about the chosen labels. Third, many of the methods and algorithms developed for general BNs can be simply adapted and used for BNCs (see e.g. Benjumeda et al., 2019). Last, they are implemented in various pieces of user-friendly software (see e.g. the `bnclassify` `R` package of Mihaljevic et al., 2018). More generally, BNCs are generative classifiers that give an estimate of the joint distribution of both the features and the class.

One of the main limitations of BNs is that they can only explicitly represent symmetric conditional independences among variables of interest. However, in many applied domains, conditional independences are context-specific, meaning that they only hold for specific instantiations of the conditioning variables. For this reason, numerous extensions of BNs have been proposed that can take into account asymmetric independences (Boutilier et al., 1996; Cano et al., 2012; Jaeger et al., 2006; Pensar et al., 2015, 2016; Poole and Zhang, 2003). Except for Jaeger et al. (2006) and Pensar et al. (2015), all these models somehow lose the intuitiveness of BNs since they cannot represent all the models' information in a unique graph.

Despite the efforts to accommodate asymmetries in BNs, the development of BNCs embedding context-specific information has been limited. Solutions proposed to address this issue use the idea of Bayesian multinets (Geiger and Heckerman, 1996), which consist of several networks each associated with a subset of the domain of one variable, often called distinguished. Bayesian multinets have been used for classification in Friedman et al. (1997), Gurwicz and Lerner (2006), Huang et al. (2003) and Hussein and Santos (2004), among others.

In this paper, a novel class of generative classifiers based on staged trees (Collazo et al., 2018; Smith and Anderson, 2008) embedding asymmetric conditional independences is considered and henceforth called *staged tree classifiers*. Staged trees are defined as probability trees (Shafer, 1996) embellished

with asymmetric conditional independence information, represented by a partitioning of the vertices of the tree. Staged tree classifiers are staged trees whose topology is specifically designed for classification problems. Although staged trees have been used in a variety of applications, including the modeling of health problems (Barclay et al., 2013; Keeble et al., 2017) and criminal activities (Collazo and Smith, 2016), their specific use for classification problems has been limited.

Staged tree classifiers are generative classifiers which, whilst extending the class of BNCs to deal with asymmetric conditional independences, share the same advantages of BNCs: first, the relationship between the random variables is still intuitively depicted in a unique graph; second, they are fully coherent probabilistic models; third, learning algorithms already defined for staged trees (e.g. Freeman and Smith, 2011; Silander and Leong, 2013; Leonelli and Varando, 2022a,b) can simply be adapted for classification purposes; fourth, the freely-available `R` package `stagedtrees` (Carli et al., 2022) gives an implementation of a variety of learning algorithms and inferential routines to apply the methods in practice.

Mirroring the theory of BNCs, staged tree classifiers of different complexity are discussed and their properties are investigated. Notably, the naive staged tree classifier is introduced which is shown to have the same complexity as the standard naive Bayes classifier. However, this model class relaxes the strict conditional independence assumptions of naive Bayes. Experimental studies demonstrate that staged tree classifiers have comparable classification rates to state-of-the-art classifiers and outperform other generative classifiers.

The paper is structured as follows. Section 2 introduces the notation and reviews BNCs. Staged trees are reviewed in Section 3. Staged tree classifiers are introduced and studied in Section 4. Section 5 discusses learning algorithms. Section 6 presents an experimental study and Section 7 discusses a classification application in detail. The paper is concluded with a discussion.

2. Bayesian network classifiers

2.1. The Bayesian network model

Let $G = ([p], E_G)$ be a directed acyclic graph (DAG) with vertex set $[p] = \{1, \ldots, p\}$ and edge set E_G. Let $\boldsymbol{X} = (X_i)_{i \in [p]}$ be categorical random variables with joint mass function P and sample space $\mathbb{X} = \times_{i \in [p]} \mathbb{X}_i$. For

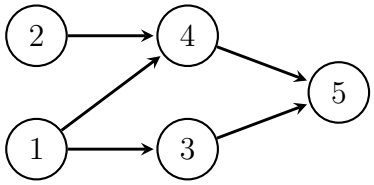


Figure 1: A simple DAG with vertex set $\{1, 2, 3, 4, 5\}$ and edge set $\{(1, 3), (1, 4), (2, 4), (3, 5), (4, 5)\}$.

$A \subset [p]$, we let $\boldsymbol{X}_A = (X_i)_{i \in A}$ and $\boldsymbol{x}_A = (x_i)_{i \in A}$ where $\boldsymbol{x}_A \in \mathbb{X}_A = \times_{i \in A} \mathbb{X}_i$. We say that P is Markov to G if, for $\boldsymbol{x} \in \mathbb{X}$,

$$P(\boldsymbol{x}) = \prod_{k \in [p]} P(x_k | \boldsymbol{x}_{\Pi_k}), \tag{1}$$

where Π_k is the parent set of k in G and $P(x_k|\boldsymbol{x}_{\Pi_k})$ is a shorthand for $P(X_k = x_k | \boldsymbol{X}_{\Pi_k} = \boldsymbol{x}_{\Pi_k})$. Henceforth, we assume the existence of a linear ordering σ of $[p]$ for which only pairs (i, j) where i appears before j in the order can be in the edge set.

The ordered Markov condition implies conditional independences of the form

$$X_i \perp\!\!\!\perp \boldsymbol{X}_{[i-1]} \,|\, \boldsymbol{X}_{\Pi_i}. \tag{2}$$

Definition 1. *Let G be a DAG and P Markov to G. The* Bayesian network *model (associated to G) is*

$$\mathcal{M}_G = \{P \in \Delta_{|\mathbb{X}|-1} \,|\, P \text{ is Markov to } G\}.$$

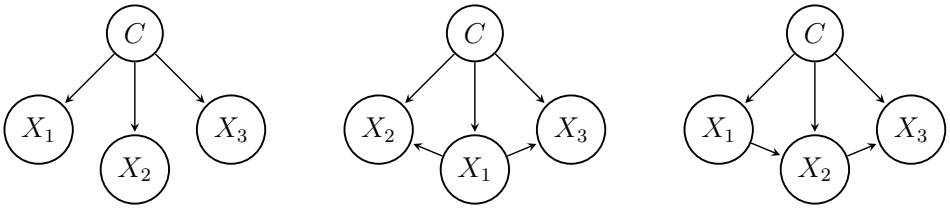
where $\Delta_{|\mathbb{X}|-1}$ is the $(|\mathbb{X}| - 1)$-dimensional probability simplex.

Figure 1 reports a simple DAG with five vertices. Any Markov distribution to this DAG must factorize as $P(x_5|x_3, x_4)P(x_4|x_1, x_2)P(x_3|x_1, x_2)P(x_2)P(x_1)$.

Let $\mathcal{G}$ be the set of DAGs with vertex set $[p]$ and ordering σ. We define the space of BN models over $\boldsymbol{X}$ as $\mathcal{M}_{\mathcal{G}} = \cup_{G \in \mathcal{G}} \mathcal{M}_G$

2.2. Bayesian networks for classification

Suppose now that $\boldsymbol{X} = (X_1, \ldots, X_p)$ is a p-dimensional vector of categorical feature variables and C a categorical class variable with sample space $\mathbb{C}$ and $c \in \mathbb{C}$. Given a training set of labeled observations $\mathcal{D} = \{(\boldsymbol{x}^1, c^1), \ldots,$



(a) Naive BNC (b) SPODE BNC (c) TAN BNC

Figure 2: Examples of BNCs with three features and one class.

$(\boldsymbol{x}^N, c^N)\}$, where $\boldsymbol{x}^i \in \mathbb{X}$ and $c^i \in \mathbb{C}$, a generative classifier aims to learn a joint probability $p(c, \boldsymbol{x})$ and assign a non-labeled instance $\boldsymbol{x}$ to the most probable a posteriori class found as

$$\arg\max_{c \in \mathbb{C}} p(c|\boldsymbol{x}) = \arg\max_{c \in \mathbb{C}} p(c, \boldsymbol{x}).$$

Such a classifier is referred to as *Bayes classifier*.

BNCs are Bayes classifiers that factorize $p(c, \boldsymbol{x})$ according to a BN over the variables $X_1, \ldots, X_p$ and C. Although any BN model could be used for classification purposes, most often the underlying DAG is restricted so that the class variable C has no parents. Therefore, in BNCs the class variable is the root of the DAG and there is a direct link from C to X_i, for $i \in [p]$, since otherwise features not connected to the class would not provide any information for classification. The simplest possible model is the so-called naive Bayes classifier (Minsky, 1961) which assumes the features are conditionally independent given the class (Figure 2a). BNCs of increasing complexity can then be defined by adding dependencies between the feature variables. For instance, the super-parent-one-dependence-estimator (SPODE) BNC (Keogh and Pazzani, 2002) assumes there is a feature parent of all others (Figure 2b). Another commonly used classifier is the tree-augmented naive (TAN) BNC (Friedman et al., 1997) for which each feature has at most two parents: the class and possibly another feature (Figure 2c).

Although BNCs of any complexity can be learned and used in practice, empirical evidence demonstrates that model complexity does not necessarily imply better classification accuracy (Bielza and Larrañaga, 2014). Despite their simplicity, naive BNCs have been shown to lead to good accuracy in classification problems (Bielza and Larrañaga, 2014; Flores et al., 2012).

Alongside these empirical evaluations of BNCs, theoretical studies about the expressiveness of such models have appeared. Recently, Varando et al.

(2015) and Varando et al. (2016) fully characterized the decision functions induced by various BNCs and consequently derived bounds for their expressive power. They built on the work of Ling and Zhang (2002) that demonstrated that any BNC whose vertices have at most k parents cannot represent any decision function containing $(k+1)$-XORs (also known as parity functions O'Donnell, 2014). Thus naive BNCs are not capable of capturing any 2-XORs. On the positive side, Domingos and Pazzani (1997) demonstrated that naive BNCs are optimal under a $0-1$ loss even when the assumption of conditional independence among features does not hold.

3. Staged trees

Because of the strict assumptions on the symmetry of independence among variables in BNs, models accommodating asymmetric conditional independence statements have been developed. Staged trees are one extension of BNs whose conditional independences can be directly read from the associated graphical representation (Collazo et al., 2018; Smith and Anderson, 2008). However, differently to BNs, whose graphical representation is a DAG, staged trees are constructed from probability trees as detailed next.

3.1. X-compatible staged trees

Consider a p-dimensional random vector $\boldsymbol{X}$ taking values in the product sample space $\mathbb{X}$. Let (V, E) be a directed, finite, rooted tree with vertex set V, root node v_0, and edge set E. For each $v \in V$, let $E(v) = \{(v, w) \in E\}$ be the set of edges emanating from v and $\mathcal{L}$ be a set of labels.

Definition 2. *An* $\mathbf{X}$*-compatible staged tree is a triple* (V, E, θ)*, where* (V, E) *is a rooted directed tree and:*

1. $V = v_0 \cup \bigcup_{i\in[p]} \mathbb{X}_{[i]}$*;*
2. *For all* $v, w \in V$*,* $(v, w) \in E$ *if and only if* $w = \boldsymbol{x}_{[i]} \in \mathbb{X}_{[i]}$ *and* $v = \boldsymbol{x}_{[i-1]}$*, or* $v = v_0$ *and* $w = x_1$ *for some* $x_1 \in \mathbb{X}_1$*;*
3. $\theta : E \to \mathcal{L}^* = \mathcal{L} \times \cup_{i\in[p]} \mathbb{X}_i$ *is a labelling of the edges such that* $\theta(v, \boldsymbol{x}_{[i]}) = (\kappa(v), x_i)$ *for some function* $\kappa : V \to \mathcal{L}$*. The function* k *is called the coloring of the staged tree* T*.*

If $\theta(E(v)) = \theta(E(w))$ *then* v *and* w *are said to be in the same* stage.

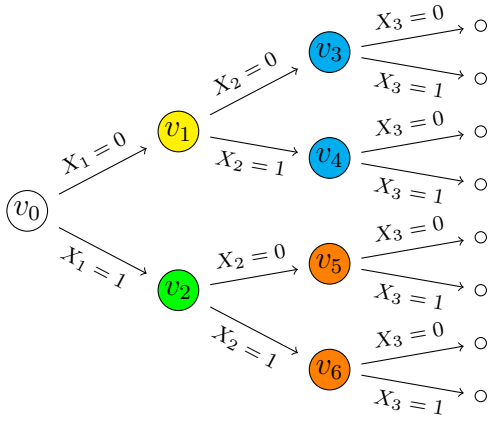


Figure 3: An example of an **X**-compatible staged tree.

Therefore, the equivalence classes induced by $\theta(E(v))$ form a partition of the internal vertices of the tree in *stages*.

Definition 2 first constructs a rooted tree where each root-to-leaf path, or equivalently each leaf, is associated with an element of the sample space $\mathbb{X}$. Then a labeling of the edges of such a tree is defined where labels are pairs with one element from a set $\mathcal{L}$ and the other from the sample space $\mathbb{X}_i$ of the corresponding variable X_i in the tree. By construction, **X**-compatible staged trees are such that two vertices can be in the same stage if and only if they correspond to the same sample space. Although staged trees can be more generally defined without imposing this condition (see e.g. Collazo et al., 2018), henceforth, and as common in practice, we focus on **X**-compatible staged trees only (see Leonelli, 2019, for an example of a non **X**-compatible tree).

Figure 3 reports an (X_1, X_2, X_3)-compatible staged tree over three binary variables. The *coloring* given by the function κ is shown in the vertices and each edge $(\cdot, (x_1, \ldots, x_i))$ is labeled with $X_i = x_i$. The edge labeling θ can be read from the graph by combining the text label and the color of the emanating vertex. The staging of the staged tree in Figure 3 is given by the partition $\{v_0\}$, $\{v_1\}$, $\{v_2\}$, $\{v_3, v_4\}$ and $\{v_5, v_6\}$.

The parameter space associated to an **X**-compatible staged tree $T = (V, E, \theta)$ with labeling $\theta : E \to \mathcal{L}^*$ is defined as

$$\Theta_T = \left\{ \boldsymbol{y} \in \mathbb{R}^{|\theta(E)|} \mid \forall\, e \in E, y_{\theta(e)} \in (0,1) \text{ and } \sum_{e \in E(v)} y_{\theta(e)} = 1 \right\} \qquad (3)$$

Equation (3) defines a class of probability mass functions over the edges emanating from any internal vertex coinciding with conditional distributions $P(x_i|\boldsymbol{x}_{[i-1]})$, $\boldsymbol{x} \in \mathbb{X}$ and $i \in [p]$. In the staged tree in Figure 3 the staging $\{v_3, v_4\}$ implies that the conditional distribution of X_3 given $X_1 = 0$, and $X_2 = 0$, represented by the edges emanating from v_3, is equal to the conditional distribution of X_3 given $X_1 = 0$ and $X_2 = 1$. A similar interpretation holds for the staging $\{v_5, v_6\}$. This in turn implies that $X_3 \perp\!\!\!\perp X_2|X_1$, thus illustrating that the staging of a tree is associated with conditional independence statements.

Let $\boldsymbol{l}_T$ denote the leaves of a staged tree T. Given a vertex $v \in V$, there is a unique path in T from the root v_0 to v, denoted as $\lambda(v)$. The number of edges in $\lambda(v)$ is called the distance of v, and the set of vertices at distance k is denoted by V_k. For any path λ in T, let $E(\lambda) = \{e \in E : e \in \lambda\}$ denote the set of edges in the path λ.

Definition 3. *The* staged tree model $\mathcal{M}_{T,\theta}$ *associated to the* $\mathbf{X}$*-compatible staged tree* (V, E, θ) *is the image of the map*

$$\begin{aligned} &\phi_T : \Theta_T \to \Delta^{\circ}_{|\boldsymbol{l}_T|-1}; \\ &\phi_T : y \mapsto p_l = \Big(\prod_{e \in E(\lambda(l))} y_{\theta(e)} \Big)_{l \in \boldsymbol{l}_T} \end{aligned} \tag{4}$$

Therefore, staged tree models are such that atomic probabilities are equal to the product of the edge labels in root-to-leaf paths and coincide with the usual factorization of mass functions via recursive conditioning.

Let Θ be the set of functions θ from E to $\mathcal{L}^*$, that is all possible partitions, or staging, of the staged tree. We define $\mathcal{M}_T = \cup_{\theta \in \Theta} \mathcal{M}_{T,\theta}$. So as $\mathcal{M}_{\mathcal{G}}$ is the union of all possible BN models given a specific ordering, $\mathcal{M}_T$ is the union of all possible staged tree models, that is of all possible stagings, given a specific ordering of the variables.

3.2. Staged trees and Bayesian networks

In Smith and Anderson (2008) it is demonstrated that any BN can be represented as an equivalent staged tree, whilst the converse is not true. We next illustrate how to construct a staged tree equivalent to a BN. Consider a DAG G and an $\mathbf{X}$-compatible staged tree with vertex set V, edge set E and labeling θ defined via the coloring $\kappa(\boldsymbol{x}_{[i]}) = \boldsymbol{x}_{\Pi_i}$ of the vertices. The staged tree T_G, with vertex set V, edge set E and labeling θ so constructed, is called

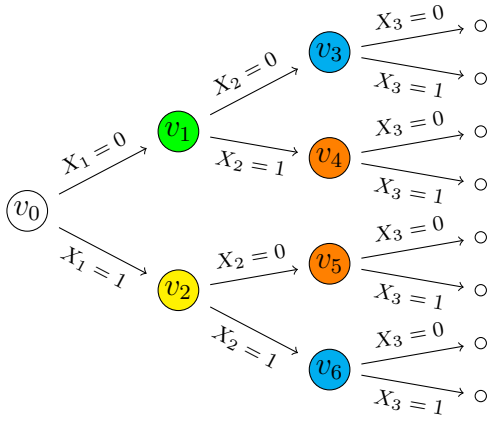


Figure 4: An example of a staged tree not associated to any BN G.

the staged tree model of G. Importantly, $\mathcal{M}_G = \mathcal{M}_{T_G,\theta}$, i.e. the two models are the same since they entail the same factorization of the joint probability (Varando et al., 2021). The staging of T_G represents the Markov conditions associated with the graph G.

For instance, the staged tree in Figure 3 can be constructed as the T_G from the BN with DAG $X_2 \leftarrow X_1 \rightarrow X_3$. Conversely, consider the staged tree in Figure 4. The blue staging implies that the conditional distribution of X_3 given $X_2 = X_1 = 0$ is equal to the conditional distribution of X_3 given $X_2 = X_1 = 1$. Such a constraint cannot be explicitly represented by the DAG of a BN and therefore there is no DAG G such that $\mathcal{M}_G = \mathcal{M}_{T_G,\theta}$, i.e. there is no BN which is equivalent to the staged tree in Figure 4. More generally, it holds that $\mathcal{M}_\mathcal{G} \subset \mathcal{M}_T$ (Varando et al., 2021).

4. Staged tree classifiers

The technology of staged trees has been refined over the years and methods to investigate causal relationships (Genewein et al., 2020; Thwaites et al., 2010), perform statistical inference (Görgen et al., 2015), check model's robustness (Leonelli et al., 2017) and carry out causal discovery (Leonelli and Varando, 2021) are now available. However, the specific use of staged trees for classification has not been investigated in the literature. Thus, just as BNCs have been defined as a specific subclass of BNs whose graph entertains some properties, the class of staged tree classifiers is defined here.

As in Section 2, suppose $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a vector of features and C is the class variable.

Definition 4. *A **staged tree classifier** for the class C and features $\boldsymbol{X}$ is a $(C, \boldsymbol{X})$-compatible staged tree.*

The requirement of C being the root of the tree follows from the idea that in most BNCs the class has no parents, so to maximize the information provided by the features for classification.

4.1. The relationship between BNCs and staged tree classifiers

The BNCs reviewed in Section 2 can now be represented as staged tree classifiers. Since a BNC is a BN with a DAG G, one can construct its equivalent staged tree T_G as in Section 3.2. For instance naive BNCs (Figure 5a), SPODE BNCs (Figure 5b) and TAN BNCs (Figure 5c) can concisely be represented as staged tree classifiers. However, the class of staged trees classifiers is much larger than that of BNCs, as formalized in Proposition 1. For a class C and features $\boldsymbol{X}$, let $\mathcal{M}_{\mathcal{G}}^{\mathrm{C}}$ be the space of BNCs (where C is the root), and $\mathcal{M}_{T}^{\mathrm{C}}$ the space of $(C, \boldsymbol{X})$-compatible staged trees.

Proposition 1. $\mathcal{M}_{\mathcal{G}}^{\mathrm{C}} \subset \mathcal{M}_{T}^{\mathrm{C}}$.

The proof follows from Varando et al. (2021).

In particular naive Bayes, SPODE, and TAN classifiers are all staged tree classifiers with a specific staging structure as described next for the naive Bayes. Recall that the set V_k includes the nodes of the tree at distance k from the root and let $T(v)$ be the subtree of T rooted at a vertex v.

Proposition 2. *Let G be the DAG of a naive BNC. Then T_G is a $(C, \boldsymbol{X})$-compatible staged tree where, for all $v \in V_1$, the subtree $T_G(v)$ is a $\boldsymbol{X}$-compatible staged tree where all nodes at the same distance from the root are in the same stage. In particular, T_G has $|\mathcal{C}|$ stages per each feature.*

4.2. Conditional independence in staged tree classifiers

For the specific task of classification, it is possible to derive two results about the dependence between the features and the class in staged tree classifiers.

Proposition 3. *If all $v \in V_k$ of a staged tree classifier are in the same stage then $(C, X_1, X_2, \ldots, X_{k-1})$ and X_k are marginally independent, i.e. $(C, X_1, \ldots, X_{k-1}) \perp\!\!\!\perp X_k$.*

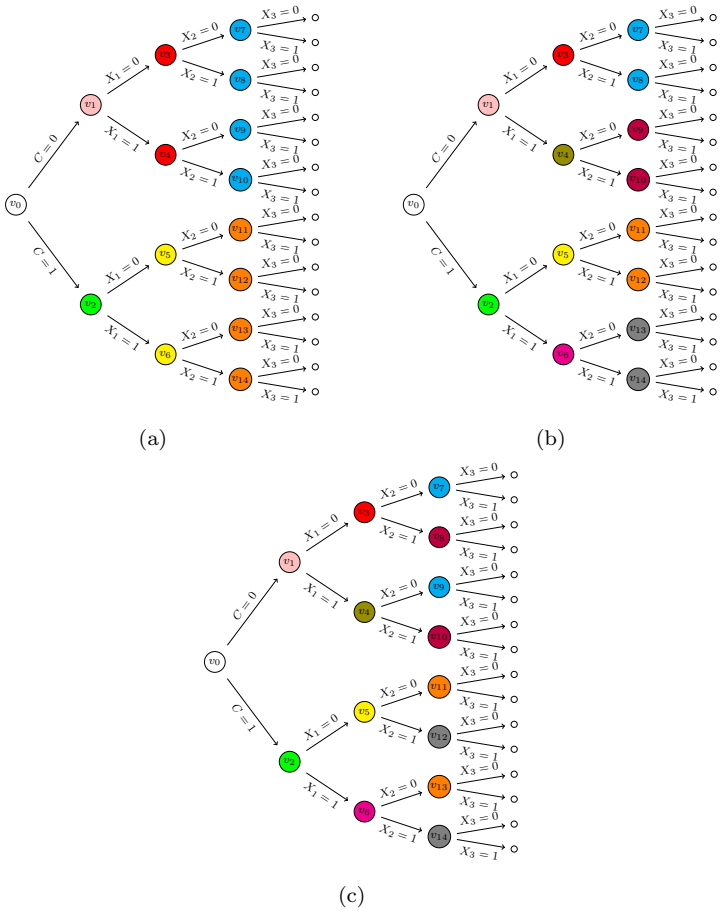


Figure 5: Representation of BNCs as staged trees classifiers: (a): naive BNC in Figure 2a as a staged tree classifier; (b): SPODE BNC in Figure 2b as a staged tree classifier; (c): TAN BNC in Figure 2c as a staged tree classifier. All variables are assumed binary.

Proposition 4. *If for all $v \in V_1$, $T(v)$ has the same stage structure over the vertices at distance $k-2$ from the root, then X_k is independent of C conditionally on $X_1, \ldots, X_{k-1}$, i.e. $C \perp\!\!\!\perp X_k \,|\, X_1, \ldots, X_{k-1}$.*

These two results are illustrated in Figure 6. For instance, consider the features associated with the last random variable in Figure 6b. The vertices in the upper half are framed as the vertices in the bottom half thus implying that the class variable is conditionally independent of the last feature given all others.

4.3. Naive staged tree classifiers

The class of staged tree classifiers is extremely rich and, for any classification task, the number of candidate models that could explain the relationship between class and features increases exponentially. One first common assumption that we make here is to consider only $(C, \boldsymbol{X})$-compatible staged trees, ones where only vertices at the same distance from the root can be in the same stage. However, even with this assumption, the model class of staged tree classifiers is still much richer than that of BNCs. Therefore, just like BNCs whose DAGs have restricted topologies (as SPODE and TAN classifiers) have been studied, next, we introduce a class of simpler staged tree classifiers.

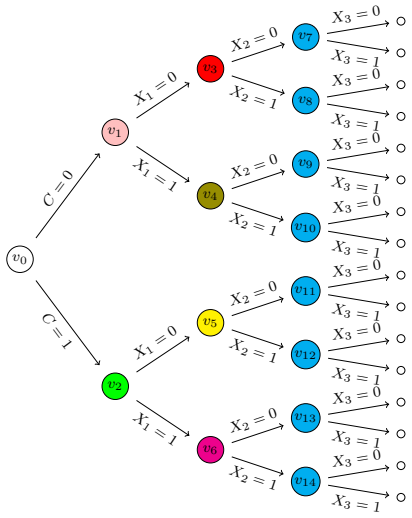
We have discussed that in many practical applications the naive BNC has very good classification performance despite its simplicity. A naive BNC has a total of $\sum_{j=1}^{n} |\mathbb{C}| \cdot (|\mathbb{X}_j| - 1) + |\mathbb{C}| - 1$ free parameters that need to be learned, whilst its DAG is always fixed. Similarly, we introduce a class of staged tree classifiers that has the constraint of having the same number of free parameters as naive BNCs and therefore has the same complexity, whilst being a much richer class of models than the naive BNC.

Definition 5. *A $(C, \boldsymbol{X})$-compatible staged tree classifier such that for every $k \leq p$, the set V_k is partitioned into $|\mathbb{C}|$ stages is called* naive.

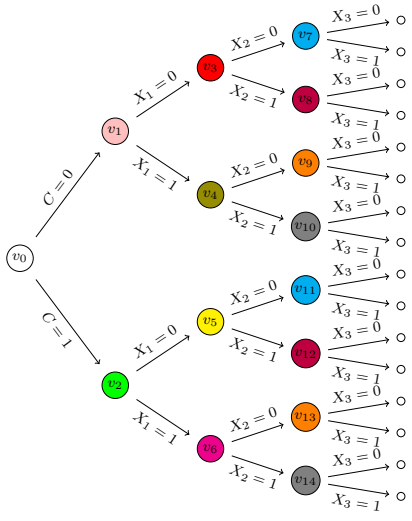
It straightforwardly follows from the definition that naive staged tree classifiers have the same number of free parameters as naive Bayes classifiers.

Despite the strict constraint on the number of parameters, the class of naive staged tree classifiers is still rich and extends naive BNCs in a non-trivial way. More formally, let $\mathcal{M}_{\mathcal{G}}^{\text{naive}}$ and $\mathcal{M}_{T}^{\text{naive}}$ be the space of naive BNCs and naive staged tree classifiers, respectively, for a class C and features $\boldsymbol{X}$.

Proposition 5. $\mathcal{M}_{\mathcal{G}}^{\text{naive}} \subset \mathcal{M}_{T}^{\text{naive}}$.



(a)



(b)

Figure 6: Staged trees classifiers embedding conditional independence statements between features and the class. (a): $X_3 \perp\!\!\!\perp C$; (b): $X_3 \perp\!\!\!\perp C \,|\, X_1, X_2$.

	ST_Naive		RFor		NB	
	-1	1	-1	1	-1	1
-1	0.4623	0.0713	0.3972	0.3123	0.4314	0.3982
1	0.0347	0.4317	0.0998	0.1907	0.0656	0.1048

Table 1: Proportion of predicted instances in the simulated XOR example for the naive staged tree classifier (ST_Naive), random forest (RFor) and naive Bayes (NB).

Differently to naive BNCs, it is not sufficient to simply learn the probabilities of the naive staged tree classifier, but also the staging structure has to be discovered. However, because of the strict restriction on the number of parameters, fast algorithms can be devised to efficiently explore the model space. Notice that in a binary classification problem the set V_k must be partitioned into two subsets.

Critically and differently to naive Bayes classifiers, naive staged trees are capable of representing complex decision rules. For instance, consider the simplest scenario of a binary class with two binary features. The naive staged tree classifier in Figure 4 (assuming the first variable is the class variable), which, as already noticed, does not have a naive BNC representation, is capturing the only 2-XOR present.

To investigate further the capabilities of naive staged tree classifiers in expressing complex decision rules, we simulate $N_{train} = 200$ observations from $n = 10$ binary variables $X_1, \ldots, X_{10}$ taking values in $\mathbb{X} = \{-1, +1\}^n$. We define the class variable as the parity (or XOR) function $C = \prod_{i=1}^{n} X_i$ and compare naive Bayes, random forest and naive staged tree classifiers over $N_{test} = 10000$ test instances, obtaining the results in Table 1. See Section 4.3 for details on the learning of naive staged tree classifiers.

As expected, the Naive Bayes classifier is unable to represent the parity function (Varando et al., 2015) and wrongly classifies the class in more than 40% of the test data. Similar performances are obtained by random forests (implemented with the `randomForest` R package using 500 trees), even if theoretically they have much larger expressive power. Conversely, the naive staged tree correctly learns the parity function with an accuracy of 90%.

5. Learning staged tree classifiers

Learning the structure of a staged tree from data is challenging due to the exponential increase in the size of the tree with the number of random variables. The first learning algorithm used for this purpose was the agglomerative hierarchical clustering proposed by Freeman and Smith (2011). Other learning algorithms were then introduced by Silander and Leong (2013) and Collazo and Smith (2016), among others. Recently, the implementation of a staged tree in `R` was made available in the `stagedtrees` package (Carli et al., 2022) with various searching algorithms to estimate stage structures from data.

In this article, we present different methods for learning staged tree classifiers. All methods follow three main steps: (i) an optimal order of the variables is identified; (ii) the full probability tree based on all random variables is pruned to speed-up computations; (iii) model search heuristics are used to identify high-scoring structures. We next give details about each of these phases. Furthermore, in Section 5.4 we introduce algorithms specifically designed for naive staged tree classifiers.

5.1. Choosing a features' order

It is well known that the ordering of the variables affects the quality of a statistical classifier (e.g. Hruschka Jr and Ebecken, 2007). Thus, choosing the order of the features in a staged tree classifier is expected to be important to obtain good performances.

To confirm this, we perform an empirical study where we compute classification accuracies for different staged tree classifiers over all possible features' orders. Given the combinatorial explosion of the number of orders, we limit this study to two illustrative datasets with a small number of features, namely `puffin` and `monks3` (see Table 2 for details). From Figure 7 we can observe that, as expected, the order of the features is highly relevant for classification performance. For instance, the naive staged tree learned with hierarchical clustering, ST_Naive (as described in Section 5.4), and the staged tree classifier learned with the fast backward-hill climbing algorithm, ST_FBHC (maximizing BIC score, see Section 5.3), exhibit a variety of accuracies (from 0.5 to 1) for the same dataset.

It is therefore critical to couple any algorithm to learn a staged tree with an appropriate method to select a good ordering of the variables. For staged tree classifiers, we tested various ordering heuristics and we propose here

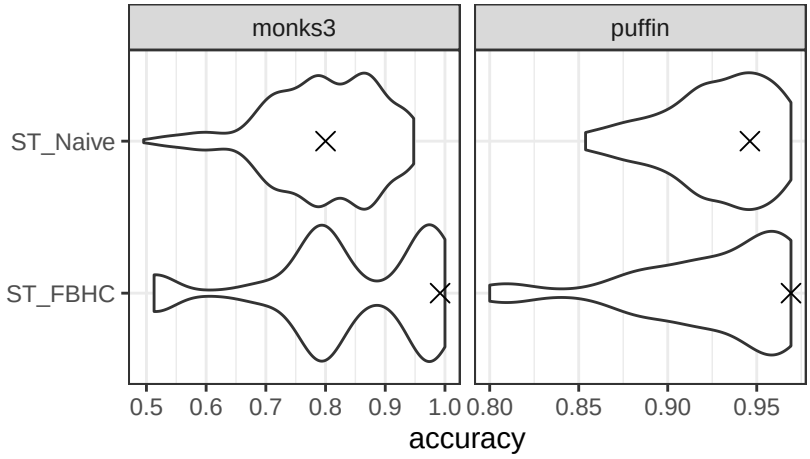


Figure 7: Distribution of accuracy for different orders of features. Accuracies obtained with the CMI ordering are shown with a cross. Results using two learning algorithms for datasets `puffin` and `monks3`.

the use of the conditional mutual information (CMI) criterion. The order of the features given by the CMI criterion is obtained by iteratively selecting the feature that maximizes the conditional mutual information for the class variable, given the features previously selected. The performance of such ordering is shown with a cross symbol in Figure 7. We can observe that in both `puffin` and `monks3` problems, the CMI order leads to a staged tree that perfrom better than the majority of the possible orders.

5.2. Dealing with unobserved instances

Once the order of the variables is selected, some of the nodes in the tree may not be observed in the training data. For such nodes, estimating the associated probabilities is not possible without further assumptions. As implemented in the `R` package `stagedtrees`, for each feature, unobserved situations are joined in a common stage, called *unobserved stage*, and a uniform probability is imposed for their distribution. Furthermore, such unobserved stages are excluded from the stage structure search. This has the effect of drastically reducing the number of nodes for which a stage structure must be learned, thus critically reducing training time.

From a statistical point of view, this does not affect the number of free parameters of the corresponding statistical model, since probability distribu-

tions of these unobserved stages have not been estimated. This procedure can be seen as the usual pruning step of learning algorithms for classification trees. From the naive staged tree classifier perspective, it follows that the number of stages in the tree is unaffected, since its complexity, i.e. number of free parameters, does not change.

5.3. Learning algorithms

Different heuristic algorithms can be used to search the space of possible stage structures. We rely on the available implementations in the `stagedtrees` package, in particular, we use greedy approaches that maximize the BIC score (Görgen et al., 2020) or iterative methods that join nodes into the same stages based on the distance between their associated probabilities. More details of the algorithms are described in Carli et al. (2022) and the `stagedtrees` documentation.

5.4. Learning naive staged tree classifiers

As described in Section 4.3, a naive staged tree classifier is a staged tree where the class is the first variable in the tree and nodes at the same distance from the root are assigned to a fixed number of stages equal to the number of possible values of the class (e.g. two for a binary classification problem).

The two algorithms based on clustering (hierarchical and k-means) available in the `stagedtrees` package can be used to learn naive staged tree classifiers. They cluster probabilities into a user-selected number of stages which, for classification, can be fixed to the number of possible values of the class variable.

5.5. Parameter estimation for staged trees classifiers

As for staged tree probability models (Carli et al., 2022), once the order of the variables and the stages structures have been chosen, the parameters, that is the conditional probabilities, have to be estimated from data. The usual maximum-likelihood parameter estimation is the state-of-the-art method, possibly with a smoothing parameter to avoid zero probabilities which could harm classification for unseen instances (Bielza and Larrañaga, 2014).

As with BNCs, different and specific parameter estimation techniques could be envisioned, such as discriminative learning (Roos et al., 2005; Feelders and Ivanovs, 2006; Pernkopf et al., 2011) and weighted schemes (Zhang and

Dataset	# observations	# variables	# atomic events	imbalance measure
asym	100	4	16	0.881
breastCancer	683	10	1024	0.934
chestSim500	500	8	256	0.994
energy1	768	9	1728	1.000
energy2	768	9	1728	1.000
fallEld	5000	4	64	0.888
fertility	100	10	15552	0.529
monks1	432	7	864	1.000
monks2	432	7	864	0.914
monks3	432	7	864	0.998
puffin	69	6	768	0.999
ticTacToe	958	10	39366	0.931
titanic	2201	4	32	0.908
voting	435	17	131072	0.997

Table 2: Details about the 14 datasets included in the experimental study.

Sheng, 2004; Frank et al., 2002). Furthermore, adjusted class prior probabilities could be used to address unbalanced datasets, when different classification errors entail distinct costs (Wong and Tsai, 2021). In this work, we restrict to the classical smoothed maximum-likelihood estimator of conditional probabilities as implemented also in the `stagedtrees` package (Carli et al., 2022). Under the assumption of the correct stage structures, it implies the Bayes classifier rule, which minimizes the probability of misclassification (Devroye et al., 2013).

6. Experimental study

The classification accuracy for binary classification of staged tree classifiers is investigated in a comprehensive simulation study involving 14 datasets, whose details are given in Table 2, which also include the normalized entropy of the class variable as a measure of dataset imbalance. Each dataset is randomly divided ten times in the train set (80% of the data) to learn the classifiers and the test set (remaining 20%) to predict the response. The reported performance measures, area under the curve (AUC), and balanced accuracy, are computed as the mean over the ten replications. The complete results are given in Tables A.3a, A.3b and A.3c, while plots of the results highlight comparisons within distinct set of competing methods as follows.

The code to replicate in full the experiments is available in the public repository `https://github.com/gherardovarando/exp_class_stagedtrees`.

First, nine model search algorithms to learn staged tree classifiers are compared, namely: ST_BHC (backward hill-climbing); ST_BJ_01 (backward joining of vertices that have Kullback-Leibler divergence between their floret probability distributions less than 0.01); ST_BJ_20 (as ST_BJ_01 but with the threshold at 0.20); ST_FBHC (a fast backward hill-climbing where two vertices are joined whenever the score is increased); ST_Full (each vertex is in its own stage); ST_HC_Full (hill-climbing algorithm starting from ST_Full); ST_HC_Indep (hill-climbing algorithm starting from a tree where all vertices associated to the same variable are in the same stage); ST_Naive_HC (naive staged tree learned with hierarchical clustering); ST_Naive_KM (naive staged tree learned with k-means). Further details about these algorithms can be found in Carli et al. (2022). Due to computational restrictions, for ST_HC_Full the model search is restricted to the first five features according to the variable ordering chosen through CMI, whilst for ST_BHC and ST_HC_Indep only the first seven are considered. The vertices corresponding to the remaining variables are still used for classification but left as in the starting tree of the model search.

The results of the experiment are reported in Figure 8, which suggests the following conclusions:

- The ST_Full model (in yellow), which does not require any model search and has the largest number of parameters, has in general lower AUC and balanced accuracy than other staged trees. This highlights the need for a model-based search of simpler models;
- Models based on hill-climbing (in blue) overall perform better than others (in particular ST_HC_Full). This is expected since these are the most refined learning algorithms and, as a consequence, they are also the slowest.
- Models based on backward joining (in green) have a satisfactory performance, often comparable to that of hill-climbing models, whilst being much quicker to learn.
- Naive staged trees (in red) can be learned extremely quickly and whilst often they have a lower performance, there are cases where it is comparable to that of much more complex trees (see e.g. the balanced accuracy for the `titanic` dataset)

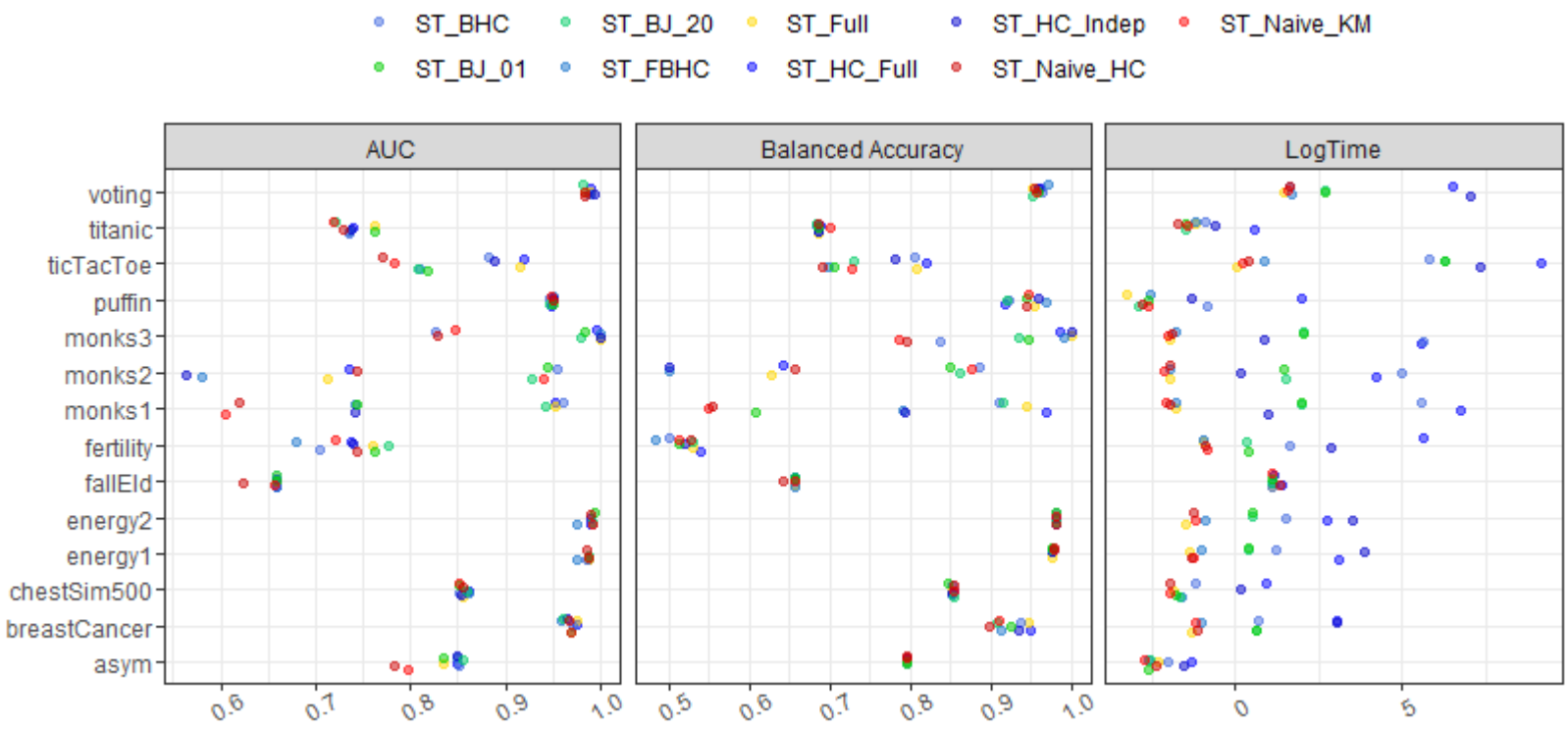


Figure 8: AUC, balanced accuracy and logarithm of time spent for structure learning for nine staged tree classifiers algorithms over fourteen datasets.

Next, we compare staged tree classifiers with their competitor generative classifiers, namely BNCs. For ease of exposition, three representative staged trees are selected (ST_BJ_01, ST_HC_Full and ST_Naive_KM) and three BNCs are fitted: (i) the TAN BNC (BNC_TAN); (ii) the 3-dependence BNC (BNC_KDB); (iii) the naive Bayes (BNC_NB). The results are reported in Figure 9. We can see that for most datasets there is one staged tree classifier (in red) that outperforms BNCs (in blue). Due to the complexity of the models, staged trees are in general slower to learn, but the ST_Naive_KM, due to its simplicity, has learning times comparable to those of generic BNCs.

Figure 10 reports the results of the simulation experiments for three staged tree classifiers (ST_BJ_01, ST_HC_Full, and ST_Naive_KM) as well as other state-of-the-art generative and discriminative classifiers, namely: (i) the naive Bayes (BNC_NB); (ii) the TAN BNC (BNC_TAN); (iii) Classification trees (CTree) (iv) Logistic regression (Logistic); (v) Neural Networks with 20 hidden layers and 0.01 weight decay (NNet); (vi) Random Forests combining 100 classification trees (RFor). Although in some cases discriminative classifiers (in green) outperform staged trees (in red), in many others, they have comparable AUC and balanced accuracy. However, as shown in the next section, staged tree classifiers have the capability of producing an understanding of the relationship between the class and the features since they are generative models. As already noticed, staged trees have an advantage over BNCs (in blue). Although the learning time for generic staged trees is larger, the learning time for naive staged tree classifiers is comparable to that of state-of-the-art classifiers.

Last, we compare the performance of naive Bayes classifiers with that of naive staged tree classifiers in Figure 11. The overall conclusion is that in most cases naive staged trees (in red) outperform naive Bayes in terms of AUC and balanced accuracy. Furthermore, although naive staged trees require more learning time since their structure has to be discovered, these can be learned very quickly and most times in less than one second.

6.1. A simulation study

We additionally perform a simulation study to analyze the performance of staged tree classifiers under different data complexity conditions.

In particular for a range of predictor variables ($p = 2, \ldots, 15$) and predictors' levels ($k = 2, 3$) we generate artificial generative models of a binary response variable with randomly generated staged tree models. For each generated model, we sample 1000 observations for training and 1000 observations

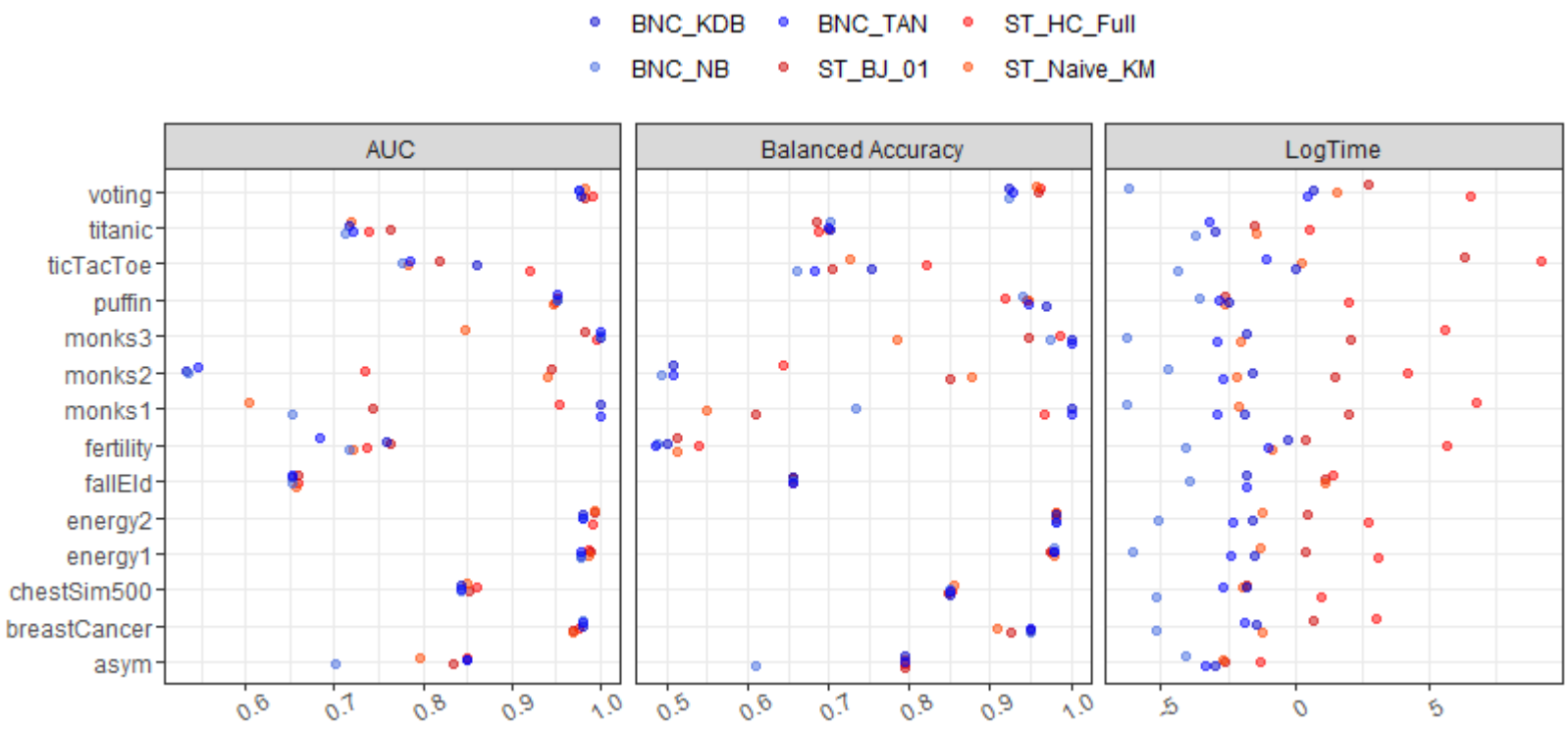


Figure 9: AUC, balanced accuracy and logarithm of time spent for structure learning for three staged tree classifiers (in red) and three BNCs (in blue) over fourteen datasets.

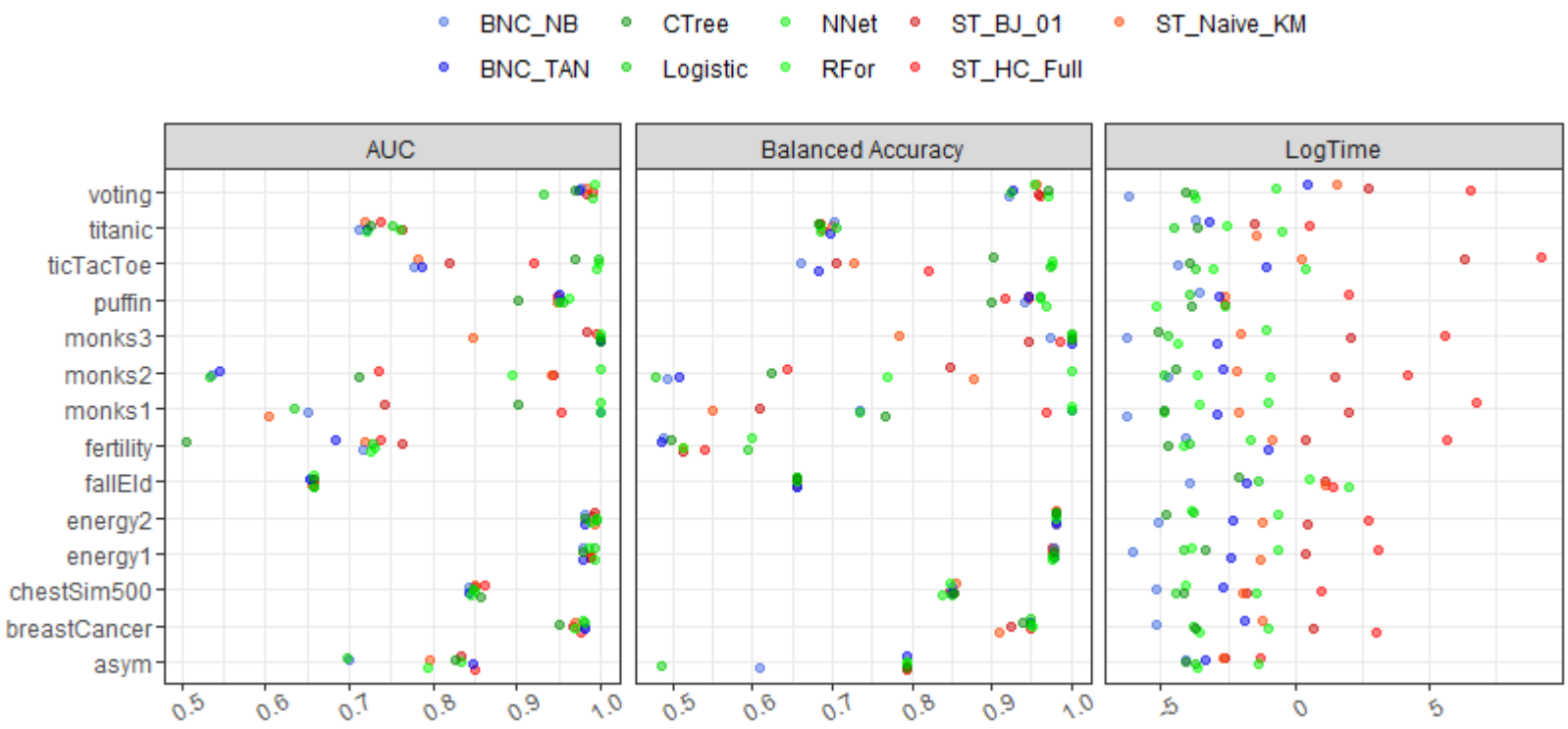


Figure 10: AUC, balanced accuracy and logarithm of time spent for structure learning for three staged tree classifiers (in red), two BNCs (in blue) and other discriminative models (in green) over fourteen datasets.

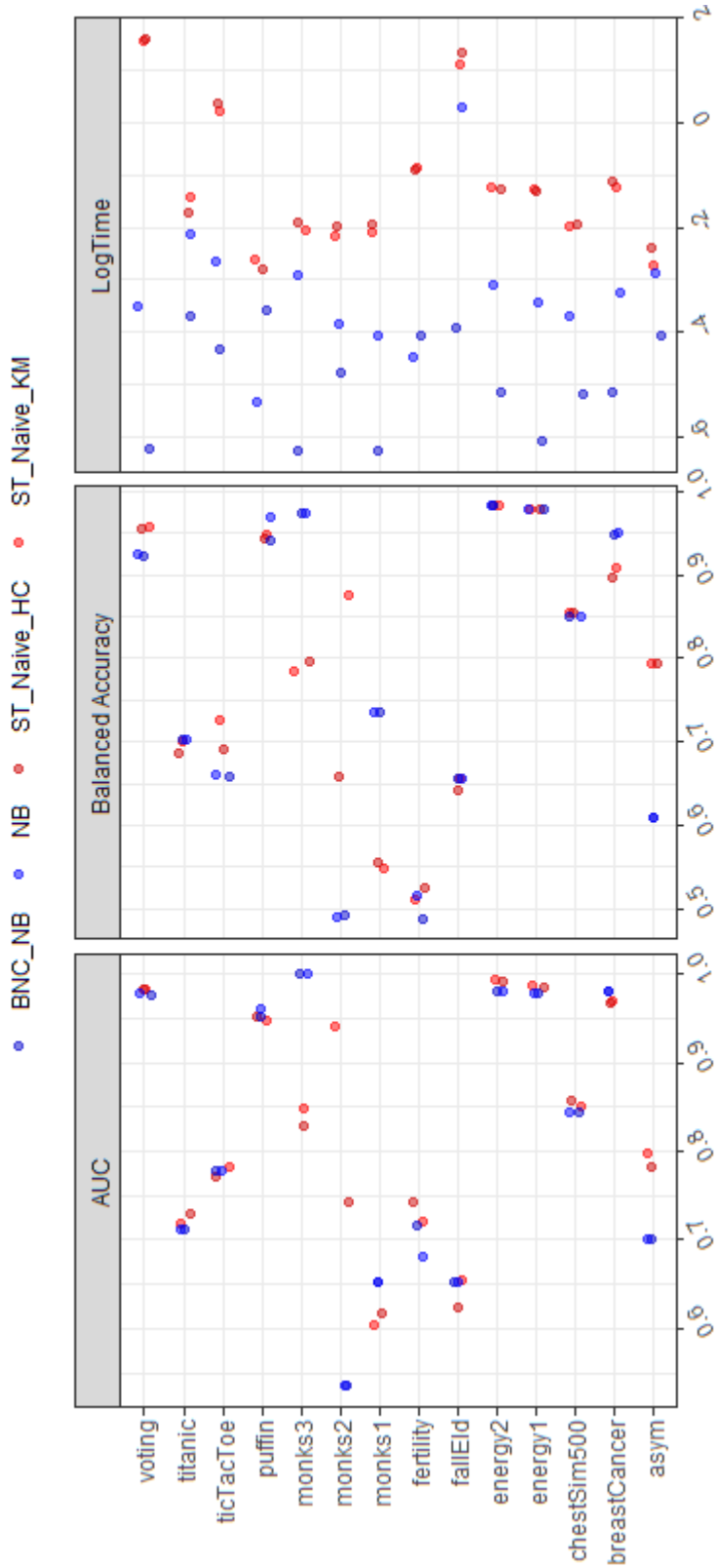


Figure 11: AUC, balanced accuracy and logarithm of time spent for structure learning for two naive staged tree classifiers (in red) and two implementations of naive Bayes classifiers (in blue) over fourteen datasets.

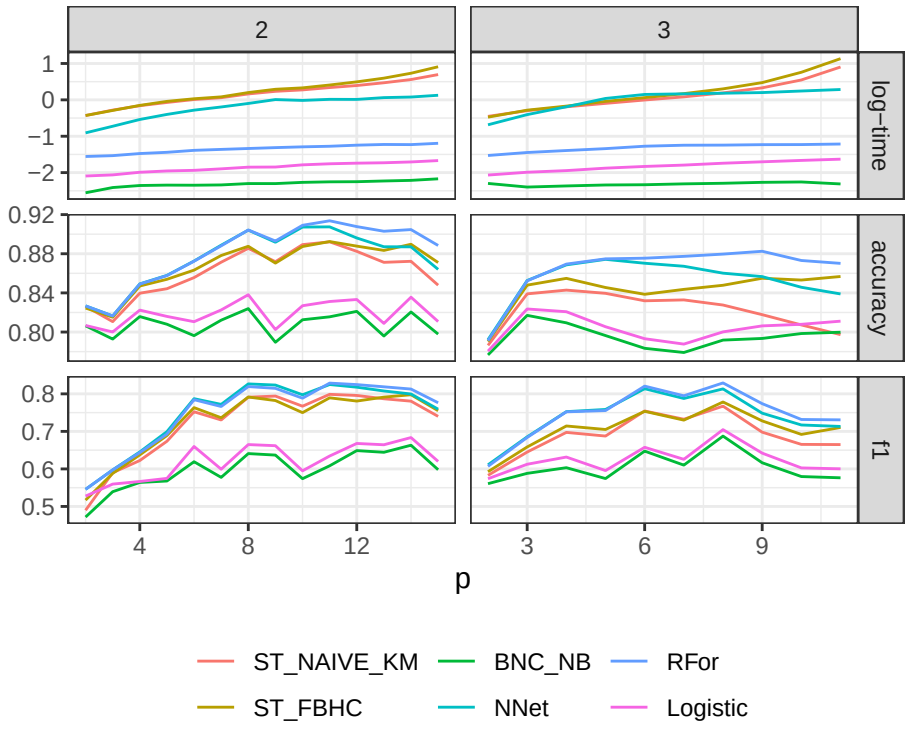


Figure 12: Average results of the simulation experiments, across 100 replications for binary and ternary predictors (columns) and for an increasing number of predictor variables (p). For 2 levels results until $p = 15$ predictors, for ternary predictors only results until $p = 10$ predictor variables are available because of limits on computation time.

for testing. We compare the execution time, accuracy and f1 score for two staged tree algorithms (ST_NAIVE_KM and ST_FBHC) and four competing methods. Average results of the experiment over 100 replications for each combination of simulation parameters are shown in Figure 12.

As the number of predictors increases, the performance of all classifiers, including staged tree classifiers, increases. However, for the larger number of predictors, there is a slight decrease in performance, as already noticed in the literature (Bielza and Larrañaga, 2014). When variables are more complex (i.e. ternary) we see a slight decrease in performance, but this is shared among all classifiers considered. The simulation study confirms that staged tree classifiers outperform other generative classifiers and are competitive with state-of-the-art methods such as artificial neural networks and random

forest classifiers.

7. An example of a staged tree classifier

To illustrate the capabilities of staged tree classifiers we next develop an example classification analysis over the freely available `titanic` dataset, which provides information on the fate of the Titanic passengers. It has three binary variables (Survived, Sex, and Age) and a categorical variable Class taking four levels (1st, 2nd, 3rd, and Crew). The aim is to correctly classify whether the Titanic passengers survived or not based on their gender, age, and traveling class.

From Figure 8 we can see that one of the best staged tree classifiers is the ST_BJ_01 learned using a backward joining of the vertices based on the Kullback-Leibler divergence and a threshold of 0.1. In Figure 13 we report the staged tree classifier ST_BJ_01 learned over the full Titanic dataset using the `R` package `stagedtrees`. By investigating the staging structure we can deduce conditional independence statements relating to the classification variable (Survived) and the features. From stages associated with Class, we can deduce that $P(\text{Class} \,|\, \text{Sex} = \text{Male}, \text{Survived}) = P(\text{Class} \,|\, \text{Sex} = \text{Male})$ since the second and the fourth vertices (starting from the top) are in the same stage. This implies the asymmetric conditional independence

$$\text{Class} \perp\!\!\!\perp \text{Survived} \,|\, \text{Sex} = \text{Male}$$

The complex staging structure over the Age variable also implies asymmetric conditional independences. We can notice that all paths going through an edge labeled Crew are in the same stage for the variable Age. This implies that

$$\text{Age} \perp\!\!\!\perp \text{Survived} \,|\, \text{Class} = \text{Crew}$$

The same conclusion can also be drawn for Class = 3rd.

As an additional illustration in Figure 14 is reported the naive staged tree classifier learned over the full Titanic dataset using the k-means hierarchical clustering algorithm. The staging structure over the variables Sex and Class implies that Sex and Survived are not independent and that Class is conditionally independent of Survived given Sex. The staging structure over the Age variable is a lot more complex describing highly asymmetric constraints on the associated probabilities. Whilst imposing much more flexible

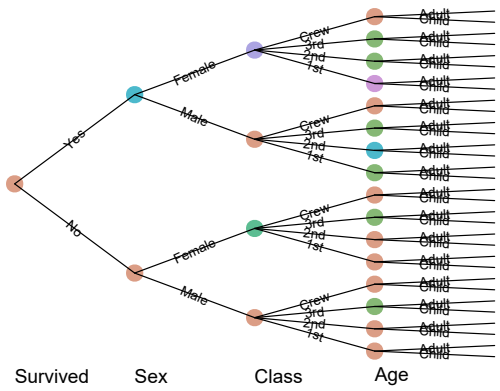


Figure 13: Staged tree classifier ST_BJ_01 learnt over the full Titanic dataset.

dependence structures, the naive staged tree classifier has the same complexity as the naive Bayes classifier, meaning they have the same number of independent parameters.

BNCs of different complexity are also learned over the full `titanic` dataset using the `bnclassify` `R` package. Irrespective of the complexity chosen, the model selection search always returns the simple naive Bayes classifier. Given that staged tree classifiers outperform BNCs in classification measures for the Titanic dataset (see Figure 9), as well as for other datasets, this observation highlights the need for context-specific generative classifiers that can more flexibly model the dependence structure between the classification variable and the features.

8. Discussion

Staged tree classifiers are a highly-expressive new class of generative classifiers with classification performance comparable to that of state-of-the-art classifiers. They embed context-specific conditional independence statements which can be easily read by the staging of the vertices of the tree. These are implemented in the freely available `stagedtrees` `R` package.

A special staged tree classifier is the naive staged tree classifier which, whilst having the same complexity as the naive Bayes classifier, can flexibly represent complex classification rules. Naive staged trees not only relax the

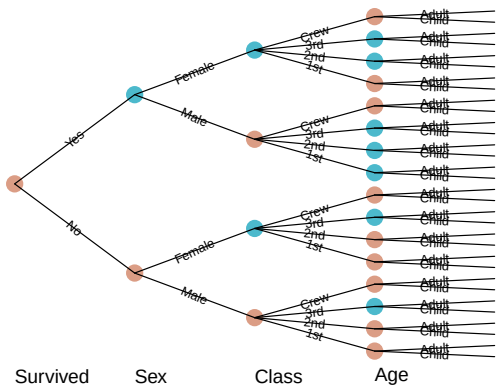


Figure 14: Naive staged tree classifier learnt over the full Titanic dataset using the `stages_kmeans` algorithm.

assumption of conditional independence of the features as in naive Bayes classifiers but also have better performances in classification, as highlighted by the simulation study.

Naive staged trees are learned from data using a clustering algorithm of the probability distributions over the non-leaf vertices of the tree. Such algorithms divide the vertices at the same distance from the root in $|\mathbb{C}|$ stages. More generally, we could devise clustering algorithms where, for each variable, the number of stages is automatically selected according to an optimality criterion. The development of these algorithms is the focus of ongoing research.

Furthermore, many model search algorithms implemented in `stagedtrees` are based on the maximization of a model score, by default the negative BIC. We are currently investigating algorithms based on the minimization of the classification error, as commonly implemented for BNCs.

Acknowledgments

Gherardo Varando's work was partly funded by the European Research Council (ERC) Synergy Grant "Understanding and Modelling the Earth System with Machine Learning (USMILE)" under Grant Agreement No 855187.

References

Barclay, L., Hutton, J., Smith, J., 2013. Refining a Bayesian network using a chain event graph. International Journal of Approximate Reasoning 54, 1300–1309.

Benjumeda, M., Luengo-Sanchez, S., Larranaga, P., Bielza, C., 2019. Tractable learning of Bayesian networks from partially observed data. Pattern Recognition 91, 190–199.

Bielza, A., Larrañaga, P., 2014. Discrete Bayesian network classifiers: a survey. ACM Computing Surveys 47, 5:1–5:43.

Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D., 1996. Context-specific independence in Bayesian networks, in: Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, pp. 115–123.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and regression trees. Wadsworth.

Cano, A., Gómez-Olmedo, M., Moral, S., Pérez-Ariza, C., Salmerón, A., 2012. Learning recursive probability trees from probabilistic potentials. International Journal of Approximate Reasoning 53, 1367–1387.

Carli, F., Leonelli, M., Riccomagno, E., Varando, G., 2022. The R package stagedtrees for structural learning of stratified staged trees. Journal of Statistical Software 102, 1–30.

Collazo, R., Görgen, C., Smith, J., 2018. Chain event graphs. Chapmann & Hall.

Collazo, R., Smith, J., 2016. A new family of non-local priors for chain event graph model selection. Bayesian Analysis 11, 1165–1201.

Devroye, L., Györfi, L., Lugosi, G., 2013. A probabilistic theory of pattern recognition. volume 31. Springer Science & Business Media.

Domingos, P., Pazzani, M., 1997. On the optimality of the simple Bayesian network classifiers. Machine Learning 29, 103–130.

Feelders, A., Ivanovs, J., 2006. Discriminative scoring of Bayesian network classifiers: A comparative study, in: Probabilistic Graphical Models, pp. 75–82.

Flores, M., Gámez, J., Martínez, A., 2012. Supervised classification with Bayesian networks: a review on models and applications, in: Intelligent data analysis for real-life applications: theory and practice, pp. 72–102.

Frank, E., Hall, M., Pfahringer, B., 2002. Locally weighted naive Bayes, in: Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, pp. 249–256.

Freeman, G., Smith, J., 2011. Bayesian MAP model selection of chain event graphs. Journal of Multivariate Analysis 102, 1152–1165.

Friedman, N., Geiger, D., Goldszmidt, M., 1997. Bayesian network classifiers. Machine Learning 29, 131–163.

Geiger, D., Heckerman, D., 1996. Knowledge representation and inference in similarity networks and Bayesian multinets. Artificial Intelligence 82, 45–74.

Genewein, T., McGrath, T., Delétang, G., Mikulik, V., Martic, M., Legg, S., Ortega, P., 2020. Algorithms for causal reasoning in probability trees. arXiv:2010.12237 .

Görgen, C., Leonelli, M., Smith, J., 2015. A differential approach for staged trees, in: European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Springer. pp. 346–355.

Gurwicz, Y., Lerner, B., 2006. Bayesian class-matched multinet classifier, in: Proceedings of the Joint IAPR International Conference on Structural, Syntactic, and Statistical Pattern Recognition, pp. 145–153.

Görgen, C., Leonelli, M., Marigliano, O., 2020. The curved exponential family of a staged tree. Electronic Journal of Statistics 16, 2607–2620.

Ho, T., 1995. Random decision forests, in: Proceedings of the 3rd International Conference on Document Analysis and Recognition, pp. 278–282.

Hruschka Jr, E.R., Ebecken, N.F., 2007. Towards efficient variables ordering for Bayesian networks classifier. Data & Knowledge Engineering 63, 258–269.

Huang, K., King, I., Lyu, M., 2003. Discriminative training of Bayesian Chow-Liu multinet classifiers, in: Proceedings of the International Joint Conference on Neural Networks, pp. 484–488.

Hussein, A., Santos, E., 2004. Exploring case-based Bayesian networks and Bayesian multi-nets for classification, in: Proceedings of the 17th Conference of the Canadian Society of Computational Studies of Intelligence, pp. 485–492.

Jaeger, M., Nielsen, J., Silander, T., 2006. Learning probabilistic decision graphs. International Journal of Approximate Reasoning 42, 84–100.

Keeble, C., Thwaites, P., Baxter, P., Barber, S., Parslow, R., Law, G., 2017. Learning through chain event graphs: The role of maternal factors in childhood type 1 diabetes. American Journal of Epidemiology 186, 1204–1208.

Keogh, E., Pazzani, M., 2002. Learning the structure of augmented Bayesian classifiers. International Journal on Artificial Intelligence Tools 11, 587–601.

Leonelli, M., 2019. Sensitivity analysis beyond linearity. International Journal of Approximate Reasoning 113, 106–118.

Leonelli, M., Görgen, C., Smith, J., 2017. Sensitivity analysis in multilinear probabilistic models. Information Sciences 411, 84–97.

Leonelli, M., Varando, G., 2021. Context-specific causal discovery for categorical data using staged trees. arXiv:2106.04416 .

Leonelli, M., Varando, G., 2022a. Highly efficient structural learning of sparse staged trees. arXiv:2206.06970 .

Leonelli, M., Varando, G., 2022b. Structural learning of simple staged trees. arXiv:2203.04390 .

Ling, C.X., Zhang, H., 2002. The representational power of discrete Bayesian networks. Journal of Machine Learning Research 3, 709–721.

Mihaljevic, B., Bielza, C., Larrañaga, P., 2018. bnclassify: learning discrete Bayesian network classifiers from data. R package version 0.4.0.

Minsky, M., 1961. Steps towards artificial intelligence, in: Computers and Thought, pp. 406–450.

O'Donnell, R., 2014. Analysis of Boolean functions. Cambridge University Press.

Pensar, J., Nyman, H., Koski, T., Corander, J., 2015. Labeled directed acyclic graphs: a generalization of context-specific independence in directed graphical models. Data Mining and Knowledge Discovery 29, 503–533.

Pensar, J., Nyman, H., Lintusaari, J., Corander, J., 2016. The role of local partial independence in learning of Bayesian networks. International Journal of Approximate Reasoning 69, 91–105.

Pernkopf, F., Wohlmayr, M., Tschiatschek, S., 2011. Maximum margin Bayesian network classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 34, 521–532.

Poole, D., Zhang, N., 2003. Exploiting contextual independence in probabilistic inference. Journal of Artificial Intelligence Research 18, 263–313.

Roos, T., Wettig, H., Grünwald, P., Myllymäki, P., Tirri, H., 2005. On discriminative Bayesian network classifiers and logistic regression. Machine Learning 59, 267.

Shafer, G., 1996. The art of causal conjecture. MIT Press.

Silander, T., Leong, T.Y., 2013. A dynamic programming algorithm for learning chain event graphs, in: Proceedings of the International Conference on Discovery Science, pp. 201–216.

Smith, J., Anderson, P., 2008. Conditional independence and chain event graphs. Artificial Intelligence 172, 42 – 68.

Specht, D., 1990. Probabilistic neural networks. Neural Networks 3, 109–118.

Thwaites, P., Smith, J., Riccomagno, E., 2010. Causal analysis with chain event graphs. Artificial Intelligence 174, 889–909.

Varando, G., Bielza, C., Larrañaga, P., 2015. Decision boundary for discrete Bayesian network classifiers. Journal of Machine Learning Research 16, 2725–2749.

Varando, G., Bielza, C., Larrañaga, P., 2016. Decision functions for chain classifiers based on Bayesian networks for multi-label classification. International Journal of Approximate Reasoning 68, 164–178.

Varando, G., Carli, F., Leonelli, M., 2021. Staged trees and asymmetry-labeled dags. arXiv:2108.01994 .

Wong, T.T., Tsai, H.C., 2021. Multinomial naïve Bayesian classifier with generalized Dirichlet priors for high-dimensional imbalanced data. Knowledge-Based Systems 228, 107288.

Zhang, H., Sheng, S., 2004. Learning weighted naive Bayes with accurate ranking, in: Fourth IEEE International Conference on Data Mining (ICDM'04), IEEE. pp. 567–570.

Appendix A. Complete Experiments Results

(a) Accuracy

	Asym	breast_cancer	chestSim500	energy1	energy2	FallEld	fertility	monks1	monks2	monks3	puffin	tic_tac_toe	Titanic	voting
st_full	0.8490	0.9559	0.8490	0.9758	0.9824	0.7668	0.8850	0.9395	0.6919	1.0000	0.9538	0.8424	0.7918	0.9543
st_indep	0.6985	0.7037	0.5340	0.5569	0.5647	0.6925	0.9000	0.4000	0.6849	0.4930	0.6615	0.6356	0.6793	0.5891
st_hc_indep	0.8490	0.9493	0.8510	0.9771	0.9824	0.7668	0.8750	0.7953	0.6791	1.0000	0.9615	0.8141	0.7868	0.9587
st_hc_full	0.8490	0.9574	0.8520	0.9765	0.9824	0.7668	0.8600	0.9651	0.7035	0.9860	0.9154	0.8503	0.7925	0.9609
st_bhc	0.8490	0.9507	0.8510	0.9758	0.9824	0.7668	0.8300	0.9105	0.8651	0.8419	0.9231	0.8215	0.7916	0.9652
st_bj_01	0.8490	0.9404	0.8460	0.9752	0.9824	0.7668	0.8500	0.6081	0.8279	0.9500	0.9462	0.7435	0.7925	0.9587
st_naive_hc	0.8490	0.9199	0.8540	0.9791	0.9824	0.7651	0.8600	0.5535	0.6802	0.8000	0.9462	0.7246	0.7868	0.9543
st_naive_km	0.8490	0.9096	0.8540	0.9791	0.9824	0.7669	0.8500	0.5535	0.8814	0.7988	0.9385	0.7613	0.7766	0.9522
bnc_3db	0.8490	0.9559	0.8480	0.9791	0.9824	0.7669	0.8750	1.0000	0.6733	1.0000	0.9692	0.7948	0.7816	0.9196
bnc_nb	0.6815	0.9559	0.8480	0.9791	0.9824	0.7669	0.8850	0.7372	0.6721	0.9733	0.9385	0.7209	0.7752	0.9196
bnc_tan	0.8490	0.9559	0.8480	0.9791	0.9824	0.7669	0.8750	1.0000	0.6698	1.0000	0.9385	0.7351	0.7816	0.9196
nnet	0.8490	0.9574	0.8390	0.9752	0.9824	0.7668	0.8650	1.0000	1.0000	1.0000	0.9692	0.9838	0.7918	0.9522
rf	0.8490	0.9588	0.8430	0.9765	0.9824	0.7667	0.8800	1.0000	0.8453	1.0000	0.9615	0.9880	0.7925	0.9717
logistic	0.6400	0.9581	0.8480	0.9791	0.9824	0.7667	0.8500	0.7372	0.6535	1.0000	0.9615	0.9812	0.7795	0.9239
ctree	0.8490	0.9441	0.8510	0.9784	0.9824	0.7668	0.9000	0.7605	0.6977	1.0000	0.8923	0.9230	0.7902	0.9696

(b) Balanced accuracy

	Asym	breast_cancer	chestSim500	energy1	energy2	FallEld	fertility	monks1	monks2	monks3	puffin	tic_tac_toe	Titanic	voting
st_full	0.7950	0.9468	0.8510	0.9759	0.9825	0.6555	0.5307	0.9438	0.6272	1.0000	0.9542	0.8074	0.6859	0.9523
st_indep	0.5000	0.5836	0.5010	0.5727	0.5794	0.5000	0.4974	0.4423	0.5000	0.4762	0.6807	0.4915	0.5000	0.5933
st_hc_indep	0.7950	0.9351	0.8529	0.9770	0.9825	0.6556	0.5200	0.7922	0.5008	1.0000	0.9604	0.7820	0.6856	0.9586
st_hc_full	0.7950	0.9491	0.8538	0.9766	0.9825	0.6556	0.5395	0.9681	0.6431	0.9867	0.9177	0.8209	0.6873	0.9614
st_bhc	0.7950	0.9385	0.8528	0.9759	0.9825	0.6556	0.5012	0.9107	0.8850	0.8376	0.9239	0.8066	0.6858	0.9641
st_bj_01	0.7950	0.9250	0.8472	0.9753	0.9825	0.6555	0.5118	0.6086	0.8493	0.9469	0.9458	0.7046	0.6862	0.9592
st_naive_hc	0.7950	0.8977	0.8554	0.9790	0.9825	0.6416	0.5260	0.5548	0.6575	0.7956	0.9437	0.6912	0.6860	0.9558
st_naive_km	0.7950	0.8849	0.8556	0.9790	0.9825	0.6561	0.5677	0.5614	0.8835	0.7936	0.9394	0.7332	0.6992	0.9524
bnc_3db	0.7950	0.9491	0.8498	0.9790	0.9825	0.6561	0.4833	1.0000	0.5132	1.0000	0.9688	0.7520	0.7014	0.9230
bnc_nb	0.6095	0.9491	0.8498	0.9790	0.9825	0.6561	0.4887	0.7347	0.4922	0.9746	0.9406	0.6596	0.7022	0.9230
bnc_tan	0.7950	0.9491	0.8498	0.9790	0.9825	0.6561	0.4973	1.0000	0.5030	1.0000	0.9406	0.6793	0.7014	0.9228
nnet	0.7950	0.9485	0.8404	0.9753	0.9825	0.6555	0.6252	1.0000	1.0000	1.0000	0.9688	0.9764	0.6859	0.9535
rf	0.7950	0.9500	0.8449	0.9766	0.9825	0.6551	0.5137	1.0000	0.7889	1.0000	0.9604	0.9824	0.6862	0.9736
logistic	0.4862	0.9497	0.8501	0.9790	0.9825	0.6557	0.5950	0.7347	0.4790	1.0000	0.9616	0.9741	0.7038	0.9246
ctree	0.7950	0.9401	0.8531	0.9785	0.9825	0.6555	0.4971	0.7678	0.6248	1.0000	0.9010	0.9024	0.6827	0.9709

(c) AUC

	Asym	breast_cancer	chestSim500	energy1	energy2	FallEld	fertility	monks1	monks2	monks3	puffin	tic_tac_toe	Titanic	voting
st_full	0.8347	0.9751	0.8547	0.9884	0.9929	0.6584	0.7594	0.9517	0.7129	1.0000	0.9507	0.9161	0.7627	0.9900
st_indep	0.5208	0.6287	0.6375	0.5653	0.5734	0.4910	0.6862	0.6190	0.5651	0.6391	0.7208	0.5137	0.5796	0.6428
st_hc_indep	0.8493	0.9660	0.8538	0.9883	0.9902	0.6585	0.7390	0.7410	0.5629	1.0000	0.9507	0.8884	0.7365	0.9935
st_hc_full	0.8493	0.9764	0.8611	0.9876	0.9903	0.6583	0.7371	0.9529	0.7352	0.9953	0.9483	0.9196	0.7386	0.9903
st_bhc	0.8515	0.9720	0.8512	0.9864	0.9914	0.6585	0.7036	0.9609	0.9549	0.8255	0.9455	0.8825	0.7364	0.9900
st_bj_01	0.8349	0.9686	0.8513	0.9887	0.9932	0.6583	0.7632	0.7436	0.9438	0.9832	0.9507	0.8190	0.7630	0.9833
st_naive_hc	0.7835	0.9666	0.8562	0.9855	0.9902	0.6244	0.7434	0.6185	0.7435	0.8288	0.9507	0.7715	0.7300	0.9831
st_naive_km	0.7972	0.9627	0.8526	0.9885	0.9924	0.6560	0.7668	0.6138	0.9537	0.8579	0.9479	0.8085	0.7181	0.9825
bnc_3db	0.8489	0.9809	0.8430	0.9786	0.9812	0.6533	0.7107	1.0000	0.5330	1.0000	0.9507	0.8727	0.7166	0.9776
bnc_nb	0.7010	0.9809	0.8439	0.9786	0.9812	0.6533	0.7165	0.6520	0.5355	0.9998	0.9507	0.7772	0.7128	0.9767
bnc_tan	0.8489	0.9809	0.8430	0.9786	0.9812	0.6533	0.7186	1.0000	0.5450	1.0000	0.9507	0.7836	0.7164	0.9761
nnet	0.8341	0.9783	0.8456	0.9940	0.9966	0.6585	0.7168	1.0000	1.0000	1.0000	0.9507	0.9966	0.7622	0.9910
rf	0.7950	0.9771	0.8520	0.9872	0.9894	0.6577	0.7378	1.0000	0.9148	1.0000	0.9688	0.9986	0.7209	0.9940
logistic	0.6975	0.9809	0.8481	0.9944	0.9956	0.6589	0.7276	0.6357	0.5337	1.0000	0.9566	0.9975	0.7525	0.9325
ctree	0.8267	0.9519	0.8579	0.9797	0.9825	0.6579	0.5049	0.9025	0.7117	1.0000	0.9010	0.9703	0.7261	0.9709

Table A.3: Accuracy (a), balanced accuracy (b) and AUC (c) for all the considered classifiers over the datasets in the experiments.