Context-Specific Refinements of Bayesian Network Classifiers

Manuele Leonelli MANUELE.LEONELLI@IE.EDU
School of Science and Technology, IE University, Madrid, Spain

Gherardo Varando GHERARDO.VARANDO@UV.ES
Image Processing Lab, University of Valencia, Valencia, Spain

Editors: J.H.P. Kwisthout & S. Renooij

Abstract

Supervised classification is one of the most ubiquitous tasks in machine learning. Generative classifiers based on Bayesian networks are often used because of their interpretability and competitive accuracy. The widely used naive and TAN classifiers are specific instances of Bayesian network classifiers with a constrained underlying graph. This paper introduces novel classes of generative classifiers extending TAN and other famous types of Bayesian network classifiers. Our approach is based on staged tree models, which extend Bayesian networks by allowing for complex, context-specific patterns of dependence. We formally study the relationship between our novel classes of classifiers and Bayesian networks. We introduce and implement data-driven learning routines for our models and investigate their accuracy in an extensive computational study. The study demonstrates that models embedding asymmetric information can enhance classification accuracy.

Keywords: Classification; Bayesian networks; Staged trees; Structural learning.

1. Introduction

We consider the problem of supervised classification of a categorical class variable C given a vector of categorical features $\boldsymbol{X} = (X_1, \ldots, X_p)$ using generative classifiers. Given a training set of labeled observation $\mathcal{D} = \{(\boldsymbol{x}^1, c^1), \ldots, (\boldsymbol{x}^n, c^n)\}$, 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, thus inducing the following decision rule

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

where $\boldsymbol{x}^i \in \mathbb{X}$ and $c^i \in \mathbb{C}$. With $\mathbb{X} = \times_{i=1}^{p} \mathbb{X}_i$ and $\mathbb{C}$, we denote the sample spaces of the feature and class variables, respectively.

Bayesian network classifiers (BNCs) (Bielza and Larrañaga, 2014; Friedman et al., 1997) are the most widely-used class of generative classifiers which factorize $P(c, \boldsymbol{x})$ according to a Bayesian network over $\boldsymbol{X}$ and C. They have been shown to have competitive classification performance with respect to black-box discriminative classifiers while being interpretable and explainable since they explicitly describe the relationship between the features using a simple graph. The famous naive Bayes classifier (Minsky, 1961) can be seen as a specific instance of BNCs with a fixed graph structure where no edges between features are allowed.

The main limitation of BNCs is that they can only formally encode symmetric conditional independence. However, there is now a growing amount of evidence that real-world scenarios are better described by more generic, asymmetric types of relationships (e.g.

Eggeling et al., 2019; Leonelli and Varando, 2023, 2024c; Rios et al., 2024), for instance, context-specific ones (Boutilier et al., 1996). There have been limited attempts to extend BNCs to embed asymmetric types of dependence, most notably Bayesian multinets (Geiger and Heckerman, 1996), but their use in practice is limited.

Staged tree classifiers have been recently introduced by Carli et al. (2023). They have been shown to extend the class of BNCs to embed complex patterns of asymmetric dependence using staged tree models (Collazo et al., 2018; Smith and Anderson, 2008). Staged trees are an explainable class of probabilistic graphical models that visually depict dependence using a colored tree. A particular type of staged tree classifier is the *naive* staged tree, which, while having the same complexity as naive Bayes, extends it to account for asymmetric dependences.

In this paper, we introduce novel classes of staged tree classifiers, which can be seen as refinements of famous sub-classes of BNCs, namely TAN (Friedman et al., 1997) and k-DB (Sahami, 1996) classifiers. We formally investigate the relationship between our novel staged tree classifiers and their BNCs' counterparts. Data-driven learning routines for these novel classes are discussed and implemented. An extensive experimental study compares the classification performance of our novel classifiers to BNCs. The results highlight that these novel classes can increase classification accuracy in some cases by explicitly modeling asymmetric and flexible relationships between features.

2. Bayesian Network Classifiers

Let $G = ([p], E_G)$ be a directed acyclic graph (DAG) with vertex set $[p] = \{1, \dots, p\}$ and edge set E_G. For $A \subset [p]$, 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 $\boldsymbol{x} \in \mathbb{X}$,

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

where Π_k is the parent set of k in G. 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

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

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

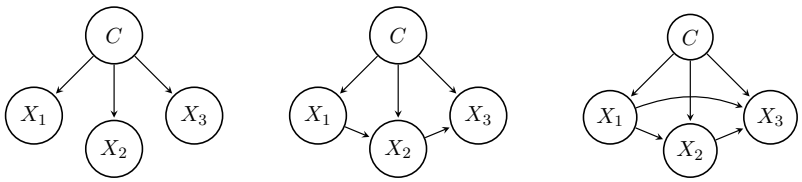
$$\mathcal{M}_G = \{P \in \Delta_{|\mathbb{X}|-1} \,|\, P \text{ is Markov to } G\},$$

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

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

2.1. Classes of Bayesian Network Classifiers

BNCs are DAGs with vertices $\boldsymbol{X}$ and C. Although any Bayesian network could be, in principle, used for classification, most commonly, the space of considered DAGs is restricted



(a) Naive BNC (b) TAN BNC (c) k-DB BNC

Figure 1: Examples of BNCs with three features and one class.

to those where C has no parents and there is an edge from C to X_i for every $i \in [p]$ (this class is sometimes referred to as Bayesian network-augmented naive Bayes, Friedman et al., 1997). By BNCs, we henceforth refer to such classifiers.

Subclasses of BNCs entertaining specific properties in the underlying DAG have been defined. The simplest possible model is the so-called naive Bayes classifier (Minsky, 1961), which assumes that the features are conditionally independent, given the class (Figure 1(a)). BNCs of increasing complexity can then be defined by adding dependencies between the feature variables. Another commonly used classifier is the TAN BNC (Friedman et al., 1997), for which each feature has at most two parents: the class and possibly another feature (Figure 1(b)). The more generic k-DB BNCs (Sahami, 1996) assume that each feature can have at most k feature parents (Figure 1(c)). Naive and TAN classifiers are k-DB BNCs for $k = 0$ and $k = 1$, respectively.

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 and TAN BNCs have been shown to lead to good accuracy in classification problems.

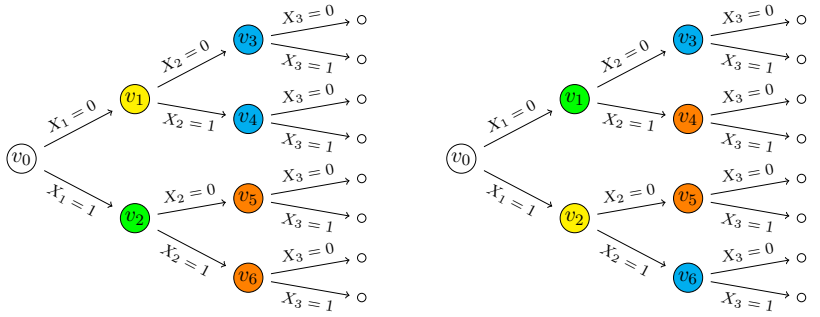
3. Staged Tree Classifiers

As before, consider a p-dimensional categorical 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.

An $\boldsymbol{X}$-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 an edge labeling such that $\theta(v, \boldsymbol{x}_{[i]}) = (\kappa(v), x_i)$ for some function $\kappa : V \to \mathcal{L}$. The function κ 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*. Therefore, the equivalence classes induced by $\theta(E(v))$ form a partition of the internal vertices of the tree in *stages*.



(*a*) A staged tree such that $\mathcal{M}_T = \mathcal{M}_{G_T}$

(*b*) A staged tree such that $\mathcal{M}_T \subseteq \mathcal{M}_{G_T}$

Figure 2: Examples of X-compatible staged trees

Points 1 and 2 above construct a rooted tree where each root-to-leaf path, or equivalently each leaf, is associated with an element of the sample space $\mathbb{X}$; similarly, each internal vertex of the tree is associated with an element of the sample space of the $\mathbb{X}_{[i]} = \times_{j\in[i]}\mathbb{X}_j$. 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, $\boldsymbol{X}$-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. Figure 2(*a*) 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 2(*a*) 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 $\boldsymbol{X}$-compatible staged tree $T = (V, E, \theta)$ with labeling $\theta : E \to \mathcal{L}^*$ is defined as

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

Equation (1) 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 2(*a*) 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.

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

$$\begin{aligned} \phi_T : \Theta_T &\rightarrow \Delta_{|\boldsymbol{l}_T|-1}; \\ y &\mapsto \Big(\prod_{e \in E(\lambda(l))} y_{\theta(e)} \Big)_{l \in \boldsymbol{l}_T} \end{aligned} \tag{2}$$

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.

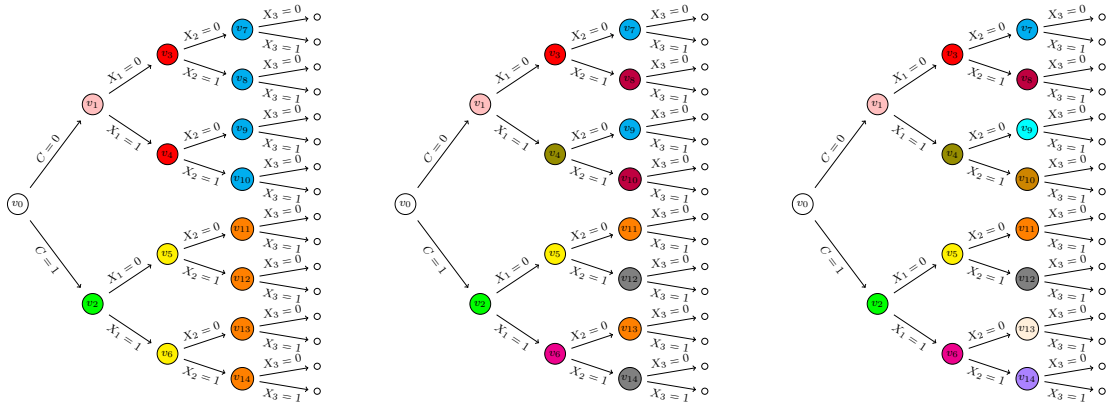
3.1. Staged Trees and Bayesian Networks

Although the relationship between Bayesian networks and staged trees was already formalized by Smith and Anderson (2008), a formal procedure to represent a Bayesian network as a staged tree has been only recently introduced (e.g. Varando et al., 2024). Assume $\boldsymbol{X}$ is topologically ordered with respect to a DAG G and consider an $\boldsymbol{X}$-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 *the staged tree model of* G. Importantly, $\mathcal{M}_G = \mathcal{M}_{T_G}$, i.e. the two models are exactly the same, since they entail exactly the same factorization of the joint probability (Smith and Anderson, 2008). Clearly, the staging of T_G represents the Markov conditions associated with the graph G.

Varando et al. (2024) approached the reverse problem of transforming a staged tree into a Bayesian network. Of course, since staged trees represent more general asymmetric conditional independences, given a staged tree T most often there is no Bayesian network with DAG G_T such that $\mathcal{M}_T = \mathcal{M}_{G_T}$. However, Varando et al. (2024) introduced an algorithm that, given an $\boldsymbol{X}$-compatible staged tree T, finds the minimal DAG G_T such that $\mathcal{M}_T \subseteq \mathcal{M}_{G_T}$. Minimal means that such a DAG G_T embeds all symmetric conditional independences that are in $\mathcal{M}_T$ and that there are no DAGs with less edges than G_T embedding the same conditional independences.

As an illustration, the staged tree in Figure 2(a) can be constructed as the T_G from the Bayesian network with DAG $X_2 \leftarrow X_1 \rightarrow X_3$, embedding the conditional independence $X_3 \perp\!\!\!\perp X_2 \mid X_1$. Conversely, consider the staged tree T in Figure 2(b). Such a staged tree does not embed any symmetric conditional independence, only non-symmetric ones, and therefore there is no DAG G_T such that $\mathcal{M}_{G_T} = \mathcal{M}_T$. Furthermore, the minimal DAG G_T such that $\mathcal{M}_T \subseteq \mathcal{M}_{G_T}$ is the complete one since the staging of the tree implies direct dependence between every pair of variables.

Leonelli and Varando (2022) introduced a subclass of staged trees based on the topology of the associated minimal DAG, which will be relevant for the definition of the novel classifiers below. A staged tree T is said to be in the class of k-parents staged trees if the maximum in-degree in G_T is less or equal to k. For instance, the staged tree in Figure 2(a) is



(a) Naive BNC (b) TAN BNC (c) k-DB BNC

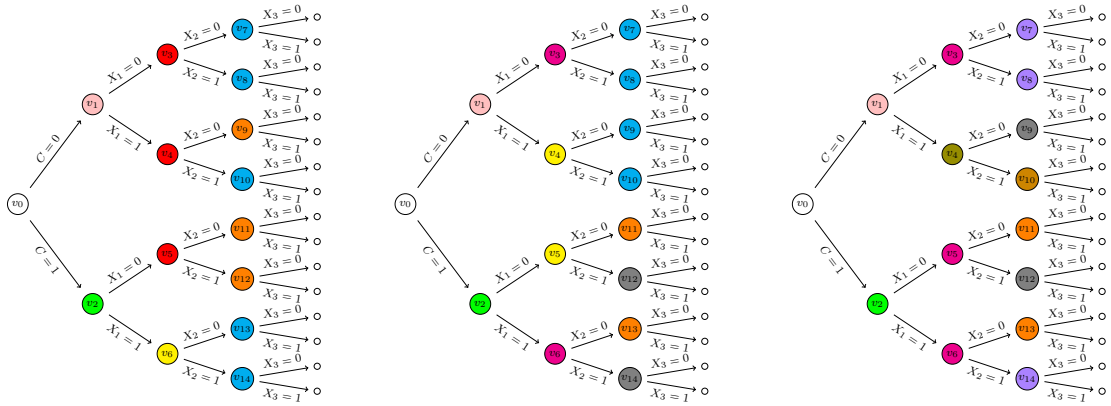
Figure 3: Staged tree representation of the BNCs in Figure 1.

in the class of 1-parent staged trees, whilst the one in Figure 2(*b*) is not, since its associated minimal DAG is such that X_3 has two parents.

3.2. Staged Trees for Classification

Carli et al. (2023) discussed how staged tree models can be used for classification purposes. As in Section 1, suppose $\boldsymbol{X}$ is a vector of features and C is the class variable. 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, to maximize the information provided by the features for classification. All BNCs reviewed in Section 2 can therefore be represented as staged tree classifiers, by constructing the equivalent T_G. Figure 3 shows the staged trees equivalent to the BNCs in Figure 1. However, Carli et al. (2023) demonstrated that the class of staged tree classifiers is much larger than that of BNCs. Formally, letting $\mathcal{M}_{\mathcal{G}}^{C}$ be the space of BNCs and $\mathcal{M}_{T}^{C}$ the space of $(C, \boldsymbol{X})$-compatible staged trees, then $\mathcal{M}_{\mathcal{G}}^{C} \subset \mathcal{M}_{T}^{C}$.

Since the class of staged tree classifiers is extremely rich, Carli et al. (2023) introduced a subclass of staged tree classifiers termed *naive*. Let V_k be the set of nodes of a tree at distance k from the root. Formally, a $(C, \boldsymbol{X})$-compatible staged tree classifiers such that for every $k \leq p$, the set V_k is partitioned into $|\mathbb{C}|$ stages is called naive. The name naive comes from these classifiers having $\sum_{j=1}^{p} |\mathbb{C}|(|\mathbb{X}_j| - 1) + |\mathbb{C}| - 1$ free parameters that need to be learned, the same number as for the standard naive Bayes model. An example of a naive staged tree is given in Figure 4(*a*). Notice that unlike naive BNCs, which have a fixed DAG structure, the coloring of the vertices must also be learned from data for naive staged trees. Carli et al. (2023) proposed using k-means and hierarchical clustering algorithms for this task. Just as for generic staged tree classifiers, naive staged trees generalize naive BNCs. Letting $\mathcal{M}_{\mathcal{G}}^{\text{naive}}$ and $\mathcal{M}_{T}^{\text{naive}}$ be the space of naive BNCs and naive staged tree classifiers, respectively, we have that $\mathcal{M}_{\mathcal{G}}^{\text{naive}} \subset \mathcal{M}_{T}^{\text{naive}}$. Importantly, Carli et al. (2023) showed via simulation experiments that naive staged tree classifiers can correctly classify



(*a*) Naive staged tree (*b*) TAN refinement (*c*) k-DB refinement

Figure 4: Examples of subclasses of staged tree classifiers.

parity functions (or 2-XORs), which cannot be captured by naive Bayes models (Varando et al., 2015).

4. Context-Specific Classifiers as Refinements of BNCs

Next, we introduce novel subclasses of staged tree classifiers that are different from naive staged tree classifiers. These subclasses are inspired by subclasses of BNCs, and they are refined to embed asymmetric patterns of dependence.

Definition 1 *A staged tree classifier T is said to be a TAN (or generally k-DB) refinement if its minimal DAG G_T is a TAN (or k-DB) BNC.*

Examples of TAN and k-DB staged tree classifiers are given in Figure 4. The coloring of the tree is much more flexible than for BNCs, embedding asymmetric patterns of dependence. It can be shown that for these staged tree classifiers T (except for the naive staged tree classifier) their associated G_T are the ones in Figure 1. The following simple result links our new classes of classifiers to k-parents staged trees. Although straightforward, this result guides the learning algorithms for the new classes of classifiers since routines already established for k-parents staged trees can be simply adapted to our classifiers.

Proposition 2 *If a staged tree classifier is a k-DB refinement, then it is in the class of $(k+1)$-parents staged trees.*

Notice that naive staged tree classifiers are not necessarily 1-parent staged trees since they are defined differently from k-DB refinements. The latter are defined through the minimal DAG, while naive staged trees are defined based on the number of parameters of the model. The naive staged tree in Figure 4 is such that its minimal DAG is complete and different from the DAG of a naive Bayes model. Conversely to naive and generic staged tree classifiers, our novel classes of classifiers are refinements of their DAG equivalent. Let $\mathcal{M}_{\mathcal{G}}^k$ and $\mathcal{M}_T^k$ be the space of k-DB BNCs and staged tree classifiers.

Proposition 3 $\mathcal{M}_{\mathcal{T}}^k \subseteq \mathcal{M}_G^k$.

An analogous result can be stated for TAN classifiers. This means that our classifiers may represent a subset of the decision rules that their BNC counterparts can. However, having a smaller number of parameters that can thus be better estimated from data, our refinements may provide better performance than BNCs when they represent the true data-generating system.

5. Learning Staged Tree Classifiers

Learning the structure of a staged tree from data is challenging due to the fast increase in the size of the tree with the number of the considered random variables. The first learning algorithm used for this purpose was the agglomerative hierarchical clustering (AHC) proposed by Freeman and Smith (2011). This starts from a staged tree where each vertex is in its own stage and joins the two stages at each iteration leading to the highest increase in a model score until no improvement is found. Initially, AHC considered only a Bayesian score, but the `stagedtrees` implementation contains a *backward hill-climbing* method, which is similar in the optimization technique, and allows using arbitrary scores such as BIC and AIC. Since then, other learning algorithms have been proposed, most often by considering some restricted space of staged trees (e.g. Leonelli and Varando, 2024a,b; Rios et al., 2024), just as in this paper. The novel learning algorithms we introduce follow three steps: (i) an initial, appropriate BNC is learned from data; (ii) the DAG associated with the learned BNC is transformed into its equivalent staged tree; (iii) the AHC algorithm is run starting from the equivalent staged tree from (ii). We give details of these phases and their implementation in the following sections.

5.1. Step (i): Learning a BNC

To learn BNCs we employ the routines implemented in the `bnclassify` R package (Mihaljević et al., 2018). In particular, we consider the following algorithms and `bnclassify` corresponding implementations:

- TAN BNCs obtained by either optimizing the log-likelihood with the Chow-Liu algorithm (`tan_cl`, Chow and Liu, 1968) or maximizing the cross-validated estimated accuracy (`tan_hc`).
- k-DB BNCs obtained by greedy optimization of the cross-validated estimated accuracy (`kdb`).

5.2. Step (ii): Transforming a DAG into a Staged Tree

Varando et al. (2024) defined a conversion algorithm to transform any Bayesian network G into its equivalent staged tree T_G. In our routines, we use the implementation provided in the `stagedtrees` R package (Carli et al., 2022) by the functions `as_sevt` and `sevt_fit`. Notice that by construction the resulting staged tree is a $(k+1)$-parents staged tree.

5.3. Step (iii): Refining the Staged Tree

Starting from the staged tree obtained in the previous step, we run the AHC algorithm which only joins stages together (no splitting of stages). We use the `stages_bhc` function from the `stagedtrees` package based on the minimization of the model BIC (Görgen et al., 2022). This step learns asymmetric dependencies between the features that were joined by an edge in step (i) without adding any new dependence between the features and the class. Therefore the resulting staged tree classifier is in the appropriate k-DB class.

For parameter estimation (i.e. the conditional probabilities of the stages) we use the state-of-the-art maximum likelihood estimation method, possibly with a smoothing parameter to avoid zero probabilities which could harm classification for unseen instances (Bielza and Larrañaga, 2014). Under the assumption of the correct stage structures, this method corresponds to the Bayes classifier rule, which minimizes the probability of misclassification (Devroye et al., 2013).

6. Experiments

We compare different instantiations of the novel paradigm for staged tree classifiers in computational experiments. Specifically, we consider staged tree classifiers refined from TAN BNCs (`sevt_tan_cl` and `sevt_tan_hc`) and k-DB (for $k = 3, 5$; `sevt_3db` and `sevt_5db`). We compare their performance to the corresponding BNC classifiers (`bnc_tan_cl`, `bnc_tan_hc`, `bnc_3db`, and `bnc_5db`). We also consider the naive staged tree classifier (`sevt_kmeans_cmi`) obtained with the k-means clustering of probabilities as proposed by Carli et al. (2023) and implemented in the `stages_kmeans` function of `stagedtrees`. For completeness, we also report results for a random forest (Breiman, 2001) classifier (`rf_1`) learned with the `randomForest` R package (Liaw and Wiener, 2002).

6.1. Benchmarks

We compare the considered classifiers across various benchmark datasets, similarly to Carli et al. (2023) but also including some additional datasets with a larger number of features. Details about the datasets are given in Table 1, which also includes the normalized entropy of the class variable as a measure of dataset imbalance. For each dataset, we repeat 10 times an 80% - 20% train-test split, and we report (Figures 5, 6, 7) median accuracies and elapsed times. Median F1 scores, balanced accuracies, and precisions are reported in Appendix B. In Figure 5 we compare all considered staged tree classifiers and a random forest classifier. While TAN and k-DB classifiers have comparable accuracy across all datasets, the naive staged tree is more variable, in some cases outperforming the others and in others having lower accuracy. All staged tree classifiers require a similar training time. In Figure 6 we compare staged trees k-DB classifiers with their corresponding k-DB BNCs and in Figure 7 we similarly compare TAN staged trees with corresponding TAN BNCs. It can be observed that overall the accuracy of all approaches is similar. In some instances, BNCs outperform staged tree refinements, while in others the reverse can be observed. Considering TAN classifiers only, the `st_bhc_tan_cl` often outperforms the others. As expected, staged tree refinements require more training time, but can still be learned in short time frames.

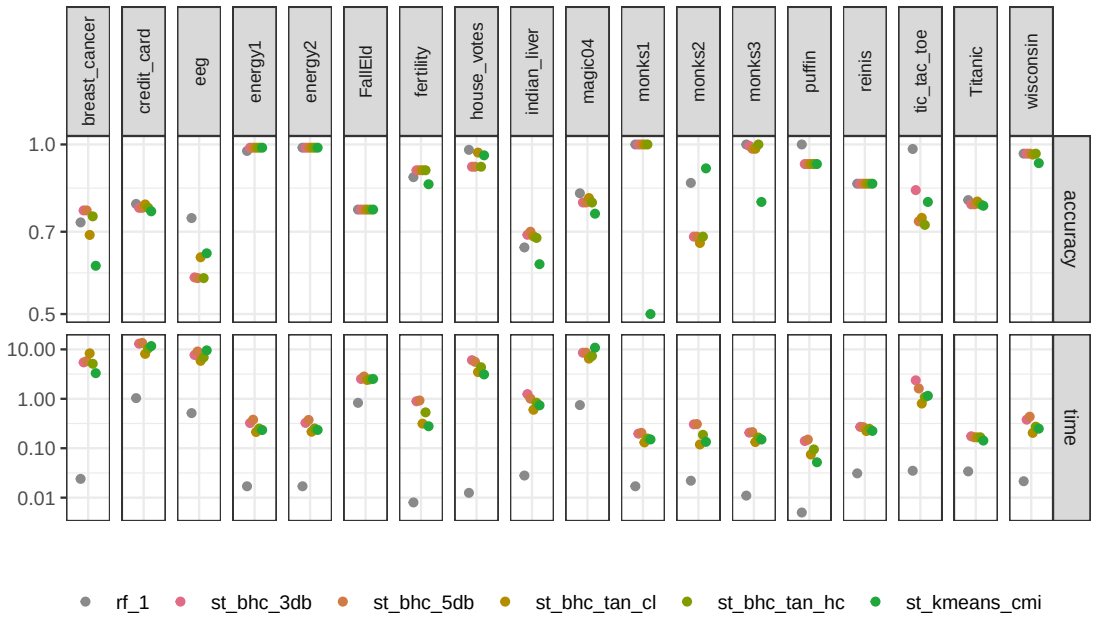


Figure 5: Comparison of median accuracies and elapsed time (in seconds) for all the staged tree classifiers considered. Additionally, results for random forest classifiers are reported.

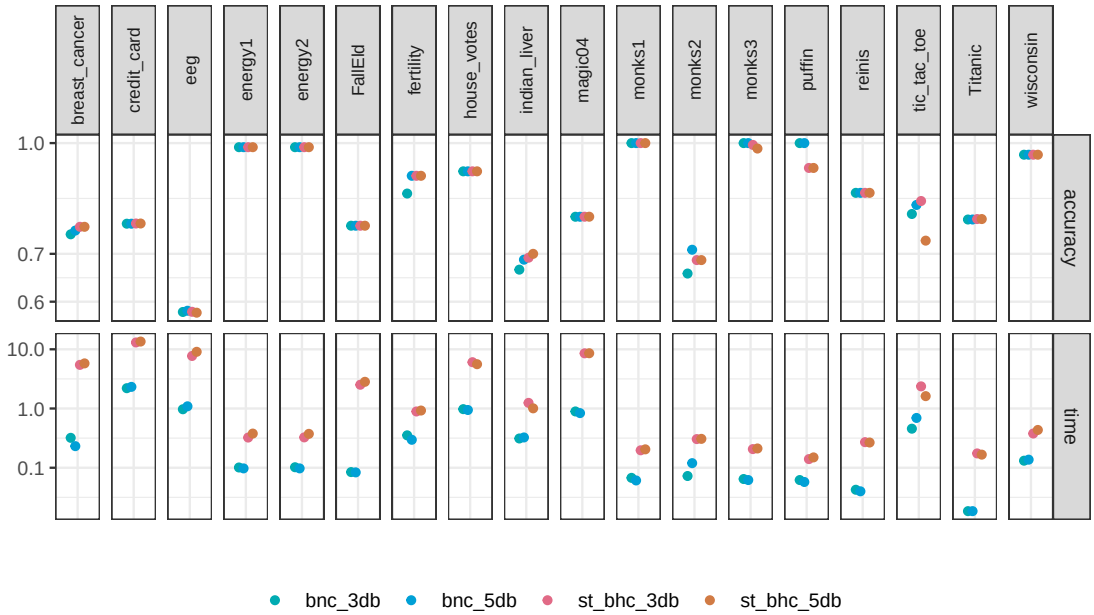


Figure 6: Comparison of median accuracies and elapsed time (in seconds) between k-DB BNCs and k-DB staged tree classifiers ($k = 3, 5$).

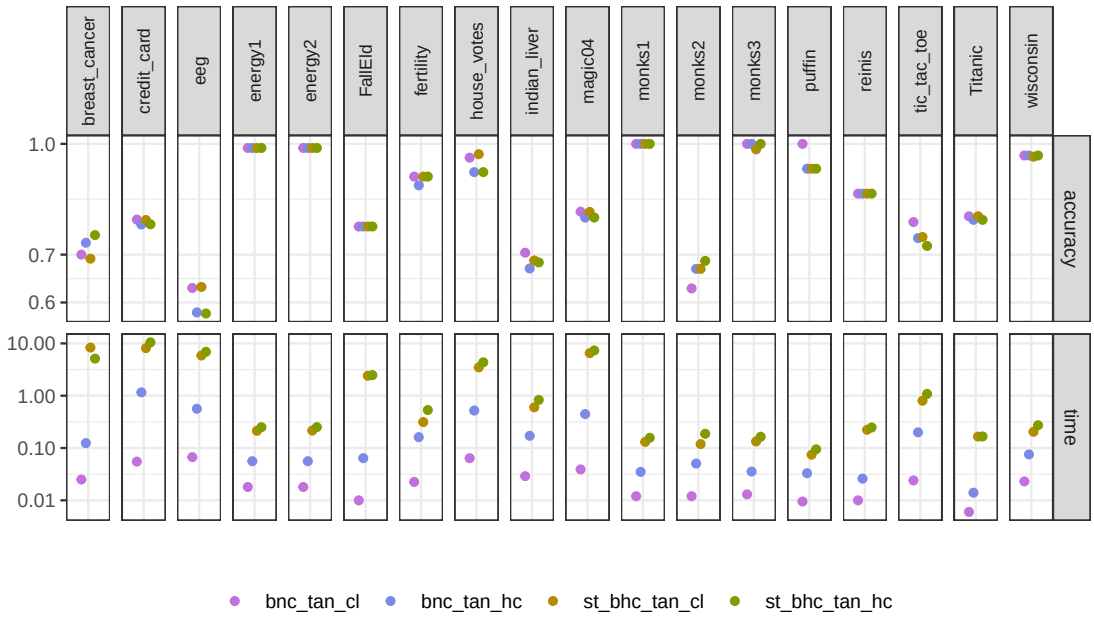


Figure 7: Comparison of median accuracies and elapsed time (in seconds) between TAN BNCs and TAN staged tree classifiers.

Furthermore, we report in Figure 11 a measure of asymmetric independence in the learned classifiers, showcasing that the learned models actually include asymmetric information.

6.2. Simulations

We perform a simulation study to explore how the classifiers behave under different scenarios. We consider three different data-generating processes:

- `random_sevt`: generates a random staged tree model over binary variables, with the ordering $C, X_1, X_2, \ldots, X_p$ and then samples from its induced joint distribution. Random staged trees are generated with the `random_sevt` function in the `stagedtrees` package. The function randomly joins stages starting from the full staged tree and assigns random conditional probabilities uniformly from the probability simplex.
- `linear`: samples independent binary predictors as Bernulli(q) where $q \sim \text{Unif}([0,1])$. The class variable C is then obtained as $C = \text{sign}\left(\sum_{i=1}^{p} \alpha_i X_i + \gamma + \varepsilon\right)$, where $\alpha_i, \gamma \sim \text{Unif}([-p, p])$ and $\varepsilon \sim \text{Unif}([-1.2, 1.2])$.
- `xor`: similar to the `linear` case, but with $C = \text{sign}\left(\prod_{i=1}^{p} X_i + \varepsilon\right)$.

We vary the number of predictors (p) ranging from 2 to 15. For each case, we generate 100 repetitions of 1000 training and 1000 test samples generated with the above three processes. In Figure 8 we report median accuracies, F1 scores, and elapsed times (in seconds) across repetitions. The naive staged tree outperforms all other approaches for all values of p under the `random_sevt` and `xor` simulation scenarios, while it has a decrease of performance in the `linear` data-generating scenario for large values of p. The other algorithms have

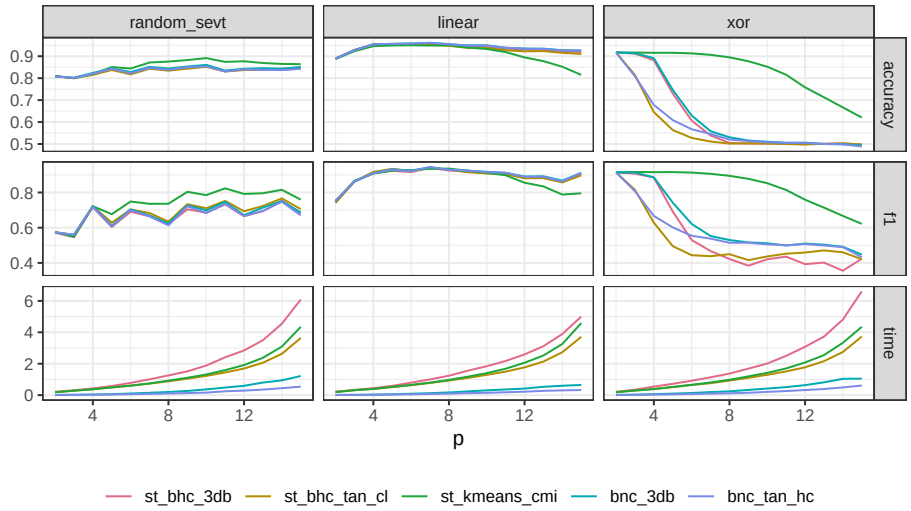


Figure 8: Median accuracy, F1, and elapsed time (in seconds) in the simulation study.

comparable performance except for the `xor` data generating scenario. The results in the `xor` scenario are partially to be expected: TAN and kDB BNC, as well as the staged tree refinements, cannot represent complex functions (Varando et al., 2015), and thus, as the number of predictors increases, their performance decrease drastically. As partially anticipated in Carli et al. (2023), the naive staged tree classifiers, here implemented with k-means clustering, can estimate optimal decision functions across a diverse range of data-generating processes, while only slightly worsening the sample-efficiency for larger p.

7. Discussion

The paper introduced novel learning routines for subclasses of staged tree classifiers, which enhance BNCs with context-specific patterns of dependence. Experimental studies showed that in some scenarios they can outperform BNCs, although the difference in accuracy appears to be only marginal. As with BNCs, different and specific parameter estimation techniques could be envisioned, such as discriminative learning (e.g. Pernkopf et al., 2011) and weighted schemes (e.g. Frank et al., 2002). Furthermore, adjusted class prior probabilities could be used to address imbalanced datasets, when different classification errors entail distinct costs (Wong and Tsai, 2021). The simulation study suggests that naive staged tree classifiers are highly expressive, with high accuracy in the `xor` data generating scenario, where all other classifiers perform poorly. This fact led us to believe that naive staged trees may be able to theoretically represent any decision rule, or at least a set of decision rules much larger than those expressible by subclasses of BNCs. A formalization of this statement and its associated proof is currently being developed.

Acknowledgments

G.V. was supported by the USMILE (grant agreement 855187) and the AI4PEX (grant agreement 101137682) ERC projects.

References

D. Aha. Tic-Tac-Toe Endgame. UCI Machine Learning Repository, 1991.

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

R. Bock. MAGIC Gamma Telescope. UCI Machine Learning Repository, 2007.

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

C. Bouveyron, G. Celeux, T. B. Murphy, and A. E. Raftery. *Model-based clustering and classification for data science: With applications in R*. Cambridge University Press, 2019.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

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

F. Carli, M. Leonelli, and G. Varando. A new class of generative classifiers based on staged tree models. *Knowledge-Based Systems*, 268:110488, 2023.

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

R. A. Collazo, C. Görgen, and J. Q. Smith. *Chain event graphs*. CRC Press, 2018.

R. J. M. Dawson. The "unusual episode" data revisited. *Journal of Statistics Education*, 3 (3), 1995.

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

R. Eggeling, I. Grosse, and M. Koivisto. Algorithms for learning parsimonious context trees. *Machine Learning*, 108:879–911, 2019.

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

G. Freeman and J. Q. Smith. Bayesian MAP model selection of chain event graphs. *Journal of Multivariate Analysis*, 102(7):1152–1165, 2011.

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

D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82(1-2):45–74, 1996.

D. Gil and J. Girela. Fertility. UCI Machine Learning Repository, 2013.

C. Görgen, M. Leonelli, and O. Marigliano. The curved exponential family of a staged tree. *Electronic Journal of Statistics*, 16(1):2607–2620, 2022.

S. Højsgaard, D. Edwards, and S. Lauritzen. *Graphical models with R*. Springer Science & Business Media, 2012.

M. Leonelli and G. Varando. Highly efficient structural learning of sparse staged trees. In *International Conference on Probabilistic Graphical Models*, pages 193–204. PMLR, 2022.

M. Leonelli and G. Varando. Context-specific causal discovery for categorical data using staged trees. In *International Conference on Artificial Intelligence and Statistics*, pages 8871–8888. PMLR, 2023.

M. Leonelli and G. Varando. Learning and interpreting asymmetry-labeled DAGs: A case study on COVID-19 fear. *Applied Intelligence*, 54(2):1734–1750, 2024a.

M. Leonelli and G. Varando. Robust learning of staged tree models: A case study in evaluating transport services. *Socio-Economic Planning Sciences*, 95:102030, 2024b.

M. Leonelli and G. Varando. Structural learning of simple staged trees. *Data Mining and Knowledge Discovery*, pages 1–25, 2024c.

A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

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

M. Minsky. Steps toward artificial intelligence. *Transactions of the Institute of Radio Engineers*, 49:8–30, 1961.

F. Pernkopf, M. Wohlmayr, and S. Tschiatschek. Maximum margin Bayesian network classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):521–532, 2011.

B. Ramana and N. Venkateswarlu. ILPD (Indian Liver Patient Dataset). UCI Machine Learning Repository, 2012.

F. L. Rios, A. Markham, and L. Solus. Scalable structure learning for sparse context-specific causal systems. *arXiv preprint arXiv:2402.07762*, 2024.

O. Roesler. EEG Eye State. UCI Machine Learning Repository, 2013.

M. Sahami. Learning limited dependence Bayesian classifiers. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

A. Shenvi, J. Q. Smith, R. Walton, and S. Eldridge. Modelling with non-stratified chain event graphs. In *Bayesian Statistics and New Generations*, pages 155–163. Springer, 2019.

J. Q. Smith and P. E. Anderson. Conditional independence and chain event graphs. *Artificial Intelligence*, 172(1):42–68, 2008.

A. Tsanas and A. Xifara. Energy Efficiency. UCI Machine Learning Repository, 2012.

UCI ML repository. Congressional Voting Records. UCI Machine Learning Repository, 1987.

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

G. Varando, F. Carli, and M. Leonelli. Staged trees and asymmetry-labeled DAGs. *Metrika*, pages 1–28, 2024.

J. Wnek. MONK's Problems. UCI Machine Learning Repository, 1992.

W. Wolberg. Breast Cancer Wisconsin (Original). UCI Machine Learning Repository, 1992.

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

I.-C. Yeh. Default of Credit Card Clients. UCI Machine Learning Repository, 2016.

M. Zwitter and M. Soklic. Breast Cancer. UCI Machine Learning Repository, 1988.

Appendix A. Benchmark Datasets Details

Dataset	# observations	# variables	$\lvert\mathbb{X}\rvert$	imbalance measure	Source
breast_cancer	277	10	332640	0.872	Zwitter and Soklic (1988)
credit_card	30000	12	18432	0.762	Yeh (2016)
eeg	14979	15	32768	0.992	Roesler (2013)
energy1	768	9	1728	1.000	Tsanas and Xifara (2012)
energy2	768	9	1728	1.000	Tsanas and Xifara (2012)
fallEld	5000	4	64	0.888	Shenvi et al. (2019)
fertility	100	10	15552	0.529	Gil and Girela (2013)
house_votes	232	17	131072	0.997	UCI ML repository (1987)
indian_liver	579	11	15552	0.862	Ramana and Venkateswarlu (2012)
magic04	19020	11	118098	0.936	Bock (2007)
monks1	432	7	864	1.000	Wnek (1992)
monks2	432	7	864	0.914	Wnek (1992)
monks3	432	7	864	0.998	Wnek (1992)
puffin	69	6	768	0.999	Bouveyron et al. (2019)
reinis	1841	6	64	0.587	Højsgaard et al. (2012)
ticTacToe	958	10	39366	0.931	Aha (1991)
titanic	2201	4	32	0.908	Dawson (1995)
wisconsin	683	10	1024	0.934	Wolberg (1992)

Table 1: Details about the 19 datasets included in the experimental study.

Appendix B. Additional Results

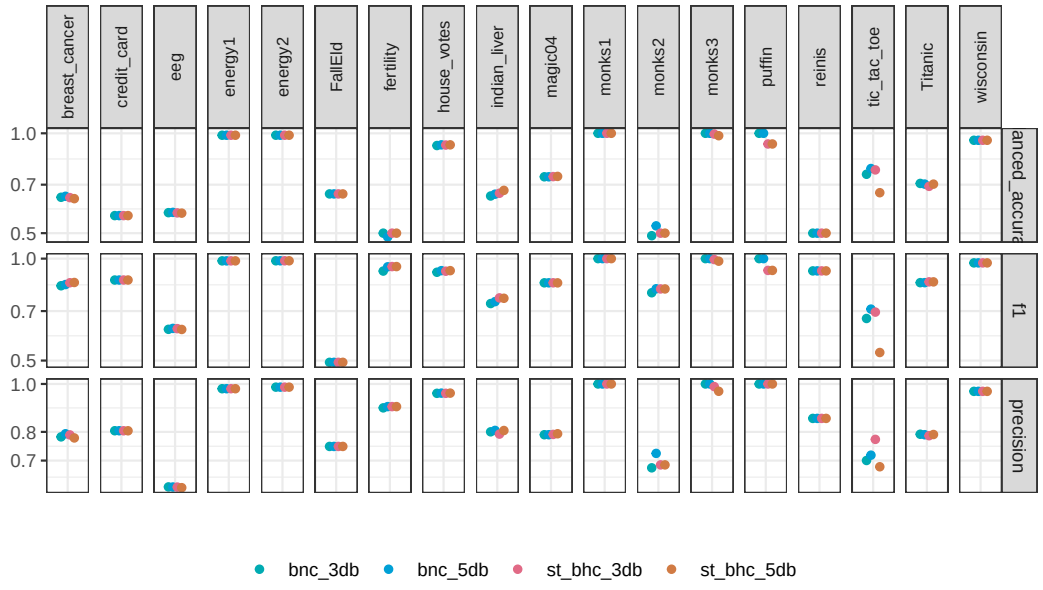


Figure 9: Median F1 scores, balanced accuracies and precisions for k-DB BN and staged tree classifiers.

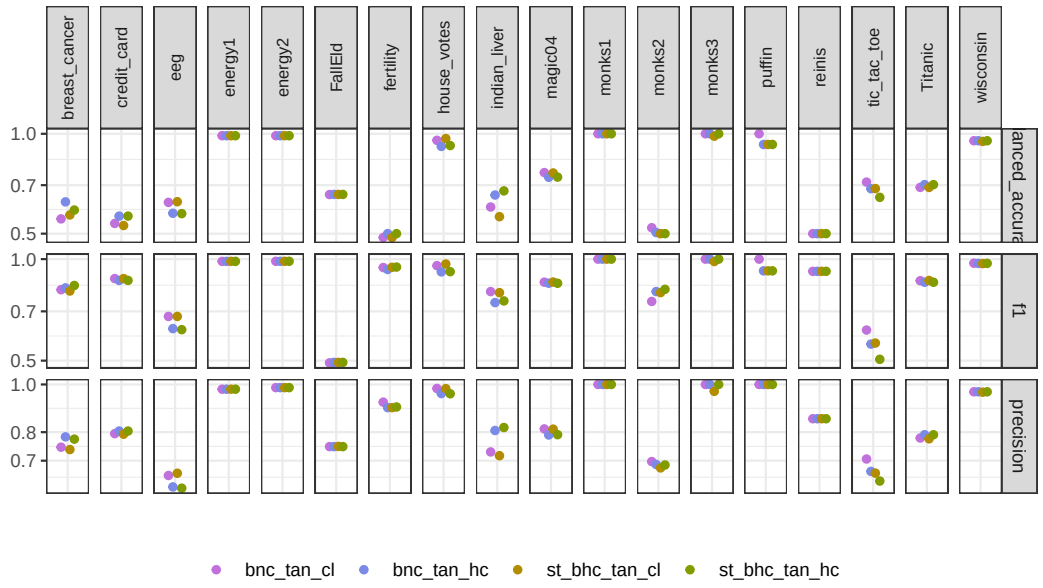


Figure 10: Median F1 scores, balanced accuracies and precisions for TAN BN and staged tree classifiers.

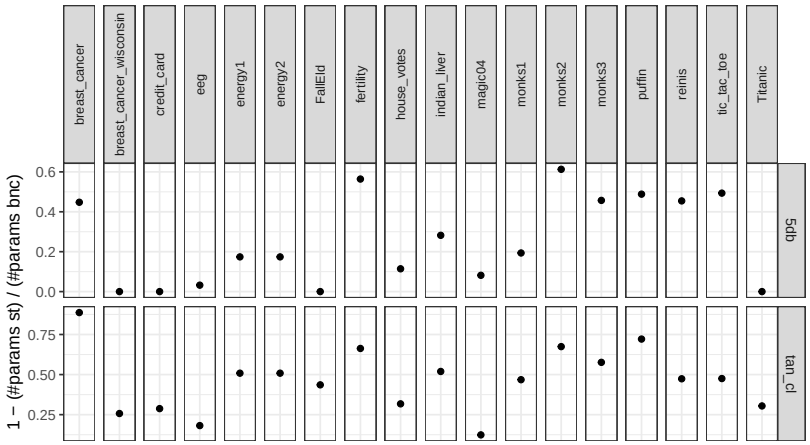


Figure 11: Measure of asymmetric conditional independence learned by refining the corresponding BN classifier (`bnc_tan_cl` or `bnc_5db`) to a staged event tree (with the `stages_bhc` method). The values reported are $1 - \frac{(\#\text{params st})}{(\#\text{params bnc})}$, where (#params bnc) is the number of free parameters in the BN classifier and (#params st) is the number of parameters in the refined staged event tree classifier. Values close to 0 indicate that the `stages_bhc` algorithm was not able to find additional asymmetric relationships, while values close to 1 hint at the presence of a large number of asymmetric independences.