Pattern Recognition 174 (2026) 112927

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Generalized Naive Bayes

Edith Alice Kovács [a,*], Anna Ország [b], Dániel Pfeifer [a], András Benczúr [b]

[a] Budapest University of Technology and Economics, Budapest, Hungary
[b] Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN), Budapest, Hungary

ARTICLE INFO

2000 MSC:
62C12
62C10
62-07

Keywords:
Naive Bayes
Generalized Naive Bayes
Conditional independence
Classification
Feature selection
Classification on health data

ABSTRACT

This paper introduces the Generalized Naive Bayes (GNB) structure as an extension of the Naive Bayes structure. We give two new greedy algorithms to find the possible closest GNB structure to the data in terms of minimizing Kullback-Leibler divergence between the probability distribution corresponding to the GNB structure and the data. Both algorithms focus on determining the best-fitting GNB structure based on the training data. The first algorithm adopts a greedy approach to find a good-fitting GNB probability distribution. Then, we introduce a second algorithm, which we prove finds the optimal (best-fitting) GNB probability distribution on the training set under a non-restrictive condition. Based on these algorithms, new feature importance scores and feature selection methods are introduced. The algorithms are compared from theoretical and practical points of view with other probabilistic graphical model-based algorithms.

1. Introduction

Naive Bayes is one of the most popular machine learning algorithms due to its simplicity, efficiency and easy interpretation, which is appealing for experts in various domains. Therefore, Naive Bayes (NB) is considered to be one of the top 10 data mining algorithms [1].

In order to show the ubiquitous interest in this algorithm we give a glimpse of some recent applications of it. A recent work [2] introduces the COVID-19 Prudential Expectation Strategy (CPES) as a new strategy for predicting the behaviour of a person's body if he has been infected with COVID-19. For the classification task, they proposed the Statistical Naive Bayes algorithm. It turned out that this methodology outperforms other competing methods. Another paper [3] uses Distance Biased Naive Bayes (DBNB) classification strategy for accurate diagnosis of Covid-19 patients which combines the Weighted Naïve Bayes Module with the Distance Reinforcement Module. Their experimental results showed that the introduced DBNB algorithm outperforms all competitors in maximum diagnosis accuracy and also minimum error. It is interesting to note that both of these COVID-related papers began with a feature selection phase. The paper [4] is concerned with using Naive Bayes in text mining, for automated literature research to select appropriate gene ontology in the MEDLINE database. The paper [5] is related to predicting lung adenocarcinoma (LUAD) from a selected gene expression. Six key genes were identified based on the tumor micro-environment, which is used as potential prognostic biomarkers and therapeutic targets of LUAD. Using these six genes, a Naive Bayes model was constructed which predicts LUAD accurately, AUC reached 92.03 %, which shows that the model used could accurately discriminate the tumor samples from the normal ones.

NB algorithm is based on the assumption that the explanatory variables are conditionally independent given the class variable (target variable). This is not generally true for real-life problems. Many papers are concerned with the improvement of this remarkable algorithm. These improvements follow mainly two directions. Some researches show that selecting some attributes beforehand might result in better classification accuracy and better generalization. Another improvement might be achieved by relaxing the conditional independence assumption.

This paper aims to relax the conditional independence assumption by introducing a structure that allows dependencies among the explanatory attributes, while the marginal distributions involved remain bounded at order three. This way, the search space for finding the best-fitting structure to the training data is also massively reduced.

The remainder of this paper is organized as follows. In Section 2, the related works are presented. Section 3 introduces the key concepts used in our approach. Section 4, presents the GNB structure the GNB probability distribution and the quantification of its fitting to the data. The next Section 5 contains the proposed algorithms and their complexities. In Section 6 we discuss the relation of the new approach to related

* Corresponding author.
E-mail addresses: kovacsea@math.bme.hu (E.A. Kovács), orszag.anna@sztaki.hun-ren.hu (A. Ország), pfeiferd@math.bme.hu (D. Pfeifer), benczur.andras@sztaki.hun-ren.hu (A. Benczúr).

https://doi.org/10.1016/j.patcog.2025.112927
Received 16 July 2025; Received in revised form 2 November 2025; Accepted 14 December 2025
Available online 18 December 2025

algorithms. In Section 7, the classification process and new feature selection method is introduced. In Section 8, we apply the algorithms to real datasets and compare the algorithms with similar algorithms by using various measures of accuracy. We close the paper with conclusions and future work.

2. Related work

Naive Bayes based methods remain a cornerstone of probabilistic machine learning due to their simplicity, interpretability, and surprisingly strong performance across diverse domains. Despite their conceptual simplicity, research in this area remains highly active, with ongoing efforts to relax the conditional independence assumption, improve calibration and robustness, and integrate Naive Bayes principles into different machine learning frameworks. Naive Bayes is a fast and effective classification method with a clear theoretical foundation, although its drawback is the conditional independence assumption. Many empirical comparisons between naive Bayes and decision trees, such as C4.5 [6], show that NB predicts equally as well as C4.5. Papers like Domingos and Pazzani [7] and Zhang [8] try to explain why are NB so effective. Domingos and Pazzani explain the good performance of NB by the 0–1 loss function, which does not penalize inaccurate probability estimations. Harry Zhang shows in [8] that the distribution of dependencies among all attributes over classes affects the classification of NB, not the dependencies themselves. The dependencies between attributes may exist but they cancel out.

Many methods were introduced to improve the NB classification algorithms. We group them as follows.

(A) Attribute selection-based methods to improve the NB performance. This category contains methods based on backward elimination or forward selection, which iteratively remove or add attributes based on cross-validation accuracy or by selecting subsets of explanatory variables. This category includes the work of Feng et al. in [9], where they introduced the Latent Selection Augmented Naive Bayes (LSAN) classifier and the Latent Selective Naive Bayes classifier with a maximum aposteriori feature selection estimator. These methods were successfully applied to text classification problems.

(B) Methods based on weighting the features by their importance. In the literature there are known many weighting possibilities: based on decision trees in [10], where it was shown that using feature weights in NB improves the quality of the model compared with the original NB. In [11] the feature weighting method is based on the KL-divergence.

A recent paper [12] introduces a novel selective Naive Bayes algorithm, which proposes an efficient method for classification. The models were built in such a way that each one is an extension of another one. The algorithm uses a pre-ranking of the attributes based on their mutual information with the classification variable. The best model is selected based on cross-validation. Empirical results show that the introduced method is superior to NB and it is comparable with the Tree Augmented Naive Bayes (TAN) [13] as accuracy and running time. Zhang and Sheng [14] proposed a weighted Naive Bayes. The weights were chosen to maximise AUC score. Another way to improve the accuracy of NB is by weakening the conditional assumption. The term “semi-Naive Bayes” was introduced by Kononenko [15]. In this paper, conditional probabilities of joint features were computed based on the training data. The method was illustrated on medical datasets with slight improvements over the classical NB.

(C) Methods that ensure the conditional independence assumption is satisfied. The idea is to use a carefully selected subset of variables to minimize violations of this independence assumption and improve accuracy.On top of this is the fact that a model with less variables is more explainable. An important paper in this direction is [16]. Their variable reduction strategy was based on clustering the features in terms of their dependence degree, and it selects combinations of features that, being as independent as possible. This lead to a good performance rate even on the high dimension (synthetic) datasets.

(D) Methods that relax the assumption of conditional independence. An important subclass of this allows to each feature to depend on the class variable and one other feature, called also semi-naive methods. An important step in this direction was realized by extending the graph structure of naive Bayes is the introduction of a tree augmentation of the NB graph in [13]. The algorithm assumes that the relationship among the explanatory variables is only tree-structure-like, in which the class variable directly points to all feature nodes and each feature has only one parent node from another feature. TAN algorithm showed significant improvement in accuracy compared to NB. An improvement for the TAN algorithm is presented in [17]. This result is obtained by averaging several TAN structures, which proves to be more accurate than classifying using just one TAN structure. Beyond TAN, one-dependence estimators (ODE) include SPODE (Super-Parent ODE) and AODE (Averaged One-Dependence Estimator) [18]. In a SPODE, all features depend on the class and a single designated “super-parent” feature. AODE takes this to an ensemble: it builds one SPODE for each attribute as super-parent, then averages their posterior predictions. In the paper [19], the author argues that the one-dependence estimator (AODE) performs exceptionally well compared to more sophisticated models. They enhance the model by applying a special weighting based on mutual information. A very interesting article which extends the NB structure by adding more edges (more parents) is presented in [20]. This approach uses information contents, and add the attributes in a greedy way. However it turned out that using 2 or rarely maximum 3 parents gave the best results on the test data. A more recent paper [21] uses a special Bayes network (directed edges) between the explanatory variables to improve results of Naive Bayes classification. In the classification procedure they use conditional probabilities derived from the Bayesian networks. However discovering the Bayes network is typically NP hard. Another recent paper [1] introduced new algorithms for classifications based on attribute weighting and instance weighting. They define a new weighting methodology that uses the correlation of the attributes with the class variable(relevance) and also with the other attributes (redundancy). The weight of an attribute is defined by taking into account relevance and average redundancy.

A very important paper that uses decomposable models to expand the classical NB was published in Ghofrani et al. in 2018 [22]. This work is also interesting because, in their comparisons with other models, they use the cherry tree probability distribution defined on the explanatory attribute set. They concluded that cherry trees were not so efficient because of the overfitting. The basic idea of their paper related to cherry trees is to replace the tree between the explanatory variables given by the TAN algorithm with a cherry tree. They also introduce their model, which aims to capture interdependencies between the explanatory attributes. Their experimental performance shows that their algorithm had a better classification performance than, NB and Cherry tree-based ones.

The new methodology and algorithms, we introduce in this paper are structure-extending algorithms that alleviate the conditional independence assumption of the NB this way, they belong to the methods of the last group. The main difference is that, although we perform structural extensions that usually lead to NP-hard problems, we are able to reduce the search space. Additionally, we have developed a methodology and algorithms that allow us to prove the optimality of the Kullback-Leibler (KL) divergence with respect to the training data.

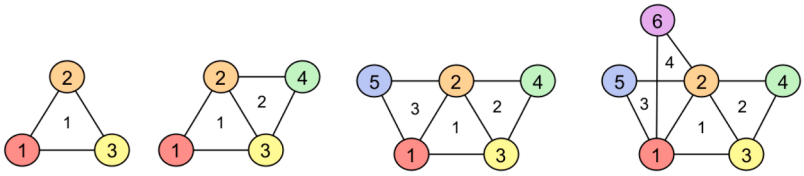

Fig. 1. Construction of a third-order cherry tree on 6 vertices.

Section 6 focuses on comparing the similarities and main differences between the algorithms introduced in this paper and those proposed earlier: TAN in [13], ATAN in [17], and LDMLCS in [22].

Our algorithms' output can be used as the base for a new feature selection method. We demonstrate that the introduced Generalized Naive Bayes structure performs at least as well as the standard Naive Bayes structure in fitting their respective probability distributions to real data. We introduce two algorithms: the first is a general greedy algorithm that provides a good fitting GNB probability distribution, while the second algorithm provides the best fitting GNB probability distribution to training data, whenever a not very restrictive condition is met.

3. Basic concepts

A series of studies, related to Naive Bayes, approach the classification problem from the perspective of directed probabilistic networks, known as Bayes Networks. In this paper, we will extend the Naive Bayes structure by employing undirected probabilistic graphical models, also known as Markov networks. To reduce the complexity of the search space, we will concentrate on a specific class known as third-order cherry trees.

Probabilistic graphical models integrate graph theory, probability theory, and information theory to provide a flexible probabilistic framework for modeling random variables with complex interactions. However, learning structures for graphical models remains challenging, due to the combinatorial search required across all potential structures. In this paper, we define a structure called Generalized Naive Bayes as a special type of cherry trees.

At the heart of probabilistic graphical models stands the conditional independence concept (CI). In 1950 Loeve [23] defined the concept of CI in terms of $\sigma-$algebras. Phi Dawid formulated the formal properties of CI from a statistical point of view [24].

The following subsection presents key concepts and preliminary results necessary for the new results presented in the paper. In this paper, we focus on a specific case of more general structures. For simplicity, we will introduce these directly rather than deriving them from the more general case.

3.1. Probabilistic graphical models used in the GNB structures

We introduce the concept of a cherry tree graph structure, which is a key component of this work. Originally referred to as a t-cherry in [25], we have chosen to call it a cherry tree graph in this context, as this terminology does not lead to any confusion. This structure is, in fact, a generalization of a tree.

We define the third-order cherry tree graph by construction.

We call **third order cherry tree graph structure**, a graph structure obtained by the following two steps: (a) The smallest cherry tree consists of 3 interconnected vertices. (b) A cherry tree on $m+1$ vertices is obtained from a cherry tree on m ($m>3$) vertices by connecting a new vertex to 2 already connected vertices. Fig. 1 shows how a third-order cherry tree is built on 6 vertices step by step.

It is easy to see that a third-order cherry tree is a triangulated (chordal) graph with the maximum cliques containing 3 interconnected vertices (fully connected subgraph).

Next, we introduce the concept of a cherry junction tree, as it establishes the connection between the cherry tree graph structure and the associated probability distribution, which we aim to fit to the training data. Intuitively, this can be seen as a cluster tree, where the clusters consist of 3 elements (we mention here that in this framework clusters may not be disjoint), and the edges comprise 2 elements that are common to the connected clusters.

We call **cherry-junction tree of order 3** the structure assigned to a cherry tree graph of order 3 in the following way:

1. The set of vertices in each 3 order maximum clique (fully connected subgraph) is called a *cluster* containing its 3 vertices (a node in the junction tree graph).
2. Two 3-element clusters are connected if the following two conditions are fulfilled: (a) the clusters share 2 elements; (b) if a set of 2 elements is contained by m clusters, these clusters will be connected tree-like by $m-1$ edges, see second row of Fig. 2.
3. The 2-element set given by the intersection of two connected clusters is called a *separator*. The number of clusters containing it is the *multiplicity of the separator*.

A cherry-junction tree is characterised by the set of indices V, the set of clusters C, the set of separators S and a set of the multiplicities of the separators $\mathcal{M}$.

For an illustration of a 3-rd order cherry-junction tree, corresponding to a 3rd-order cherry tree graph, see Fig. 2. The junction tree associated with a cherry tree graph structure does not have a unique graph representation.

An important property of the junction trees is the Running intersection property, i.e. if a vertex is contained in two different clusters it is contained in any cluster on the path between the two clusters. For a good overview of the connection between these concepts, see [26].

Let $\mathbf{X}=\left(X_1,\dots,X_d\right)^T$ a random vector and $V=\{1,\dots,d\}$ the set of indices, and let us consider a junction tree over V given by the set of clusters C and the set of separators S denoted by (V,C,S). For the probability distribution of a random vector having its components $(X_{i_1},\dots,X_{i_k})$, we use the following abbreviations: $P(\mathbf{X}_{i_1,\dots,i_k})$ and $P(\mathbf{X}_A)$ where $A=\{i_1,\dots,i_k\}\subset V$.

The **cherry tree probability distribution** corresponding to a cherry-junction tree (V,C,S) is given by the following formula:

$$P_{ch}(\mathbf{X})=\frac{\prod_{C\in C}P(\mathbf{X}_C)}{\prod_{S\in S}P(\mathbf{X}_S)^{\upsilon_S-1}} \tag{1}$$

where υ_S is the number of clusters that are connected through the separator S (multiplicity of S); the formula stands for all realizations of $\mathbf{X}$.

One can note that, regardless of which graphical representation of a junction tree we use (see, for example, Fig. 2), the cherry tree probability distribution formula is the same:

$$P_{ch}(X)=\frac{P(\mathbf{X}_{1,2,3})P(\mathbf{X}_{2,3,4})P(\mathbf{X}_{1,2,5})P(\mathbf{X}_{1,2,6})}{(P(\mathbf{X}_{1,2}))^2P(\mathbf{X}_{2,3})}.$$

In this paper, the generalization of the NB structure is derived from probabilistic graphical models.

The probability distribution, assigned to a Naive Bayes structure which encodes the conditional independence of the attributes X_i given

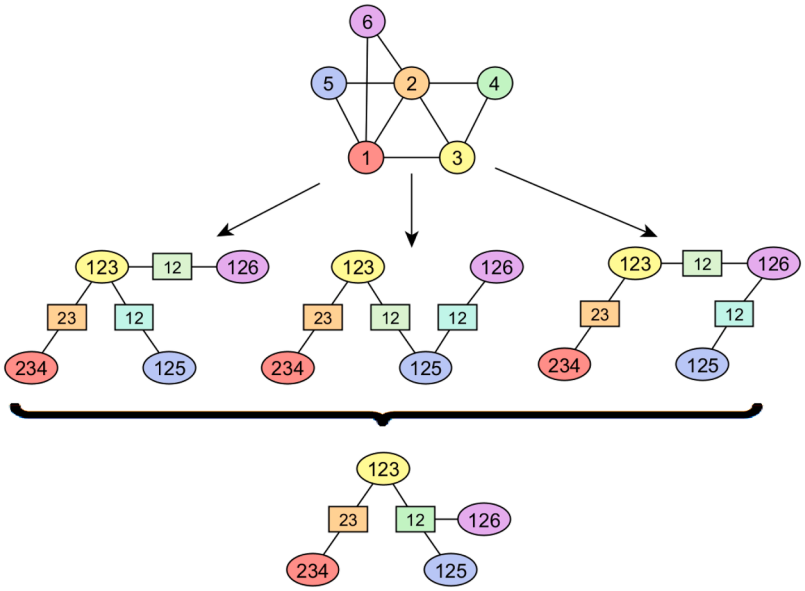

Fig. 2. Cherry tree structure of order 3 represented as a chordal graph (first row), junction trees (second row) and represented in a compact junction tree form (third row).

the class variable Y, has the following expression:

$$P_{NB}(\mathbf{X}, Y) = P(Y)\prod_{i=1}^{d} P(X_i|Y) \tag{2}$$

It is easy to see that Formula (2) is equivalent to the following formula:

$$P_{NB}(\mathbf{X}, Y) = \frac{\prod_{i=1}^{d} P(X_i,Y)}{P(Y)^{d-1}} \tag{3}$$

which is a cherry tree probability distribution of order 2. The clusters consist of two elements X_i, Y, $i = 1, \ldots, d$, and the separators consist of only one element, namely Y.

The Naive Bayes classifier assigns to a new realization $\mathbf{x}$ the most probable value $y(\mathbf{x})$ of the class variable: $y(\mathbf{x}) = arg\max_{y_i}\left(p_{NB}(y_i, \mathbf{x})\right)$ where y_i, $i = 1, \ldots, s$ denote the classes of the class variable Y. It is clear that this classifier relies on conditional independencies among the explanatory variables given the class variable. These conditional independencies often appear to be conservative, as they are typically not satisfied in real-life scenarios. This observation also motivates our research.

We close this subsection with the observation that the expression in Formula (1) will be used to generalize the probability distribution assigned to a Naive Bayes structure, in Formula (3).

3.2. Information theoretical concepts

Since the goodness of fitting of the probability distributions we introduce, to the data is quantified by the KL divergence, we will need to introduce some information-theoretical concepts that we recall in this subsection. For more details, see the fundamental book [27].

We will use an equivalent formulation of conditional entropy. The **conditional entropy** quantifies the amount of information needed to describe the outcome of a random variable Y given a random variable X: $H(Y|X) = H(X, Y) - H(X)$.

If $P(\mathbf{X})$ is approximated by a probability distribution $P_{app}(\mathbf{X})$, a quantification of the goodness of the approximation is given by the **Kullback-Leibler divergence** (KL-divergence) [28] between a given joint probability distribution $P(\mathbf{X})$ and its approximation $P_{app}(\mathbf{X})$:

$$\mathrm{KL}(P_{app}(\mathbf{X}), P(\mathbf{X})) = \sum_{\mathbf{x}} P(\mathbf{x}) \log_2 \frac{P(\mathbf{x})}{P_{app}(\mathbf{x})}.$$

KL divergence is a positive number unless $P_{app} \equiv P$, when KL = 0. The smaller the value of the KL-divergence is the better the approximation is.

The **mutual information** can be seen as a special type of KL divergence between a given $P(X_{i_1}, X_{i_2})$ and an approximation given by independence: $P_{app} = P(X_{i_1})P(X_{i_2})$. One of the possible generalizations is straight forward as follows.

The **information content** of a random vector $\mathbf{X} = (X_{i_1}, \ldots, X_{i_k})^T$ is given by

$$I(X_{i_1}, \ldots, X_{i_k}) = \sum_{\mathbf{x}} P\left(x_{i_1}, \ldots, x_{i_k}\right) \log_2 \frac{P\left(x_{i_1}, \ldots, x_{i_k}\right)}{P(x_{i_1}) \cdot \ldots \cdot P(x_{i_k})}.$$

We mention that the information content defined above, and used in the papers [25] and [29] is a quantification of the strongness of dependence between the random variables, which was called also multiinformation.

A fundamental theorem we use in this paper expresses the KL divergence between a real probability distribution (which can be given by a dataset) and a cherry-tree probability distribution approximation (1). The random vectors assigned to the indices in the clusters and separators are marginals of the real probability distribution.

Theorem 1. *[29] The Kullback-Leibler divergence between the cherry tree approximation (1) and the real distribution $P(\mathbf{X})$ is:*

$$\mathrm{KL}\left(P_{ch}(\mathbf{X}), P(\mathbf{X})\right) = -H(\mathbf{X}) - \left[\sum_{C\in\mathcal{C}} I(\mathbf{X}_C) - \sum_{S\in\mathcal{S}} (\nu_S - 1)(I(\mathbf{X}_S))\right] + \sum_{i=1}^{d} H(X_i). \tag{4}$$

These were the basic information theoretical concepts and results that we will use in the next part.

4. Generalized Naive Bayes

This section introduces the new concepts, specifically the GNB structure as a special case of the third-order cherry tree, along with the

probability distribution associated with it. We provide a formula for the Kullback-Leibler divergence when fitting a GNB probability distribution to the data. Additionally, we demonstrate that the GNB probability distribution fits the data as well as, or even better than, the standard NB probability distribution.

Let us introduce the Generalized Naive Bayes graph structure, which is an NB graph structure augmented with certain edges that provide conditional dependencies and independencies between the explanatory variables.

As we saw earlier, the NB structure can be assigned to a junction tree with two elements in each cluster, see (3). The new concept introduced in this paper is basically a generalization of the above idea, i.e., we will use special junction trees, with exactly three elements in a cluster (two besides Y) and two elements in the separator (one besides Y). This way, we will use a cherry tree structure for the generalization.

4.1. Generalized Naive Bayes graph structure

We introduce the Generalized Naive Bayes graph structure and then define the probability distribution associated with it.

Definition 2. A **Generalized Naive Bayes graph-structure** on $V \cup \{0\} == \{0, 1, \dots, d\}$ (vertex 0 is assigned to Y), is constructed as follows:

- Step 1. The smallest GNB structure on three vertices is given by the interconnected triplet $(0, i_1, i_2)$, where $i_1, i_2 \in V$.
- Step k. Let us suppose that the vertices $i_1, i_2, \dots, i_k \in V$ are already in the GNB structure. We add a new vertex $i_{k+1} \in V$ to the existing GNB structure by connecting it to vertex 0 and an already connected vertex $i_m \in \{i_1, i_2, \dots, i_k\}$. The vertex i_m is called the *mother of vertex* $i_{k+1}, k > 0$, denoted by $\mu(i_{k+1})$. Vertex 0 is called the *father vertex* of all vertices.

It is easy to see that a GNB is a special type of cherry-junction tree structure; all clusters contain the index 0 corresponding to the classification variable Y; each vertex $i_{k+1}, k > 0$, has one mother vertex $\mu(i_k)$ and the same father vertex 0. The first connected vertex i_1 has only the father vertex 0 by definition.

The set of all possible GNB structures on $V \cup \{0\}$ defines the **search space** for our algorithms.

The permutation $i_1, i_2, \dots, i_d$ of the indices $\{1, \dots, d\}$, given by the steps of Definition 2 is called c-**ordering**, where the first two indices in the smallest cherry tree are arranged increasingly.

Without loss of generality, we denote a c-ordering $i_1, i_2, \dots, i_d$ by $1, \dots, d$.

Definition 3. The *GNB probability distribution* associated to the GNB graph structure and a c-ordering $1, \dots, d$ is the following:

$$P_{GNB} = \frac{\prod_{i=2}^{d} P\left(Y, X_{\mu(i)}, X_i\right)}{\prod_{i=3}^{d-1} P\left(Y, X_{\mu(i)}\right)}.$$

Now we want to prove that a GNB probability distribution given by Definition 3 gives a better approximation than the NB probability distribution, in terms of KL-divergence. For this, we need the expressions of the KL-divergences for the NB and GNB approximations.

Theorem 4. *If the joint probability distribution $P(\mathbf{X},Y)$ is approximated by a Naive Bayes distribution (3) the Kullback-Leibler divergence between the true distribution and the approximation is given by the formula:*

$$KL(P(\mathbf{X},Y), P_{NB}(\mathbf{X},Y)) = \sum_{i=1}^{d} H(X_i) + H(Y) - H(\mathbf{X},Y) - \sum_{i=1}^{d} I(X_i,Y) \tag{5}$$

This is a consequence of the Chow-Liu tree method [30]. The goodness of fit of the NB approximation depends on the $\sum_{i=1}^{d} I(X_i,Y)$. The larger this sum, the better the approximation. This result highlights why it is worth choosing for feature selection the explanatory variables based on their mutual information with the class variable.

Theorem 5. *If $P(\mathbf{X})$ is approximated by a Generalized Naive Bayes distribution $P_{GNB}(\mathbf{X})$ (see Definition 3) then the KL divergence is given by the formula:*

$$KL(P(\mathbf{X},Y), P_{NB}(\mathbf{X},Y)) = \sum_{i=1}^{d} H(X_i) + H(Y) - H(\mathbf{X},Y) - \left(\sum_{i=2}^{d} I\left(Y, X_{\mu(i)}, X_i\right) - \sum_{i=3}^{d} I\left(Y, X_{\mu(i)}\right)\right)$$

Proof. This follows straightforwardly from the formula (4) of KL divergence applied for an approximation associated to a cherry tree approximation, as a special case of it. □

We saw in Theorems 4 and 5 that the corresponding KL divergences depend on two parts, one of them being common to both approximations. Therefore the goodness of fit will depend on the part that differs from one model to another, from NB to GNB. This specific part depends only on the graph structure and is defined as follows.

Let $P_{app}(\mathbf{X})$ be an approximation for $P(\mathbf{X})$; The **weight of the approximation** is defined by:

$$W(P(\mathbf{X}), P_{app}(\mathbf{X})) = \sum_{i=1}^{d} H\left(X_i\right) - H(\mathbf{X}) - KL\left(P(\mathbf{X}), P_{app}(\mathbf{X})\right). \tag{6}$$

The larger the weight of the approximation, the better the approximation is.

The weight of the $P_{NB}P(\mathbf{X},Y)$ approximation is:

$$W(P(\mathbf{X},Y), P_{NB}P(\mathbf{X},Y)) = \sum_{i=1}^{d} I\left(Y, X_i\right) \tag{7}$$

and the weight of the $P_{GNB}P(\mathbf{X},Y)$ approximation is:

$$W(P(\mathbf{X},Y), P_{GNB}P(\mathbf{X},Y)) = \sum_{i=2}^{d} I\left(Y, X_{\mu(i)}, X_i\right) - \sum_{i=2}^{d} I\left(Y, X_{\mu(i)}\right). \tag{8}$$

Now we will show that the weight of a GNB approximation is at least as large as the weight of an NB approximation. To prove this we need first the following two theorems.

Theorem 6. *The weight of the NB approximation can be written as:*

$$W(P, P_{NB}) = \sum_{i=1}^{d} H\left(X_i\right) - \sum_{i=1}^{d-1} H\left(X_i|Y\right) - H\left(Y, X_d\right) \tag{9}$$

Proof.

$$W(P, P_{NB}) = \sum_{i=1}^{d} I\left(Y, X_i\right) =$$

$$= (H(Y) + H(X_1) - H(Y, X_1)) + (H(Y) + H(X_2) - H(Y, X_2)) + \dots$$

$$\dots + (H(Y) + H(X_{d-1}) - H(Y, X_{d-1})) + (H(Y) + H(X_d) - H(Y, X_d))$$

First we add the entropies $H(Y), H(X_1), H(X_2), \dots, H(X_d)$, then we apply the formula $H(Y, X_i) - H(Y) = H(X_i|Y)$ starting on from the first row, for $i = 1, \dots, d-1$. This way we get the formula (9). □

To prove a similar theorem regarding the weight of a GNB approximation, we need a result related to the GNB graph structure.

First, we introduce a special cherry tree called **chain of clusters** where each cluster is connected to at most two different clusters by at most two different separators.

Lemma 7. *A GNB graph structure can be represented as a chain of clusters.*

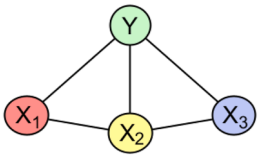

Fig. 3. The first three attributes which define two clusters of a cherry tree.

Proof. Let us suppose that we have a GNB structure with the property that there exists a cluster that is connected to three clusters by different separators. We denote this cluster by $C_1(0,i,j)$ and its three neighbours by C_2, C_3, C_4. This implies that it is possible to connect C_1 with two clusters by two different separators S_{12} $(0,i)$ and S_{13} $(0,j)$, but we have no other different subset in C_1 which contains the vertex 0 (which corresponds to class variable Y). Therefore any cluster can be connected to the other clusters by at most two different separators, which completes the proof. □

Since the GNB probability distribution associated to the GNB graph can be defined as a junction tree associated to a chain of clusters, we will assign a numbering to the random variables involved.

The **chain numbering of the attributes** based on the chain structure of GNB is defined as follows. Let us suppose $C_1, C_2, \ldots, C_{d-1}$ are the clusters of the chain. $C_1 = (Y, X_{i_1}, X_{i_2})$, where i_1 and i_2 are chosen in increasing order, $C_2 \backslash C_1$ is denoted by X_{i_3}, ..., $C_t \backslash C_{t-1}$ is denoted by $X_{i_{t+1}}$, ...,$C_{d-1} \backslash C_{d-2}$ is denoted by X_{i_d}. (Where $\varepsilon A \backslash B \varepsilon$ denotes the elements that belong to set A but not to set B).

The chain numbering $\left\{X_{i_1}, \ldots, X_{i_d}\right\}$ of the attributes is a permutation of the indices of the attributes and it is not unique.

For simplicity, without loss of generality, we will denote the chain numbering $\{i_1, \ldots, i_d\}$ by $\{1, \ldots, d\}$ but whenever we use this we mention that this is a chain numbering.

Theorem 8. *Let $\{1, \ldots, d\}$ be a chain numbering. The weight of the basic GNB approximation is given by the formula:*

$$W_{GNBbasic}^{(d)} = \sum_{i=1}^{d} H(X_i) + H(Y) - \sum_{i=1}^{d-1} H(X_i|Y, X_{i+1}) - H(Y, X_d). \qquad (10)$$

Proof. We do the proof by induction.

Let us consider a chain numbering of the attributes $\left\{X_1, X_2, \ldots, X_d\right\}$.

First we consider the class variable Y and the attributes X_1, X_2, X_3, as illustrated on Fig. 3, and we show that the formula is true for $d = 3$.

As $\left(Y, X_1, X_2, X_3\right)$ is a cherry junction tree and the formula of KL divergence is given as in (8) the following formula holds

$$\begin{aligned} W_{GNB}^{(3)} &= I\left(Y, X_1, X_2\right) - I\left(Y, X_2\right) + I\left(Y, X_2, X_3\right) = \\ &= H(Y) + H(X_1) + H(X_2) - H(Y, X_1, X_2) + H(Y) + H(X_2) \\ &\quad + H(X_3) - H(Y, X_2, X_3) - \left(H(Y) + H(X_2) - H(Y, X_2)\right) \end{aligned}$$

After reduction $W_{GNB}^{(3)} = \sum_{i=1}^{3} H(X_i) + H(Y) - H(Y, X_1, X_2) - H(Y, X_2, X_3) + H(Y, X_2)$

We add and substract $H(Y, X_3)$ and get the following:

$$\begin{aligned} W_{GNB}^{(3)} &= \sum_{i=1}^{3} H(X_i) + H(Y) + H(Y, X_2) - H(Y, X_1, X_2) \\ &\quad + H(Y, X_3) - H(Y, X_2, X_3) - H(Y, X_3) \\ &= \sum_{i=1}^{3} H(X_i) + H(Y) - H(X_1|Y, X_2) - H(X_2|Y, X_3) - H(Y, X_3), \end{aligned}$$

which is exactly the formula (10) for $d = 3$.

Now, we suppose that the formula (10) is true for d attributes and we will prove that it is true for $d + 1$ attributes, see Fig. 4. Without loss of generality, we will suppose that X_{d+1} is connected to X_d and Y.

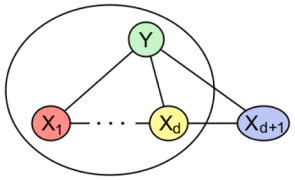

Fig. 4. The inductive construction of the GNB structure as a cherry tree in step d.

Since this has also a cherry-junction tree structure with a chain structure, it's weight can be written by adding a new cluster (Y, X_d, X_{d+1}) as follows:

$$\begin{aligned} W_{GNB}^{(d+1)} &= W_{GNB}^{(d)} + I\left(Y, X_d, X_{d+1}\right) - I(Y, X_d) = \\ &= W_{GNB}^{(d)} + H(Y) + H(X_d) + H(X_{d+1}) - H(Y, X_d, X_{d+1}) \\ &\quad - \left[H(Y) + H(X_d) - H(Y, X_d)\right] \end{aligned}$$

After reduction, we obtain:

$$W_{GNB}^{(d+1)} = W_{GNB}^{(d)} + H(X_{d+1}) - H(Y, X_d, X_{d+1}) + H(Y, X_d).$$

Now we use the induction hypothesis and get:

$$\begin{aligned} W_{GNB}^{(d+1)} &= \sum_{i=1}^{d} H(X_i) + H(Y) - \sum_{i=1}^{d-1} H(X_i|Y, X_{j+1}) - H(Y, X_d) + \\ &\quad + H(X_{d+1}) - H(Y, X_d, X_{d+1}) + H(Y, X_d) \end{aligned}$$

After reduction and adding and subtracting $H(Y, X_{d+1})$ we use the formula

$$H(Y, X_d, X_{d+1}) - H(Y, X_{d+1}) = H\left(X_d|Y, X_{d+1}\right)$$

and obtain

$$\begin{aligned} W_{GNB}^{(d+1)} &= \sum_{i=1}^{d+1} H(X_i) + H(Y) - \sum_{i=1}^{d-1} H(X_i|Y, X_{i+1}) - H\left(X_d|Y, X_{d+1}\right) \\ &\quad - H(Y, X_{d+1}) = \\ &= \sum_{i=1}^{d+1} H(X_i) + H(Y) - \sum_{i=1}^{d} H(X_i|Y, X_{i+1}) - H(Y, X_{d+1}) \end{aligned}$$

which completes the proof. □

Theorem 9. *The GNB approximation gives an approximation at least as good as NB approximation (the weight of the approximation is larger or equal).*

Proof. First, let us observe that formula (8) is true for any permutation of indices; therefore, this is particularly true for the case of chain numbering.

We compare the weight of the two approximations.

$$\begin{aligned} W_{NB}^{(d)} &= \sum_{i=1}^{d} H\left(X_i\right) + H(Y) - \sum_{i=1}^{d-1} H\left(X_i|Y\right) - H\left(Y, X_d\right) \leq \\ &\leq \sum_{i=1}^{d} H(X_i) + H(Y) - \sum_{i=1}^{d-1} H(X_i|Y, X_{i+1}) - H(Y, X_d) = W_{GNB}^{(d)} \end{aligned}$$

This is true because $H\left(X_i|Y\right) \geqslant H(X_i|Y, X_{i+1})$ $i = 1, \ldots, d$.

Since a higher weight indicates a better approximation, GNB provides an approximation that is at least as accurate as that of NB. □

5. Greedy algorithms for finding a good fitting GNB structure

In this section, we introduce two algorithms that aim to find the best-fitting GNB structure to the training data. In the first subsection, we give a greedy algorithm (GNBA) that aims to minimize the KL divergence, then we introduce the theory necessary for the second algorithm (GNBO) that guarantees the minimization of the KL-divergence. For both algorithms, the complexities are calculated.

5.1. The greedy algorithm GNB-A

The basic idea of the algorithm called GNB-A is to find an approximation with a maximum weight (8), which is equivalent with minimizing KL divergence. After calculating the information of all triplets that contain the class variable Y we choose the first two attributes such that the information content of (Y, X_{i_1}, X_{i_2}) is maximum. Then we add greedily a new variable X_{new} at a time, such that the maximum of $I(X_{new}, X_j, Y) - I(Y, X_j)$ is achieved, where the variables X_j are already connected. The algorithm selects the most relevant variable at each step, considering the variables already connected. This way tries to avoid redundancy.

The pseudo-algorithm is given in Algorithm 1.

Algorithm 1 GNB-A.

procedure GNB-A(D)
Input: Training samples D with d features $X_1, \dots, X_d$ and class variable Y
Output: A sorted list of triplets
$\mathcal{X} := \{X_1, X_2, \dots, X_d\}$
$X_{i_1}, X_{i_2} := \arg\max_{X_i, X_j \in \mathcal{X}} I(Y, X_i, X_j)$
$L_3 := [(Y, X_{i_1}, X_{i_2})]$
$L_2 := [(Y, X_{i_1}), (Y, X_{i_2})]$
$\mathcal{X} := \mathcal{X} \backslash \{X_{i_1}, X_{i_2}\}$
while $\mathcal{X}$ is not empty **do**
$e^*, X_{new} := \arg\max_{e \in L_2, X_j \in \mathcal{X}} (I(e, X_j) - I(e))$
Append (e^*, X_{new}) to L_3
Append (Y, X_{new}) to L_2
$\mathcal{X} := \mathcal{X} \backslash \{X_{new}\}$
end while
return L_3
end procedure

Complexity of the GNB-A algorithm

Overall, the Algorithm requires the following number of additions, subtractions, multiplications, divisions, or comparisons:

$$O(n^2 d^2) + O(n^2\ d) + O(d^2) + O(d^3) \leq O(n^2 d^2 + d^3) = O(d^2(n^2 + d))$$

If d (the number of variables or columns) is less than n^2 (the number of entries or rows, squared), then $O(d^2(n^2 + d)) < O(d^2(n^2 + n^2)) = O(d^2 n^2)$. So if we have at least $\sqrt{d}$ entries in our dataset, then the runtime of the algorithm is $O(d^2 n^2)$, otherwise it is $O(d^2(n^2 + d))$. The detailed calculation can be found in Appendix A.

5.2. The optimal algorithm GNB-O

Our second algorithm GNBO also aims to find a GNB approximation with the maximum weight (8). We will also prove that this finds the optimal GNB structure i.e. best fitting to the training data, under the assumption that the best fitting structure contains the cluster(triplet) with maximum information content. To achieve this, we have to introduce a new methodology, which will be outlined as follows.

The information content of all triplets which contain the classifier is calculated and it is chosen the one with the maximum information content is chosen.

$$(Y, X_{i_1}, X_{i_2}) = \arg\max_{i,j \in V} I(Y, X_i, X_j). \tag{11}$$

Having the first cluster $(0, i_1, i_2)$, we define a new directed and weighted graph on $V \cup \{0\} = \{0, 1, \dots, d\}$ as follows.

Definition 10. We call **auxiliary directed graph structure** on the set of vertices $\{0, 1, \dots, d\}$, a directed graph with the following properties:

(i) vertex 0 has no parent, it is called the root of the graph;
(ii) the parent of vertex i_1 is vertex 0;
(iii) the parent of vertex i_2 is vertex i_1
(iv) from vertex i_1 and vertex i_2 to all other vertices we have only one-directional edges;
(v) for all vertices i, j different from i_1 and i_2 we have an edge (i, j) and an edge (j, i),
(vi) there are no self-pointed edges;

Shortly written the set of edges of the auxiliary graph structure is as follows:

$$\mathcal{E} = \{(0, i_1)\} \cup \{(i_1, j) | j \in V \setminus \{i_1\}\} \cup \{(i_2, j) | j \in V \setminus \{i_1, i_2\}\} \cup \{(i, j) | i, j \in V \setminus \{i_1, i_2\}, i \neq j\}. \tag{12}$$

Now we define a key component, namely the weights of the edges of the auxiliary graph.

Definition 11. The **weights of the edges of the auxiliary graph structure** are defined as follows: (a) $w(0, i_1) = I(Y, X_{i_1})$; (b)$w(i_1, i_2) = I(X_{i_1}, X_{i_2})$; (c) For any other edge $(i, j) \in \mathcal{E} \setminus \{(0, i_1), (i_1, i_2)\}$ the weight of the edge is defined as follows:

$$w(i, j) = I(Y, X_i, X_j) - I(Y, X_i). \tag{13}$$

Definition 12. The weighted graph $G(V, \mathcal{E})$, where the set of edges $\mathcal{E}$ is defined by (12) and the weights of the edges are defined in (13) is called **auxiliary graph of the GNB graph**.

We recall the definition of an arborescence in a directed graph, as we want to maximize its weight.

Definition 13. An **arborescence** is a directed graph with a root vertex 0 and the property that to all other vertices v there is exactly one directed path from 0 to v.

For finding the maximum weighted arborescence in a directed graph, Chu Liu Edmond's algorithm [31] was proposed.

Observe that the maximum arborescence of the auxiliary graph contains the edges $(0, i_1)$ and (i_1, i_2). This is true because of Definition 13 and since vertex i_1's single parent is vertex 0, and vertex i_2's single parent is vertex i_1.

A consequence of Definition 13 is that each vertex of an arborescence has only one parent. Easy to see, because if a vertex v would have two parents $p_1(v)$ and $p_2(v)$, then by definition, for each of the parents we have a single directed path from the root.

We apply Chu Liu Edmonds algorithm [31] to the auxiliary graph.

The algorithm's output is a maximum weighted arborescence with the property that for each vertex $i \in V$ a unique directed path from 0 to i is defined.

Definition 14. We define the **Optimized Generalized Naive Bayes** structure based on the maximal weighted arborescence as follows:

(i) The triplet (Y, X_{i_1}, X_{i_2}) defined by formula (11) is the first cluster of the cherry tree.
(ii) For any $k \in V \setminus \{i_1, i_2\}$ there exists a unique directed path $0, i_1, i_{k_1} \dots i_{k_s} = k$ from 0 to k, such that $i_{k_1}, \dots i_{k_s} \in V$. Based on this we will define the other clusters of the cherry tree as follows $(Y, X_{p(k_i)}, X_{k_i})$, for all $k_i \notin \{0, i_1, i_2\}$, $k_i \in \{i_{k_1}, \dots i_{k_s} = k\}$

Now we can state an important result of our paper.

Theorem 15. *The Optimized Generalized Naive Bayes given in Definition 14 is the maximum weighted GNB approximation if this structure contains the triplet (Y, X_{i_1}, X_{i_2}) with maximum information content.*

Proof. We know that the triplet (Y, X_{i_1}, X_{i_2}) with the maximum information content is in the GNB structure. From Definition 14 we know that any arborescence of the auxiliary graph contains $(0, i_1)$ and $(0, i_2)$; therefore in the weight of the arborescence contains $I(Y, X_{i_1})$ and $I(X_{i_1}, X_{i_2})$. Any new k_s is connected to an existing $p(k_s)$ by a directed edge $(p(k_s), k_s)$ which adds to the weight of arborescence the

term $I(Y, X_{p(k_s)}, X_{k_s}) - I(Y, X_{p(k_s)})$. This way, the algorithm finds the maximum weighted arborescence. To each of the two nodes contained by a directed edge of an arborescence, we adjoint the vertex Y, in this way, the clusters (triplets) of the GNB structure are defined. If we had a GNB with a larger weight, then the maximum weighted arborescence would not be optimal, which contradicts the Chu-Liu-Edmonds algorithm, which guarantees finding the maximum weighted arborescence. □

The weight of the arborescence (output of Chu-Liu-Edmonds alg.) differs from the weight of the optimal approximation by the weight of the first two edges, which are present in the weight of the arborescence, and are not present in the weight of GNB, therefore we have to subtract them. Although the weight of the GNB approximation contains the information content of the first triplet, which is not contained by the weight of arborescence. This is why we have to add it to it.

We emphasize here that GNB-O (Algorithm 2) finds the optimal GNB structure in the training data, on the test data we have no such guarantee.

Algorithm 2 GNB-O.

procedure GNB-O(D)

Input: Training samples D with d features $X_1, \dots, X_d$ and class variable Y

Output: A list of triplets and a list of pairs.

$\mathcal{X} := \{X_1, X_2, \dots, X_d\}$

$X_{i_1}, X_{i_2} := \arg\max_{X_i, X_j \in \mathcal{X}} I(Y, X_i, X_j)$

We create a $(d+1) \times (d+1)$ score matrix S, where the (i, j)-th cell is the score for the edge $j \to i$:

- Row 0 and column 0 belongs to Y, row i and column i belongs to X_i.
- $S[i_1, 0] := I(Y, X_{i_1})$
- $S[i_2, i_1] := I(X_{i_1}, X_{i_2})$
- $S[j_1, j_2] := I(Y, X_{j_1}, X_{j_2}) - I(Y, X_{j_2})$, $\forall j_1 \in \{1, 2, \dots, d\} \backslash \{i_1, i_2\}$, $\forall j_2 \in \{1, 2, \dots, d\}$, $j_1 \neq j_2$
- The remaining elements are zero.

We apply the Chu-Liu-Edmonds algorithm to find the maximum weighted arborescence in the directed graph belonging to S.

From the maximum weighted arborescence we create the list of triplets L_3 and the list of pairs L_2.

return L_3, L_2

end procedure

Complexity of the GNB-O algorithm

The GNB-O algorithm first fills out the S matrix by calculating all pair and triplet information contents, both of which we have seen can be done in $O(n^2 d^2)$ time. Then we run the Chu Liu Edmonds algorithm on the directed graph belonging to S. Its runtime is $O(|E| \cdot |V|)$, where $|E|$ are the number of edges and $|V|$ are the number of vertices. In our case, $|V| = d + 1 = O(d)$ and $|E| = (d-2)^2 + 2 = O(d^2)$, so the runtime of the Chu-Liu-Edmonds algorithm is $O(|E| \cdot |V|) = O(d^2 \cdot d) = O(d^3)$ Similarly to GNB-A, the total runtime of GNB-O is $O(n^2 d^2 + d^3) = O(d^2(n^2 + d)) \overset{(*)}{=} O(n^2 d^2)$ where $(*)$ holds true if $d < n^2$.

6. Relations of the introduced GNB approach to former improvements of NB

This section is dedicated to exploring the similarities and differences between structures, algorithms, and their complexities. From a structural point of view, we prove that the graph structure defined on the explanatory variables is a tree, which is also the case in the graph structures used by TAN and ATAN algorithms.

An important link, that connects this work to the structure extension introduced in [13] is expressed by the following theorem.

Theorem 16. *A graph structure defined on the vertex set* $\{0, 1, \dots, d\}$ *has a GNB structure (Definition 2) if and only if it has the property that its restriction to the vertices* $\{1, \dots, d\}$ *form a tree.*

For the proof of the theorem see Appendix B

The closest structures and algorithms to ours are TAN, ATAN, and Ghofrani's algorithm, which are presented in [13] and [22], while their complexities are calculated in [22] (Page 6). The time complexity of the TAN algorithm is $O(nd^2) + O(k(dv)^2) + O(d^2 \log_2(d))$, while the time complexity of the ATAN algorithm is $O(nd^2) + O(k(dv)^2) + O(d^3 \log_2(d))$, where n is the number of training data (rows), d is the number of features (columns), v is the average number of values per feature (which we have set to 5 in our evaluation), and k is the number of classes (which in our evaluation setting is 2, a binary $0-1$ value).

The time complexity of the LDMLCS algorithm is $O(nd^{l_c}) + O(k(dv)^2) + O(d^2 \log_2(d)) + O(d^3 l_c^3) + O(nkn_{clq})$ where n, d, v, k are the same as before, l_c is the dimensionality of the table containing probability estimates for each class (in our case, 2), and n_{clq} is the number of cliques (clusters) generated by the algorithm (which in our case is d).

To be able to compare the runtime of our algorithm with TAN, ATAN, and Ghofrani's LDMLCS algorithm, we will use the substitutions $v = 5$, $k = 2$, $l_c = 2$ and $n_{cql} = d$; We obtain for the runtime of TAN

$O(nd^2) + O(d^2) + O(d^2 \log_2(d)) \leq O(d^2(n + \log_2(d)))$;

the runtime of ATAN

$O(nd^2) + O(d^2) + O(d^3 \log(d)) \leq O(d^2(n + d \log_2(d)))$.

and the runtime of Ghofrani's algorithm

$$O(nd^2) + O(d^2) + O(d^2 \log(d)) + O(d^3) + O(nd) \leq O(nd^2 + d^3) =$$
$$= O(d^2(n+d)) \overset{(*)}{\leq} O(d^2 \cdot 2n) = O(d^2 n)$$

where $(*)$ holds true if there is more training data (rows) than attributes (columns). We have already seen that the runtime of GNB-A is $O(d^2 n^2)$, which means that TAN is faster as we can assume that $\log_2(d) < n$, while ATAN is also slightly faster if $n + d \log_2(d) < n^2$, which holds true for datasets which have more training data than features, since $n + d \log_2(d) < n + d\frac{d}{2} < n + n\frac{n}{2} = n + \frac{n^2}{2} < n^2$. Furthermore, Ghofrani's LDMLCS is also faster if we set $l_c = 2$, as its runtime is $O(d^2 n)$.

The novelty of the LDMLCS algorithm introduced by Ghofrani et.al.in [22] was that it exploits the concept of conditional independence between random variables in the context of probabilistic graphical models. The cherry tree-based algorithms, the GNB-A and GNB-O algorithms exploit conditional independence to maximize the information content of the model and to reduce redundancy.

The main difference, from an information-theoretical point of view, between LDMLCS algorithm in [22], the TAN [13] and the ATAN in [17] and the algorithms introduced here is that the LDMLCS and TAN and ATAN consider only the bivariate conditional mutual information to weight the edges between the explanatory attributes whereas, in our algorithms, we use the fact that any structure that has a tree-like structure between the explanatory variables has a special cherry tree probability distribution. Based on this the expression of KL divergence (5) between this probability distribution and the sample data depends on the graph structure. By minimizing the KL-divergence we find this graph structure and exploit the conditional independence relation of the approximation, this way the redundancy is also reduced. We emphasize here that in Ghofrani's paper [22], as an alternative method they apply the cherry tree algorithm to the set of attributes, which is different from our methods introduced here, in which each cluster contains the Y classification variable.

7. The classification process and feature selection methods

Based on the training data, we fit a GNB probability distribution. As a result, we obtain a set of clusters and a set of separators that define the

GNB structure. With these components as inputs, we outline the main steps of the classification process.

Input: The set of clusters and the set of separators which contain the index 0 corresponding the Y classifier: $(0, i_k, j_k) \in C$ and $(0, i_k) \in S$ and the corresponding marginal probability distributions: (Y, X_{i_k}, X_{j_k}) and (Y, X_{i_k}).

Output: A probability distribution over the classes $y \in \{1, \dots, m\}$ and the classification of the element into the most probable class.

Step 1: For $y = 1, \dots, m$ calculate the approximation of the probability distribution of the training data by using the cherry tree approximation:

$$\widetilde{P}(y, x_1, \dots, x_d) = \frac{\prod_{(0,i_k,j_k)\in C} P\left(y, x_{i_k}, x_{j_k}\right)}{\prod_{(0,i_k)\in S} P\left(y, x_{i_k}\right)^{v_{k-1}}}.$$

Step 2: If there exist $y \in \{1, \dots, s\}$ such that $\widetilde{P}(y, x_1, \dots, x_d) \neq 0$ then the vector $\mathbf{x} = (x_1, \dots, x_d)$ is classified in the class y with probability

$$\widetilde{P}(y|x_1, \dots, x_d) = \frac{\widetilde{P}(y, x_1, \dots, x_d)}{\sum_{w=1}^{s} \widetilde{P}(w, x_1, \dots, x_d)}, \quad y(\mathbf{x}) = \arg\max_y \widetilde{P}(y, x_1, \dots, x_d).$$

Else, go to Step 3.

Step 3: For all k with $P\left(y, x_{i_k}, x_{j_k}\right) = 0$ calculate $P\left(y, x_{i_k}\right)$ and $P\left(y, x_{j_k}\right)$.

If for any i_k: $P\left(y, x_{i_k}\right) = 0$, substitute it with respect to $P(Y = y) \cdot P\left(X_{i_k} = x_{i_k}\right)$, then go to Step 4. Else, go to Step 4.

Step 4: Calculate $P\left(y, x_{i_k}, x_{j_k}\right) := \frac{P\left(y, x_{i_k}\right) P\left(y, x_{j_k}\right)}{P(y)}$.

Feature selection and feature importance score

The feature selection can be achieved in two stages: during the learning process and the validation process. On the training data, the relevant features can be selected based on how much weight (information) is gained by adding a new feature (by adding a new cluster) through algorithm GNB-A. The weight (8) of the cherry tree approximation never decreases (typically increases) on the training data when a new cluster is added, see Appendix C. This added weight also depends on the features that have been added so far. This provides control over redundancy. When one observes that the total weight of the cherry tree does not increase significantly after adding the $k+1$-th cluster to the graph, then we use only the features contained by the first k clusters ($k+2$ explanatory variables).

Feature selection can also be achieved in the second stage, based on the validation set, using all the features in the order designed by our GNBA algorithm and the output graphs (like in Fig. 5) for precision, recall and F1-score, as a function of triplets connected step by step to the model. The users/ experts can decide, based on the most important accuracy scores for their interests, on the relevant clusters, starting from the first one and taking them in the order they appear on the OX axis. The features which are in the reunion of these clusters should be used for classification. We have to underscore here, that the role of a new attribute X_{i_k} added by a new triplet is mirrored by how the given accuracy measures changes by adding the attribute X_{i_k} to the model consisting of the $X_{i_1} \dots X_{i_{k-1}}$ attributes. We regard these procedures as new methods for feature selection that can be used as a pre-processing step before applying other classification algorithms. In the case when feature selection based on accuracy scores is used, the model has to be tested on unseen test data.

Table 1
Short description of the datasets we used in this paper. The usage of the datasets is described in 8.1.

Dataset	Num. of attributes (columns)	Num. of continuous attributes	Num. of discrete attributes	Num. of entries (rows)
WDBC [35]	30	30	0	569
Heart Disease [36]	13	5	8	297
Diabetes [37]	16	1	15	519
Thyroid Ann Train [38]	21	15	6	3771
Parkinson [39]	22	22	0	197

Table 2
The number of same triplets identified in all five runs and the number of same triplets identified in 4 runs out of 5.

Dataset	Num.of: same triplets /all triplets /all possible triplets in all 5 random splits	Num.of: same triplets /all triplets /all possible triplets in 4 out of 5 random splist
WDBC [35]	22/29/4060	2/29/4060
Heart Disease[36]	7/12/286	2/12/286
Diabetes[37]	8/14/560	2/14/560
Thyroid [38]	11/18/1330	2/18/1330
Parkinson[39]	15/21/1540	6/21/1540

8. Numerical results

The introduced algorithms are designed for the classification task. We will use evaluation metrics specific to this problem: accuracy, precision, recall, F1-score Appendix E. GNB-A and GNB-O algorithms were implemented in Python, and are publicly available for testing at https://github.com/annaorszag/GeneralizedNaiveBayes.

8.1. Data preparation methods

The algorithms introduced in this paper are developed for finite discrete data. In general, the datasets (Table 1) contain discret, categorical, and continuous attributes. The categorical variables can be treated similarly to the discrete ones. The algorithms introduced in this paper are based on information content, which depends only on the probabilities, not on the "values" taken on by the variables. This way, our algorithms are more flexible in handling different kinds of attributes (categorical and discrete). For discretizing the continuous attributes, we chose a quite general method based on quantiles. However, there are many interesting new discretization methods, even based on NB, such as [32] and [33]. The quantile discretization that we use [34] is described in detail in the Appendix D. In the illustrative examples of this paper, we discretized the continuous random variables in 5 intervals. If a variable X_i takes on more than 5 distinct values, then it is discretized into at most 5 values.

8.2. Data description and classification results

To have a general setting and to be able to apply the same methodology to all datasets, we choose to use the quantile discretization into five intervals. All of our results are based on the same type of discretization, therefore these results should be not compared with other results based on other discretization. We omitted all rows with missing values. All the compared algorithms were run 5 times on each dataset with a random 15 % test set selection, having the same random five splits (seeds: 0, 3, 6, 9, 12) for all compared algorithms: GNB-A, GNB-O, NB, TAN . The results are compared by the means of accuracy, precision, recall and F1 score, in Table 3. This way the problems which occur because of unbalanced data can also be observed and compared. It is essential to note that for all algorithms, accuracy measures in the tables were calculated using all explanatory variables.

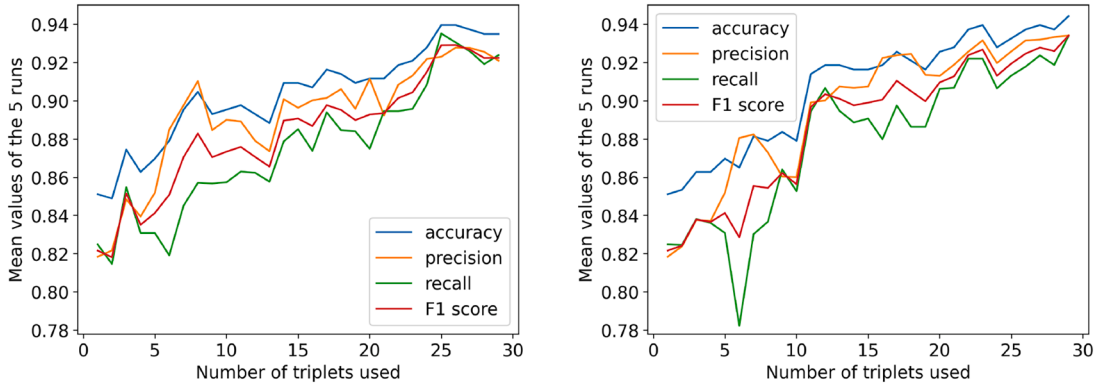

Fig. 5. Accuracy results on WDBC test data: the mean over the same 5 splits, of the different accuracy measures (coloured differently) as a function of the number of clusters added via GNB-A (left), respectively via GNB-O (right). One can observe that by using the greedy algorithm GNB-A the highest F1-score is achieved with 25 clusters (26 features) whereas in the case of GNB-O the accuracy scores have a improving tendency until it uses all explanatory variables.

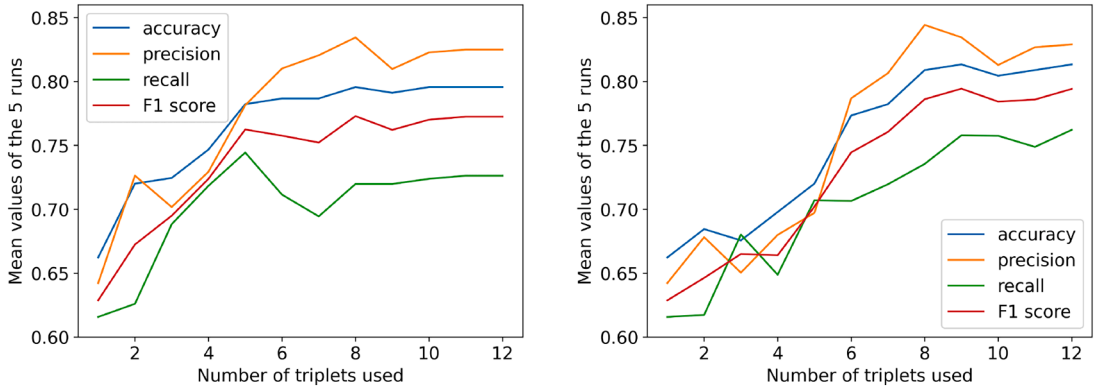

Fig. 6. Accuracy results on Heart Disease test data: the mean over the same 5 splits, of the different accuracy measures (coloured differently) as a function of the number of clusters added via GNB-A (left), respectively via GNB-O (right). One can observe that 8 clusters (9 explanatory variables) appear to yield the best scores, in terms of F1.

Table 3
Accuracy results for algorithms GNB-A, GNB-O, NB, TAN on five datasets. The table contains the mean over the same 5 splits, for each dataset, for all accuracy measures. GNB-A performs in some cases better than GNB-O on the test data; This may be due to the average of five runs or overfitting.

Accuracy	GNB-A	GNB-O	NB	TAN
WDBC dataset [35]	0.9349	**0.9442**	0.8419	0.8884
Heart Disease dataset [36]	0.7956	**0.8133**	0.7156	0.7511
Diabetes dataset [37]	0.8692	0.8692	0.8692	**0.9154**
Thyroid dataset [38]	0.9212	0.9208	0.9283	**0.941**
Parkinson dataset [39]	**0.8933**	0.8867	0.7533	0.76
Precision	GNB-A	GNB-O	NB	TAN
WDBC dataset [35]	0.921	0.9341	0.8688	**0.9777**
Heart Disease dataset [36]	0.8249	**0.828**	0.6913	0.7753
Diabetes dataset [37]	0.9419	0.9419	0.8243	**0.9605**
Thyroid dataset [38]	**0.9717**	**0.9717**	0.9286	0.971
Parkinson dataset [39]	**0.9628**	0.9462	0.8158	0.8838
Recall	GNB-A	GNB-O	NB	TAN
WDBC dataset [35]	0.9238	**0.9339**	0.7316	0.7542
Heart Disease dataset [36]	0.7262	**0.7621**	0.7214	0.6405
Diabetes dataset [37]	0.845	0.845	0.8261	**0.9035**
Thyroid dataset [38]	0.9425	0.9421	**0.9996**	0.9655
Parkinson dataset [39]	0.8968	**0.9045**	0.8718	0.7916
F1 score	GNB-A	GNB-O	NB	TAN
WDBC dataset [35]	0.9213	**0.9333**	0.7928	0.8501
Heart Disease dataset [36]	0.7677	**0.7887**	0.6989	0.7008
Diabetes dataset [37]	0.8908	0.8908	0.8243	**0.9305**
Thyroid dataset [38]	0.9569	0.9567	0.9628	**0.9682**
Parkinson dataset [39]	**0.9271**	0.924	0.8406	0.8315

The results of our two new algorithms GNB-A and GNB-O are presented on different datasets. The accuracy measures (accuracy, F1, precision and recall) are calculated as a function of the number of triplets (clusters) used in the GNB stucture. The average results of these runs can be seen in this subsection (Figs. 5–8).

We want to highlight here that the graphs of GNB-A have the advantages to give the user information on overfitting, and on the relevant features to be used. GNB-O constructs the structure so that by adding all the features, the structure becomes optimal, maximizing its weight among all GNB structures.

In the **Diagnostic Wisconsin Breast Cancer Database (WDBC)** [35] the aim is to classify a tumor as benign or malignant. The data is slightly unbalanced 0.63 are benign (B) and 0.37 are malign (M). The average accuracy results of the 5 runs on the test data can be seen in Fig. 5.

The **Heart Disease** database [36] contains 4 datasets from which we used the Cleveland dataset. The class variable takes on integer values from 0 (no presence of heart disease) to 4. We concentrated on distinguishing the presence (values 1, 2, 3, 4) from the absence (value 0) of the disease in a patient. The average results over 5 runs on the test data are shown in Fig. 6.

The **Diabetes** dataset [37] contained no missing values and only one continuous attribute (Age), which was discretized with the method explained in Subsection 8.1. Fig. 7 shows the average accuracy over 5 runs of the two algorithms on the same test data.

In the **Thyroid** dataset [38] we have selected the Thyroid Ann Train dataset, as it contained mostly binary values. The six continuous attributes were discretized, and the classes 1 and 2 (irregular thyroid level) were joined into a single class. We have then removed column $b14$ as it contained 3771 1's but only a single 0. In Fig. 8, one can see, how

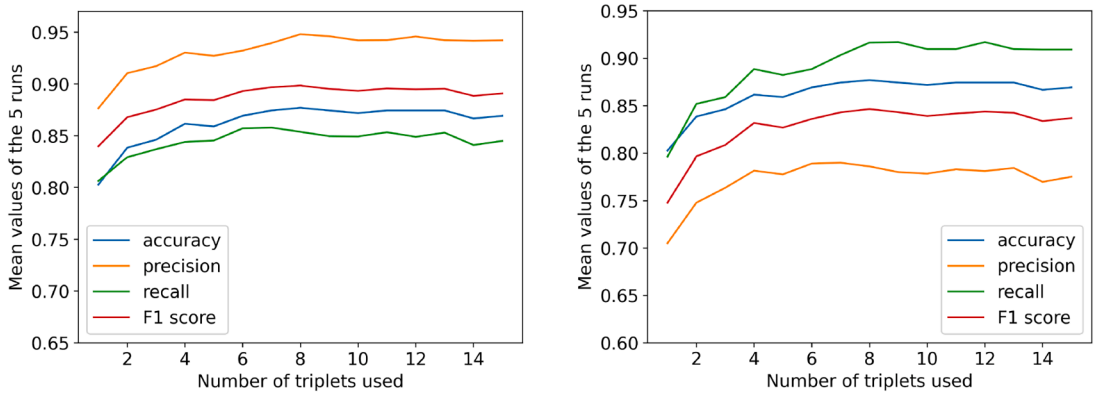

Fig. 7. Accuracy results on Diabetes test data: the mean over the same 5 splits, of the different accuracy measures (coloured differently) as a function of the number of clusters added via GNB-A (left), respectively via GNB-O (right).One can observe that by using 8 clusters (9 variables), we obtain a high accuracy, precision, recall, and F1 score on the test data.

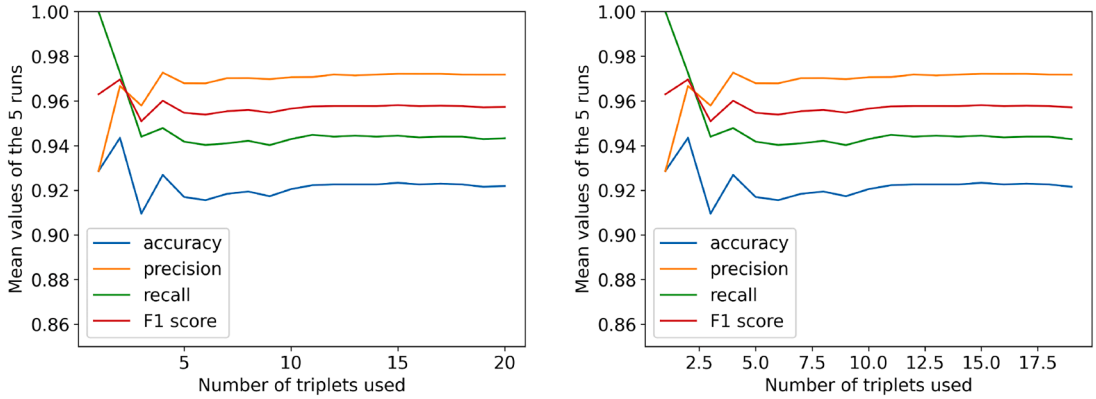

Fig. 8. Accuracy results on Thyroid test data: the mean over the same 5 splits, of the different accuracy measures (coloured differently) as a function of the number of clusters added via GNB-A (left), respectively via GNB-O (right).One can observe that by using 4 clusters (5 features), we obtain a high accuracy, precision, recall, and F1 score on the test data. Adding new cluster does not improve the accuracy.

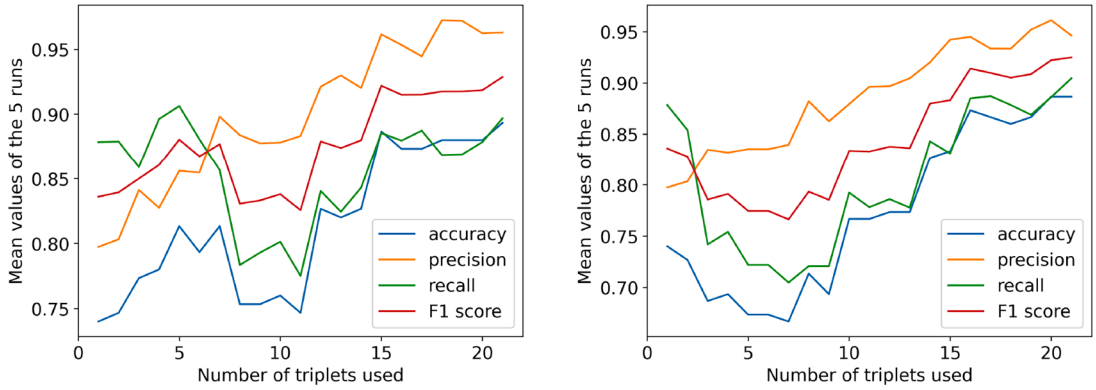

Fig. 9. Accuracy results on Parkinson test data: the mean over the same 5 splits, of the different accuracy measures (coloured differently) as a function of the number of clusters added via GNB-A (left), respectively via GNB-O (right). One can observe that the accuracy scores were the highest when nearly all features were included.

the average accuracy measures depend on the number of clusters on the same test data.

The **Parkinson's Disease (Parkinson)** data set [39] contains biomedical voice measurements. The aim is to discriminate healthy people from those with PD.The average results of the runs can be seen in Fig. 9.

In addition to the accuracy graphs, we assessed the robustness of the GNB structure across five random samplings. We recorded the number of clusters obtained in all five runs, as well as in four out of the five runs. Refer to Table 2 for details. It is quite interesting to note that the number of common triplets closely corresponds to the number of how many triplets should be use based on the accuracy Figs 5–9. One

explanation may be that the common triplets (clusters) are the most important for classification.

Comparison of GNB-A and GNB-O with NB and TAN

All average accuracy measures were calculated using all explanatory features.

9. Conclusion

This paper introduced the Generalized Naive Bayes structure as a particular case of third-order cherry trees. Two new algorithms were introduced to construct a well fitting GNB structure to a dataset (in the sense of KL minimization), which can then be utilized for classification. GNB-A is a greedy algorithm which can also be used for feature selection in the training phase and for observing when overfitting occurs on the validation set. The GNB-O algorithm provides the optimal GNB structure on a given input feature set, when the most informative cluster is contained in the best-fitting GNB structure.

Advantages: A significant strength of the new algorithms, in contrast to black-box algorithms, is their transparency. Therefore the GNB structures can be considered an interpretable machine learning tool. Moreover, the algorithms provide new feature selection mechanisms, one during the learning phase and the other based on the graphical representation of the accuracy measures on the validation data as a function of the number of clusters (features). Additionally, selecting relevant features can significantly enhance subsequent research. A huge advantage of our approach is that all marginal probability distributions used for classification are of order 3; therefore, our algorithms also work well in cases with many features and relatively small datasets. We proved that the approximation assigned to the GNB structure gives at least as good an approximation as the approximation associated with the NB structure. We also proved that our GNB-O algorithm gives the optimal GNB-approximation in terms of minimizing the KL-divergence, over the GNB family. All the tested algorithms were tested and compared under the same conditions. It turned out that our algorithms work better in many cases than the NB and TAN algorithms. Our algorithms were implemented in Python and are publicly available for testing at https://github.com/annaorszag/GeneralizedNaiveBayes.

Limitations: The new algorithms were not compared to other algorithms concerning overfitting, although overfitting was discussed for the two new algorithms based on output graphs that included the accuracy measures as a function of the number of clusters (features). A drawback of the new algorithms is that the structures depend on how many intervals are used for discretisation. It is not straightforward to decide on the number of intervals. A very large dataset, i.e. large number of entries ($> 10^6$) may slow down the algorithms.

Future work: We presume that our feature selection algorithms can be used as a first step to improve the performance of other algorithms, too. This analysis is one of our plans for future work. The methods presented here can also be easily adapted to a broader range of classification tasks, beyond just binary classifications. We also plan to release a Python package containing these algorithms.

We hope that this work will have a great impact on a wide range of classification problems related to different fields, especially where obtaining large sample data is very costly or time-consuming.

CRediT authorship contribution statement

Edith Alice Kovács: Writing – review & editing, Writing - original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization; **Anna Ország:** Writing – review & editing, Validation, Software, Methodology, Investigation; **Dániel Pfeifer:** Writing – review & editing, Writing - original draft, Visualization, Validation, Software, Investigation; **András Benczúr:** Writing – original draft, Supervision, Resources, Project administration.

Data availability

We have shared the links to all data and code used in the paper.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Edith Alice Kovacs reports financial support was provided by Ministry of Innovation and Technology of Hungary. Daniel Pfeifer reports financial support was provided by Ministry of Innovation and Technology of Hungary. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We are grateful to anonymous reviewers for their detailed and thoughtful comments and suggestions, which have helped to improve the quality of this paper. The research reported in this paper is part of project no. BME-NVA-02, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021 funding scheme.

Appendix A. Complexity of GNBA Algorithm

We calculate the runtime of Algorithm 1. Firstly we know that Python 3 calculates $\log_2$ in $O(1)$ time [40]. To calculate $P(X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2}, X_{i_3} = x_{i_3})$ based on a dataset with n entries, we need to evaluate the number of realizations of the form $(X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2}, X_{i_3} = x_{i_3})$, which takes $3n$ comparisons and 1 division, which is again an $O(n)$ amount. In particular, to calculate $P(X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2})$ and $P(X_{i_1} = x_{i_1})$ we have the same $O(n)$ amount of operations. For simplicity we use $P(x_{i_1}, x_{i_2}, x_{i_3})$ instead of $P(X_{i_1} = x_{i_1}, X_{i_2} = x_{i_2}, X_{i_3} = x_{i_3})$. To calculate the argument of the information content, we have:

$$\underbrace{P(x_{i_1}, x_{i_2}, x_{i_3})}_{O(n)} \underbrace{\log_2}_{O(1)} \left(\frac{\overbrace{P(x_{i_1}, x_{i_2}, x_{i_3})}^{O(n)}}{\underbrace{P(x_{i_1})}_{O(n)} \underbrace{P(x_{i_2})}_{O(n)} \underbrace{P(x_{i_3})}_{O(n)}} \right)$$

which takes $O(n) + O(1) + 4O(n) = O(n)$ steps to evaluate. Using this, we can show the complexity of calculating the information content for three- and respectively two-order marginals is $O(n^2)$. Let us consider

$$I_3(X_1, X_2, X_3) = \sum_{(x_1,x_2,x_3)} P(x_1, x_2, x_3) \log_2 \left(\frac{P(x_1, x_2, x_3)}{P(x_1)P(x_2)P(x_3)} \right)$$

where the summation runs through all the realizations of (X_1, X_2, X_3). These triplets can be obtained by going through the dataset of n rows (sample size), and storing the unique (x_1, x_2, x_3) values in a dictionary, with each value of the dictionary being the number of occurrences. Such a dictionary can be constructed in $O(n^2)$ steps, since for each row of the dataset (x_1, x_2, x_3), we will (potentially) need to go through the entire dictionary we have constructed so far, to see if the triplet (x_1, x_2, x_3) already exists or not. If it does, then increase its value (occurrence) by one, otherwise, add it to the dictionary with value 1.

Next, we need to sum up an $O(n)$ amount on an index set of size $O(n)$, which can also be done in $O(n^2)$ steps.

Similarly, the information content for a pair of random variables can also be calculated in $O(n^2)$ steps. The difference is that the keys of the dictionary will be of size 2 instead of 3.

Now we can move onto the steps of Algorithm 1.

In **Step** 1, the Algorithm calculates all triplet-information contents where one of the elements is Y, namely $I_3(Y, X_i, X_j)$, for all $i, j \in$

$\{1,\dots,d\}$. There are $\binom{d}{2}$ choices for (X_i, X_j), so there are $\binom{d}{2}$ of such triplets. We have already shown that calculating an I_3 information content can be done in $O(n^2)$ steps, so calculating $\binom{d}{2} = O(d^2)$ of them takes $O(n^2 d^2)$ operations. In **Step** 2, the Algorithm calculates all pair-information contents where one of the elements is Y, namely $I_2(Y, X_i)$. There are d options for X_i, so similarly to the previous case, this can be done in $O(n^2\ d)$ steps. In **Step** 3, the Algorithm calculates all information content differences $I_3(.) - I_2(.)$ where the arguments of I_2 are part of the arguments of I_3, which are of the form: $I_3(Y, X_i, X_j) - I_2(Y, X_i)$ and $I_3(Y, X_i, X_j) - I_2(Y, X_j)$. So there are $2\binom{d}{2}$ of them, which is an $O(d^2)$ amount. Therefore in Step 3, the Algorithm calculates $O(d^2)$ differences. Finally in **Step** 4, for all $k \in \{1,\dots,d-1\}$, the Algorithm chooses the $\arg\max$ of the $I_3(.) - I_2(.)$ values in $O(d)$ steps.

In a general **step** $k = 4,\dots,d-1$, the algorithm has already selected k $\arg\max$'s, so there are $d-k$ that remain. In order to calculate $\arg\max(I_3 - I_2)$, we choose the highest information content nodes (X_a, X_b) to add, where $X_a \in \{X_1,\dots,X_k\}$ and $X_b \in \{X_{k+1},\dots,X_d\}$. There are $k(d-k)$ such choices.

The maximum and the argmax of $k(d-k)$ elements can be obtained by going through each element once, and saving the maximum so far, and the argmax so far, which takes $k(d-k)$ comparisons.

So in steps $k = 1,\dots,d-1$, the total number of comparisons is

$$\sum_{k=1}^{d-1} k(d-k) = \sum_{k=1}^{d-1} kd - \sum_{k=1}^{d-1} k^2 = d\frac{(d-1)d}{2} - \frac{(d-1)d(2d-1)}{6} =$$
$$= d(d-1)\Big(\frac{d}{2} - \frac{2d-1}{6}\Big) = d(d-1)\Big(\frac{3d-2d+1}{6}\Big) = \frac{d(d-1)(d+1)}{6}$$
$$= O(d^3)$$

So overall in Steps 1 to 4, the Algorithm requires the following number of additions, subtractions, multiplications, divisions, or comparisons:

$$O(n^2d^2) + O(n^2\ d) + O(d^2) + O(d^3) \le O(n^2d^2 + d^3) = O(d^2(n^2 + d))$$

If d (the number of variables or columns) is less than n^2 (the number of entries or rows, squared), then $O(d^2(n^2+d)) < O(d^2(n^2+n^2)) = O(d^2n^2)$. So if we have at least $\sqrt{d}$ entries in our dataset, then the runtime of the algorithm is $O(d^2n^2)$, otherwise it is $O(d^2(n^2+d))$.

Appendix B. The proof of Theorem 16

Theorem 16 *A graph structure defined on the vertex set* $\{0,1,\dots,d\}$ *has a GNB structure (Definition 2) if and only if it has the property that its restriction to the vertices* $\{1,\dots,d\}$ *form a tree.*

Proof. First we prove that a GNB on $\{0,1,\dots,d\}$ implies the tree structure on $\{1,\dots,d\}$. We do the proof by reductio ad absurdum. Let us suppose that the vertices in $\{1,\dots,d\}$ in a GNB graph structure are not connected tree-wise. This means that there exists a path $\{i_1, i_2, \dots, i_r = i_1\}$, $r > 3$, where $i_1, i_2, \dots, i_r \in V$. We prove this by considering the following two cases:

(i) if $r = 4$ then $0, i_1, i_2, i_3, i_4$, are interconnected, which implies that the maximum clique is of size four (see Fig. B.10), which is in contradiction, with Definition 2.

(ii) if $r > 4$ then we have at least three connected triplets (which share two common vertices): $(0, i_1, i_2)$, $(0, \mu(i_3), i_3)$, $(0, \mu(i_4), i_4)$, where $\mu(i_3)$ is the mother vertex of i_3, and $\mu(i_4)$ is the mother vertex of i_4, and $\mu(i_3) \in \{i_1, i_2\}$, $\mu(i_4) \in \{i_1, i_2, i_3\}$ Since $i_1 = i_r$ it belongs to the first and last triplet, and obviously it does not belong to the intermediate triplets. Therefore the running intersection property is not fulfilled.

Now we will prove the other implication. Let us consider a tree defined on the vertices $\{1,\dots,d\}$. This tree can be expressed in terms of junction tree, with two vertices in each cluster, and one vertex in each separator, see the first row in Fig. B.11. Then we add to each cluster

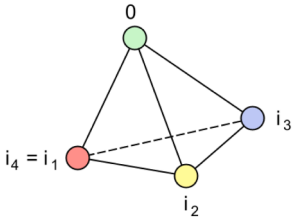

Fig. B.10. The maximum clique is of size four.

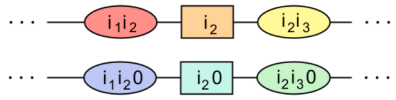

Fig. B.11. Adding the vertex 0 to each cluster.

the vertex 0, because this is connected by definition with all vertices $\{1,\dots,d\}$. It is easy to see that the new junction tree will have three vertices in a cluster and two in each separator (the old vertex and the vertex 0, which is now in all the clusters (see Fig. B.11, the second row). □

Appendix C. Weight increase on training data as a function of number of triplets

We show on the Heart Disease [36] train data how the weight of the approximation (8) grows by adding a new cluster at each step in the training phase. This is a non-decreasing function of the number of clusters/features (Fig. C.12).

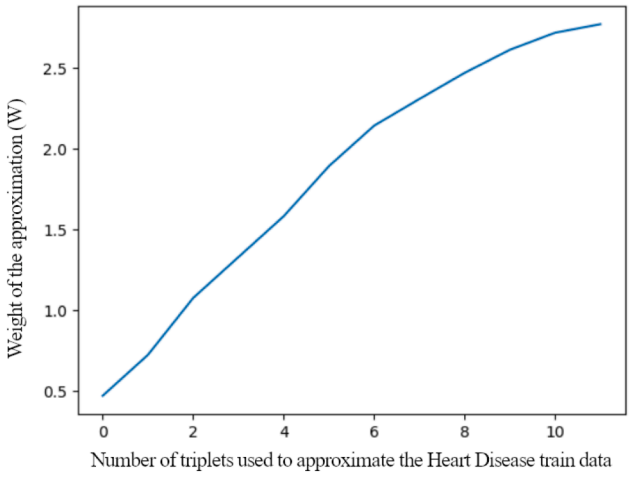

Fig. C.12. The non-decreasing weight of the cherry tree as a function of the clusters added by the algorithm GNB-A in each step in the learning phase, on the Heart Disease [36] training set.

Appendix D. Discretization algorithm

Denoting by $[x_1, x_2, \dots, x_n]$ the values taken on by X_i in the sample, let us first make an ordered list $\hat{X}_i := [x_{\phi(1)}, \dots, x_{\phi(n)}]$ with $x_{\phi(j)} \le x_{\phi(j+1)}$ $\forall j \in \{1,\dots,n-1\}$. Then the quantiles of this attribute are $Q_j = x_{\lfloor \phi(nj/5) \rfloor}$

For $j \in \{1,\dots,4\}$, in our case (4 quantiles along with the 0 %-quantile and the 100 %-quantile split the interval into 5 parts). If for any $i \in \{1,\dots,5\}$, $Q_i = Q_{i+1}$, then we threw away Q_{i+1}, and only used Q_i. Let us denote the number of quantiles remaining after this process by $M \le 5$. And let $Q_0 = \min X_i$, $Q_{M+1} = \max X_i$.

Then the representative value of each interval will be the average of each discretized interval: $a_j = \frac{1}{|\{i : Q_j \le x_i < Q_{j+1}\}|} \sum_{i=0}^{n} x_i \mathbf{1}_{(Q_j \le x_i < Q_{j+1})}$ $\forall j \in$

$\{1, \ldots, M\}$, where $\mathbf{1}$ is the indicator function. Finally, the discretized elements $[x_1, x_2, \ldots, x_n]$ are $[d_1, d_2, \ldots, d_n]$, where $d_k = \sum_{s=0}^{M} a_s \mathbf{1}_{(Q_s \leq x_k < Q_{s+1})}$.

Appendix E. Evaluation metrics

In binary classification, beyond accuracy, precision and recall are often used metrics to evaluate the models. Let the labels we want to predict be 'positive' and 'negative', then we can define the *confusion matrix*, in which the columns show the actual values and the rows show the predicted values:

	Actual		
	Positive	Negative	
Predicted	Positive	#{**True Positive**}	#{**False Positive**}
	Negative	#{**False Negative**}	#{**True Negative**}

There are four possible outcomes according to the actual and the predicted label. The elements of the matrix show the occurrence of each outcome. Based on these accuracy, precision and recall can be defined as follows:

$$\text{Accuracy} = \frac{\#\{\text{True Positive}\} + \#\{\text{True Negative}\}}{\#\{Total\}}$$

$$\text{Precision} = \frac{\#\{\text{True Positive}\}}{\#\{\text{True Positive}\} + \#\{\text{False Positive}\}}$$

$$\text{Recall} = \frac{\#\{\text{True Positive}\}}{\#\{\text{True Positive}\} + \#\{\text{False Negative}\}}$$

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

References

[1] H. Zhang, L. Jiang, L. Yu, Attribute and instance weighted Naive Bayes, Pattern Recognit. 111 (2021) 107674.
[2] A.H. Rabie, N.A. Mansour, A.I. Saleh, A.E. Takieldeen, Expecting individuals body reaction to COVID-19 based on statistical Naïve Bayes technique, Pattern Recognit. 128 (2022) 108693.
[3] W.M. Shaban, A.H. Rabie, A.I. Saleh, M. Abo-Elsoud, Accurate detection of COVID-19 patients based on distance biased Naïve Bayes (DBNB) classification strategy, Pattern Recognit. 119 (2021) 108110.
[4] H. Kim, S.S. Chen, Associative Naive Bayes classifier: automated linking of gene ontology to medline documents, Pattern Recognit. 42 (9) (2009) 1777–1785.
[5] Z. Ye, P. Song, D. Zheng, X. Zhang, J. Wu, A Naive Bayes model on lung adenocarcinoma projection based on tumor microenvironment and weighted gene co-expression network analysis, Infect. Dis. Modell. 7 (3) (2022) 498–509.
[6] J.R. Quinlan, C4.5: Programs for Machine Learning, Elsevier (2014).
[7] P. Domingos, M. Pazzani, Beyond independence: conditions for the optimality of the simple Bayesian classifier, in: Proc. 13th Intl. Conf. Machine Learning, 1996, pp. 105–112.
[8] H. Zhang, The optimality of Naive Bayes, in: Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS, 2004, p. 3.
[9] G. Feng, J. Guo, B.-Y. Jing, T. Sun, Feature subset selection using Naive Bayes for text classification, Pattern Recognit. Lett. 65 (2015) 109–115.
[10] M. Hall, A decision tree-based attribute weighting filter for Naive Bayes, in: International Conference on Innovative Techniques and Applications of Artificial Intelligence, Springer, 2006, pp. 59–70.
[11] C.-H. Lee, F. Gutierrez, D. Dou, Calculating feature weights in Naive Bayes with Kullback-Leibler measure, in: 2011 IEEE 11th International Conference on Data Mining, IEEE, 2011, pp. 1146–1151.
[12] S. Chen, G.I. Webb, L. Liu, X. Ma, A novel selective Naïve Bayes algorithm, Knowl. Based Syst. 192 (2020) 105361.
[13] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (1997) 131–163.
[14] H. Zhang, S. Sheng, Learning weighted Naive Bayes with accurate ranking, in: Fourth IEEE International Conference on Data Mining (ICDM'04), IEEE, 2004, pp. 567–570.
[15] I. Kononenko, Semi-Naive Bayesian classifier, in: Machine LearningEWSL-91: European Working Session on Learning Porto, Portugal, March 6–8, 1991 Proceedings 5, Springer, 1991, pp. 206–219.
[16] R. Blanquero, E. Carrizosa, P. Ramírez-Cobo, M.R. Sillero-Denamiel, Variable selection for Naïve Bayes classification, Comput. Oper. Res. 135 (2021) 105456.
[17] L. Jiang, Z. Cai, D. Wang, H. Zhang, Improving tree augmented Naive Bayes for class probability estimation, Knowl. Based Syst. 26 (2012) 239–245.
[18] F. Zheng, G.I. Webb, Averaged one-dependence estimators, in: Encyclopedia of Machine Learning and Data Mining, Springer, 2017, pp. 85–87.
[19] L. Wang, Y. Xie, M. Pang, J. Wei, Alleviating the attribute conditional independence and iid assumptions of averaged one-dependence estimator by double weighting, Knowl. Based Syst. 250 (2022) 109078.
[20] M. Sahami, Learning limited dependence bayesian classifiers, KDD vol. 96 (1996) 335–338.
[21] L. Wang, X. Zhang, K. Li, S. Zhang, Semi-supervised learning for k-dependence Bayesian classifiers, Appl. Intell. 52 (4) (2022) 3604–3622.
[22] F. Ghofrani, A. Keshavarz-Haddad, A. Jamshidi, A new probabilistic classifier based on decomposable models with application to internet traffic, Pattern Recognit. 77 (2018) 1–11.
[23] M. Loéve, Probability theory, in: Foundations, Random Processes, D. van Nostrand, 1955.
[24] A.P. Dawid, Conditional independence in statistical theory, J. R. Stat. Soc. Ser. B (Methodological) 41 (1) (1979) 1–15.
[25] E. Kovács, T. Szántai, On the approximation of a discrete multivariate probability distribution using the new concept of t-cherry junction tree, in: Coping with Uncertainty, Springer, 2010, pp. 39–56.
[26] D. Pfeifer, E.A. Kovács, Vine copula structure representations using graphs and matrices, Inf. Sci. (2024) 120151.
[27] T.M. Cover, J.A. Thomas, Elements of Information Theory, John Wiley & Sons, 2012.
[28] S. Kullback, R.A. Leibler, On information and sufficiency, Anna. Math. Stat. 22 (1) (1951) 79–86.
[29] T. Szántai, E. Kovács, Hypergraphs as a mean of discovering the dependence structure of a discrete multivariate probability distribution, Ann. Oper. Res. 193 (1) (2012) 71–90.
[30] C. Chow, C. Liu, Approximating discrete probability distributions with dependence trees, IEEE Trans. Inf. Theory 14 (3) (1968) 462–467.
[31] J. Edmonds, et al., Optimum branchings, J. Res. Natl. Bureau Stand. B 71 (4) (1967) 233–240.
[32] H. Zhang, L. Jiang, G.I. Webb, Rigorous non-disjoint discretization for Naive Bayes, Pattern Recognit. 140 (2023) 109554.
[33] S. Wang, J. Ren, R. Bai, Y. Yao, X. Jiang, A max-relevance-min-divergence criterion for data discretization with applications on Naive Bayes, Pattern Recognit. 149 (2024) 110236.
[34] The quantile discretization function in python 3.11, https://github.com/DaniPfeifer/QuantileDiscretize.
[35] https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic.
[36] https://archive.ics.uci.edu/dataset/45/heart+disease.
[37] https://archive.ics.uci.edu/dataset/529/early+stage+diabetes+risk+prediction+dataset.
[38] https://archive.ics.uci.edu/dataset/102/thyroid+disease.
[39] https://www.kaggle.com/datasets/thecansin/parkinsons-data-set.
[40] The implementation of the log base-2 function in C, https://github.com/sifive/picolibc/blob/master/newlib/libm/common/log2.c, line 92, an 8-element Taylor series expansion.