International Journal of Applied Data Science
Volume 2, Issue 1, 2026

A COPULA-BASED SUPERVISED LEARNING CLASSIFICATION FOR CONTINUOUS AND DISCRETE DATA

M. Hammon[1], and P Myeris[2]
Human-Computer Interaction Lab, UMIACS, Univ. of Maryland, College Park, MD 20742

Article Info

Received: 22-09-2025 **Revised:23-10-2025** **Accepted:27-01-2026** **Published:19-03-2026**

Abstract: The naive Bayes classifier offers a straightforward method for allocating an observation to a class given the characteristics seen, yet it competes effectively with more advanced classifiers under the zero-one loss function despite the irrational feature independence requirement. However, it has been shown that when the features are correlated, the naive Bayes performs badly in both estimation and classification. In order to overcome this fundamental but seldom met assumption for the naive Bayes, researchers have created a variety of strategies. In this research, we present a novel classifier that does not rely on the independence assumption by assessing feature dependency using pair copulas built using a graphical model known as the D-Vine tree. Even for data with high dimensions and huge sample sizes, this tree form facilitates the easy and fast evaluation of feature dependency by breaking down the multivariate dependence into several bivariate dependencies. For features with discrete-valued entries, we expand the suggested approach. The suggested approach works effectively in both continuous and discrete situations, according to experimental research.

Keywords: Classification, Copulas, D-Vine Tree, Multivariate Dependence, Naive Bayes, Supervised Learning outlier.

Introduction

For the purpose of categorizing new cases given a vector of features, the naive Bayes method is one of the most extensively used learning algorithms in machine learning and data mining. Even though the model is based on a crucial and seldom met assumption—that is, all characteristics are independent given a class—its simplicity and competitive performance are impressive. Furthermore, it is appropriate for fitting high-dimensional data because to its computational efficiency, which is attained by the training process's linearity in features. While certain studies (Domingos and Pazzani, 1997; Rish, 2001; Zhang, 2004) have shown why naive Bayes performs very well in some situations when even the characteristic

Even if the independence assumption is broken, the naïve Bayes algorithm still performs badly in estimate and in other situations. Friedman et al. (1997) created a tree-like augmented naive Bayes (TAN) to expand naive Bayes to account for feature reliance. In TAN, a feature may have just one parent from another feature in addition to the class node, and the class node directly refers to all feature nodes. A more broad augmented naive Bayes (ANB), a specific example of Bayesian networks in which no node is designated as a class node, is generated by easing the restriction on linkages among features. Then, Zhang and Ling (2001) demonstrated that an ANB may represent any Bayesian network and, thus, any joint probability distribution. In this work, we extend the naive Bayes classifier with a novel tree-based classifier that uses pair copulas built using a D-Vine tree structure to assess feature dependency.

A copula is a multivariate probability distribution where the dependency structure between random variables

may be "separate" from each variable's marginal probability distribution. However, the selection of suitable copula families for high dimensional multivariate dependency is fairly constrained in real applications. The multivariate Gaussian and other standard multivariate copulas are unable to effectively describe the interdependence between several variables. In order to simulate multivariate dependency using a cascade of bivariate copulas that only operate on two variables at a time, a method known as pair-copula building (Aas et al., 2006) was devised. However, as the number of variables increases, the number of alternative structures rises very quickly. Vine copula, first presented by Joe (1996) and expanded by Bedford and Cooke (2001, 2002) and Kurowicka and Cooke (2005, 2006), is designed as a versatile graphical model for building pair copulas in a hierarchical manner in order to efficiently manage the construction process. Our suggested

approach uses D-Vine copulas to construct a dependent classification model. While pair copulas are used to assess the dependency of features (variables), the margins in the suggested model might be thought of as functioning just as in the naive Bayes. Additionally, since just two variables are taken into account at once, its bivariate structure enables fitting for a big p and small n dataset. By using a parallel computing approach, the computational efficiency for massive data may be attained at each D-Vine tree level. Theoretically, our suggested approach may effectively fit data with high dimensions and a big sample size. We use an enhanced approach with latent variables added for assessing the feature dependency in order to expand the suggested technique for features with discrete-valued entries. For discrete instances, we thereby adapt the standard D-Vine method.

This is how the paper is structured. The D-Vine tree structure, which aids in organizing the decomposition on a multivariate joint density, is introduced in Section 2 along with the ideas of copula and pair-copula creation. Our D-Vine copula-based dependency classifier is developed in Section 3, and Section 4 provides a detailed illustration of the learning process. We expand the suggested approach for discrete features in Section 5, and experimental investigations are carried out for continuous and discrete situations, respectively, in Sections 6 and 7. Section 8 discusses the findings.

1. Background

1.1. Copulas

Sklar's theorem (1959) states: Given p random variables X = (X_1 , … , X_p), for any p- dimensional multivariate joint distribution F with continuous marginal distribution functions F_1 , … , F_p, there exists a unique copula C such that for all x = (x_1 , … , x_p) ∈ R^p, where x is a realization of X,

$$F(x_1, \ldots, x_p) = C(F_1(x_1), \ldots, F_p(x_p)). \quad (1)$$

In words, it states that any p-dimensional multivariate distribution function F can be written in terms of univariate margins and a copula C which describes the dependence structure for variables. Another common used expression for Eq. (1) is.

$$C(u_1, \ldots, u_p) = F(F_1^{-1}(u_1), \ldots, F_p^{-1}(u_p)), \quad (2)$$

where F_K^{-1} is the inverse distribution function associated with F_k and $u_k = F_k(x_k)$ ∈ (0,1), for k= 1,2,…,p. The joint density associated with F thus can be written as

$$f(x_1, x_2, \ldots, x_p) = \prod_{k=1}^{p} f_k(x_k) \cdot c(F_1(x_1), F_2(x_2), \ldots, F_p(x_p)), \quad (3)$$

where f_k is the marginal density corresponding to F_k and c is the copula density associated with C.

Addition, a p-dimensional joint density can be factored by claims.

$$f(x_1, x_2, \ldots, x_p) = f_p(x_p) \cdot f(x_{p-1}|x_p) \cdot f(x_{p-2}|x_{p-1}, x_p) \cdot \cdots \cdot f(x_1|x_2, \ldots, x_p). \quad (4)$$

Following the notation from Aas et al. (2006), each right-hand side term in Eq. (4) can be further decomposed into the associated pair copula times a conditional density, written as.

$$f(x|\boldsymbol{\nu}) = c_{x,\nu_k|\boldsymbol{\nu}_{-k}}(F(x|\boldsymbol{\nu}_{-k}), F(\nu_k|\boldsymbol{\nu}_{-k})) \cdot f(x|\boldsymbol{\nu}_{-k}), \quad (5)$$

here $\boldsymbol{\nu}$ is a vector, ν_k is an arbitrarily chosen component of $\boldsymbol{\nu}$, and ν_{-k} denotes the $\boldsymbol{\nu}$-vector excluding ν_k . Iteratively decomposing $f(x|\nu_{-k})$ till $\nu_{-k} = \emptyset$, we finally obtain a general decomposition for a p-dimensional density which only involves margins and pair copulas. As shown in Joe(1996), the involved conditional distributions in the form of $F(x|\boldsymbol{\nu})$ can be obtained

$$F(x|\boldsymbol{\nu}) = \frac{\partial C_{x,\nu_k|\boldsymbol{\nu}_{-k}}(F(x|\boldsymbol{\nu}_{-k}), F(\nu_k|\boldsymbol{\nu}_{-k}))}{\partial F(\nu_k|\boldsymbol{\nu}_{-k})}, \quad (6)$$

And when $\nu_{-k} = \emptyset$, Eq.(6) reduces to

$$F(x|\nu) = \frac{\partial C_{x,\nu}(F_x(x), F_\nu(\nu))}{\partial F_\nu(\nu)}. \quad (7)$$

When x and ν are uniformly distributed, Eq.(7) can be further simplified as

$$h(x, \nu) \overset{def}{=} F(x|\nu) = \frac{\partial C_{x,\nu}(x, \nu)}{\partial \nu}, \quad (8)$$

Where $h(\cdot)$, called h-function, is the function used for generating pseudo observations which we will use later for fitting a model with D-Vine tree structure. For the details of bivariate copulas, such as the Gaussian and Student-t, see the reference (Aas et al.,2006).

2.2.D-Vine Tree Structure

Decomposing a p-dimensional joint density via Eq. (4) is not unique since it depends on how to factorize the component ν_k in $\boldsymbol{\nu}$ given in Eq. (5), and the number of possible decompositions increases extremely fast as the dimension p grows. To help manipulate decomposition for data with high dimension, a technique called *regular vine* was introduced in Bedford and Cooke (2001; 2002). Two special cases, *canonical vine* and D-Vine (Kurowicka and Cooke, 2004; Kurowicka and Joe, 2011), received even more attention due to their simplicity in organizing decomposition. In this paper, we focus on the D-Vine structure. However, our method straightforwardly extends to other types of vines. Figure 1 shows a decomposition example based on a 5-dimensional D-Vine. The total number of the nested trees in a p-dimensional D-Vine is p − 1, and each tree T_j, for j = 1, 2, . . . , p − 1, has p + 1 − j nodes and p − j edges. Each edge corresponds to a pair copula, and an edge in the tree T_j becomes to a node in T_j+1. The whole decomposition is obtained by the product of p margins and p(p−1)/2 copulas involving only two random variables in each copula. We should notice that the total number of ways to decompose a p-dimensional joint density via a p-dimensional D-Vine is p!/2. However, once the order of the nodes in T1 is determined, the decomposition is unique. Following the notation in Aas et al. (2006), a p-variate density $f(x_1, \ldots, x_p)$ corresponding to a p-dimensional D-Vine is given by

$$f(x_1, \ldots, x_p) = \prod_{j=1}^{p-1} \prod_{i=1}^{p-j} c_{i,i+j|i+1,\ldots,i+j-1}(F(x_i|x_{i+1}, \ldots, x_{i+j-1}), F(x_{i+j}|x_{i+1}, \ldots, x_{i+j-1})) \times \prod_{k=1}^{p} f_k(x_k), \quad (9)$$

where index j identifies the nested trees and i is for the edges in the tree T_j. $F(\cdot|\cdot)$ can be obtained by Eq. (8) cascaded down from T_1 to T_{p-1}, and $c_{i,i+j|i+1,\ldots,i+j}$ is a bivariate copula density corresponding to the pair nodes connected via the edge i in T_j.

2. D-Vine Copula Bayes Classifier

A D-Vine copula Bayes classifier (DVCBC) is a function to assign an observation to a class. According to the Bayes rule, the probability of a feature vector $X = (X_1, \ldots, X_p)$ with a realization $x = (x_1, \ldots, x_p)$

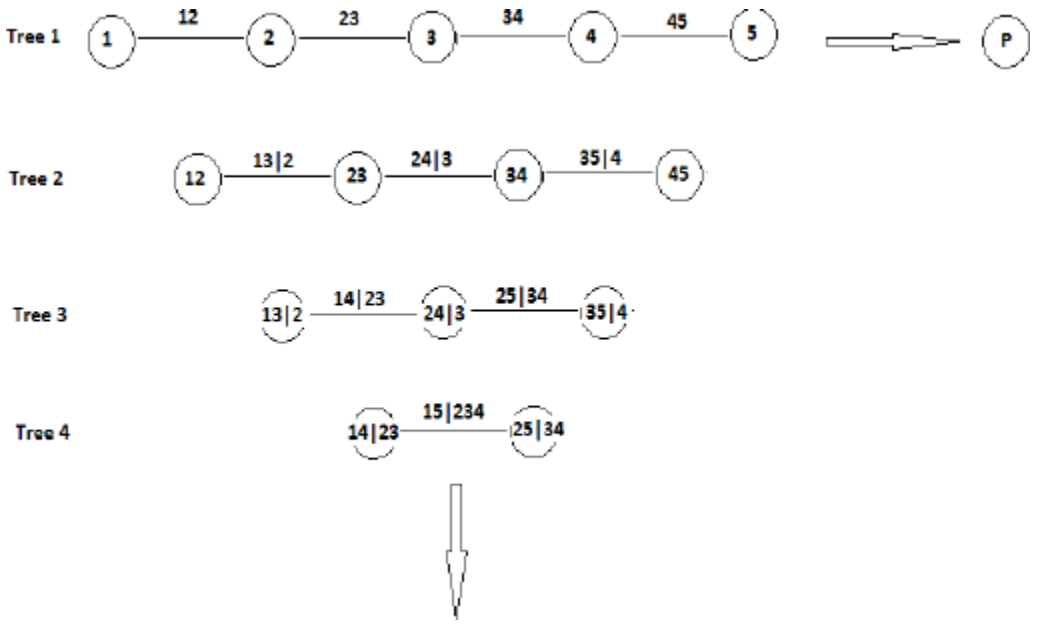


Figure 1: A 5-dimensional D-vine Example.

being in class $e \in \mathbf{E}$, where $\mathbf{E}$ is a class vector with $\mathbf{E} = (e_1, e_2, \ldots, e_l)'$ and l is the total number of classes, is written as

$$Pr(e|\mathrm{x}) \propto Pr(\mathrm{x}|e) \cdot Pr(e). \tag{10}$$

For continuous features, via Eq. (9), an augmented copula dependent Bayes probability model derived upon Eq. (10) is then given by

$$Pr(e|\mathrm{x}) \propto \underbrace{\prod_{k=1}^{p} f_k(x_k|e) \prod_{j=1}^{p-1} \prod_{i=1}^{p-j} c_{i,i+j|i+1,\ldots,i+j-1}(F(x_i|x_{i+1},\ldots,x_{i+j-1}), F(x_{i+j}|x_{i+1},\ldots,x_{i+j-1})|e)}_{f(\mathbf{x}|e)} \times Pr(e). \tag{11}$$

Combing the model (11) with the maximum a posterior (MAP) decision rule, the corresponding classifier to assign x to a class $e \in \mathbf{E}$ is then given as follows

$$\text{classify}(\mathrm{x}) = \{e : \underset{e \in \mathbf{E}}{\text{argmax}}\ f(\mathrm{x}|e) \cdot Pr(e)\}, \tag{12}$$

where $f(x|e)$ is given in (11). With the assumption that each continuous feature X_i is conditionally independent of every other feature X_j for $i \neq j$ and given a class e, Eq. (10) reduces to

$$Pr(e|\mathbf{x}) \propto \prod_{k=1}^{p} f_k(x_k|e) \cdot Pr(e), \tag{13}$$

which is the naive Bayes model. Consequently, we can consider Eq. (13) as a special case of our proposed method in Eq. (11).

3. Learning Process

3.1. Learning Optimization Structure of D-Vine

A D-Vine consists of a sequence of trees so that optimization over a D-Vine can involve algorithms for graphs. We know that an order of the nodes in the tree T_1 uniquely determines a decomposition of a joint density $f(x_1, \ldots, x_p)$ defined in Eq. (9). Thus, optimizing a D-Vine structure can be simply thought as optimizing the tree T_1. Here, optimizing T_1 means to capture as much dependence as possible for the pairs of features in T_1. However, the number of the possible orders of the nodes in T_1 in a p-dimensional D-Vine structure is p!/2, which implies that optimization via enumeration becomes impossible when p is large. One possible way of optimizing T_1 is to maximize a sum of transforms of the dependence, such as absolute Kendall tau (|τ|), absolute Pearson correlation ($|\rho_p|$), and absolute Spearman correlation ($|\rho_s|$), from all pair nodes in T_1 using the greedy search algorithm. We note, for a D-Vine structure, the greedy search is equivalent to solve the traveling salesman problem (TSP), see (Brechmann, 2010) for details. However, TSP is a well known NP-hard problem. For computational efficiency, we use an insertion algorithm provided in an R package **TSP** (Hahsler and Hornik, 2015) with the weights obtained from |τ|, or $|\rho_p|$, or $|\rho_s|$. Although the optimization through this algorithm might be suboptimal, it runs much faster for a large p with the running time $O(p^2 \log(p))$. Throughout this paper, we optimize T_1 for a D-Vine structure via the TSP with insertion algorithm.

3.2. Learning Univariate Margin

Given a class e ∈ **E**, denote the corresponding data structure as De = $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_{n_e}})$, where $\mathbf{x_s} = (\mathbf{x_{s1}}, \mathbf{x_{s2}}, \ldots, \mathbf{x_{sp}})$ with s = 1, 2, . . . , n_e and n_e is the total number of data points in the class e. For continuous mar- gins, they can be estimated by using either parametric or nonparametric approaches. For parametric approaches, the parameter(s) ψ_k in an chosen parametric marginal density f_{N_k} for the feature X_k can be estimated by maximizing the log-likelihood function given by

$$ll_e^k = \sum_{s=1}^{n_e} \log(f_{\psi_k}(x_{sk}|e)), \quad \text{for} \quad k = 1, 2, \ldots, p. \tag{14}$$

For example, if f_{N_k} is chosen to be a Gaussian density, i.e., $f_{N_k} = \emptyset_{N_k}$, then $\psi_k = (\mu_k, \sigma^2)$.

Maximizing the likelihood given in Eq. (14), we could obtain the estimated parameters $\hat{\mu_k}$ and $\hat{\sigma}_k$. For nonparametric approaches, the marginal density f_k for X_k can be estimated by using a

kernel-based approach (Parzen, 1962). Briefly, $f_k(x \mid e) = \frac{1}{n_e h(e)} \sum_{s=1}^{n_e} K(\frac{x - x_s}{h(e)})$, where K is a kernel function and h is the bandwidth smooth parameter corresponding to the class e.

Recently, Chen, Hanson, and Zhang (2014) proposed a new parametric based density estimation approach, named transformed Bernstein polynomial prior (TBPP). Briefly, the density $f_k(x|e)$ can be estimated through $f_k(x \mid e) = \sum_{t=1}^{T} G_{T,t} b_{T,t} \{F_{N_k}(x|e)\} f_{N_k}(x|e)$, where T decides the order of Bernstein polynomial, $G_{T,t}$ is the Bernstein coefficient, $b_{T,t}(\cdot)$ is a beta density with two parameters t and T-t+1, and F_{N_k} is a parametric distribution with the corresponding density f_{N_k}. By centering F_k, associated with f_k, at a parametric distribution F_{N_k}, the estimated density through TBPP can behavior like its centering when data actually follows F_{N_k} and can be adjusted like a nonparametric approach when data departs from the centering F_{N_k}. For details, see the reference (Chen, Hanson, and Zhang; 2014). Throughout this paper, we use the Gaussian parametric, Kernel, and TBPP methods for continuous marginal density estimation, and we always report the best result in our experimental studies.

3.3. Learning Dependence Structure

Given a class e and the pair nodes, $A_e^{i,j}$, connected by the edge i in the tree T_j with j = 1,…,p-1 and i = 1,…,p-j ,denote the corresponding data for the pair $A^{i,j}$ as $D_e^{i,j} = \{\boldsymbol{u}_s^{i,j,e} : s = 1, \ldots, n_e\}$, where n_e is the total number of data points in the class e and

$$\mathbf{u}_s^{i,j,e} = \left(u_{(s,i|i+1,\ldots,i+j-1)}, u_{(s,i+j|i+1,\ldots,i+j-1)} \middle| e\right)$$

With $i|i$+1,…, i_j-1 and i+$j|i$+1,…, i_j-1 indicating the indexes of two nodes connected by the i-th edge in T_j, see Section2.2. For notation convenience, we define $u_{s,i}^{i,j,e} \stackrel{\text{def}}{=} (u_{s,i}^{j,e}, u_{s,i+j}^{j,e})$.We should note that at T_1, the $u_s^{i,j,e}$ is directly estimated via true observations by $u_{s,k}^{j,e} = F_k(x_{s,k}|e)$, for k= $\{i,i$+1$\}$, here $\widehat{F}_k$ is the estimated univariate marginal distribution for feature X_k, see Section 4.2. Starting from T_2 to T_{p-1}, $u_{s,k}^{i,j,e}$ (a pseudo observation) is obtained via the h-function defined in Eq. (8), which thus corresponds to the chosen bivariate copula $C_{\theta_e^{i,j}}$ for $A_e^{i,j}$. To estimate the parameter(s) $\theta_e^{i,j}$ in $C_{\theta_e^{i,j}}$, we need to use the lemma given the below.

Let we first define a loss function via *Kullback-Leibler divergence*,written as

$$\text{LOSS}(\boldsymbol{\theta}) = \text{KL}(f_0, f_{\boldsymbol{\theta}}) \stackrel{def}{=} \int \log \frac{f_0}{f_{\boldsymbol{\theta}}} dF_0, \qquad (15)$$

where f_0 is the unknown true joint density with the distribution F_0, written as(by Eq. (3))

$$f_0(\mathbf{x}) = \prod_{k=1}^{p} f_k^0(x_k) \cdot c_0(F_1^0(x_1), F_2^0(x_2), \ldots, F_p^0(x_p)), \tag{16}$$

where f_k^0 and F_k^0 are the unknown univariate true marginal density and distribution functions for the feature X_k, respectively, and c_0 is the unknown true copula density with the copula function C_0. Similarly,

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \prod_{k=1}^{p} f_k^0(x_k) \cdot c_{\boldsymbol{\theta}}(F_1^0(x_1), F_2^0(x_2) \ldots, F_p^0(x_p)), \tag{17}$$

where c_θ is a chosen parametric copula density with the parameter(s) $\boldsymbol{\theta}$. We then give the lemma as follows

Lemma. *Let p random variables $X = (X_1, \ldots, X_p)$ with a realization $x = (x_1, \ldots, x_p)$ and F_k^0 be the true marginal distribution for variable X_k with density f_k^0 for k = 1,...,p. Let $U = (U_1, \ldots, U_p)$ are p random variables with $U_k = F_k^0(X_k)$, where U_k is uniformly distributed with $U_k \in (0,1)$, and $u = (u_1, \ldots, u_p)$ is a realization of U. Let data structure be $D = \{u_s: s = 1,\ldots,n\}$, where $u_s = (u_{s1}, \ldots, u_{sp})$ is one realization of U. Denote the unknown true joint density as f_0 given in (16) with the unknown true copula density c_0, and f_θ is a joint density given in (17) with an arbitrarily chosen parametric copula density c_θ. By defining a loss function depend on θ, LOSS(θ),through Kullback-Leibler divergence given in (15), we then have: minimizing LOSS(θ) for $\boldsymbol{\theta}$ is equivalent to maximizing the copula log-likelihood(CLL) given by*

$$\boldsymbol{CLL}(\boldsymbol{\theta}; \mathcal{D}) = \sum_{s=1}^{n} log(c_{\boldsymbol{\theta}}(u_{s1}, u_{s2}, \ldots, u_{sp})). \tag{18}$$

Proof.

$$\begin{aligned}
\mathrm{LOSS}(\boldsymbol{\theta}) &\overset{def}{=} \int \log\frac{f_0}{f_{\boldsymbol{\theta}}} dF_0 \\
&= \int \log(f_0) dF_0 - \int \log(f_{\boldsymbol{\theta}}) dF_0, \\
&= \int \log(c_0(F_1^0(x_1), \ldots, F_p^0(x_p))) dF_0 + \sum_{k=1}^{p} \int \log(f_k^0(x_k)) dF_0 \quad \text{(By Eq. (16))} \\
&- \left(\int \log(c_{\boldsymbol{\theta}}(F_1^0(x_1), \ldots, F_p^0(x_p))) dF_0 + \sum_{k=1}^{p} \int \log(f_k^0(x_k)) dF_0 \right) \quad \text{(By Eq. (17))} \\
&= \int \log\frac{c_0(F_1^0(x_1), \ldots, F_p^0(x_p))}{c_{\boldsymbol{\theta}}(F_1^0(x_1), \ldots, F_p^0(x_p))} dF_0 \\
&= \int \log\frac{c_0(u_1, \ldots, u_p)}{c_{\boldsymbol{\theta}}(u_1, \ldots, u_p)} dC_0 \quad \text{(By Eq. (1) and Eq. (2))} \\
&= \mathrm{KL}(c_0, c_{\boldsymbol{\theta}}).
\end{aligned} \tag{19}$$

We note that minimizing the loss function LOSS(θ) = KL(f_0, f_θ) with respect to (w.r.t.) $\boldsymbol{\theta}$ is equivalent to minimizing the alternative loss function KL(c_0, c_θ) given in Eq. (19). With the fact that c_0 is a copula density nothing about $\boldsymbol{\theta}$, then minimizing Eq. (19) w.r.t. $\boldsymbol{\theta}$ is thus equivalent to maximizing the function given by

$$\begin{aligned}\boldsymbol{\theta} &= \text{argmax}_{\boldsymbol{\theta}} \int \log(c_{\boldsymbol{\theta}}(u_1 \ldots, u_p)) dC_0 \\ &= \text{argmax}_{\boldsymbol{\theta}} E\left[\log(c_{\boldsymbol{\theta}}(u_1, \ldots, u_p))\right] \\ &\approx \text{argmax}_{\boldsymbol{\theta}} \sum_{s=1}^{n} \log(c_{\boldsymbol{\theta}}(u_{s1}, u_{s2}, \ldots, u_{sp})) \\ &\overset{def}{=} \text{argmax}_{\boldsymbol{\theta}} \mathbf{CLL}(\boldsymbol{\theta}; \mathcal{D}). \end{aligned} \quad (20)$$

3.4. Model Selection

One of the advantages of using D-Vine for assessing dependence structure is that different bivariate copulas might be used to evaluate the dependences for different pairs in down cascaded trees starting from *T1*. It thus makes possible for the whole dependence structure constructed by different copulas, such as Gaussian, Student-t, and Clayton copulas. Typically, compared to Gaussian copulas, Student-t copulas can capture more dependence in- formation for tails and Clayton has more power in evaluating unsymmetric dependence. The best dependence for each pair does not rely on a priori specific dependent type, it is instead obtained by voting for the best one among all of the considered copulas using a BIC-based score criterion given by

$$\mathbf{score}(\mathcal{C}(\mathcal{A}_e^{i,j}) : \mathcal{D}_e^{i,j})) = \mathbf{CLL}(\hat{\boldsymbol{\theta}}_e^{i,j}; \mathcal{D}_e^{i,j}) - \frac{1}{2}|\boldsymbol{\Theta}(\mathcal{A}_e^{i,j})| \log(n_e), \quad (21)$$

Where $\mathcal{C}(A^{i,j})$ is a bivariate copula for $A^{i,j}$ $\theta^{i,j}$, obtained by maximizing $\mathbf{CLL}(\theta^{i,j}, D^{i,j})$, is the corresponding estimated parameter(s) for $\mathcal{C}(A^{i,j})$, and $|\Theta(A^{i,j})|$ is the number of free parameters(s) corresponding to $A^{i,j}$. The task of voting for the best dependence for the pair $A^{i,j}$ is then equivalent to finding a bivariate copula $\mathcal{C}(A^{i,j})$ which maximizes score $(\mathcal{C}(A^{i,j}): D^{i,j})$.

3.5. Computational Issues

For a big-data, the computational efficiency could be achieved at each T_j, for j = 1, . . . , p−1, by utilizing a parallel computing technique. Specifically, the tasks of evaluating dependences for the pairs in T_j can be distributed out to different computing nodes along with the data points (observations for T1 or pseudo-observations for T_2 , . . . , T_{p-1}) only involved for the pairs evaluated at each computing node. The results from all computing nodes are then collected for the next tree T_{j+1} when j < p − 1. Theoretically, our proposed method can be used in fitting for a

data with extremely high dimension and large sample size. We are currently coding for our high performance computing (HPC) algorithm by using R packages **snow** (Tierney et al., 2015) and **Rmpi** (Yu, 2015), and we will release our codes as an R package later on. Throughout this paper, we do not use the HPC technique for our experimental studies.

By considering k different bivariate copulas for each pair, the total system running time needed for evaluating D-Vine dependence structure for an $n \times p$ dataset is then given by

$$\mathcal{O}(p^2 \log(p)) + p\mathcal{O}(T_n^m) + k\frac{p(p-1)}{2}\mathcal{O}(T_n^{c_k}),$$

One where $G(p^2\log(p))$ is the time for solving the TSP problem for T_1 optimization; $pG(T_n^m)$ is the time used for p margins estimation, here we use m to denote that this time also depends on the marginal method we used; $G(T^{ck})$ is the time needed for a bivariate copula estimation, and $k\frac{p(p-1)}{2}$ is the total copulas to be evaluated. We should note, for each pair, the selection of the bivariate dependence has no extra cost since the score in Eq. (21) is obtained directly after evaluating Eq. (18) for copula estimation. For a $p > n$ problem, the running time for evaluating pair copulas can be expected to be quadratically related to the dimension p, i.e. $G(p^2)$.

4. Extension to Discrete Cases

Now, let us consider p-random variables X with discrete-valued entries. Let $a_k = F_k(x_k^-)$ be the left hand limit of the margin F_k at x_k and $b_k = F_k(x_k)$. Then the probability mass function of features X can be written as(Smith and Khaled, 2012)

$$f(x_1, \ldots, x_p) = Pr(X_1 = x_1, \ldots, X_p = x_p) = \mathcal{I}_{a_1}^{b_1} \ldots \mathcal{I}_{a_p}^{b_p} C(u_1, \ldots, u_p),$$

where

$$\begin{aligned}\mathcal{I}_{a_k}^{b_k} C(u_1, \ldots, u_{k-1}, u_k, u_{k+1}, \ldots, u_p) &= C(u_1, \ldots, u_{k-1}, b_k, u_{k+1}, \ldots, u_p) \\ &- C(u_1, \ldots, u_{k-1}, a_k, u_{k+1}, \ldots, u_p).\end{aligned}$$

For a bivariate case, we have

$$\begin{aligned}f(x_1, x_2) &= \mathcal{I}_{a_1}^{b_1}\mathcal{I}_{a_2}^{b_2} C(u_1, u_2) \\ &= C(b_1, b_2) - C(a_1, b_2) - C(b_1, a_2) + C(a_1, a_2).\end{aligned}$$

Following the Chen and Hanson's (2016+) approach, let consider an augmented joint distribution of $(X_{k_1}, X_{k_2}, U_{k_1}, U_{k_2})$ with a bivariate latent variable (U_{k_1}, U_{k_2}) and each U_{k_t} is uniformly distributed on [0, 1], for t = 1, 2, and $k_t \in \{1, \ldots, p\}$. Then the mixed probability density for $(X_{k_1}, X_{k_2}, U_{k_1}, U_{k_2})$ written as

$$f(x_{k_1}, x_{k_2}, u_{k_1}, u_{k_2} | \boldsymbol{\theta}, \boldsymbol{\theta}_{k_1}, \boldsymbol{\theta}_{k_2}) = c(u_{k_1}, u_{k_2} | \boldsymbol{\theta}) \prod_{t=1}^{2} \mathcal{I}(a_{k_t} \le u_{k_t} < b_{k_t}; \boldsymbol{\theta}_{k_t}), \quad (22)$$

where $c(u_{k_1}, u_{k_2} | \boldsymbol{\theta})$ is the copula density corresponding to a parametric copula function $C(u_{k_1}, u_{k_2} | \boldsymbol{\theta})$ with parameter(s) $\boldsymbol{\theta}$, $a_{k_t} = F_{k_t}(x_{k_t}^-; \theta_{k_t})$, and $b_{k_t} = F_{k_t}(x_{k_t}; \theta_{k_t})$, for t = 1, 2,

where θ_{k_t} is the parameter(s) of the margin F_{k_t} . The joint probability of (X_{k_1}, X_{k_2}) is thus given by

$$\begin{aligned} f(x_{k_1}, x_{k_2}|\boldsymbol{\theta}, \boldsymbol{\theta}_{k_1}, \boldsymbol{\theta}_{k_2}) &= \int_0^1 \int_0^1 f(x_{k_1}, x_{k_2}, u_{k_1}, u_{k_2}|\boldsymbol{\theta}, \boldsymbol{\theta}_{k_1}, \boldsymbol{\theta}_{k_2}) du_{k_1} du_{k_2} \\ &= \int_0^1 \int_0^1 c(u_{k_1}, u_{k_2}|\boldsymbol{\theta}) \prod_{t=1}^{2} \mathcal{I}(a_{k_t} \leq u_{k_t} < b_{k_t}; \boldsymbol{\theta}_{k_t}) du_{k_1} du_{k_2} \\ &= \int_{a_{k_2}}^{b_{k_2}} \int_{a_{k_1}}^{b_{k_1}} c(u_{k_1}, u_{k_2}|\boldsymbol{\theta}) du_{k_1} du_{k_2}. \end{aligned} \tag{23}$$

Particularly, if we consider a bivariate Gaussian copula density for $c(\cdot)$ in Eq. (23), we get

$$f(x_{k_1}, x_{k_2}|\rho, \boldsymbol{\theta}_{k_1}, \boldsymbol{\theta}_{k_2}) = \int_{\Phi^{-1}(a_{k_2})}^{\Phi^{-1}(b_{k_2})} \int_{\Phi^{-1}(a_{k_1})}^{\Phi^{-1}(b_{k_1})} \phi_\Sigma(z_{k_1}, z_{k_2}) dz_{k_1} dz_{k_2}, \tag{24}$$

here,

$$\phi_\Sigma(z_{k_1}, z_{k_2}) = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{z_{k_1}^2 - 2\rho z_{k_1} z_{k_2} + z_{k_2}^2}{2(1-\rho^2)}\right) \tag{25}$$

With $z_{k_t} = \Phi^{-1}(u_{k_t})$, for t = 1,2. Since we note that $\phi_{\Sigma(z_{k_1}, z_{k_2})}$ in Eq. (25) is a bivariate Gaussion density with the $\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and covariance matrix $\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ and Eq. (24) can be written as

$$\begin{aligned} f(x_{k_1}, x_{k_2}|\rho, \boldsymbol{\theta}_{k_1}, \boldsymbol{\theta}_{k_2}) &= \int_{-\infty}^{\Phi^{-1}(b_{k_2})} \int_{-\infty}^{\Phi^{-1}(b_{k_1})} \phi_\Sigma(z_{k_1}, z_{k_2}) dz_{k_1} dz_{k_2} \\ &- \int_{-\infty}^{\Phi^{-1}(b_{k_2})} \int_{-\infty}^{\Phi^{-1}(a_{k_1})} \phi_\Sigma(z_{k_1}, z_{k_2}) dz_{k_1} dz_{k_2} \\ &- \int_{-\infty}^{\Phi^{-1}(a_{k_2})} \int_{-\infty}^{\Phi^{-1}(b_{k_1})} \phi_\Sigma(z_{k_1}, z_{k_2}) dz_{k_1} dz_{k_2} \\ &+ \int_{-\infty}^{\Phi^{-1}(a_{k_2})} \int_{-\infty}^{\Phi^{-1}(a_{k_1})} \phi_\Sigma(z_{k_1}, z_{k_2}) dz_{k_1} dz_{k_2}, \end{aligned} \tag{26}$$

thus each term in the right hand side of Eq. (26) can be computed by the function *pmvnorm()* in an R package **mvtnorm** (Genz et al., 2012). Similary, when considering $c(\cdot)$ in Eq. (23) as a bivariate Student-t copula, then $\phi_{\Sigma(z_{k_1}, z_{k_2})}$ in Eq. (24) is then written as

$$\phi_\Sigma(z_{k_1}, z_{k_2}) = \frac{\Gamma(\frac{\nu+2}{2})}{\Gamma(\frac{\nu}{2})\sqrt{(\pi\nu)^2(1-\rho^2)}} \left(1 + \frac{z_{k_1}^2 - 2\rho z_{k_1} z_{k_2} + z_{k_2}^2}{\nu(1-\rho^2)}\right)^{-\frac{\nu+2}{2}} \tag{27}$$

Given n data points, the log-likelihood function is then written as

$$ll(\boldsymbol{\theta}) = \sum_{s=1}^{n} \log(f(x_{sk_1}, x_{sk_2}|\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}_{k_1}, \hat{\boldsymbol{\theta}}_{k_2})), \tag{28}$$

where $f(x_{sk_1}, x_{sk_2}|\boldsymbol{\theta}, \widehat{\boldsymbol{\theta}_{k_1}}\widehat{\boldsymbol{\theta}_{k_2}})$ is given in Eq. (23) with the estimated $\widehat{\theta_{k_1}}$ and $\widehat{\theta_{k_2}}$ for the margins F_{k_1} andF_{k_2} , respectively. The estimated $\widehat{\theta}$ for $c(\cdot)$ then can be obtained by maximizing Eq. (28). The whole dependence structure of a D-Vine tree with discrete-valued entries then can be evaluated via the following algorithm: Algorithm 0.0.1.

[1] For $k = 1, \ldots, p$: Estimate the parameter $\boldsymbol{\theta}_k$ for the margin $F_k(\cdot; \boldsymbol{\theta}_k)$

[2] For $k = 1, \ldots, p$: Compute $a_k = F_k(x_k^-, \hat{\boldsymbol{\theta}}_k)$ and $b_k = F_k(x_k, \hat{\boldsymbol{\theta}}_k)$

[3] Estimate the order of the features $\mathbf{X}$ in $\mathcal{T}_1$ using the method provided in Section 4.1

[4] In $\mathcal{T}_1$:

> *[i] For each pair $\mathcal{A}_{i,i+1}$, for $i = 1, \ldots, p-1$, maximize Eq. (28) to obtain $C_{i,i+1}(u_i, u_{i+1}|\hat{\boldsymbol{\theta}})$*
>
> *[ii] Generate $u_1 \sim Unif(a_1, b_1)$*
>
> *[iii] For $i = 1, \ldots, p-1$:*
>
> > *[a] Compute $A_{i+1} = h_{i,i+1}(a_{i+1}, u_i|\hat{\boldsymbol{\theta}})$ and $B_{i+1} = h_{i,i+1}(b_{i+1}, u_i|\hat{\boldsymbol{\theta}})$*
> >
> > *[b] Generate $w_{i+1} \sim Unif(A_{i+1}, B_{i+1})$ and compute $u_{i+1} = h_{i,i+1}^{-1}(w_{i+1}, u_i)$*
> >
> > *[c] Obtain pseudo observations for $\mathcal{T}_2$ by $u_{i|i+1} = h_{i,i+1}(u_i, u_{i+1}|\hat{\boldsymbol{\theta}})$ and $u_{i+1|i} = h_{i,i+1}(u_{i+1}, u_i|\hat{\boldsymbol{\theta}})$*

[5] In $\mathcal{T}_j$, for $j = 2, \ldots, p-1$:

> *[i] For each pair $\mathcal{A}_{i,i+j|i+1,\ldots,i+j-1}$, for $i = 1, \ldots, p-j$, maximize Eq. (18) to obtain $C_{i,i+j|i+1,\ldots,i+j-1}(u_{i|i+1,\ldots,i+j-1}, u_{i+j|i+1,\ldots,i+j-1}|\hat{\boldsymbol{\theta}})$*
>
> *[ii] Obtain pseudo observations for $\mathcal{T}_{k=j+1}$ by*
>
> > *[a] $u_{i|i+1,\ldots,i+k-1} = h_{i,i+j|i+1,\ldots,i+j-1}(u_{i|i+1,\ldots,i+j-1}, u_{i+j|i+1,\ldots,i+j-1}|\hat{\boldsymbol{\theta}})$*
> >
> > *[b] $u_{i+k|i+1,\ldots,i+k-1} = h_{i+1,i+k|i+2,\ldots,i+k-1}(u_{i+1|i+2,\ldots,i+k-1}, u_{i+k|i+2,\ldots,i+k-1}|\hat{\boldsymbol{\theta}})$*

5. Experimental Studies

To assess the merits of our model (DVCBC), we compare it to the existing popular supervised learning classification methods including the naive Bayes classifier (NBC), a network-based classifier (TAN), the support vector machine (SVM), and an ensemble classifier random forests (RF). The naive Bayes model assumes independence of the features and is fitted by using the function *naiveBayes()* provided in an R package **e1071** (Meyer et al., 2014). Also in the same R

on data, we found that it is better to consider a relative large parameter space for the SVM model. Specifically, we consider the cost parameter 9 ∈ $\{10^{-3}, 10^{-2}, 10^{-1}\}$ and kernel parameter γ ∈ $\{10^{2}, 10^{1}, 10^{0}, 10^{-1}, 10^{-2}\}$ for radial basis kernel. The TAN model is considered with a linear Gaussian conditional distribution given by $X_k|e \sim N(\beta_0 + \beta_1 x_r, \sigma^2)$, where the feature x_r is the parent of x_k in the network structure with the realization x_r. The RF model is fitted by using the Breiman approach (Breiman, 2001), and the according R package **randomForest** (Breiman et al., 2015) can be installed from the official R website.

Given a class e, we use the transformed Bernstein polynomial centering at a univariate Gaussian to estimate all margins, given by $f_k(x|e) = \sum_{t=1}^{T} w_{T,t}\, b_{T,t}\{(\Phi_{\mu_k,\sigma_k})(x|e)\}\phi_{(\mu_k,\sigma_k)}(x|e)$ associated with the distribution $F_k(x|e) = \sum_{t=1}^{T} w_{T,t}\, B_{T,t}\{(\Phi_{\mu_k,\sigma_k})(x|e)\}$, where $B_{T,t}$ is the beta cumulative distribution function with the beta density $b_{T,t}$, for k = 1, . . . , p. We found that this marginal estimation method works well either for Gaussian or non-Gaussian margins. The D-Vine structure is optimized by the TSP algorithm introduced in Section 4.1. We consider three different types of bivariate copulas, i.e, Gaussian, Student-t, and Clayton copulas (Aas et al., 2006). The best dependence among those three is picked up with the highest score obtained by Eq. (21). We evaluate all methods for five datasets from the UCI ma- chine learning repository, see Table 1 for the details of the datasets.

Table 1: 5 datasets from UCI machine learning repository.

Name	Feature-Type	n	p	Classes	Classification Task
Glass_Ident	Continuous	214	10	6	glass identification
Iris	Continuous	150	4	3	specie identification
Wine	Continuous	178	13	3	wine identification
Parkinsons	Continuous	197	23	2	Parkinson identification
Hill_Valley	Continuous	606	101	2	hill or valley identification

In Table 2, we summarize the average classification accuracy upon 2-fold cross validation repeated 50 times for all methods. Also, in Figure 2 and Figure 3, we plot the ROC curves for binary classifications – Parkinsons and Hill Valley. It is not surprised that our model is always superior to the NBC since our method allows the feature dependence and thus makes it possible to get better predictions by adding more dependence information in the structure through copulas. The same phenomenon can be observed for the TAN compared to the NBC since the TAN model also relaxes the independence assumption for features by adding directional edges between them for the dependencies. We also note that our method performs better than the TAN model for all five experimental studies. It might imply that, compared to the TAN, our method can more effectively integrate dependence information into the structure, even if the both are the tree-based models and free of the feature independence assumption. The SVM model has a better classification for "Glass Ident" dataset, but our method performs better than the SVM for the other 4 datasets. The RF model performs the best.

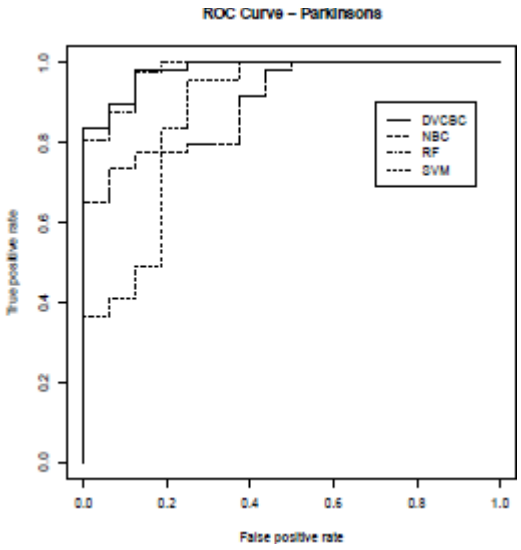


Figure 2: ROC: Parkinsos.

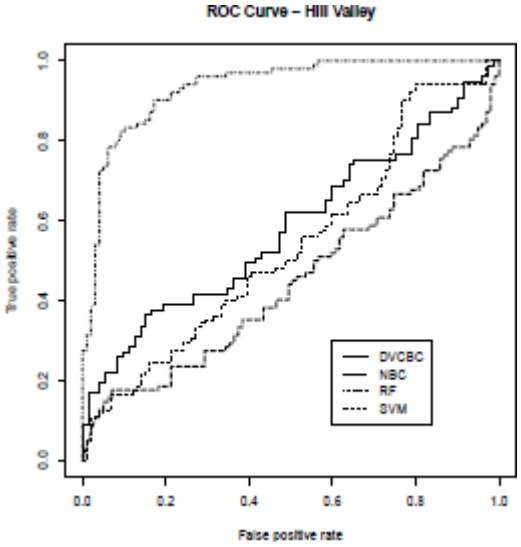


Figure 2: ROC: Parkinsos.

among all considered methods in terms of the average classification accuracy for “Glass Ident” and “Hill Valley” datasets. It is only slightly worse than the DVCBC for “Iris” and “Wine” datasets. Some literatures, e.g., Caruana and Niculescu-Mizil (2006), have commented on the reasons of the outperformance of the random forests for supervised learning classification compared to other classifiers prior to calibration. For example, the outputs of an SVM are normalized distances to the decision boundary and thus it is not originally desinged to predict probabilities, and the naive Bayes models are well known to predict poorly because of the unrealistic independence assumption. Although our proposed method relaxes the independence

assumption via copula techniques, it still has several restrictions such as the types of copulas specified for dependence structure. Misspecification on it will eventaully result in inaccurate predictions and thus wrong classifications. Following the discussion in Caruana and Niculescu- Mizil (2006), although on average some classifiers including the NBC are not competitive with the best methods such as random forests, those conclusions do not always hold since they probably perform much better for some metrics on some particular problems. It also turns out that our proposed model has its own advantages when used on some datasets considered here, see Table 2 and Figure 2 for "Iris", "Wine", and "Parkinsons".

Table 2: 2-fold classification accuracy for the 5 datasets.

Classifier	Measure	Glass_Ident	Iris	Wine	Parkinsons	Hill_Valley
DVCBC	Min	0.46	0.96	0.97	0.88	0.59
	Mean	0.58	**0.98**	**0.99**	**0.93**	0.61
	Max	0.68	1.00	1.00	1.00	0.63
NBC	Min	0.19	0.92	0.96	0.60	0.49
	Mean	0.42	0.96	0.98	0.70	0.53
	Max	0.54	1.00	1.00	0.78	0.56
SVM	Min	0.67	0.94	0.97	0.83	0.49
	Mean	0.76	0.96	0.98	0.89	0.52
	Max	0.84	1.00	1.00	0.95	0.55
TAN	Min	0.45	0.90	0.96	0.66	0.57
	Mean	0.55	0.97	0.98	0.73	0.60
	Max	0.67	1.00	1.00	0.83	0.65
RF	Min	0.75	0.93	0.96	0.89	0.79
	Mean	**0.85**	0.97	0.98	**0.93**	**0.84**
	Max	0.92	1.00	1.00	0.98	0.88

6. Studies for Discrete Cases

We consider two simulation scenarios for all features with binary entries, see the following simulation setup. This is quite common for data from survey with answers either "Yes" or "No". We should note that with binary entry, if $x = 0$, then $a = F(0^{-}) = 0$ and $b = F(0) = 1-\pi$; and if $x = 1$, then a = F $(1^{-}) = 1 - \pi$ and b = F (1) = 1, where π is the probability of success ($x = 1$) on each trial. In Scenario I, all three features are independent to each other, however, in Scenario II, the feature *X2* depends on the feature *X3*. The number of the data points in each class e_k, for $k = 1, 2, 3$, is n = 100, so the total number is $n_T = 3 * 100 = 300$.

[Scenario I]:

$$[Y=e_1]:\ X_1 \sim \text{Bernoulli}(\pi = 0.1),\ X_2 \sim \text{Bernoulli}(\pi = 0.6),\ X_3 \sim \text{Bernoulli}(\pi = 0.7)$$
$$[Y=e_2]:\ X_1 \sim \text{Bernoulli}(\pi = 0.1),\ X_2 \sim \text{Bernoulli}(\pi = 0.1),\ X_3 \sim \text{Bernoulli}(\pi = 0.1)$$
$$[Y=e_3]:\ X_1 \sim \text{Bernoulli}(\pi = 0.1),\ X_2 \sim \text{Bernoulli}(\pi = 0.9),\ X_3 \sim \text{Bernoulli}(\pi = 0.1)$$

[Scenario II]:

$$[Y=e_1]:\ X_1 \sim \text{Bernoulli}(\pi = 0.1),\ X_3 \sim \text{Bernoulli}(\pi = 0.7),$$
$$X_2|X_3 = 1 \sim \text{Bernoulli}(\pi = 0.6)$$
$$X_2|X_3 = 0 \sim \text{Bernoulli}(\pi = 0.1)$$
$$[Y=e_2]:X_1 \sim \text{Bernoulli}(\pi = 0.1),\ X_3 \sim \text{Bernoulli}(\pi = 0.1),$$
$$X_2|X_3 = 1 \sim \text{Bernoulli}(\pi = 0.6)$$
$$X_2|X_3 = 0 \sim \text{Bernoulli}(\pi = 0.1)$$
$$[Y=e_3]:\ X_1 \sim \text{Bernoulli}(\pi = 0.1),\ X_3 \sim \text{Bernoulli}(\pi = 0.1),$$
$$X_2|X_3 = 1 \sim \text{Bernoulli}(\pi = 0.9)$$
$$X_2|X_3 = 0 \sim \text{Bernoulli}(\pi = 0.1)$$

We estimate Bernoulli margins for all three features. Similar to a continous case, the D-Vine structure is optimized by the TSP algorithm, and three types of copulas are considered with the best one having the highest score from Eq. (21). We report the classification accuracy from the DVCBC, the NBC, the SVM, and the TAN for Scenario I and II in Table 3. Apparently, our method is superior to others in the two simulated scenarios. The SVM model is always better than other tree based classifiers, the NBC and TAN, and the NBC model performs worst in all two scenarios. To compare their computational efficiencies, we also record the CPU time for all four methods. With Intel Core i5-4570R CPU @2.70GHz, the system running time in one run for the DVCBC, the NBC, the SVM, and the TAN are 0.52s, 0.01s, 0.03s, and 0.03s, respectively.

Table 3: Features with Binary Entries..

Scenario I				
	DVCBC	NBC	SVM	TAN
Min	0.82	0.33	0.76	0.33
Mean	**0.89**	0.33	0.79	0.38
Max	0.96	0.33	0.82	0.42
Scenario II				
	DVCBC	NBC	SVM	TAN
Min	0.67	0.33	0.50	0.33
Mean	**0.79**	0.34	0.54	0.35
Max	0.97	0.39	0.57	0.37

We further use the four methods to fit for *hartigan* data from an R package **homals** (Mair and De Leeuw, 2015). The main task is to classify a number of bolts, nails, screws, and tacks according to three criteria, i.e., Thread (Yes or No), Bottom (Sharp and Flat), and Brass (Yes or No). The total number of data points is n_T = 24. Specifically, 9 from nail, 6 from screw, 6 from

bolt, and 3 from tack. We use 2-fold cross validation repeated 50 times for computing the classification accuracy and report the results in Table 4. Compared to the NBC, the SVM, and the TAN, our method is (0.912 − 0.870)/0.870 × 100% ≈ 4.8%, (0.912 − 0.894)/0.894 × 100% ≈2.0%, and (0.912 − 0.879)/0.879 × 100% ≈ 3.8% better accuracy on average, respectively.

Table 4: Hartigan Data

Hartigan				
	DVCBC	NBC	SVM	TAN
Min	0.824	0.824	0.824	0.818
Mean	**0.912**	0.870	0.894	0.879
Max	1.000	0.941	0.941	0.941

7. Conclusion

We provide a novel supervised learning approach in this study that does not rely on the feature independence assumption. Pair copulas created using a graphical model known as D-Vine are used to assess the dependencies between characteristics. Due to the fact that just two features (variables) are taken into account at a time, fitting for data with high dimensions and a big sample size is made simple and effective. We see that the suggested approach is basically a particular instance of the widely used naive Bayes classifier. Using an enhanced approach, we further expand the model to assess the dependencies between discrete-valued characteristics. It should be noted that during the whole research, we only looked at three different copula types for multivariate dependency: Gaussian, Student-t, and Clayton copulas. However, in reality, we might utilize additional copula types, including Gumbel and Frank copulas, to measure the reliance of features.

References

[1]. Aas, K., Czado, C., Frigessi, A., and Bakken, H. (2006). Pair-copula constructions of multiple dependence. Insurance: Mathematics and Economic, 44, 182–198.

[2]. Bedford, T. and Cooke, R. (2001). Probability density decomposition for conditionally dependent random variables modeled by vines. Annals of Mathematics and Artificial Intelligence, 32, 245–268.

[3]. Bedford, T. and Cooke, R. (2002). Vines - a new graphical model for dependent random variables. Annals of Statistics, 30, 1031–1068.

[4]. Brechmann, E. (2010). Truncated and simplified regular vines and their applications. Diploma thesis. Technische Universitaet Muenchen.

[5]. Breiman, L. (2001). Random forests. Machine Learning, 45, 5–32. Breiman, L., Cutler, A., Liaw, A., and Wiener, M. (2015). Breiman and Cutler's random forests for classification and regression. R package randomForest.

[6]. Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. ICML '06 Proceedings of the 23rd International Conference on Machine Learning, 161–168.

[7]. Chen, Y. and Hanson, T. (2016+). Copula regression models for discrete and mixed bivariate responses. Submitted to The American Statistician.

[8]. Chen, Y., Hanson, T., and Zhang, J. (2014). Accelerated hazards model based on parametric families generalized with Bernstein polynomials. Biometrics, 70, 192–201.
[9]. Domingos, P. and Pazzani, M. (1997). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. Machine Learning, 29, 103–130.
[10]. Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. Machine Learning, 29, 131–163.
[11]. Genz, A., Bretz, F., Miwa, T., Mi, X., Leisch, F., Scheipl, F., Bornkamp, B., Maechler, M., and Hothorn, T. (2012). Multivariate normal and t distributions. R package mvtnorm.
[12]. Hahsler, M. and Hornik, K. (2015). Traveling salesperson problem (TSP).R package TSP. [13].
Joe, H. (1996). Families of m-variate distributions with given margins and m(m-1)/2 bivariate dependence parameters. Lecture Notes - Mono- graph Series, 28, 120–141.
[14]. Kurowicka, D. and Cooke, R. (2004). Distribution-free continuous Bayesian belief nets. In Fourth International Conference on Mathematical Meth- ods in Reliability Methodology and Practice, Santa Fe, New Mexico.
[15]. Kurowicka, D. and Cooke, R. (2005). Sampling algorithms for generat- ing joint uniform distributions using the vine-copula method. In 3rd IASC world conference on Computational Statistics & Data Analysis, Limassol, Cyprus.
[16]. Kurowicka, D. and Cooke, R. (2006). Uncertainty Analysis with High Dimensional Dependence Modeling. John Wiley & Sons, Chichester.
[17]. Kurowicka, D. and Joe, H. (2011). Dependence Modeling: Vine Copula Handbook. World Scientific Publishing Co., Singapore.
[18]. Mair, P. and De Leeuw J. (2015). Gifi Methods for Optimal Scaling. R package homals.
[19]. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C., and Lin, C. (2014). Misc functions of the department of Statistics, probability theory group (formerly: E1071), TU Wien. R package e1071.
[20]. Parzen, E. (1962). On estimation of a probability density function and model. Annals of Mathematical Statistics, 33, 1065–1076.
[21]. Rish, I. (2001). An empirical study of the naive Bayes classifier. In IJCAI- 01 Workshop on Empirical Methods in AI, pages 41–46, Sicily, Italy.
[22]. Sklar, A. (1959). Fonctions de rpartition n dimensions et leurs marges. [23].
Publ. Inst. Stat. Univ. Paris, 8, 229–231.
[24]. Smith, M. and Khaled, M. (2012). Estimation of copula models with discrete margins via Bayesian data augmentation. Journal of the American Statistical Association, 107, 290–303.
[25]. Tierney, L., Rossini, A., Li, N., and Sevcikova, H. (2015). Simple network of workstations. R package snow.
[26]. Yu, H. (2015). Interface (Wrapper) to MPI (message-passing interface). R package Rmpi.