UCLouvain

FACULTE DES SCIENCES

Faculté des sciences

Using truncated vine copulas in Supervised Probabilistic Classification

Auteur·es : Kim Vogelaere

Promoteur·rices : Johan Segers

Lecteur·rices : Anouar El Ghouch, Christian Hafner

Année académique 2019-2020

I wish to express my deepest gratitude to my advisor, Professor Johan Segers, for the valuable support and guidance he provided me with during this last year. His help, insight and encouragements were never missing. It's not too much to say that without his patience and understanding, I would never have been able to bring this project to completion. I would also like to extend my appreciation to Professors Anouar El Ghouch and Christian Hafner for their participation to my thesis defense committee.

Contents

Introduction **5**

1 Bayesian classifiers **8**

1.1 Bayesian classifiers 8

1.2 The naive Bayes classifier 9

2 Copulas **11**

2.1 Background 11

2.1.1 Definition 12

2.1.2 Sklar's theorem 12

2.1.3 Parametric copula families 13

2.1.3.1 Meta elliptical copulas 13

2.1.3.2 Archimedean copulas 14

2.1.3.3 Extreme-value copulas 16

2.1.3.4 Rotated copulas 17

2.2 Bivariate copula estimation 17

2.2.1 Computing Pseudo-observations 17

2.2.1.1 Parametric estimation 18

2.2.1.2 Non-parametric estimation 18

2.2.2 Bivariate copula density estimation 20

2.2.2.1 Parametric estimation 20

2.2.2.2 Parametric model selection 21

2.2.2.3 Non-parametric estimation 22

2.3 Multivariate copula estimation 23

2.3.1 Standard multivariate copula models 24

2.3.2 Vine copula models 24

2.3.2.1 A pair-copula decomposition of a general multivariate distribution 24

2.3.2.2 Regular vines 27

2.3.2.3 Regular vine copulas 28
2.3.2.4 Estimation procedure 29
2.3.2.5 Structure selection 31
2.3.2.6 Vine copula models for high dimensional data 33

3 A Copula-based Bayes classifier for high-dimensional classification 41
3.1 Definition 41
3.2 Learning 42

4 Experimental results 44
4.1 Experimental setup 44
4.2 Performance 45
4.3 Running time 50

Conclusion 52

Bibliography 53

Introduction

Classification is the problem of assigning categories to new observations. In the context of supervised classification, the possible categories are known a priori and the assignations are made on the basis of a training set of data which contains observations and their respective categories. The concepts discussed in this thesis belong to both machine learning and statistics. In this regard, we believe it is important to point out that the terminology varies across fields. In statistics, the properties of observations are usually referred to as explanatory or independent variables (sometimes also regressors) and the categories, which are the possible values of the dependent variable, are termed outcomes. In machine learning, the observations are often referred to as instances, the properties of observations are termed features and the categories are known as classes. The same is true for the notion of classifier. In machine learning, a classifier is an algorithm that implements classification (i.e., a classification algorithm) while in statistics, a classifier is a mathematical function mapping an observation to a category implemented by a classification algorithm (i.e., a hypothesis). Note that those terms might be used interchangeably in the remainder of this paper.

Learning classifiers able to make accurate predictions in various domains is one of the main purposes of machine learning. However, as stated by the no-free-lunch theorem (Wolpert, 1996) and shown by practice, there is no single classifier able to outperform all other classifiers on all given problems. That is because the performance of a classifier highly depends on the characteristics of the problem/data. This had led to the development of a large set of classification algorithms to choose from. Even if choosing an appropriate classifier for a given problem is not an easy task (i.e., there is no standard procedure), the first step still consists in defining the problem at hand. Therefore, we start by defining the type of problem we wish to work on by going over some of its main characteristics.

Two subtypes of classification can be distinguished depending on the number of categories: binary (2) vs multiclass (>2) classification. Multiclass classification problems have two main disadvantages compared to binary classification problems. First, they often involves

using a combination of several binary classifiers as many classification algorithms have been developed only for binary classification. Also, the metrics used for model evaluation are much more limited for multiclass classifiers. Therefore, we choose to address the problem of learning effective binary classifiers. Considering the instances, the features can be categorical, ordinal, integer-valued or real-valued. The type of the features is an important characteristic of the data that has to be taken into consideration. That is because some classification algorithms can only handle certain types of features. As we believe continuous domains are the most abundant in real-life and the most convenient to work with, we choose to address the problem of learning effective classifiers in domains with continuous features. The last characteristic we consider is certainly one of the most important and consists in the number of features. Depending on the complexity of the decision boundary and the amount of training data available, all classification algorithms do not handle high-dimensional problems equally. In other terms, some classifiers are more affected by the curse of dimensionality than others. This has been perfectly illustrated by Bickel, Levina, et al. (2004) who showed that when the number of features grows faster than the number of instances, even if the Gaussian assumption is met, the naive Bayes classifier (NB) outperforms the classical Linear Discriminant Analysis (LDA) which is asymptotically equivalent to random guess. This phenomenon explains why classifiers based on the often invalid assumption that the features are independent given the class often perform better than other classifiers when the number of features is large (Domingos & Pazzani, 1997; Lewis, 1998). In the light of the above, high dimensional classification seems to be of greater interest. That is why we choose to address the problem of learning effective high-dimensional classifiers.

So, all in all, the objective of this paper is to address the problem of learning effective high-dimensional binary classifiers in domains with continuous features. In this context, the already mentioned and widely used naive Bayes classifier naturally arises as an appealing solution. However, as shown by Rish et al. (2001), the conditional independence assumption limits its performance when the features are neither completely conditionally independent nor functionally dependent. Other probabilistic generative network-based classifiers have been developed to relax the independence assumption and overcome this limitation. For example, the Tree-Augmented Naive Bayes classifier (TAN) (Friedman, Geiger, & Goldszmidt, 1997) which incorporates some dependencies between the features by allowing a directed tree structure over the features or the Bayesian network Augmented Naive Bayes classifier (BAN) (Cheng & Greiner, 2001) which further expands the dependency relationship between any two features by allowing a directed acyclic

graph on the features. In certain domains, these Bayesian network-based classifiers can offer non-negligible gains in terms of predictive performance (Grossman, Domingos, & Domingos, 2004). Unfortunately, this comes at the expense of higher computational costs, especially in continuous domains. Consequently, in practice, these classifiers usually rely on simple parametric forms (e.g., Gaussian) and may show suboptimal performance (Elidan, 2012).

In an attempt to develop a more competitive Bayesian classifier overcoming the limitation of the naive Bayes, we propose a new classifier which relaxes the independence assumption by evaluating the class-conditional probability distributions of the features through pair-copula constructions. The use of copula functions in supervised classification is not new (e.g., Y. Chen, 2016; Elidan, 2012; Han, Zhao, & Liu, 2013; Pérez Díaz, 2018; Salinas-Gutiérrez, Hernández-Aguirre, Rivera-Meraz, & Villa-Diharce, 2010). The use of copula functions to extend the naive Bayes classifier is not new either (e.g., Y. Chen, 2016; Pérez Díaz, 2018; Salinas-Gutiérrez et al., 2010). However, only a small fraction of the possibilities have been explored and to the best of our knowledge, we are the first to present a copula-based extension of the naive Bayes which has been specifically developed for high-dimensional and large sample size data.

1 Bayesian classifiers

To avoid any further confusion, we start by introducing some notation and properly formulate a binary classification problem. We denote variables by uppercase letters (e.g., X_i), their values by lowercase letters (e.g., x_i) and vectors by boldface letters (e.g., $\boldsymbol{X}$). Suppose we have a training set $\{(\boldsymbol{x}^{(m)}, y^{(m)}), m = 1, \ldots, n\}$ independently drawn from a joint distribution of $(\boldsymbol{X}, Y)$. $\boldsymbol{X} = (X_1, \ldots, X_d)$ is the feature vector where each feature takes values from its domain D_i. The set of all instances is denoted $\Omega = D_1 \times \ldots \times D_d$. $Y \in \{0, 1\}$ is the class vector. The target of the classification is to determine the value of Y given a new instance $\boldsymbol{x}$. Consequently, the implicit objective of any binary classifier is to learn a concept $g : \Omega \to \{0, 1\}$ where $g(\boldsymbol{x}) = Y$. In practice, the concept to be learnt is approximated by a hypothesis which is a function $h : \Omega \to \{0, 1\}$ that assigns a class to any given example. The approach used in Bayesian classifiers consists in associating each class $k = 0, 1$ with a discriminant function $f_k(\boldsymbol{x})$ and assign the class with maximum discriminant function: $h(\boldsymbol{x}) = argmax_{k \in \{0,1\}} f_k(\boldsymbol{x})$.

1.1 Bayesian classifiers

Bayesian classifiers are part of the probabilistic generative classifiers. They are said to be naturally probabilistic because they use the class posterior probabilities as discriminant functions, i.e., $f_k(\boldsymbol{x}) = \Pr(Y = k \mid \boldsymbol{X} = \boldsymbol{x})$. They are also said to be generatively trained because they don't optimize the class posterior probabilities directly on the training set (in opposition to conditionally trained classifiers such as logistic regression). Instead, the training phase consists in finding the class-conditional probability distributions $\Pr(\boldsymbol{X} = \boldsymbol{x} \mid Y = k)$ and the class priors $\Pr(Y = k)$. Then, the class posterior probabilities given a feature vector are derived using Bayes rule

$$f_k(\boldsymbol{x}) = \Pr(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) = \frac{\Pr(\boldsymbol{X} = \boldsymbol{x} \mid Y = k) \cdot \Pr(Y = k)}{\Pr(\boldsymbol{X} = \boldsymbol{x})},$$

where $\Pr(\boldsymbol{X} = \boldsymbol{x})$, the evidence, is a scaling factor that can be ignored in the context of hard classification since it is identical for all classes (Bishop, 2006, p. 43). Thus, Bayesian classifiers are said to find the maximum a posterior probability (MAP) hypothesis given instance $\boldsymbol{x}$:

$$h(\boldsymbol{x}) = argmax_k \Pr(\boldsymbol{X} = \boldsymbol{x} \mid Y = k) \cdot \Pr(Y = k).$$

When the class-conditional probability distributions and the class priors are known, the hypothesis used in Bayesian classifiers is the optimal decision rule in the sense that it minimizes the probability of misclassification (see Bayes-optimal classifier (Devroye, Györfi, & Lugosi, 2013)). But in practice, they are not and the direct application of the Bayes rule becomes a hard and computationally expensive problem when dealing with high-dimensional feature spaces. That is because the number of parameters in the class-conditional probability distribution estimates increases exponentially with the number of features. Therefore, approximations are commonly used.

1.2 The naive Bayes classifier

By assuming the features are independent given the class, one can approximate the class-conditional probability distributions by

$$\Pr(\boldsymbol{X} = \boldsymbol{x} \mid Y = k) \propto \prod_{i=1}^{d} \Pr(X_i = x_i \mid Y = k).$$

This yields the naive Bayes classifier which uses the following discriminant functions:

$$f_k^{NB}(\boldsymbol{x}) = \prod_{i=1}^{d} \Pr(X_i = x_i \mid Y = k) \cdot \Pr(Y = k).$$

The naive Bayes classifier is the most widely used implementation of the Bayes classifier. Two main factors explain its popularity. First, the naive Bayes formulation significantly reduces the complexity of the Bayes classifier as the complex task of estimating k d-dimensional class-conditional probability distributions is transformed into the simpler task of estimating $k \times d$ one-dimensional class-conditional probability distributions independently. This makes the training procedure linear in features and explains why the naive Bayes classifier suffers less from the curse of dimensionality. Secondly, NB is remarkably successful in practice despite the fact that the simplifying conditional independence assumption is often violated. In many practical applications, NB often

competes with much more sophisticated classifiers (e.g., Domingos & Pazzani, 1997; Friedman et al., 1997; Hellerstein, Jayram, Rish, et al., 2000; Mitchell, 1997; Ng & Jordan, 2002). Yet, it often fails to produce good estimates for the true class probabilities (Niculescu-Mizil & Caruana, 2005). So why then can the naive Bayes classifier still be successful in the presence of feature dependencies? This is because the classification error is not necessarily related to the probability estimates. As long as the correct class is the most probable, an optimal classifier is obtained no matter the accuracy of the probability estimates (Domingos & Pazzani, 1997). However, the fact remains that the conditions for the naive Bayes to be optimal when the conditional independence assumption is violated are rarely met (Rish et al., 2001; Zhang, 2004). That's why the classifier often performs poorly. To overcome this limitation, other probabilistic generative network-based classifiers accounting for the dependence of features such as the TAN and the BAN have been developed (e.g., Cheng & Greiner, 2001; Friedman et al., 1997) but their performances are not always satisfactory (Elidan, 2012).

2 Copulas

Instead of approximating the class-conditional probability distributions on the basis of the conditional independence assumption in order to be able to derive the class posterior probabilities, another idea would be to model and estimate them with copula functions. For continuous features, this would result in a classifier whose discriminant functions are given by

$$\begin{aligned} f_k^C(\boldsymbol{x}) &= \Pr(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) \\ &\propto c\left(F(x_1 \mid k), \ldots, F(x_d \mid k); k\right) \cdot \prod_{i=1}^{d} f_i(x_i \mid k) \cdot \Pr(Y = k), \end{aligned} \tag{2.1}$$

where $c(\cdot)$ is a copula density function. In this chapter, we give the necessary theoretical background of copulas and introduce the main copula models available for both the bivariate and multivariate cases. In parallel, we review the different estimation methods.

2.1 Background

Copulas are a means of modeling the dependence structure of a multivariate random vector. The term "copula" has been introduced by Sklar in the context of its theorem and comes from the Latin word "copulare", to join or to link (Sklar, 1959). Copulas are popular in high-dimensional statistical applications because they facilitate the modeling and the estimation of a multivariate distribution by splitting it into two parts that can be estimated separately: the marginal distributions and the dependence structure. Copulas have been rediscovered relatively recently in applied sciences and are widely used in quantitative finance (Li, 2000; Low, Alcock, Faff, & Brailsford, 2013; Low, Faff, & Aas, 2016; Patton, 2001).

2.1.1 Definition

A function $C : [0,1]^d \to [0,1]$ is a d-dimensional copula if there exists a random vector $(U_1, \ldots, U_d)$ with $U_i \sim U[0,1], i = 1, \ldots, d$, such that

$$\Pr(U_1 \leq u_1, \ldots, U_d \leq u_d) = C(u_1, \ldots, u_d),$$

that is, C is the cumulative distribution function of $(U_1, \ldots, U_d)$. Consequently, a function $C : [0,1]^d \to [0,1]$ is a d-dimensional copula if:

- $C(u_1, \ldots, u_{i-1}, 0, u_{i+1}, \ldots, u_d) = 0$,
- $C(1, \ldots, 1, u_i, 1, \ldots, 1) = u_i$,
- $\Pr(U_1 \in [a_1, b_1], \ldots, U_d \in [a_d, b_d]) \geq 0, \qquad \forall 0 \leq a_j \leq b_j \leq 1$

That is, C must be grounded and d-non-decreasing.

2.1.2 Sklar's theorem

The theoretical foundation of copula theory is provided by Sklar's theorem (Sklar, 1959). The latter states that any multivariate cumulative distribution function can be expressed in terms of its marginals and a copula function. Let F be a d-dimensional cumulative distribution function with continuous margins[1] $F_1, \ldots, F_d$. The theorem states that it exists an unique d-dimensional copula such that

$$F(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)), \qquad \forall (x_1, \ldots, x_d) \in \mathbb{R}^{\mathrm{d}}.$$

The converse is also true. If C is a d-dimensional copula and $F_1, \ldots, F_d$ are univariate cumulative distributions functions, F is a d-dimensional cumulative distribution function. In the case F has a density f and the latter is available, it holds further that

$$f(x_1, \ldots, x_d) = c(F_1(x_1), \ldots, F_d(x_d)) \cdot \prod_{i=1}^{d} f_i(x_i). \tag{2.2}$$

Consider a random vector $(X_1, \ldots, X_d)$ with continuous marginal cumulative distribution functions $F_i(x) = \Pr(X_i \leq x)$. By applying the probability integral transform to each

[1] If the marginals F_i are not all continuous, the copula is uniquely determined only on $Ran(F_1) \cdot \ldots \cdot Ran(F_d)$ where $Ran(F_i)$ denotes the range of the cumulative distribution function F_i.

element X_i, we get the random vector $(U_1, \ldots, U_d) = (F_1(x_1), \ldots, F_d(x_d))$ with uniformly distributed marginals $U_i \sim U[0,1]$. So, the copula of $(X_1, \ldots, X_d)$ is defined as the joint cumulative distribution function of $(U_1, \ldots, U_d)$,

$$C(u_1, \ldots, u_d) = \Pr(U_1 \leq u_1, \ldots, U_d \leq u_d).$$

This explains why a copula is defined as the distribution of uniformly distributed random variables. As the marginals F_i were assumed to be continuous, we can apply inverse transform sampling and the copula function can be rewritten as

$$\begin{aligned} C(u_1, \ldots, u_d) &= \Pr(X_1 \leq F_1^{-1}(u_1), \ldots, X_d \leq F_d^{-1}(u_d)) \\ &= F(F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d)). \end{aligned}$$

2.1.3 Parametric copula families

The most trivial copulas allow to model independent structures (Independent copula), perfect positive dependence structures (Comonotonicity copula) and perfect negative dependence structures (Countermonotonicity copula). But there exist a large selection of copula models available in the literature (Cherubini, Luciano, & Vecchiato, 2004; Frees & Valdez, 1998; Joe, 1997; Nelsen, 2007). Among the most prominent ones, we selected a decent-sized set of copula models already well implemented in R such that we would be able to investigate a great variety of dependence structures (symmetric vs asymmetric, light vs heavy-tailed, weak vs. strong, etc). They fall into three categories: Meta elliptical copulas (Gaussian copula and t-copula), Archimedean copulas (Clayton copula, Frank copula, Gumbel copula, Independence copula and Joe copula) and Extreme-value copulas (Tawn copula).

2.1.3.1 Meta elliptical copulas

Meta elliptical copulas can be defined as distribution functions of component-wise transformed elliptically distributed random vectors (Embrechts, Lindskog, & McNeil, 2001). Consequently, Meta elliptical copulas present two relatively important drawbacks. First, they're restricted to have radial symmetry, meaning that they can only model symmetric dependence structures. Then, their function $C_{\mathbf{\Sigma}}^{MEC}(\mathbf{u})$ don't have closed form expressions and have to be numerically approximated (Genz & Bretz, 2009).

Gaussian Let $\mathbf{X} \sim \mathcal{N}_d(\mathbf{0}, \mathbf{\Sigma})$, where $\mathbf{\Sigma}$ is the correlation matrix. Then the corresponding Gaussian copula is given by

$$C_{\mathbf{\Sigma}}^{Gauss}(u_1, \ldots, u_d) = \mathbf{\Phi}_{\mathbf{\Sigma}}(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d)),$$

where $\Phi(\cdot)$ is the univariate standard normal c.d.f. and $\mathbf{\Phi}_{\mathbf{\Sigma}}(\cdot)$ the joint c.d.f. of $\mathbf{X}$.

Student Similarly, let $\mathbf{X} \sim \sqcup_d(\nu, \mathbf{0}, \mathbf{\Sigma})$, where ν is the degrees-of-freedom parameter and $\mathbf{\Sigma}$ is the association matrix. Then, the corresponding Student copula is given by

$$C_{\mathbf{\Sigma}}^{\sqcup}(u_1, \ldots, u_d) = \boldsymbol{t}_{\nu,\mathbf{\Sigma}}(\sqcup_{\nu}^{-1}(u_1), \ldots, \sqcup_{\nu}^{-1}(u_d)),$$

where $\sqcup_v(\cdot)$ is the univariate c.d.f. of a t-distribution with ν degrees of freedom and $\sqcup_{v,\mathbf{\Sigma}}(\cdot)$ the joint c.d.f. of $\mathbf{X}$.

Despite those drawbacks, because we're particularly interested in copula densities, Meta elliptical copulas might still be an attractive option in our case as their density hold a general formula for any d dimensions and can thus be easily generalized and applied to the multivariate case. For example, the density of a d-dimensional Gaussian copula can be written as

$$c_{\mathbf{\Sigma}}^{Gauss}(\mathbf{u}) = \frac{1}{\det(\mathbf{\Sigma})^{1/2}} \exp(-\frac{1}{2}\mathbf{z}'(\mathbf{\Sigma}^{-1} - \boldsymbol{I})\mathbf{z}),$$

where $\mathbf{z}_i = \Phi^{-1}(\mathbf{u}_i)$ for $i = 1, \ldots, d$ and $\boldsymbol{I}$ is the identity matrix (Dalla Valle, 2009). For further details on the selected Meta elliptical copulas, we refer to Fang, Fang, and Kotz (2002).

2.1.3.2 Archimedean copulas

The second considered set of copulas are the Archimedean copulas. In contrast to meta elliptical copulas, they're not derived from multivariate distributions using Sklar's Theorem. In this respect, all commonly encountered Archimedean copulas have closed form expressions and allow to model a great variety of dependence structures. Also, in arbitrarily high dimensions, Archimedean copulas allow to model dependence structures with a significantly lower number of parameters (usually from one to three) than meta elliptical copulas. While this can be an advantage in terms of complexity, it also means that multivariate extensions of Archimedean copula models can be prone to underfitting

because of the lack of free parameter choice. As we won't technically model dependence structures with multivariate Archimedean copulas, we will only consider the bivariate case. For multivariate extensions of the presented bivariate Archimedean copulas, we refer to Nelsen (2007) and Joe (1997). Let $\phi : [0,1] \to [0,\infty]$ be a continuous, strictly decreasing function such that $\phi(1) = 0$. Denote further ϕ^{-1} as the general inverse of ϕ. Then,

$$C(u_1, u_2) = \phi^{-1}(\phi(u_1) + \phi(u_2))$$

is called an Archimedean copula. Depending on the chosen function ϕ and the parameter space for θ, we get different one-parameter Archimedean copulas.

Clayton With $\phi_\theta(x) = (x^{-\theta} - 1)/\theta$ and $0 < \theta < \infty$, the Clayton copula (Clayton, 1978) is given by

$$C_\theta^{Clayton}(u_1, u_2) = (u_1^{-\theta} + u_2^{-\theta} - 1)^{-1/\theta}.$$

The density of the bivariate Clayton copula is given by (Venter, 2002)

$$c_\theta^{Clayton}(u_1, u_2) = (1+\theta)(u_1 \cdot u_2)^{-1-\theta}(u_1^{-\theta} + u_2^{-\theta} - 1)^{-1/\theta-2}.$$

Frank With $\phi_\theta(x) = -\ln((\exp(-\theta x) - 1)/(\exp(-\theta) - 1))$ and $\theta \in \mathbb{R}\setminus\{0\}$, the Frank copula (Frank, 1979) is given by

$$C_\theta^{Frank}(u_1, u_2) = -\frac{1}{\theta}\ln\left(1 + \frac{(\exp(-\theta u_1) - 1)(\exp(-\theta u_2) - 1)}{\exp(-\theta) - 1}\right).$$

The density of the bivariate Frank copula is given by (Venter, 2002)

$$c_\theta^{Frank}(u_1, u_2) = \frac{-\theta\,(\exp(-\theta) - 1)\,(1 + \exp(-\theta(u_1 + u_2)) - 1)}{((\exp(-\theta u_1) - 1)(\exp(-\theta u_2) - 1) + \exp(-\theta) - 1)^2}.$$

Gumbel With $\phi_\theta(x) = (-\ln(x))^\theta$ and $\theta \geq 1$, the Gumbel copula (Gumbel, 1960) is given by

$$C_\theta^{Gumbel}(u_1, u_2) = \exp(-((-\ln(u_1)^\theta)(-\ln(u_2)^\theta))^{1/\theta}).$$

The density of the bivariate Gumbel copula is given by (Venter, 2002)

$$\begin{aligned} c_\theta^{Gumbel}(u_1, u_2) = C_\theta^{Gumbel}(u_1, u_2) &\left((-\ln(u_1)^\theta) + (-\ln(u_2)^\theta)\right)^{-2+2/\theta} u_1^{-1}u_2^{-1} \\ &\times (\ln(u_1)\ln(u_2))^{\theta-1}\left(1 + (\theta-1)((-\ln(u_1)^\theta) + (-\ln(u_2)^\theta))^{-1/\theta}\right). \end{aligned}$$

Independence With $\phi(x) = -\ln(x)$, the Independence copula is given by

$$\Pi(u_1, u_2) = u_1 u_2.$$

The density of the bivariate Independence copula is given by

$$c\Pi(u_1, u_2) = 1.$$

Joe With $\phi_\theta(x) = -\log(1-(1-x)^\theta)$ and $\theta \in [1, \infty)$, the Joe copula (Joe, 1997) is given by

$$C_\theta^{Joe}(u_1, u_2) = 1 - \Big((1-u_1)^\theta + (1-u_2)^\theta - (1-u_1)^\theta(1-u_2)^\theta\Big)^{1/\theta}.$$

2.1.3.3 Extreme-value copulas

The third and last considered class of copulas are the Extreme-value copulas. With higher probability concentrated in the tails, Extreme-value copulas allow to model dependence structures between rare events (Gudendorf & Segers, 2010). Compared to our selected Archimedean and Meta elliptical copulas, they're not constrained to be symmetric. Formally, a d-dimensional copula C is an Extreme value copula if it is max-stable, i.e. it satisfies

$$C(u_1, \ldots, u_d) = C(\sqrt[m]{u_1} \ldots, \sqrt[m]{u_d})^m,$$

where m is an integer ≥ 1. We refer to Gudendorf and Segers (2010) for further details on Extreme-value copulas.

Gumbel The Gumbel copula is a special case as it belongs to both Archimedean and Extreme value copula families but doesn't allow for asymmetry (Genest & Rivest, 1989). To provide a model allowing for asymmetry in its components, we also included the Tawn copula (Tawn, 1988) in our selection. With $(\theta, \alpha_1, \alpha_2) \in (1, \infty) \times [0, 1]^2$, the latter is a three-parameter copula given by

$$C_{\theta,\alpha_1,\alpha_2}^{Tawn}(u_1, u_2) = \exp((\ln(u_1) + \ln(u_2))G(\ln(u_2)/\ln(u_1 u_2))),$$

where

$$G(x) = (1-\alpha_1)x + (1-\alpha_2)(1-x) + \Big((\alpha_1(1-x))^\theta + (\alpha_2 x)^\theta\Big)^{1/\theta}.$$

Note that the asymmetry is determined by α_1, α_2 and we recover the Gumbel copula when $\alpha_1 = \alpha_2$.

2.1.3.4 Rotated copulas

With their restricted parameter space, some of the selected copula models allow only for positive dependence while some other allow only for either upper or lower tail dependence. To overcome this lack of flexibility, we also include their 90, 180 and 270 degrees rotated versions in our selection. Let $c(u_1, u_2)$ be a two-dimensional copula density. To better fit the underlying dependence structure of the data, c can be rotated counterclockwise by:

- 90°: $c_{90}(u_1, u_2) = c(1 - u_1, u_2)$
- 180°: $c_{180}(u_1, u_2) = c(1 - u_1, 1 - u_2)$
- 270°: $c_{270}(u_1, u_2) = c(u_1, 1 - u_2)$.

2.2 Bivariate copula estimation

A d-dimensional copula involve d marginal cumulative distribution functions and a joint cumulative distribution function. The estimation of a copula function can then be split into two separate estimation procedures: estimating the margins and estimating the joint law. Depending on the assumptions made, some of these functions have to be estimated parametrically or non-parametrically. From the full empirical method (Deheuvels, 1980; Doukhan, Fermanian, Lang, et al., 2005) to the fully smoothed copula (Fermanian & Scaillet, 2003) to the semi-parametric estimator (Silvapulle, Kim, Silvapulle, et al., 2004) to the fully parametric estimator, there exist a wide range of possibilities for estimating copula functions. Of course, the estimation precision depends on these choices and the correspondence between the assumptions made for both the margins and the joint law.

2.2.1 Computing Pseudo-observations

From its definition, a copula is a cumulative distribution function of a random vector $(U_1, \ldots, U_d)$ with uniformly distributed marginals. Yet, we're usually not provided with

copula samples $(u_1^{(m)}, \ldots, u_d^{(m)})$ but with samples $(x_1^{(m)}, \ldots, x_d^{(m)})$ from a multivariate distribution. Then, the first step is to get copula samples. According to Sklar's theorem, provided an iid sample $(x_1^{(m)}, \ldots, x_d^{(m)})$ from a random vector $(X_1, \ldots, X_d)$ with marginal cumulative distribution functions $F_1, \ldots, F_d$, we can define the copula sample as $F_i(x_i^{(m)}) = u_i^{(m)}$ for all $i = 1, \ldots, d$ and for all $m = 1, \ldots, n$. But in practice, the true marginals F_i are rarely known, so copula samples must be obtained through their estimate $\widehat{F}_i(x_i^{(m)}) = \widehat{u}_i^{(m)}$. Copula samples obtained this way are generally called pseudo-observations because they aren't true copula samples. From parametric to non-parametric estimations, from step to smoothed functions, there exist many different methods to estimate continuous univariate (cumulative distribution) functions. The most popular options in the context of copula models are presented below.

2.2.1.1 Parametric estimation

Maximum likelihood estimator The idea behind the parametric approach is to assume the data is generated from a particular family and estimate its parameter(s) by maximum likelihood. Let $X_i \sim F_{\boldsymbol{\theta}_i}^{PAR}$, where $\boldsymbol{\theta}_i \in \Theta_i$ is the family parameter space. Let $f_{\boldsymbol{\theta}_i}^{PAR}$be the density of $F_{\boldsymbol{\theta}_i}^{PAR}$. The maximum likelihood estimator is the parameter vector that maximizes the likelihood:

$$\hat{\boldsymbol{\theta}}_{i,n}^{MLE} = argmax_{\boldsymbol{\theta}_i \in \Theta_i} \prod_{m}^{n} f_{\boldsymbol{\theta}_i}^{PAR}(x_i^{(m)}).$$

Then, the natural estimator for $F_i(x)$ is the chosen family's cumulative distribution function with the found parameter(s) value(s) $\hat{F}_i^{PAR}(x; \hat{\boldsymbol{\theta}}_{i,n}^{MLE})$. A well specified underlying parametric model for the margins can greatly impact the precision of the copula function estimation, but the reverse is also true and a poorly fitted model may have disastrous consequences. That's why, especially for marginal estimation, non-parametric methods should be preferred over the parametric approach except if provided with valuable prior information (Charpentier, Fermanian, & Scaillet, 2007).

2.2.1.2 Non-parametric estimation

The ranked-based method The most popular estimation method for the margins is to estimate them with the empirical cumulative distribution function. In this case, the

margins $F_i(x_i)$ are estimated by

$$\hat{F}_i^{ECDF}(x) = \frac{1}{n+1} \sum_{m=1}^{n} \mathbb{I}(x_i^{(m)} \leq x).$$

The computed pseudo-observations are then usually scaled by a multiplicative factor $\frac{n}{n+1}$ to force them to fall inside the open unit hypercube. Despite it's popularity, the empirical cumulative distribution function method has a major drawback: the computation of pseudo-observations is done component-wisely. Indeed, the empirical cumulative distribution function must be applied on each of the d components X_i and queried for each of the n realizations within each component. That's why in practice, pseudo-observations $(\hat{u}_1^{(m)}, \ldots, \hat{u}_d^{(m)})$ are often computed using the ranked-based method and defined via

$$\hat{u}_i^{(m)} = \frac{r_i^{(m)}}{n+1},$$

for $i = 1, \ldots, d$ and $m = 1, \ldots, n$, where $r_i^{(m)}$ is the rank of $x_i^{(m)}$ among the n realizations of X_i. Due to modern sorting algorithms and vectorization, the ranked based approach is significantly faster than the empirical cumulative distribution function method while providing the same results.

Smoothing methods The previous estimation method produce discrete functions. But for some applications, one might wish to produce continuous estimated margins$\widehat{F}_i(x_i)$. In those cases, a large set of non-parametric smoothing methods are available (e.g., see Härdle and Linton (1994) for a review). Because of its simplicity, the most common method is the kernel method. Given an univariate kernel function K, a continuous non-negative function which integrates to 1, and a positive bandwidth h, the marginal cumulative distribution functions $F_i(x_i)$ can be estimated by

$$\hat{F}_i^K(x, h_i) = \frac{1}{nh_i} \sum_{m=1}^{n} \mathbb{K}\left(\frac{x - x_i^{(m)}}{h_i}\right),$$

where $\mathbb{K}$is the primitive function of K: $\mathbb{K}(x) = \int_{-\infty}^{x} K$. In addition to offer continuity, the kernel-based estimator is known to outperform the empirical cumulative distribution function (Dutta, 2015). However, this statement holds true only with appropriate bandwidth and in practice, the latter's value is unknown and must be selected. So, depending on the kernel function (e.g., the Normal kernel for convenience or the Epanechnikov kernel for efficiency (Epanechnikov, 1969)), the type of bandwidth (fixed v.s. variable)

and the bandwidth selection method (e.g., plug-in v.s. cross-validation selectors, see Schindler (2012) for an introduction), the kernel method can be computationally expensive compared with the empirical distribution function.

2.2.2 Bivariate copula density estimation

After the computation of the pseudo-observations, the second step is the estimation of the copula function. As our objective is to estimate a joint probability distribution function rather than a joint cumulative distribution function, we focus on estimating the copula density. The most popular methods are presented below.

2.2.2.1 Parametric estimation

Maximum likelihood estimator The idea behind the parametric approach is to assume the copula sample $\{(u_1^{(m)}, \ldots, u_d^{(m)}), m = 1, \ldots, n\}$ is drawn from a random vector following a distribution from a particular copula family $(U_1, \ldots, U_d) \sim C$ and estimate its parameter(s) by maximum likelihood. Let $(U_1, \ldots, U_d) \sim C_{\boldsymbol{\theta}}^{PAR}$, where $\boldsymbol{\theta} \in \Theta$, where $\Theta \subset \mathbb{R}^{\mathrm{p}}$ is the family parameter space. Let $c_{\boldsymbol{\theta}}^{PAR}$ be the density of $C_{\boldsymbol{\theta}}^{PAR}$. The maximum likelihood estimator is the parameter vector that maximizes the likelihood:

$$\hat{\boldsymbol{\theta}}_n^{MLE} = argmax_{\boldsymbol{\theta} \in \Theta} \prod_m^n c_{\boldsymbol{\theta}}^{PAR}(u_1^{(m)}, \ldots, u_d^{(m)}).$$

Then, the natural estimator for the copula density is the chosen family's probability distribution function with the found parameter(s) value(s) $\hat{c}^{PAR}(u_1^{(m)}, \ldots, u_d^{(m)}; \hat{\boldsymbol{\theta}}_n^{MLE})$.

Inversion of empirical Kendall's τ In the bivariate case, many one-parameter copula families show a correspondence between their parameter and Kendall's τ (e.g., see Czado, Schepsmeier, and Min (2012); Joe (1997) for a selection of correspondences). Thus, it is also possible, for some families, to estimate their parameter by estimating Kendall's τ and exploiting their relationship. Assume $(U_1, U_2) \sim C_{\theta}^{PAR}$, where $\theta \in \Theta$, where $\Theta \subset \mathbb{R}$, is the family parameter space. Assume there exists a bijective function $g : \Theta \to [-1, 1]$, such that $g(\theta) = \tau(U_1, U_2)$. Then, given $\tau(U_1, U_2)$, the bijective function g can be inverted to find the family parameter θ:

$$\theta^{\tau^{-1}} = g^{-1}(\tau(U_1, U_2)). \tag{2.3}$$

Kendall's τ, also called Kendall's rank correlation measure, is a concordance measure introduced by Kendall (1938) which is defined as the probability of concordance minus the probability of discordance. Given two independent and identically distributed two-dimensional random vectors (X_1, X_2) and $(\widetilde{X}_1, \widetilde{X}_2)$, Kendall's τ is defined as

$$\tau(X_1, X_2) = \Pr((X_1 - \widetilde{X}_1)(X_2 - \widetilde{X}_2) > 0) - \Pr((X_1 - \widetilde{X}_1)(X_2 - \widetilde{X}_2) < 0).$$

But in practice, the probabilities of concordance and discordance and, consequently, $\tau(U_1, U_2)$ are unknown. By estimating the two probabilities by the empirical proportions of concordant and discordant pairs in a sample, we obtain a non-parametric estimator of Kendall's τ : the empirical Kendall's τ. Let $\{(u_1^{(m)}, u_2^{(m)}), m = 1, \ldots, n\}$ be two-dimensional copula samples. The empirical Kendall's τ is given by

$$\widehat{\tau}_n(U_1, U_2) = \frac{N_c - N_d}{n(n-1)/2},$$

where N_c is the number of concordant pairs, N_d is the number of discordant pairs and the denominator is the total number of pair combinations in the sample (Hazewinkel, 1994). So, if we replace Kendall's τ by its estimate in equation (2.3), we get the estimated family parameter $\hat{\theta}$:

$$\hat{\theta}_n^{\hat{\tau}_n^{-1}} = g^{-1}(\widehat{\tau}_n(U_1, U_2)).$$

2.2.2.2 Parametric model selection

In contrast to marginal estimation, copula estimation is often approached parametrically without any valuable prior information. The usual procedure consists in estimating the parameter(s) for several copula families and select the model that minimizes a chosen information criterion. The two most commonly used criteria are the Akaike's information criterion (AIC) (Akaike, 1974)

$$AIC_n = -2 \sum_{m=1}^{n} \ln(\hat{c}_{\hat{\boldsymbol{\theta}}_n}^{PAR}(u_1^{(m)}, \ldots, u_d^{(m)})) + 2p,$$

and the Bayesian information criterion (BIC) (Schwarz et al., 1978)

$$BIC_n = -2 \sum_{m=1}^{n} \ln(\hat{c}_{\hat{\boldsymbol{\theta}}_n}^{PAR}(u_1^{(m)}, \ldots, u_d^{(m)})) + \ln(n)p,$$

where p is the number of parameters. Both criteria select the model with the highest penalized likelihood. The penalty depends on the number of parameters and acts as a balance to avoid overfitting. Note that, for large samples, BIC's penalty becomes significantly higher for a given number of parameters as $\ln(n) > 2$ for $n \geq 8$. Many other selection criteria can be used for bivariate copula selection but we chose to work with the AIC. Two factors motivated our choice. First, it is undeniably one of the most computationally efficient selection criteria. Then, even if the AIC is not theoretically suitable for comparing different copula families (due to the increased variability of the estimated AIC difference when comparing pairs of non-nested models (Ripley, 2007)), a large scale simulation study showed that the AIC is the most reliable selection criterion for copula selection (E. Brechmann, 2010).

2.2.2.3 Non-parametric estimation

The first approach consists in estimating the copula function C by $\hat{C}$ and defining an estimate of the copula density c at every $\mathbf{u} \in [0,1]^d$ by

$$\hat{c}(\mathbf{u}) = \frac{\partial^d}{\partial_1 \cdots \partial_d} \hat{C}(\mathbf{u}).$$

For example, one could use the empirical copula $\hat{C}^n$(Deheuvels, 1980) to estimate the copula function C and perform differentiation to get the associated copula density $\hat{c}^n$. While this is tempting, discontinuities in the data can lead to differentiability issues as the copula estimator isn't differentiable when involving an empirical cumulative distribution function. The second approach consists in estimating the copula density c directly from data using kernel methods. In the context of copula density estimation, traditional kernel density estimation techniques generally present two major drawbacks. First, kernel methods suffer from the curse of dimensionality. As the dimension d increases, the complexity increases exponentially and the rate of convergence decreases, which might lead to a poor estimation of c. One way to overcome this problem is to restrict oneself to the bivariate case by the use of vine copula models (further details below). Secondly, they suffer from boundary bias. When applied to a bivariate copula sample $(\widetilde{U}_1, \widetilde{U}_2)$, the kernel density estimator is given by

$$\hat{c}^{KDE}(u_1, u_2) = \frac{1}{n} \sum_{m=1}^{n} K_{h_n}(u_1 - \widetilde{U}_{1,m}) K_{h_n}(u_2 - \widetilde{U}_{2,m}), \quad \forall\, (u_1, u_2) \in [0,1]^2\,,$$

where $K_{h_n}(\cdot) = \frac{1}{h_n}K(\cdot/h_n)$ is a kernel function and h_n a bandwidth > 0. If we consider a point close to a boundary of the unit square, a relatively important part of its probability mass (kernel's bump centered around this point) lies outside the unit square. Consequently, $\hat{c}^{KDE}$ isn't a density function on $[0,1]^2$ as it doesn't integrate to one over the unit square. Plus, depending on the bandwidth, the estimator might be severely biased in the neighborhood of the boundaries. To cope with such issues, there exist several more advanced kernel density estimation techniques with boundary bias correction. After a short review of the literature, we've found three categories of techniques for direct bivariate copula density estimation. The first one consists of techniques aiming to redistribute the probability mass outside the unit square into $[0,1]^2$ by a reflection process (e.g., the mirror-reflection technique (Gijbels & Mielniczuk, 1990) and its improved version (Omelka, Gijbels, Veraverbeke, et al., 2009)). The second set of techniques consist in varying the kernel shape for each estimated point such that its support matches the bounded support of the density (e.g., Beta kernels (Charpentier et al., 2007; S. X. Chen, 1999)). The third category regroups techniques that transform the data so that its distribution is supported on the full $\mathbb{R}^2$ (e.g., the transformation estimator (Charpentier et al., 2007) and its generalized version, the local likelihood transformation estimator (Geenens, Charpentier, Paindaveine, et al., 2017)). A simulation study proposed by Nagler (2014) showed that the third category, and more specifically the local likelihood transformation estimator, outperforms all the other kernel-based estimators for the estimation of ten out of twelve copula models in terms of integrated squared error. Even if kernel estimators can offer significantly better estimates than parametric estimators when the dependence structures cannot be captured properly by any parametric families, they tend to become too computationally expensive to be a viable option when estimating multivariate copulas (see Subsection 2.3.2.6).

2.3 Multivariate copula estimation

In the last Section, we introduced bivariate estimation techniques. However, the indirect objective of this study is to estimate multivariate copula densities. Hereafter, we present the two most prominent approaches for multidimensional copula estimation: standard multivariate copula models and vine copula models.

2.3.1 Standard multivariate copula models

As we have seen, the literature is replete with parametric bivariate copula models but the set of models for multivariate copulas is rather limited. There have been some attempts to construct multivariate extensions of bivariate Archimedean copulas such as fully and partially nested Archimedean copulas (see, e.g., Joe (1997); McNeil (2008); Savu and Trede (2010); Whelan et al. (2004)), hierarchical Archimedean copulas (see, e.g., Savu and Trede (2010)) or multiplicative Archimedean copulas (see, e.g., Morillas (2005)). However, building these extensions is generally more difficult as they require additional parameter restrictions. Therefore, we choose not to include multivariate Archimedean copulas in our model comparison. Another more popular way to model multivariate copulas is to use Meta elliptical copulas. That is because both the Gaussian and the t copulas can be easily extended to the multivariate case (Fang et al., 2002). Both models are defined in Subsection 2.1.3.1 and can be fitted by Maximum Likelihood (see Subsection 2.2.2.1). The only difference with the bivariate case is that, as the optimization process becomes significantly more complex when the dimension increases, a sequential estimation approach might be used to find good starting values for the parameters. For the sake of comparison, we choose to include multivariate Gaussian copulas in our model selection.

2.3.2 Vine copula models

The major drawback of standard multivariate copula models is that they can become inflexible in high dimensions for modeling dependence structures. A more flexible alternative is to construct multivariate copulas using only bivariate copulas as building blocks with a method called pair copula construction (PCC) (Bedford & Cooke, 2001; Bedford, Cooke, et al., 2002; Joe, 1996; Kurowicka & Cooke, 2006). Due to their high flexibility, pair copula constructions, also called regular vine copulas, allow to model a wide range of complex dependence structures.

2.3.2.1 A pair-copula decomposition of a general multivariate distribution

As a starting point, consider a random vector $\mathbf{X} = (X_1, \ldots, X_d)$ with joint probability density function f and marginal probability density functions f_i for $i = 1, \ldots, d$. This density can be factorized as

$$f(x_1, \ldots, x_d) = f_d(x_d) \cdot f(x_{d-1} \mid x_d) \cdot f(x_{d-2} \mid x_{d-1}, x_d) \cdots f(x_1 \mid x_2, \ldots, x_d). \quad (2.4)$$

Each term in equation (2.4) can be decomposed into the product of the appropriate pair-copula and a conditional marginal density, using the general formula

$$f(x \mid \boldsymbol{v}) = c_{x,v_j|\boldsymbol{v}_{-j}} \left(F(x \mid \boldsymbol{v}_{-j}), F(v_j \mid \boldsymbol{v}_{-j})\right) \cdot f(x \mid \boldsymbol{v}_{-j}), \tag{2.5}$$

where $\boldsymbol{v}$ is a d-dimensional vector, v_j is an arbitrarily chosen component of $\boldsymbol{v}$, and $\boldsymbol{v}_{-j}$ denotes $\boldsymbol{v}$ excluding v_j. Note that, in general, the pair copula $c_{x,v_j|\boldsymbol{v}_{-j}}$ might be different depending on the value of the conditioning variables $\boldsymbol{v}_{-j}$. However, we assume the pair copula $c_{x,v_j|\boldsymbol{v}_{-j}}$ is independent of the set of conditioning variables $\boldsymbol{v}_{-j}$, that is

$$c_{x,v_j|\boldsymbol{v}_{-j}} \left(F(x \mid \boldsymbol{v}_{-j}), F(v_j \mid \boldsymbol{v}_{-j})\right) \equiv c_{x,v_j|\boldsymbol{v}_{-j}} \left(F(x \mid \boldsymbol{v}_{-j}), F(v_j \mid \boldsymbol{v}_{-j}); \boldsymbol{v}_{-j}\right).$$

This simplifying assumption allows for the construction of less complex and more flexible models as only unconditional copulas are involved in the pair-copula construction. Such PPCs are called simplified PCCs and have been shown to be good approximations to the correct decomposition. We refer to Haff, Aas, and Frigessi (2010) and Stoeber, Joe, and Czado (2013) for further details on the appropriateness of this assumption. We will assume the simplifying assumption holds for the remainder of this paper.

The pair copulas obtained from the decomposition of equation 2.4 via equation 2.5 are applied to transformed variables, which are marginal conditional distributions of the form $F(x \mid \boldsymbol{v})$. When $\boldsymbol{v}$ has more than one element, such distributions can be difficult to work with. Fortunately, Joe (1996) showed that for every arbitrarily chosen component $v_j \in \boldsymbol{v}$, these can be obtained via

$$F(x \mid \boldsymbol{v}) = \frac{\partial C_{x,v_j|\boldsymbol{v}_{-j}} \left(F(x \mid \boldsymbol{v}_{-j}), F(v_j \mid \boldsymbol{v}_{-j})\right)}{\partial F(v_j \mid \boldsymbol{v}_{-j})}, \tag{2.6}$$

where $C_{x,v_j|\boldsymbol{v}_{-j}}$ is a bivariate copula distribution function. By recursively applying equation 2.6 to rewrite the conditional distributions appearing in the arguments, we end up working with a chain of univariate conditional distributions of the form $F(x \mid v)$ where v has a single element. In this case, we have

$$F(x \mid v) = \frac{\partial C_{x,v}(F(x), F(v))}{\partial F(v)}. \tag{2.7}$$

In the context of vine copula models where x and v are uniform (i.e., pseudo observations), such functions are called h-functions and represent the conditional distribution functions

of the bivariate copula $C_{x,v}(x, v)$. They are defined as

$$h_{x|v}(x \mid v) = C_{x|v}(x \mid v) = F(x \mid v) = \frac{\partial C(x,v)}{\partial v} = \Pr(X \leq x \mid V = v),$$
$$h_{v|x}(v \mid x) = C_{v|x}(v \mid x) = F(v \mid x) = \frac{\partial C(x,v)}{\partial x} = \Pr(V \leq v \mid X = x).$$

In a similar manner, when x and v are uniform, equation 2.6 can be reformulated in terms of copula functions, in which case we have

$$\begin{aligned} C_{x|v}(x \mid \boldsymbol{v}) &= C_{x|v_j;v_{-J}}\left(C_{x|v_{-j}}(x \mid \boldsymbol{v}_{-j}) \mid C_{v_j|v_{-j}}(v_j \mid \boldsymbol{v}_{-j})\right) \\ &= h_{x|v_j;v_{-j}}\left(C_{x|v_{-j}}(x \mid \boldsymbol{v}_{-j}) \mid C_{v_j|v_{-j}}(v_j \mid \boldsymbol{v}_{-j})\right), \end{aligned} \tag{2.8}$$

where $h_{x|v_j;v_{-j}}$is the h-function corresponding to the random vector

$$\left(C_{x|v_{-j}}(X \mid \boldsymbol{v}_{-j}) \mid C_{v_j|v_{-j}}(V_j \mid \boldsymbol{v}_{-j})\right).$$

All in all, this means we can factorize a joint density function $f(x_1, \ldots, x_d)$ as a product of pair-copula densities and marginal densities. For example, in the three-dimensional case, one possible decomposition of $f(x_{1,}x_2, x_3)$ is given by

$$f(x_{1,}x_2, x_3) = f(x_3 \mid x_1, x_2) \cdot f(x_2 \mid x_1) \cdot f_1(x_1),$$

where $f(x_2 \mid x_1)$ can be factorized as

$$f(x_2 \mid x_1) = \frac{f(x_1, x_2)}{f_1(x_1)} = c_{1,2}(F_1(x_1), F_2(x_2)) \cdot f_2(x_2),$$

and where $f(x_3 \mid x_1, x_2)$ can be factorized as

$$f(x_3 \mid x_1, x_2) = \frac{f(x_1, x_3 \mid x_2)}{f(x_1 \mid x_2)} = c_{1,3|2}(F(x_1 \mid x_2), F(x_3 \mid x_2)) \cdot f(x_3 \mid x_2).$$

Given that $f(x_3 \mid x_2)$ can be factorized as

$$f(x_3 \mid x_2) = \frac{f(x_2, x_3)}{f_2(x_2)} = c_{2,3}(F_2(x_2), F_3(x_3)) \cdot f(x_3),$$

$f(x_1,x_2,x_3)$ can be further decomposed into

$$\begin{aligned} f(x_1,x_2,x_3) = f_1(x_1) \cdot f_2(x_2) \cdot f(x_3) \\ \times c_{1,2}(F_1(x_1), F_2(x_2)) \cdot c_{2,3}(F_2(x_2), F_3(x_3)) \\ \times c_{1,3|2}(F(x_1 \mid x_2), F(x_3 \mid x_2)). \end{aligned}$$

If we consider the already mentioned relationship between the density and the copula density in equation 2.2, we can rewrite this expression in terms of the copula density:

$$\begin{aligned} c(F_1(x_1), F_2(x_2), F_3(x_3)) = c_{1,2}(F_1(x_1), F_2(x_2)) \cdot c_{2,3}(F_2(x_2), F_3(x_3)) \\ \times c_{1,3|2}(F(x_1 \mid x_2), F(x_3 \mid x_2)). \end{aligned}$$

2.3.2.2 Regular vines

It is important to notice that the decomposition of the density is not unique. There are a significant number of possible decompositions for the density when working in high dimensions. Bedford et al. (2002) introduced a graphical method called regular vines (R-vine) to help classify all valid decompositions in terms of linked trees $T_i = (N_i, E_i)$, $i = 1, \ldots, d-1$, where N_i is the set of nodes and E_i is the set of undirected edges in tree T_i. According to Kurowicka and Cooke (2006), a sequence of trees $\mathcal{V} := (T_1, \ldots, T_{d-1})$ with nodes N_i and edges E_i for $i = 1, \ldots, d-1$, is called a regular vine on d variables if it satisfies the following:

1. T_1 has nodes $N_1 = \{1, \ldots, d\}$ and edges E_1.
2. For $i = 2, \ldots, d-1$, the tree T_i has nodes $N_i = E_{i-1}$.
3. (Proximity condition) For $i = 2, \ldots, d-1$, two nodes of T_i can only be connected by an edge if the corresponding edges in T_{i-1} share a common node.

The edges of an R-vine can be uniquely identified by two sets of nodes: the two conditioned nodes $\{j, k\}$ and the set of conditioning nodes D. That means the edges in tree T_i can be identified by $j, k \mid D$ where $j < k$ for convenience. Note that in tree T_1, the edges can be identified by j, k only as $D = \emptyset$ for all edges. Therefore, it is common to use a specific scheme to annotate the edges in each tree (Czado, 2010). The annotation of an edge $e \in T_i$, $i \geq 2$, is done as follows:

1. Consider all the numbers appearing in the annotation of the two nodes joined by e in T_i.

2. Le the numbers appearing in both nodes be the conditioning set $D(e)$ of e.

3. Let the numbers appearing in only one the nodes be the two conditioned nodes $j(e), k(e)$ of e.

4. Annotate e by $j(e), k(e) \mid D(e)$.

As an example, imagine we wish to annotate the topmost edge in T_3 in Figure 2.1 (that we denote by $a = j(a), k(a) \mid D(a)$). To do so, we first dress a list of all the numbers appearing in the two nodes, i.e., $\{1, 1, 2, 3, 3, 4\}$. Then, we attribute the duplicated entries in the list to the conditioning set, i.e., $D(a) = 1, 3$. Finally, we let the two left numbers be the conditioned nodes, i.e., $j(a) = 2, k(a) = 4$. So, the edge a is annotated by $2, 4 \mid 1, 3$.

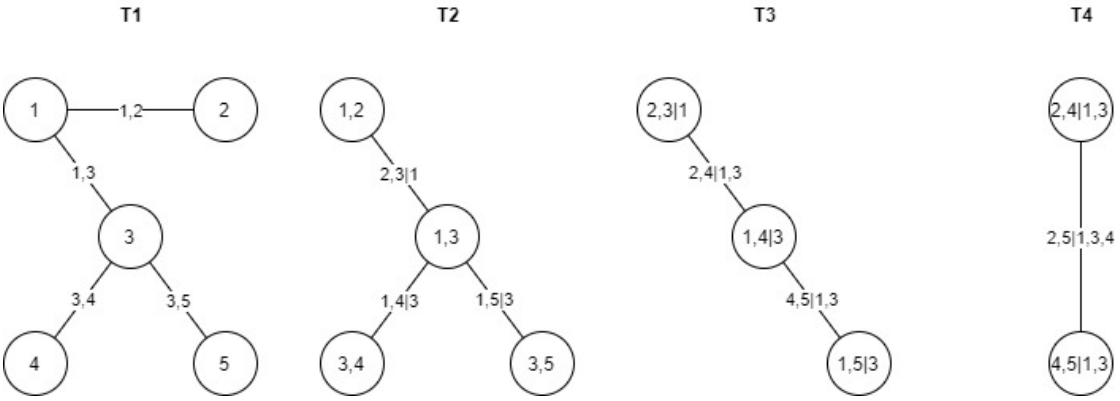

Figure 2.1: Example of a R-vine tree sequence for $d = 5$.

2.3.2.3 Regular vine copulas

Given a R-vine tree with the node set $\mathcal{N} := \{N_1, \ldots, N_{d-1}\}$ and the edge set $\mathcal{E} := \{E_1, \ldots, E_{d-1}\}$, one can construct a statistical model by associating each edge $e = j(e), k(e) \mid D(e)$ in E_i with a conditional bivariate copula density $c_{j(e),k(e)|D(e)}$ where $j(e)$ and $k(e)$ are the conditioned nodes and $D(e)$ is the conditioning set. Then, a copula C corresponding to a random vector $(U_1, \ldots, U_d)$ where $U_j \sim U[0, 1], j = 1, \ldots, d$, is called a regular vine copula if there is a tuple $(\mathcal{V}, \mathcal{C})$ that satisfies the following:

1. $\mathcal{V}$ is a regular vine tree on d variables.

2. $\mathcal{C} = \left\{ C_{j(e),k(e)|D(e)} \mid e \in E_i, i = 1, \ldots, d-1 \right\}$, where $C_{j(e),k(e)|D(e)}$ is the short notation for the bivariate copula $C_{j(e),k(e)|D(e)} \left(C_{j(e)|D(e)}(u_{j(e)} \mid \boldsymbol{u}_{D(e)}), C_{k(e)|D(e)}(u_{k(e)} \mid \boldsymbol{u}_{D(e)}) \right)$ where $\boldsymbol{u}_{D(e)}$ denotes the subvector of $\boldsymbol{u} = (u_1, \ldots, u_d)$ determined by the indices in $D(e)$.

If the $d\,(d-1)\,/2$ bivariate copulas in $\mathcal{C}$ admit a density, the density of the full R-vine copula C is given by their product:

$$c(\boldsymbol{u}) = \prod_{i=1}^{d-1} \prod_{e \in E_i} c_{j(e),k(e)|D(e)} \left(C_{j(e)|D(e)}(u_{j(e)} \mid \boldsymbol{u}_{D(e)}), C_{k(e)|D(e)}(u_{k(e)} \mid \boldsymbol{u}_{D(e)}) \right).$$

For example, the density of the R-vine copula corresponding to the tree sequence in Figure 2.1 is given by

$$\begin{aligned} c(u_1, u_2, u_3, u_4, u_5) = {} & c_{1,2}(u_1, u_2) \cdot c_{1,3}(u_1, u_3) \cdot c_{3,4}(u_3, u_4) \cdot c_{3,5}(u_3, u_5) \\ & \times c_{2,3|1}(C_{2|1}(u_2 \mid u_1), C_{3|1}(u_3 \mid u_1)) \cdot c_{1,4|3}(C_{1|3}(u_1 \mid u_3), C_{4|3}(u_4 \mid u_3)) \cdot c_{1,5|3}(C_{1|3}(u_1 \mid u_3) \\ & \times c_{2,4|1,3}(C_{2|1,3}(u_2 \mid \boldsymbol{u}_{1,3}), C_{4|1,3}(u_4 \mid \boldsymbol{u}_{1,3})) \cdot c_{4,5|1,3}(C_{4|1,3}(u_4 \mid \boldsymbol{u}_{1,3}), C_{5|1,3}(u_5 \mid \boldsymbol{u}_{1,3})) \\ & \times c_{2,5|1,3,4}(C_{2|1,3,4}(u_2 \mid \boldsymbol{u}_{1,3,4}), C_{5|1,3,4}(u_5 \mid \boldsymbol{u}_{1,3,4})). \end{aligned}$$

Thus, as proved in Theorem 4.2 of Kurowicka and Cooke (2006), the joint density of a d-dimensional random vector $\boldsymbol{X}$is uniquely determined by

$$f(x_1, \ldots, x_d) = \left[\prod_{i=1}^{d} f_i(x_i) \right] \times \left[\prod_{i=1}^{d-1} \prod_{e \in E_i} c_{j(e),k(e)|D(e)} \left(F(x_{j(e)} \mid \boldsymbol{x}_{D(e)}), F(x_{k(e)} \mid \boldsymbol{x}_{D(e)}) \right) \right],$$

where the rightmost part is the R-vine density.

2.3.2.4 Estimation procedure

If C is an R-vine copula corresponding to the tuple $(\mathcal{V}, \mathcal{C})$ and if we are provided with pseudo observations $\{(\hat{u}_1^{(m)}, \ldots, \hat{u}_d^{(m)}), m = 1, \ldots, n\}$ from C, its density c can be sequentially estimated by conducting only bivariate estimations:

1. For all $e \in E_1$:

   Use samples $\left(\hat{u}_{j(e)}^{(m)}, \hat{u}_{k(e)}^{(m)} \right)_{m=1,\ldots,n}$ to obtain estimates for $c_{j(e),k(e)}$.

2. For $i = 2, \ldots, d-1$:

   For all $e \in E_i$:

   a) For $v = j(e), k(e)$:

      i. Choose another index $v' \in D(e)$ such that $C_{v,v'|D(e)\setminus\{v'\}} \in \mathcal{C}$.

      ii. Obtain an estimate of $C_{v|v';D(e)\setminus\{v'\}}$ denoted $\hat{h}_{v|v';D(e)\setminus\{v'\}}$ based on the sample $\left(\hat{u}^{(m)}_{v|D(e)\setminus\{v'\}}, \hat{u}^{(m)}_{v'|D(e)\setminus\{v'\}}\right)_{m=1,\ldots,n}$, where $u_{v|D(e)\setminus\{v'\}}$is the short notation for $C_{v|D(e)\setminus\{v'\}}(u_v \mid \boldsymbol{u}_{D(e)\setminus\{v'\}})$.

      iii. Compute sample $\hat{u}^{(m)}_{v|D(e)} := \hat{h}_{v|v';D(e)\setminus\{v'\}}\left(\hat{u}^{(m)}_{v|D(e)\setminus\{v'\}} \mid \hat{u}^{(m)}_{v'|D(e)\setminus\{v'\}}\right), \qquad m = 1, \ldots, n.$

   b) Obtain an estimate of $c_{j(e),k(e)|D(e)}$ based on the sample $\left(\hat{u}^{(m)}_{j(e)|D(e)}, \hat{u}^{(m)}_{k(e)|D(e)}\right)_{m=1,\ldots,n}$.

For the first tree, we are provided with samples from the random variables associated to each node, so we can directly obtain estimates for all pair-copulas corresponding to the edges of the tree. For the next trees, we must first estimate and apply the required h-functions in order to obtain the pseudo-samples corresponding to each node and be able to estimate all pair-copulas corresponding to the edges of the trees. After completion, the procedure provides us with all the ingredients required to evaluate the density of the full R-vine copula, namely all estimated bivariate copula densities and h-functions.

As an example, let's consider the R-vine copula density decomposition from Figure 2.1. Imagine we have already estimated the pair-copulas corresponding to edges in the two first trees and we now wish to estimate the pair-copula corresponding to the topmost edge in T_3, i.e., $c_{2,4|1,3}(C_{2|1,3}(u_2 \mid \boldsymbol{u}_{1,3}), C_{4|1,3}(u_4 \mid \boldsymbol{u}_{1,3}))$. To do so, we first need an estimate for both arguments in order to compute the sample required for its estimation. This can be done by exploiting the relationship given in equation 2.8. Let's start with the first argument, i.e., $C_{2|1,3}(u_2 \mid \boldsymbol{u}_{1,3})$. Its estimate can be obtained by estimating

$$\begin{aligned} C_{2|1,3}(u_2 \mid \boldsymbol{u}_{1,3}) &= C_{2|1;3}\left(C_{2|3}(u_2 \mid u_3) \mid C_{1|3}(u_1 \mid u_3)\right) \\ &= C_{2|3;1}\left(C_{2|1}(u_2 \mid u_1) \mid C_{3|1}(u_3 \mid u_1)\right). \end{aligned}$$

We choose to use $C_{2|3;1}$as $C_{2,3|1} \in \mathcal{C}$, which means we already have samples from its two arguments (we have already computed samples from $\hat{C}_{2|1}(u_2 \mid u_1)$ and $\hat{C}_{3|1}(u_3 \mid u_1)$ to obtain the estimate for $c_{2,3|1}$). Therefore, we can obtain an estimate of $C_{2|3;1}$ based on the

sample $\{(\hat{u}_{2|1}^{(m)}, \hat{u}_{3|1}^{(m)}), m = 1, \ldots, n\}$. With this estimate which we denote $\hat{h}_{2|3;1}$, we define the sample $\hat{u}_{2|1,3}^{(m)}$ based on the sample $\{(\hat{u}_{2|1}^{(m)} \mid \hat{u}_{3|1}^{(m)}), m = 1, \ldots, n\}$. We then repeat the same procedure for the second argument, i.e., $C_{4|1,3}(u_4 \mid \boldsymbol{u}_{1,3})$. In short, we obtain an estimate of $C_{4|1;3}$ based on the previously obtained sample $\{(\hat{u}_{4|3}^{(m)}, \hat{u}_{1|3}^{(m)}), m = 1, \ldots, n\}$ that we use to define the sample $\hat{u}_{4|1,3}^{(m)}$ based on the sample $\{(\hat{u}_{4|3}^{(m)} \mid \hat{u}_{1|3}^{(m)}), m = 1, \ldots, n\}$. Now that we have samples for both arguments, i.e., $\{(\hat{u}_{2|1,3}^{(m)}, \hat{u}_{4|1,3}^{(m)}), m = 1, \ldots, n\}$, we obtain an estimate for $c_{2,4|1,3}$.

2.3.2.5 Structure selection

In the estimation procedure described above, the user has to specify an R-vine tree sequence $\mathcal{V}$ beforehand to tell the algorithm which pair-copulas and h-functions have to be estimated. This tree sequence is commonly called the structure of the tree. However, even if a good structure can sometimes be provided by a priori expert knowledge, it is usually unknown to the user. The structure selection procedure is an important step of vine copula estimation as the structure of the tree can have a significant influence on the estimator's performance (Garcia & Tsafack, 2011). When the structure is unknown, the user is offered two main options. Firstly, he may estimate a R-vine copula model for every possible R-vine tree structure and select the one maximizing the full likelihood. Naturally, this approach offers the most performant estimator but becomes infeasible when dealing with high-dimensional problems. The alternative approach is to build the tree structure through automatic structure selection procedures. There are many heuristic and approximative algorithms available (e.g., see Czado, Jeske, and Hofmann (2013) for a review) but we chose to work with Dissmann, Brechmann, Czado, and Kurowicka (2013)'s heuristic because of its popularity. The basic idea behind Dissmann et al. (2013)'s heuristic is to model the highest dependencies in the lower trees by sequentially selecting the tree maximizing the cumulative pairwise dependencies between the corresponding variables. To do that:

1. We first build a complete[2] graph with d nodes corresponding to each of the d variables. Note that this is only for the first tree. For the subsequent trees, we build graphs based on the R-vine construction principles, in particular on the proximity condition.

[2] A complete graph is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge (Wikipedia contributors, 2019).

2. We then attribute a weight according to a measure of pairwise dependence between the two respective nodes i,j to all edges of the graph: $|\hat{\tau}_{i,j}|$. Among the variety of dependence measures available, we chose to use the absolute value of empirical Kendall's τ for two reasons. First, in contrast to other criteria such as the AIC, the BIC or the corrected AIC (cAIC), its computation doesn't require estimation and model selection for all possible pairs. Finally, we chose it over empirical Spearman's ρ as we were going to compute it to estimate most of the bivariate copula function parameters anyway.

3. We then use Prim (1957)'s greedy algorithm to find the spanning tree maximizing the cumulative pairwise dependencies for all nodes. The procedure can be summarized as follows

   a) Initialize a tree with a single arbitrarily chosen node from the graph built in (1).

   b) Select the maximum-weight edge from the edges connecting the tree to nodes not yet in the tree and transfer it to the tree.

   c) Repeat (b) until all nodes are in the tree.

   Note that in this case, the use of a greedy algorithm has no cost in terms of performance as Prim's algorithm has been proven to yield the global optimum for the maximum spanning tree (MST) problem.

4. Finally, we obtain estimates for all pair-copulas corresponding to the edges of the tree and for all required h-functions and compute the pseudo-observations for the next level via the estimated h-functions.

5. We then repeat the same procedure with the new $d-1$ variables until we get the complete R-vine copula model.

Even if the sequential method proposed by Dissmann et al. (2013) is not guaranteed to find a global optimum, i.e., the best-fitting model, we believe it is the most suitable method available for high-dimensional vine copula estimation. That is because we believe the idea behind the heuristic is senseful. Indeed, if we assume that the model fit is often greatly influenced by the copulas families specified in the first tree and that it is more important to model the dependence structures associated with the highest dependencies correctly, it seems appropriate to model the highest dependencies in the lower trees. Also,

if we consider real applications for which it is natural to assume it is the dependence of some variables and not all that drive the randomness, such a method should allow us to work with simpler models as the transformed variables for the later trees should be rather independent. As we'll see in a later Section, this is particularly interesting in the context of truncated vines.

2.3.2.6 Vine copula models for high dimensional data

When estimating vine copulas, one of the main advantages of parametric over non-parametric models is that h-functions do not have to be estimated as they are directly obtained through the copula family density and its estimated parameters (see Aas, Czado, Frigessi, and Bakken (2009); Schepsmeier and Stöber (2014) for a list of h-functions for the main copula families). So even if there exist suitable and scalable non-parametric estimators for h-functions (e.g., Hall, Wolff, and Yao (1999)'s local linear estimator), the estimation of full vine copula models will always include an additional computational cost in the non-parametric case. This is well illustrated by Nagler (2014)'s experiment which shows the linear vs quadratic growth of computation time in the sample size n for the parametric vs kernel estimator. While the difference in computation times can be ignored for small and low dimensional data sets, it quickly widens as the sample size n and the dimension d increase. Considering our objective and assuming time is an important factor in practice, we believe the non-parametric approach for vine copula estimation might not be a viable option. Even in a parametric setting, the estimation of R-vine copula models for large and high dimensional data sets might be too time-consuming to be really useful in practice. This can be imputed to both the estimation and structure selection procedures.

Structure selection procedure Concerning the structure selection procedure, although Dissmann et al. (2013)'s approach is an heuristic, it can become quite expensive in terms of computational resources as the dimension d grows. That is because it requires to run Prim (1957)'s algorithm for each of the $d-1$ trees. Then the first way to reduce computation time for vine copulas estimation would be to use structure selection procedures with lower computational cost. To our knowledge, the only way to do that would be to use more restricted vine structures. Until now, we have focused our attention on the general regular vine, however, Bedford and Cooke (2001); Bedford et al. (2002)

have also identified two special cases of PCC models called drawable vine (D-vine) and canonical vine (C-vine). A R-vine tree is called

- D-vine tree if each node in T_i has at most 2 edges for $i = 1, \ldots, d-1$. The D-vine distribution of a d-dimensional random vector $\boldsymbol{X}$ may be written as

$$f(x_1, \ldots, x_d) = \prod_{i=1}^{d-1} \prod_{j=1}^{d-i} c_{j,j+i|j+1,\ldots,j+i-1} \left(F(x_j \mid x_{j+1}, \ldots, x_{j+i-1}), F(x_{j+i} \mid x_{j+1}, \ldots, x_{j+i-1})\right) \times \prod_{i=1}^{d} f_i(x_i),$$

  where i identifies the trees, j iterates the edges in each tree and the rightmost part is the density of the D-vine copula C (Aas et al., 2009).

- C-vine tree if each tree T_i has a unique node with $d-i$ edges for $i = 1, \ldots, d-1$. The node with $d-1$ edges in T_1 is called the root. The C-vine distribution of a d-dimensional random vector $\boldsymbol{X}$ may be written as

$$f(x_1, \ldots, x_d) = \prod_{i=1}^{d-1} \prod_{j=1}^{d-i} c_{i,i+j|1,\ldots,i-1} \left(F(x_i \mid x_1, \ldots, x_{i-1}), F(x_{i+j} \mid x_1, \ldots, x_{i-1})\right) \times \prod_{i=1}^{d} f_i(x_i),$$

  where i identifies the trees, j iterates the edges in each tree, x_1 is the root and the rightmost part is the density of the C-vine copula C (Aas et al., 2009).

As the dimension d grows, R-vines embrace a large number of possible pair-copula constructions. Because of their more restricted structure, both D and C-vines embrace a significantly lower number of possible pair-copula constructions. This has been illustrated by Aas et al. (2009) who showed that, among the 240 possible pair-copula decompositions for a five-dimensional density, there are 60 different D-vines, 60 different C-vines and 120 other R-vines. If we extend this idea and consider a d-dimensional density, we have the following

- For a d-dimensional D-vine, there are $d!$ possible ways of ordering the nodes in the tree T_1. Since we work with undirected edges (i.e., $c_{j(e),k(e)|D(e)} = c_{k(e),j(e)|D(e)}$ for all edges), we can conclude that there are only $d!/2$ different possible trees on the first level. Given the specific structure of D-vines, the trees $T_2, \ldots, T_{d-1}$ are

completely determined by the tree T_1. Consequently, there are $d!/2$ distinct D-vines on d nodes.

- For a d-dimensional C-vine, there are d possible ways of ordering the nodes in the tree T_1. Then, given the tree T_{i-1}, there are $d-(i-1)$ possible ways of ordering the nodes in the tree T_i for $i = 2, \ldots, d-2$. Consequently, there are $d!/2$ distinct C-vines on d nodes.

- Counting the number of regular vines is a more challenging task. Therefore, we refer to Morales Napoles, Cooke, and Kurowicka (2010) who showed there are $\binom{d}{2} \times (d-2)! \times 2^{\binom{d-2}{2}}$ regular vines on d nodes.

Thus, both D and C-vine copulas can be estimated using less computationally expensive automatic structure selection procedures than Dissmann et al. (2013)'s heuristic.

A D-vine copula model can be estimated as follows:

1. Build a complete graph with d nodes corresponding to each of the d variables.

2. Attribute a weight according to a measure of pairwise dependence between the two respective nodes i,j to all $d\,(d-1)\,/2$ edges: $1 - |\hat{\tau}_{i,j}|$.

3. Identify the shortest Hamiltonian path (i.e., the shortest path visiting each node exactly once) in terms of weights $1 - |\hat{\tau}_{i,j}|$ to determine the order of the nodes in the first tree T_1. Basically, this means solving the symmetric Traveling Salesman Problem (TSP). There are many TSP heuristic and approximation algorithms (e.g., see Rego, Gamboa, Glover, and Osterman (2011) for a review) but we chose to use the nearest neighbor algorithm (Rosenkrantz, Stearns, & Lewis, 1977) for convenience. The procedure can be summarized as follows

   a) Initialize a tree with a single arbitrarily chosen node from the graph built in (1).

   b) Select the minimum-weight edge from the edges connecting the current node to nodes not yet in the tree and transfer it to the tree.

   c) Repeat (b) until all nodes are in the tree.

   Note that in this case, the use of a greedy algorithm has a cost in terms of performance as the nearest-neighbor heuristic yields a path 25% longer than the

shortest possible path on average (Johnson & McGeoch, 1997). Moreover, the algorithm is also known to produce the unique worst possible solution for some specially arranged node distributions (Gutin, Yeo, & Zverovich, 2002). To partially overcome the problem, we finally decided to use the repetitive nearest neighbor algorithm (RNN). As its name suggests, the RNN algorithm constructs a nearest neighbor path for each node as the starting point and returns the shortest path found.

4. With the so-constructed tree T_1, obtain the D-vine tree sequence $\mathcal{V}$.

5. Estimate the complete D-vine copula model with the estimation procedure described in Subsection 2.3.2.4.

A C-vine copula model can be estimated as follows:

1. Build a complete graph with d nodes corresponding to each of the d variables for the first tree, build a graph based on the R-vine construction principles for the subsequent trees.

2. Attribute a weight according to a measure of pairwise dependence between the two respective nodes i,j to all edges: $|\hat{\tau}_{i,j}|$.

3. Identify the node with strongest dependencies to all other nodes as the root node to determine the order of the nodes in the tree. This can be done by taking the node with maximum weight (the weight of a node being the sum of the weights of its incident edges):
$$\max_{i \in \{1,\dots,d\}} \sum_{j \in \{1,\dots,d\} \setminus i} |\hat{\tau}_{i,j}| \,.$$

4. Obtain estimates for all pair-copulas corresponding to the edges of the tree and for all required h-functions and compute the pseudo-observations for the next level via the estimated h-functions.

5. Repeat the same procedure with the new $d-1$ variables until the complete C-vine copula model is estimated.

In short, the estimation process of a D-vine tree requires fewer computational resources because it uses a less or equally expensive algorithm and runs it only once as the first tree determines all the following trees. On the other hand, estimating a C-vine tree is

less computationally expensive as the order of the nodes in each tree is selected via a simple sum maximization. So, for high-dimensional problems, it might be reasonable to restrict to D and C-vine trees to reduce time complexity. However, this requires to make an additional assumption about the relationship between the variables, which may or may not be true. In practice, it was usually advised to fit a C-vine when a key variable is known to drive all other variables or a D-vine otherwise (Aas et al., 2009; Czado, 2010). Nonetheless, this advice might have become less relevant since the new sequential methods were discovered (Dissmann et al., 2013).

Estimation procedure The structure selection procedure set aside, high computation time for vine copulas estimation is mainly explained by the amount of computational effort required to estimate all parameters which grows exponentially with the dimension. That is because the estimation of a standard vine copula model involves the estimation of $l \times d\,(d-1)\,/2$ pair-copulas, where $d\,(d-1)\,/2$ is the number of pair-copulas composing the vine copula and l the size of the pool of tested copula families. In other words, the flexibility of vine copulas comes at the cost of a large number of model parameters. This poses two major challenges : the computational cost and the risk of overfitting. Fortunately, both issues can be handled by inducing sparsity. According to Nagler, Bumann, and Czado (2019), a vine copula model is called sparse "when a large number of pair-copulas are the independence copula". If we consider only the state-of-the-art solutions, there are two kinds of sparse vine copula models : truncated vine copula models (E. C. Brechmann, Czado, & Aas, 2012) and thresholded vine copula models Nagler et al. (2019). In this paper, we'll restrict ourselves to truncated vine copulas.

Truncated vines A truncated R-vine at level $K \in \{1, \ldots, d-1\}$, denoted $tV(K)$, is a R-vine copula where all pair-copulas $c_{j(e),k(e)|D(e)}$ with conditioning set $D(e)$ equal or larger than K are replaced by independence copulas (E. C. Brechmann et al., 2012). Thus, K-truncated R-vine copula models induce sparsity by setting the pair-copulas in the last $d-K-1$ trees of the vine to the independence copula. As the density of the independence copula is always 1, the density of a K-truncated R-vine copula model is given by

$$c_{tRV(K)}(\boldsymbol{u}) = \prod_{i=1}^{K} \prod_{e \in E_i} c_{j(e),k(e)|D(e)} \left(C_{j(e)|D(e)}(u_{j(e)} \mid \boldsymbol{u}_{D(e)}), C_{k(e)|D(e)}(u_{k(e)} \mid \boldsymbol{u}_{D(e)}) \right).$$

Depending on the truncation level K, the difference between the $d(d-1)/2$ pair-copulas composing the full vine copula and the $K(2d-K-1)/2$ pair-copulas left to estimate can be significant. Note that alternatively, one can use the Gaussian copula rather than the independence copula. In that case, we talk about simplified vines (E. C. Brechmann et al., 2012). Further note that truncated vines can be seen as a special case of simplified vines, using Gaussian copulas with correlation parameter equal to zero. However, we chose to focus on truncation rather than simplification. That is because it requires no estimation at all but also because it suits the idea behind Dissmann et al. (2013)'s heuristic better as the dependence in higher trees should be practically irrelevant.

Truncation level selection As the concept of truncated vines is fairly simple, the main challenge lies in identifying the most appropriate truncation level. The natural way to select the most appropriate truncation level would be to fit a K-truncated R-vine copula model $tRV(K)$ for each possible value of $K = 1, \ldots, d-1$ and select the best fitting model using statistical model selection techniques. However, this method might not be an attractive option when attempting to reduce the computational cost as it requires to estimate the full R-vine copula model. Alternatively, the truncation level can be selected using a stepwise strategy highly inspired by the stepwise semi-parametric estimator proposed by Aas et al. (2009) and formally introduced by Haff et al. (2013):

1. Start with $K = 1$.
2. Fit the $tRV(K)$ model.
3. Fit the new $tRV(K+1)$ model by fitting an extra tree.
4. Assess the gain from moving from model $tRV(K)$ to model $tRV(K+1)$
5. If the gain is sufficient:
   a) Increase K by one.
   b) If $K < d-1$, continue with 3. Else, declare $tRV(K)$ as the best fitting model among the nested model sequence $tRV(j)$, $j = 1, \ldots, d-1$.

   If the gain is not sufficient, declare $tRV(K)$ as the best fitting model among the nested model sequence $tRV(j)$, $j = 1, \ldots, d-1$.

Note that it is also possible to use a computation time limit as stopping criteria. The main advantage of such a sequential approach is that the amount of computational resources spent on model fitting and selection is comparable to the one required for fitting the best fitting model, not the full model. This means that, when K is small, one can expect to achieve significant time savings. Several statistical model selection techniques can be used to assess whether there is a gain to move from model $tRV(K)$ to model $tRV(K+1)$. For example, since $tRV(K)$ is nested within $tRV(K+1)$, the marginal gain provided by fitting an extra tree can be quantified by the difference in terms of AIC or BIC values between the two models. Alternatively, the model can be selected based on a likelihood-ratio-based test called the Vuong closeness test (Vuong, 1989). In their extensive simulation studies, E. C. Brechmann et al. (2012) showed that the Vuong test-based procedure is significantly more performant than the AIC/BIC-based procedures as the latter tend to select models that are too large. However, the studies also showed the Vuong test-based procedure is considerably slower. For further details on these procedures, we refer to E. C. Brechmann et al. (2012).

In this paper, we decided use a procedure based on a modified Bayesian Information Criterion specifically adapted for sparse Vine copula models (mBICV) recently proposed by Nagler et al. (2019). The modification brought by the mBICV has two objectives. The first one is to take out the assumption that all models are equally likely a priori and set higher prior probabilities for sparse models. That is because under this assumption, only half of the $d\,(d-1)\,/2$ pair-copulas are expected to be independence and the resulting model could not really be described as sparse. The second one is to relax the conditions under which the criterion can consistently distinguish between the true and alternative models. As mentioned in Nagler et al. (2019), the BIC is not able to distinguish the true model from an incorrect model when the number of parameters grows too fast with the sample size n, i.e., $d \geq \sqrt[4]{n \ln n}$ if we assume there is only one parameter per pair-copula. Yet, the authors argue that "the mBICV can consistently distinguish between a finite number of models provided that $d = o(\sqrt{n})$" (Nagler et al., 2019, p. 7). The mBICV for $tRV(K)$ is given by

$$\begin{aligned} mBICV(tRV(K)) = &-2 \times l_{tRV(K)}(\boldsymbol{\theta}_{tRV(K)} \mid \boldsymbol{u}) + \ln(n) \times n_{tRV(K)} \\ &- 2 \times \sum_{i=1}^{d-1} \left\{ q_i \times \ln(\psi_i) + (d - m - q_i) \times \ln(1 - \psi_i) \right\}, \end{aligned}$$

where $l_{tRV(K)}(\boldsymbol{\theta}_{tRV(K)} \mid \boldsymbol{u})$ denotes the log-likelihood, $n_{tRV(K)}$ the number of effective parameters, q_i the number of non-independence copulas in tree T_i and ψ_i the prior

probability of non-independence copula in tree T_i.

3 A Copula-based Bayes classifier for high-dimensional classification

We can now define our copula-based Bayes classifier. We propose a new tree-based classifier extending the naive Bayes by evaluating the dependence structure through pair-copula constructions: the truncated R-Vine Copula Bayes classifier (tRVCB). Hereafter, we define the tRVCB classifier, describe its learning process and compare it with some existing copula-based classifiers on some of its properties.

3.1 Definition

The proposed classifier use a probability model derived upon equation 2.1 where the copula density function is modeled and estimated via a truncated R-vine. Consequently, the discriminant functions used in the tRVCB classifier are given by

$$\begin{aligned} f_k^{tRVCB}(\boldsymbol{x}) &= \Pr(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) \\ &\propto \prod_{i=1}^{K} \prod_{e \in E_i} c_{j(e),k(e)|D(e);k} \left(F(x_{j(e)} \mid \boldsymbol{x}_{D(e)}; k), F(x_{k(e)} \mid \boldsymbol{x}_{D(e)}; k); k \right) \\ &\times \prod_{i=1}^{d} f_i(x_i \mid k) \times \Pr(Y = k). \end{aligned}$$

Similar to any Bayesian classifiers, when hard classification is required, the probability model is combined with the MAP decision rule and the classifier assigns the class with maximum discriminant function:

$$h(\boldsymbol{x}) = argmax_{k \in \{0,1\}} f_k^{tRVCB}(\boldsymbol{x}).$$

Note that, when the truncation level K is set to zero, equation **??** reduces to the naive Bayes model:

$$\Pr(Y = k \mid \boldsymbol{X} = \boldsymbol{x}) \propto \prod_{i=1}^{d} f_i(x_i \mid k) \cdot \Pr(Y = k).$$

Therefore, the naive Bayes can be considered as a special case of the tRVCB.

3.2 Learning

The naive Bayes probability model Given a training set $\{(\boldsymbol{x}^{(m)}, y^{(m)}), m = 1, \ldots, n\}$ where $\boldsymbol{X} \in \mathbb{R}^{\mathrm{d}}$ and $Y \in \{0, 1\}$, we start by estimating the class prior probabilities $\Pr(Y = k)$ for $k = 0, 1$ by the proportion of training instances belonging to each class:

$$\hat{\Pr}(Y = k) = \frac{1}{n} \sum_{m=1}^{n} \mathbb{I}(y^{(m)} = k) = \frac{n_k}{n}.$$

Then, using the *density* function from base R (R Core Team, 2019), the marginal densities $f_i(x_i \mid k)$ are estimated for $i = 1, \ldots, d$ and $k = 0, 1$ by using a kernel-based approach:

$$\hat{f}_i(x \mid k) = \frac{1}{n_k h_{i,k}} \sum_{m:y^{(m)}=k}^{n_k} K\left(\frac{x - x^{(m)}}{h_{i,k}}\right),$$

where K is the Gaussian kernel function and $h_{i,k}$ is the bandwidth estimated via Silverman's rule of thumb (Silverman, 1986, p.48). We chose a plug-in bandwidth estimator to avoid having hyper-parameters to tune. For more details on kernel density estimation, see Subsection 2.2.1.2.

Pseudo-observations Next, the pseudo-observations $\{\hat{u}_i^{(m)}, m = 1, \ldots, n\}$ are computed for $i = 1, \ldots, d$ using the rank-based method on the full training set:

$$\hat{u}_i^{(m)} = \frac{r_i^{(m)}}{n + 1},$$

where $r_i^{(m)}$ is the rank for $x_i^{(m)}$. To do this, we used the *pseudo_obs* function from the *rvinecopulib* R package (Nagler & Vatter, 2019). For more details on the rank-based method, see Subsection 2.2.1.2.

The truncated R-vine copula model Then, using the *vinecop* function from the *rvinecopulib* R package (Nagler & Vatter, 2019), a truncated R-vine copula model $tRV_k(K)$ is fitted for $k = 0, 1$. Unlike Salinas-Gutiérrez et al. (2010), we decided to evaluate the conditional dependence structure between the features with Vine copula models rather than standard multivariate copula models such as the multivariate Gaussian copula. That is because standard multivariate copulas can become quite inflexible and often lack the ability of accurately modeling dependence structures in high-dimensional settings. Furthermore, we chose to use R-vine structures rather than C or D-vine structures (e.g., see Y. Chen (2016) for a D-vine copula based classifier) in order to minimize the set of assumptions on the conditional dependence structure between the features. The structure of the R-vines are obtained using Dissmann et al. (2013)'s heuristic (see Subsection 2.3.2.5). Within each tree, estimation of pair-copulas is done parallelly for computational efficiency. The truncation level K is automatically selected using a stepwise strategy based on the mBICV as described in Subsection 2.3.2.6. To the best of our knowledge, we are the first to propose the use of truncated vines in supervised classification. We believe this constitutes an important step towards the development of effective high-dimensional copula-based classifiers since truncated vines allow to simultaneously reduce the amount of computational resources and the risk of overfitting.

Pair-copulas For each pair-copula, six parametric families are considered, namely the Gaussian, Student, Clayton, Gumbel, Frank and Joe copulas (see Subsection 2.1.3). For the Clayton, Gumbel, Frank and Joe copulas, their rotated versions are also included to make them able to cover negative dependence as well (see Subsection 2.1.3.4). First, all available copulas are fitted using the Kendall's τ inversion-based estimation method (see Subsection 2.2.2.1). Then, the AIC is computed for all fitted copulas and the copula model with minimum AIC is chosen (see Subsection 2.2.2.2).

4 Experimental results

4.1 Experimental setup

To assess the merit of our (tRVCB) model, we compare it to the naive Bayes classifier (NB) and two other copula-based classifiers: the Multivariate Gaussian Copula Bayes classifier (MGC) and the truncated D-Vine Copula Bayes classifier (tDVCB). As their name suggest, the MGC evaluates the dependence structure with a multivariate Gaussian copula model (see Subsection 2.1.3.1) while the tDVCB evaluates it with a truncated D-vine copula model (see 2.3.2.6). The objective behind this comparison is to investigate how the strength of the assumptions on the conditional dependence structure between the features impacts the performance and the computational cost of copula-based Bayes classifiers. As in practice, running time matters almost as much as performance, we are particularly interested in identifying the best copula model in terms of balanced speed and performance. However, the task is not trivial as the performance of a classifier highly depends on the characteristics of the problem/data such as the sample size, the number of features, the complexity of the decision boundary, etc.

To do that, we evaluate all methods on eighteen simulated balanced data sets differing in terms of sample size $n \in \{10k, 20k, 30k, 40k, 50k\}$, number of features $d \in \{10, 20, 30, 40, 50\}$ and decision boundary complexity $s \in \{easy, hard\}$. Each generated data set has d features among which $d_{inf} = 2d/5$ are informative, $d_{red} = d/5$ are redundant (i.e., random linear combinations of the informative features) and $2d/5$ are filled with random noise. Each generated data set has $n/100$ instances with randomly assigned classes in order to introduce some noise. Half of the data sets are "easily" separable given the informative features while the other half is not. The data sets have been generated using the *make_classification* function from the *scikit-learn* Python library (Pedregosa et al., 2011). The idea behind the underlying algorithm is that the two classes are composed of multiple clusters (two in our case) located around the vertices of a d_{inf}-dimensional hypercube. Within each cluster, informative features are randomly

drawn from a standard Normal distribution before to be randomly linearly combined to add covariance. When the process is done, the clusters are relocated on the vertices of the hypercube. The difficulty of the classification task is mainly controlled by a factor multiplying the size of the d_{inf}-dimensional hypercube. In our case, we used the default value 1.0 to generate the "easy classification task" data sets and 0.5 to generate the "hard classification task" data sets. For more details on the the underlying algorithm, we refer to Guyon, Gunn, Hur, and Dror (2006). For each data set, we perform a 10-fold cross-validation and report the averaged test results.

4.2 Performance

The predictive ability of a classification algorithm can be evaluated by a large set of metrics but is typically measured by its predictive accuracy on the testing instances. This seems to be particularly true in the context of copula-based classifiers as we note that all existing copula-based classifiers have been evaluated using predictive accuracy (e.g., Y. Chen, 2016; Elidan, 2012; Han et al., 2013; Pérez Díaz, 2018; Salinas-Gutiérrez et al., 2010). As a reminder, accuracy measures, for a given threshold, the percentage of correctly classified instances regardless of which class they belong to. From its definition, it is easy to see that accuracy present several important limitations (Huang, Lu, & Ling, 2003). One of the most important being that accuracy varies from different thresholds. As the classifiers we wish to compare are binary classifiers and naturally probabilistic (in the sense that they produce probability estimations of the class prediction), we have a better measure at our disposal : the Area Under the Receiver Operating Characteristic Curve (ROC AUC). In short, the ROC AUC, often simply referred to as AUC, measures the likelihood that given an instance from the positive class and an instance from the negative one, the classifier will rank the instance from the positive class higher than the instance from the negative one (Bradley, 1997). In theory, AUC is a better measure than accuracy as it has been proved to be statistically consistent and more discriminant than accuracy (Ling, Huang, Zhang, et al., 2003). In practice, the advantage of AUC is that it measures ranking and does not depend on thresholds.

Therefore, the average AUC and its standard deviation for all methods and all data sets are reported in Table 4.2. Note that two additional models have been included in the comparison. That is because we noticed the automatic truncation level selection procedure used in our tRVCB and tDVCB models tends to select abnormally high

	N	D	Difficulty	tDVCB	tRVCB
1	10000	10	easy	2.30	1.50
2	10000	10	hard	5.00	4.80
3	10000	20	easy	10.80	11.00
4	10000	20	hard	10.40	11.20
5	10000	30	easy	16.40	18.60
6	10000	30	hard	20.40	21.50
7	10000	40	easy	22.40	25.60
8	10000	40	hard	22.40	26.00
9	10000	50	easy	29.80	40.40
10	10000	50	hard	28.60	39.20
11	20000	10	easy	5.00	5.40
12	20000	10	hard	5.00	5.00
13	30000	10	easy	4.80	4.00
14	30000	10	hard	5.00	5.00
15	40000	10	easy	4.10	4.00
16	40000	10	hard	5.40	5.00
17	50000	10	easy	5.00	4.70
18	50000	10	hard	5.00	5.00

Table 4.1: Automatically selected truncation levels in the tDVCB and tRVCB models averaged over the 2 classes and the 10 test folds for the 18 data sets.

truncation levels (see Table 4.1). We strongly suspect it has something to do with the implementation of the automatic truncation level selection procedure in the *vinecop* function from the *rvinecopulib* R package (Nagler & Vatter, 2019) as the mBICV has been shown to select satisfactory truncation levels in high-dimensional settings (Nagler et al., 2019). An better solution would have been to compare the truncation levels obtained using different selection criteria (e.g., AIC, BIC, Vuong closeness test, etc. (see Subsection 2.3.2.6)). However, this would have meant to give up the high-performance implementations provided by the *rvinecopulib* package. The two additional models consist in simple variants of the tRVCB and tDVCB models where the truncation levels are arbitrarily fixed to $d/5$ instead of being automatically selected. Therefore, they'll be referred to as ftRVCB and ftDVCB in the remainder of this paper.

The first thing to notice is that the estimates of the model's predictive performance obtained on the 10 test folds are stable for all our models and all the data sets. That's particularly important since we couldn't compute repeated cross-validation in order to reduce the variability of the estimates due to time constraints. As expected, the mean AUC scores of each classifier are always lower on the "hard classification task" data sets

	NB	MGC	ftDVCB	ftRVCB	tDVCB	tRVCB
1	0.99 (0.003)	0.94 (0.011)	0.98 (0.004)	0.98 (0.004)	0.98 (0.005)	0.98 (0.004)
2	0.91 (0.009)	0.85 (0.012)	0.93 (0.014)	0.91 (0.008)	0.9 (0.013)	0.9 (0.008)
3	0.95 (0.011)	0.85 (0.015)	0.98 (0.004)	0.98 (0.004)	0.98 (0.004)	0.98 (0.004)
4	0.83 (0.012)	0.78 (0.031)	0.96 (0.004)	0.95 (0.004)	0.95 (0.005)	0.95 (0.006)
5	0.91 (0.009)	0.72 (0.025)	0.97 (0.005)	0.98 (0.004)	0.98 (0.006)	0.98 (0.004)
6	0.8 (0.014)	0.68 (0.042)	0.96 (0.005)	0.96 (0.006)	0.96 (0.008)	0.96 (0.005)
7	0.93 (0.007)	0.85 (0.017)	0.99 (0.004)	0.98 (0.005)	0.99 (0.004)	0.99 (0.004)
8	0.8 (0.016)	0.71 (0.033)	0.97 (0.006)	0.97 (0.009)	0.96 (0.007)	0.96 (0.007)
9	0.91 (0.009)	0.76 (0.017)	0.99 (0.002)	0.99 (0.002)	0.99 (0.002)	0.99 (0.002)
10	0.8 (0.013)	0.73 (0.022)	0.99 (0.002)	0.98 (0.003)	0.99 (0.001)	0.99 (0.003)
11	0.95 (0.005)	0.95 (0.004)	0.95 (0.004)	0.95 (0.005)	0.95 (0.005)	0.95 (0.005)
12	0.86 (0.01)	0.83 (0.012)	0.88 (0.011)	0.88 (0.008)	0.86 (0.009)	0.87 (0.01)
13	0.96 (0.004)	0.92 (0.003)	0.96 (0.005)	0.96 (0.003)	0.96 (0.004)	0.96 (0.003)
14	0.86 (0.007)	0.71 (0.014)	0.87 (0.009)	0.86 (0.015)	0.83 (0.011)	0.84 (0.013)
15	0.98 (0.002)	0.89 (0.008)	0.97 (0.003)	0.97 (0.003)	0.96 (0.003)	0.96 (0.002)
16	0.87 (0.005)	0.8 (0.007)	0.86 (0.005)	0.87 (0.003)	0.86 (0.01)	0.87 (0.005)
17	0.96 (0.002)	0.86 (0.013)	0.96 (0.004)	0.96 (0.002)	0.96 (0.003)	0.96 (0.005)
18	0.83 (0.005)	0.75 (0.011)	0.87 (0.003)	0.88 (0.003)	0.87 (0.004)	0.87 (0.007)

Table 4.2: 10-fold test AUC score (mean and standard deviation) for the different models on the 18 data sets.

than on the "easy classification task" data sets when controlling for the sample size and the number of features.

	NB	MGC	tRVCB
1	0.5 (0.015)	1 (0)	1 (0)

Table 4.3: 10-fold test AUC score (mean and standard deviation) for the NB, MGC and tRVCB models on a data set with normally distributed features given the class.

On a side note, the MGC model is the worst performing classifier across all data sets. That can be explained by the fact that in our simulated data sets, the interactions between the features given the class are far from Gaussian. We quickly verified that assumption by generating a balanced binary classification data set with 10k instances and 20 highly correlated features normally distributed given the class (i.e., $(\boldsymbol{X} \mid Y = k) \sim \mathcal{N}_{20}(\mathbf{0}, \boldsymbol{\Sigma}_k)$) and performing a 10-fold cross-validation. The obtained results, reported in Table 4.3, confirm it as both the MGC and tRVCB models predict 100% of the testing instances correctly while the NB classifier performs poorly. If we also consider the fact that by definition, the NB and MGC models should perform equally well when the conditional

independence assumption is valid, we believe choosing the MGC model over the NB model might offer some gains in terms of performance in very specific situations only (i.e., when there is some dependence between the features given the class and the class-conditional probability distributions are not too far from normal).

Next, we note that all the truncated vine copula-based models show similar AUC scores on all the data sets, no matter the type of vine tree and the truncation level selection procedure. The similar performances of the four models show that D-vine copulas are already flexible enough to model almost any dependence structure and that only a small amount of nested trees is necessary to capture the most important part of the dependence structure in high-dimensional settings. We believe that only the large amount of training instances prevented the tDVCB and tRVCB models from overfitting. That being said, the truncated vine copula-based models are the best performing models overall (on 15 out of the 18 data sets) but their advantage relative to the naive Bayes is not always substantial, especially on "easy classification task" and/or low-dimensional data sets. Indeed, the improvements in terms of AUC vary from none to .19.

After conducting a deeper analysis, we see an interesting pattern emerging. The latter is shown in Figure 4.1. If we consider only the ten first data sets which differ only in the number of features and the complexity of the decision boundary, we see that the gap in terms of AUC scores between the truncated vine copula-based models and the naive Bayes model tend to widen as the number of features increases. Furthermore, the wideness of the gap seems to evolve faster when the decision boundary is complex. Obviously, we would have to make sure the observed results are not due to unconsidered particular characteristics of the data sets in order to draw any safe conclusion. Therefore, larger simulation studies have to be conducted. However, the obtained results for the naive Bayes classifier make sense. First, it is known that adding completely uninformative or redundant features decreases the predictive ability of the naive Bayes classifier (Ng & Jordan, 2002). Since our data sets always contain $d/5$ redundant features and $2d/5$ supposedly uninformative features, data sets with higher number of features d also have higher number of redundant/uninformative features. That could explain why the naive Bayes model's performance decreases when the dimension increases. Additionally, the naive Bayes model, with its restricted number of parameters, might not be powerful enough and produce decision rules that are too simple when the true decision boundary is complex.

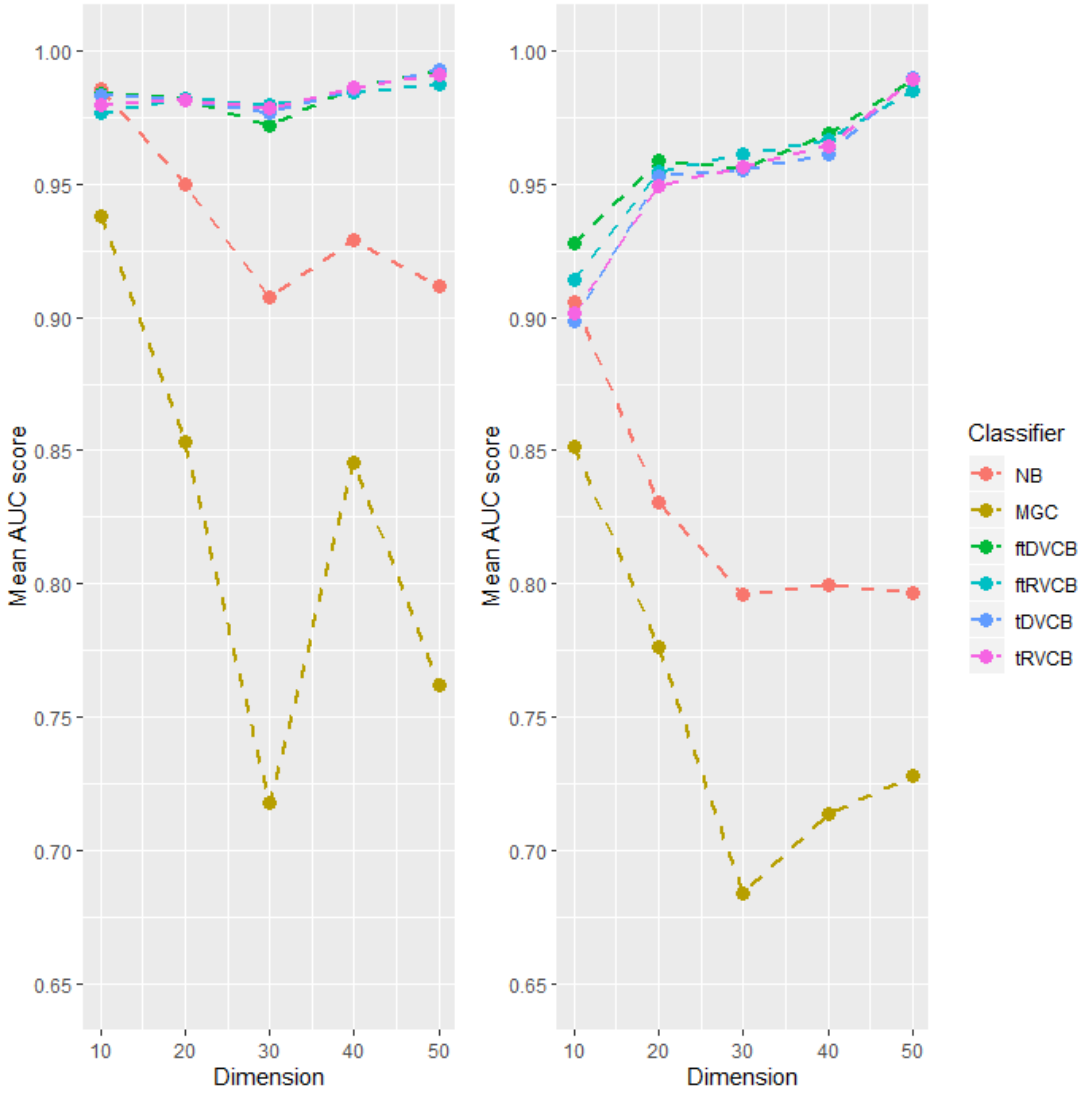

Figure 4.1: AUC mean scores vs Number of features for the different models on the data sets with sample size $n = 10k$. The left graph consist of "easy classification task" data sets, the right graph consist of "hard classification task" datasets.

In summary, the truncated vine copula-based models (i.e., the ftDVCB, ftRVCB, tDVCB and tRVCB models), which show similar performances, perform better than the baseline NB classifier in most of the cases (15/18 data sets). The difference in terms of AUC scores varies from 0 to .19. This shows that taking the dependence structure into account when the conditional independence assumption is violated might be highly beneficial depending on the particular characteristics of the data to be classified. When evaluating

the dependence structure, vine copulas are a better alternative than multivariate Gaussian copulas since the MGC model might perform quite poorly when the Gaussian assumption is violated.

4.3 Running time

As all our presented copula-based models extend the naive Bayes model (i.e., the prior probabilities and the marginal class-conditional probability distributions are also estimated in order to make predictions) by fitting a copula model for each class. Therefore, they are obviously more computationally expensive options. But to what extent? To answer that question, we computed the mean user CPU time across the 10 test folds for each model on each data set. As the complexity of the decision boundary does not matter in this case, we further averaged the results over the "easy classification task" and "hard classification task" data sets for each $\{n, d\}$ combinations. The computations were conducted on a customary Windows 10 PC with AMD Ryzen 5 3600X CPU @ 3.79 GHz and 16 GB RAM. The results are shown in Figure 4.2. Note that the reported times include both the training and classification processes.

As expected for one of the fastest classification algorithm, the naive Bayes model is extremely fast and highly scalable. The same applies to the MGC model which shows only slightly larger computation times. Again, this is not surprising since fitting a Gaussian copula can be carried out in closed form. More interestingly, Figure 4.2 shows that only the truncation level of the fitted truncated vine copula model has a significant impact the computation time. Even if the time grows approximately linearly in the sample size n and approximately quadratrically in the dimension d for both the $d/5$ and the mBICV-optimal truncated vine copula-based models, the ftDVCB and ftRVCB models are at least twice as fast as the tDVCB and tRVCB models when controlling for n or d.

Ultimately, time is an important factor in practice. In this regard, the truncated vine copula-based models have a clear disadvantage over the computationally more simple naive Bayes model. However, this is the case for most classification algorithms and in the end, the time required for computing the slowest of our models should be in the order of minutes in most practical situations.

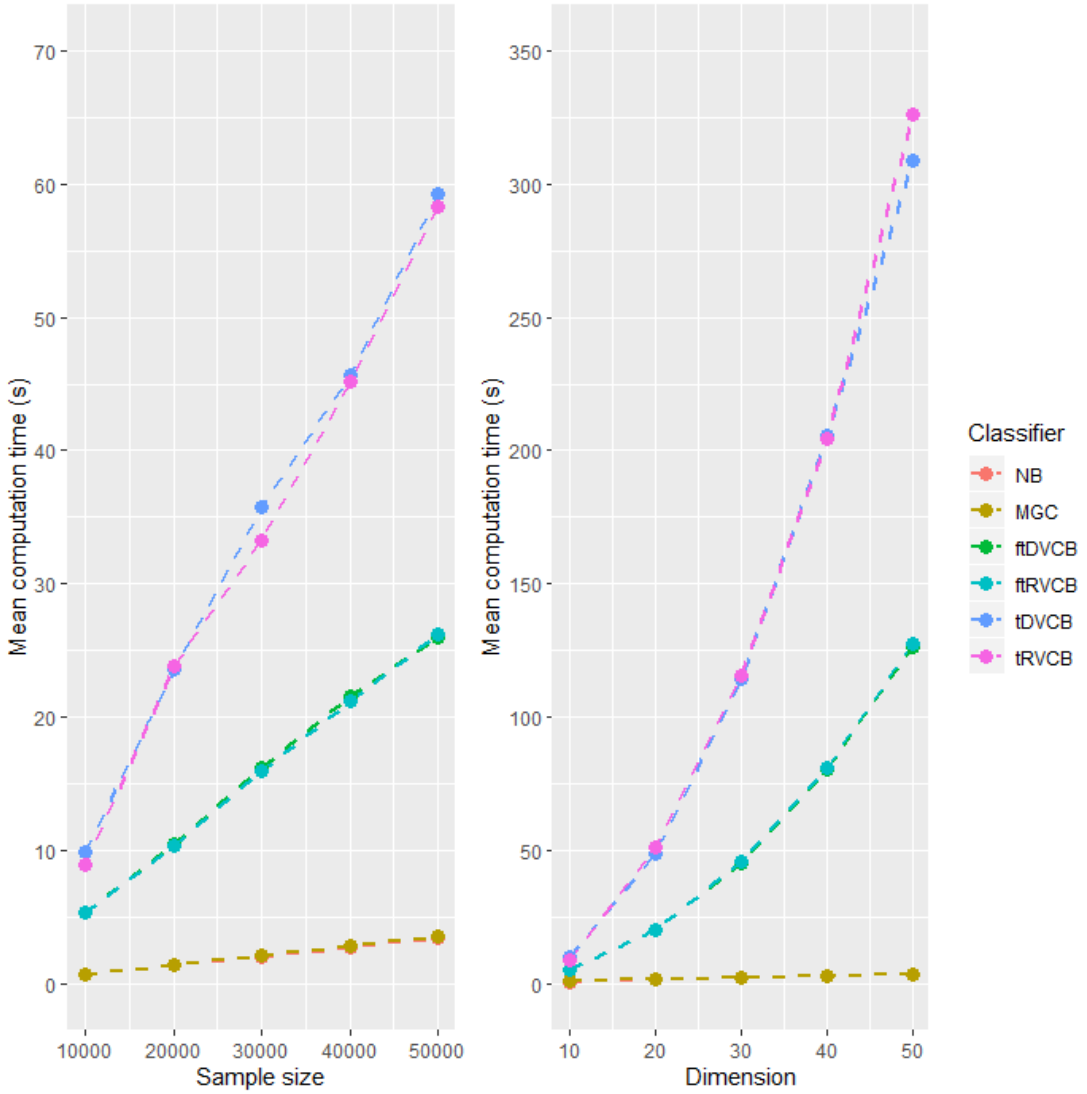

Figure 4.2: Mean computation time for varying sample size when $d = 10$ (left graph) and Mean computation time for varying number of features when $n = 10k$ (right graph) of the different models

Conclusion

The objective of this paper was to address the problem of learning effective high-dimensional binary classifiers in domains with continuous features. Given the large set of existing classification algorithms, we chose to focus on the Bayesian framework. In this context, we saw that existing Bayesian classifiers are already highly competitive depending on the characteristics of the dependence structure between the features given the class. On one hand, the traditional naive Bayes model performs amazingly well when there is little to no interaction and can even compete with more complex models when the conditional independence assumption is violated (under certain specific conditions). On the other hand, network-based models such as the TAN and BAN classifiers (or standard multivariate copula-based models such as the MGC (Salinas-Gutiérrez et al., 2010)) constitute good options when the interactions are Gaussian.

However, more often than not, the interactions are non-Gaussian and the mentioned classifiers may show suboptimal performances. To overcome this limitation (i.e., relax the conditional independence and Gaussian assumptions), Y. Chen (2016) presented the D-vine copula Bayes classifier, a tree-based classifier extending the naive Bayes by evaluating the dependence structure with a D-vine copula model. The classifier delivers satisfactory performances on small n moderate d data sets when compared with the naive Bayes and TAN models. However, despite their claim, we strongly believe the classifier is not tailored for high-dimensional and large sample size data because its learning process requires to fit a full D-vine copula model per class. Yet, we know that full vine copula models are computationally expensive and prone to overfitting. To overcome these limitations, we present the truncated R-vine copula Bayes classifier (tRVCB). As its name suggests, the tRVCB evaluates the dependence structure with a truncated R-vine copula model. Relative to D-vine copula models, R-vine copula models present the advantage of not imposing any restrictions on the dependence structure. Additionally, the computational price of this increased flexibility has become negligible since vine copula models are fitted using Dissmann et al. (2013)'s heuristic. But the

most innovative feature of the tRVCB model is the use of truncated vines which allows to significantly reduce the amount of necessary computation resources and potentially reduce the risk of overfitting.

We assessed the merits of our model by comparing it with the naive Bayes (NB), multivariate Gaussian copula Bayes (MGC) and truncated D-vine copula Bayes (tDVCB) models on 18 simulated data sets. As we were not satisfied with the automatically selected truncation levels which we found to be abnormally high, we also included two truncated vine copula-based models with truncation levels arbitrarily set to $d/5$, namely the ftDVCB and ftRVCB models. We demonstrated the consistent predictive effectiveness of our model relative to the naive Bayes and MGC models. We also showed that the type of structure of the vine copula has no impact neither on the predictive ability nor on the computation time while the truncation level only impacts the computation time. However, this last statement has to be taken with a grain of salt as it might be due to the particular characteristics of our data. For example, it might be possible that the important size of our training sets prevented the large truncation level models from overfitting and that the truncation levels were not low enough to cause the models to underfit. It is also possible that the true decision boundaries were not complex enough to highlight the advantage of R-vine structures over D-vine structures. Thus, the comparison has to be conducted on a wider variety of data sets.

All in all, we believe the tRVCB model constitutes a solid option for performing binary classification given continuous features with non-Gaussian interactions. It is obviously computationally expensive when compared with the naive Bayes and the MGC models but is still tolerable in most practical situations. This is because it has been specifically designed to be tailored for high-dimensional and large sample size data. Nonetheless, an important problem remains: how can one select an appropriate truncation level K? Setting an arbitrarily chosen value between 1 and $d-1$ is obviously not appropriate and the automatic truncation level selection procedure based on the mBICV criterion seems to select models that are too large for an unknown reason. Of course, one could consider the truncation level as a hyper-parameter to be tuned but the computational cost would explode as the nested cross-validation procedure is much more demanding than the flat one. Thus, future work should focus on finding appropriate automatic truncation level selection procedures in the context of supervised classification.

Bibliography

Aas, K., Czado, C., Frigessi, A., & Bakken, H. (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and economics*, *44*(2), 182–198.

Akaike, H. (1974). A new look at the statistical model identification. In *Selected papers of hirotugu akaike* (pp. 215–222). Springer.

Bedford, T., & Cooke, R. M. (2001). Probability density decomposition for conditionally dependent random variables modeled by vines. *Annals of Mathematics and Artificial intelligence*, *32*(1-4), 245–268.

Bedford, T., Cooke, R. M., et al. (2002). Vines–a new graphical model for dependent random variables. *The Annals of Statistics*, *30*(4), 1031–1068.

Bickel, P. J., Levina, E., et al. (2004). Some theory for fisher's linear discriminant function,naive bayes', and some alternatives when there are many more variables than observations. *Bernoulli*, *10*(6), 989–1010.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, *30*(7), 1145–1159.

Brechmann, E. (2010). Truncated and simplified regular vines and their applications.

Brechmann, E. C., Czado, C., & Aas, K. (2012). Truncated regular vines in high dimensions with application to financial data. *Canadian Journal of Statistics*, *40*(1), 68–85.

Charpentier, A., Fermanian, J.-D., & Scaillet, O. (2007). The estimation of copulas: Theory and practice.

Chen, S. X. (1999). Beta kernel estimators for density functions. *Computational Statistics & Data Analysis*, *31*(2), 131–145.

Chen, Y. (2016). A copula-based supervised learning classification for continuous and discrete data. *Journal of Data Science*, *14*(4), 769–782.

Cheng, J., & Greiner, R. (2001). Learning bayesian belief network classifiers: Algorithms and system. In *Conference of the canadian society for computational studies of intelligence* (pp. 141–151).

Cherubini, U., Luciano, E., & Vecchiato, W. (2004). *Copula methods in finance*. John Wiley & Sons.

Clayton, D. G. (1978). A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence. *Biometrika*, *65*(1), 141–151.

Czado, C. (2010). Pair-copula constructions of multivariate copulas. In *Copula theory and its applications* (pp. 93–109). Springer.

Czado, C., Jeske, S., & Hofmann, M. (2013). Selection strategies for regular vine copulae. *Journal de la Société Française de Statistique*, *154*(1), 174–191.

Czado, C., Schepsmeier, U., & Min, A. (2012). Maximum likelihood estimation of mixed c-vines with application to exchange rates. *Statistical Modelling*, *12*(3), 229–255.

Dalla Valle, L. (2009). Bayesian copulae distributions, with application to operational risk management. *Methodology and Computing in Applied Probability*, *11*(1), 95–115.

Deheuvels, P. (1980). Non parametric tests of independence. In J.-P. Raoult (Ed.), *Statistique non paramétrique asymptotique* (pp. 95–107). Berlin, Heidelberg: Springer Berlin Heidelberg.

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

Dissmann, J., Brechmann, E. C., Czado, C., & Kurowicka, D. (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, *59*, 52–69.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, *29*(2-3), 103–130.

Doukhan, P., Fermanian, J.-D., Lang, G., et al. (2005). *Copulas of a vector-valued stationary weakly dependent process*. INSEE.

Dutta, S. (2015). Local smoothing for kernel distribution function estimation. *Communications in Statistics - Simulation and Computation*, *44*(4), 878-891. Retrieved from `https://doi.org/10.1080/03610918.2013.795591` doi: 10.1080/03610918.2013.795591

Elidan, G. (2012). Copula network classifiers (cncs). In *Artificial intelligence and statistics* (pp. 346–354).

Embrechts, P., Lindskog, F., & McNeil, A. (2001). Modelling dependence with copulas. *Rapport technique, Département de mathématiques, Institut Fédéral de Technologie de Zurich, Zurich*.

Epanechnikov, V. (1969). Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, *14*(1), 153-158. Retrieved from

https://doi.org/10.1137/1114019 doi: 10.1137/1114019

Fang, H.-B., Fang, K.-T., & Kotz, S. (2002). The meta-elliptical distributions with given marginals. *Journal of Multivariate Analysis*, *82*(1), 1–16.

Fermanian, J.-D., & Scaillet, O. (2003). Nonparametric estimation of copulas for time series. fame research paper series. *International Center for Financial Asset Management and Engineering*, *73*.

Frank, M. J. (1979). On the simultaneous associativity off (x, y) andx+y- f (x, y). *Aequationes mathematicae*, *19*(1), 194–226.

Frees, E. W., & Valdez, E. A. (1998). Understanding relationships using copulas. *North American actuarial journal*, *2*(1), 1–25.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, *29*(2-3), 131–163.

Garcia, R., & Tsafack, G. (2011). Dependence structure and extreme comovements in international equity and bond markets. *Journal of Banking & Finance*, *35*(8), 1954–1970.

Geenens, G., Charpentier, A., Paindaveine, D., et al. (2017). Probit transformation for nonparametric kernel estimation of the copula density. *Bernoulli*, *23*(3), 1848–1873.

Genest, C., & Rivest, L.-P. (1989). A characterization of gumbel's family of extreme value distributions. *Statistics & Probability Letters*, *8*(3), 207–211.

Genz, A., & Bretz, F. (2009). *Computation of multivariate normal and t probabilities* (Vol. 195). Springer Science & Business Media.

Gijbels, I., & Mielniczuk, J. (1990). Estimating the density of a copula function. *Communications in Statistics-Theory and Methods*, *19*(2), 445–464.

Grossman, D., Domingos, P., & Domingos, P. (2004). Learning bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the twenty-first international conference on machine learning* (p. 46).

Gudendorf, G., & Segers, J. (2010). Extreme-value copulas. In *Copula theory and its applications* (pp. 127–145). Springer.

Gumbel, E. J. (1960). Bivariate exponential distributions. *Journal of the American Statistical Association*, *55*(292), 698–707.

Gutin, G., Yeo, A., & Zverovich, A. (2002). Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the tsp. *Discrete Applied Mathematics*, *117*(1-3), 81–86.

Guyon, I., Gunn, S., Hur, A. B., & Dror, G. (2006). Design and analysis of the nips2003 challenge. In *Feature extraction* (pp. 237–263). Springer.

Haff, I. H., Aas, K., & Frigessi, A. (2010). On the simplified pair-copula construction-

simply useful or too simplistic? *Journal of Multivariate Analysis*, *101*(5), 1296–1310.

Haff, I. H., et al. (2013). Parameter estimation for pair-copula constructions. *Bernoulli*, *19*(2), 462–491.

Hall, P., Wolff, R. C., & Yao, Q. (1999). Methods for estimating a conditional distribution function. *Journal of the American Statistical association*, *94*(445), 154–163.

Han, F., Zhao, T., & Liu, H. (2013). Coda: High dimensional copula discriminant analysis. *Journal of Machine Learning Research*, *14*(Feb), 629–671.

Härdle, W., & Linton, O. (1994). Applied nonparametric methods. *Handbook of econometrics*, *4*, 2295–2339.

Hazewinkel, M. (1994). *Encyclopaedia of mathematics (set)*. Springer Netherlands. Retrieved from `https://books.google.be/books?id=2-z0tAEACAAJ`

Hellerstein, J. L., Jayram, T., Rish, I., et al. (2000). *Recognizing end-user transactions in performance management*. IBM Thomas J. Watson Research Division Hawthorne, NY.

Huang, J., Lu, J., & Ling, C. X. (2003). Comparing naive bayes, decision trees, and svm with auc and accuracy. In *Third ieee international conference on data mining* (pp. 553–556).

Joe, H. (1996). Families of m-variate distributions with given margins and m (m-1)/2 bivariate dependence parameters. *Lecture Notes-Monograph Series*, 120–141.

Joe, H. (1997). *Multivariate models and multivariate dependence concepts*. CRC Press.

Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, *1*(1), 215–310.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81–93. Retrieved from `http://www.jstor.org/stable/2332226`

Kurowicka, D., & Cooke, R. M. (2006). *Uncertainty analysis with high dimensional dependence modelling*. John Wiley & Sons.

Lewis, D. D. (1998). Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning* (pp. 4–15).

Li, D. X. (2000). On default correlation: A copula function approach. *The Journal of Fixed Income*, *9*(4), 43–54.

Ling, C. X., Huang, J., Zhang, H., et al. (2003). Auc: a statistically consistent and more discriminating measure than accuracy. In *Ijcai* (Vol. 3, pp. 519–524).

Low, R. K. Y., Alcock, J., Faff, R., & Brailsford, T. (2013). Canonical vine copulas in the context of modern portfolio management: Are they worth it? *Journal of Banking & Finance*, *37*(8), 3085–3099.

Low, R. K. Y., Faff, R., & Aas, K. (2016). Enhancing mean–variance portfolio selection by modeling distributional asymmetries. *Journal of Economics and Business*, *85*, 49–72.

McNeil, A. J. (2008). Sampling nested archimedean copulas. *Journal of Statistical Computation and Simulation*, *78*(6), 567–581.

Mitchell, T. M. (1997). *Machine learning.* McGraw hill.

Morales Napoles, O., Cooke, R. M., & Kurowicka, D. (2010). About the number of vines and regular vines on n nodes.

Morillas, P. M. (2005). A method to obtain new copulas from a given one. *Metrika*, *61*(2), 169–184.

Nagler, T. (2014). Kernel methods for vine copula estimation.

Nagler, T., Bumann, C., & Czado, C. (2019). Model selection in sparse high-dimensional vine copula models with an application to portfolio risk. *Journal of Multivariate Analysis*, *172*, 180–192.

Nagler, T., & Vatter, T. (2019). rvinecopulib: High performance algorithms for vine copula modeling [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=rvinecopulib` (R package version 0.5.1.1.0)

Nelsen, R. B. (2007). *An introduction to copulas.* Springer Science & Business Media.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems* (pp. 841–848).

Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on machine learning* (pp. 625–632).

Omelka, M., Gijbels, I., Veraverbeke, N., et al. (2009). Improved kernel estimation of copulas: weak convergence and goodness-of-fit testing. *The Annals of Statistics*, *37*(5B), 3023–3058.

Patton, A. J. (2001). Modelling time-varying exchange rate dependence using the conditional copula.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pérez Díaz, Á. P. (2018). Supervised classification based on copula functions.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, *36*(6), 1389–1401.

R Core Team. (2019). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Rego, C., Gamboa, D., Glover, F., & Osterman, C. (2011). Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, *211*(3), 427–441.

Ripley, B. D. (2007). *Pattern recognition and neural networks*. Cambridge university press.

Rish, I., et al. (2001). An empirical study of the naive bayes classifier. In *Ijcai 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, pp. 41–46).

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M., II. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, *6*(3), 563–581.

Salinas-Gutiérrez, R., Hernández-Aguirre, A., Rivera-Meraz, M. J., & Villa-Diharce, E. R. (2010). Using gaussian copulas in supervised probabilistic classification. In *Soft computing for intelligent control and mobile robotics* (pp. 355–372). Springer.

Savu, C., & Trede, M. (2010). Hierarchies of archimedean copulas. *Quantitative Finance*, *10*(3), 295–304.

Schepsmeier, U., & Stöber, J. (2014). Derivatives and fisher information of bivariate copulas. *Statistical Papers*, *55*(2), 525–542.

Schindler, A. (2012). Bandwidth selection in nonparametric kernel estimation.

Schwarz, G., et al. (1978). Estimating the dimension of a model. *The annals of statistics*, *6*(2), 461–464.

Silvapulle, P., Kim, G., Silvapulle, M. J., et al. (2004). Robustness of a semiparametric estimator of a copula. In *Econometric society 2004 australasian meetings.*

Silverman, B. (1986). *Density estimation for statistics and data analysis*. Taylor & Francis. Retrieved from `https://books.google.be/books?id=e-xsrjsL7WkC`

Sklar, M. (1959). *Fonctions de répartition à n dimensions et leurs marges*. Université Paris 8. Retrieved from `https://books.google.be/books?id=nreSmAEACAAJ`

Stoeber, J., Joe, H., & Czado, C. (2013). Simplified pair copula constructionsâlimitations and extensions. *Journal of Multivariate Analysis*, *119*, 101–118.

Tawn, J. A. (1988). Bivariate extreme value theory: models and estimation. *Biometrika*, *75*(3), 397–415.

Venter, G. G. (2002). Tails of copulas. In *Proceedings of the casualty actuarial society* (Vol. 89, pp. 68–113).

Vuong, Q. H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica: Journal of the Econometric Society*, 307–333.

Whelan, N., et al. (2004). Sampling from archimedean copulas. *Quantitative finance*, *4*(3), 339–352.

Wikipedia contributors. (2019). *Complete graph.* In Wikipedia, The Free Encyclopedia. Retrieved from `https://en.wikipedia.org/wiki/Complete_graph`

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation*, *8*(7), 1341–1390.

Zhang, H. (2004). The optimality of naive bayes. *AA*, *1*(2), 3.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté des sciences
Place des sciences, 2 bte L6.06.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/sc**