UCLouvain

FACULTE DES SCIENCES

Faculté des sciences

Anomaly detection using dependence trees and copulas

Author: **Arnaud Briol**

Supervisor: **Prof. Johan Segers**

Reader**: Prof. Christian Hafner**

Academic year 2020-2021

Contents

Acknowledgements

First, I would like to thank my supervisor, Prof. Johan Segers, for his support, guidance and the numerous meetings that enabled me to progress in this master's thesis.

I am extremely thankful to my parents and my partner who supported me during the two years of my master's degree.

Finally, I would like to thank UCLouvain and the Faculty of Sciences for the quality of their teaching and the accommodations they have put in place to ensure the continuity of the master program despite the sanitary measures.

Introduction and objectives

Identifying anomalous instances in datasets is a common machine learning challenge. This task is generally unsupervised or semi-supervised and referred to as anomaly detection or outlier detection. It has been the subject of intensive research over the years (Goldstein and Uchida 2016; Shyu et al. 2003; Goldstein and Dengel 2012; Breunig et al. 2000; Angiulli and Pizzuti 2002; He, Xu, and Deng 2003). The reasons to identify instances that diverge from the norm can be diverse; either to exclude them from the dataset in an attempt to data cleansing, or to respond to a specific domain related problem such as fraud detection, intrusion detection, medical application, etc.

Recently, Horváth et al. (2020) published a semi-supervised algorithm that combined anomaly detection and copula dependence trees. This new algorithm aims at discovering and locating anomalies in high-dimensional continuous data using copulas and dependence trees. The algorithm is characterized by two interesting features for anomaly detection. First, it precisely locates the pairs of features from which the anomaly originates. Secondly, it is able to provide an anomaly score even if the observation is incomplete, which makes it particularly suitable for high-dimensional data where missing values are frequent.

In Horváth et al. (2020), anomaly scores are based on an estimated joint probability density underlying the multidimensional dataset; observations from low density areas receive high anomaly scores and conversely. This approach, although intuitive, can be challenging because it requires estimating the joint density function which grows in complexity with the dimension of the dataset. The approach of this new algorithm is to decompose the problem using dependence trees where these dependencies are modeled by copulas. The authors argue that copulas are convenient tools to model tail dependencies where anomalies are likely to be located. Similar decompositions have been performed in the literature; Chow and Liu (1968) already approximated a d-dimensional discrete probability distribution by decomposing it into a set of conditional probability distributions based on a tree structure. Horváth et al. (2020) adopted a similar approach and adapted it to continuous features. This decomposition is possible by combining the work of Chow and Liu (1968) and the work of Joe (1996), Bedford and Cooke (2001), Bedford and Cooke (2002), and Cooke and Kurowicka (2006) where copula-trees are used in vines to model high-dimensional probability distributions. Indeed, the idea of combining dependence trees and copulas described by Horváth et al. (2020) is actually a well-known concept called copula-trees in pair copula constructions, although it has never been used for anomaly detection.

The objective of this master's thesis will be to study the effectiveness of copula-trees for anomaly detection. To do so, we will implement the algorithm presented by Horváth et al. (2020), that we will call HKMN in reference to its authors' names. Moreover, another algorithm for anomaly detection based on copula-trees and pair copula constructions will be introduced, the vine copula anomaly detection (VCAD). The concept is similar to Horváth et al. (2020) except that not one, but several trees will be stacked to score anomalies. The goal of these additional trees is to model more complex conditional dependencies such as described in Joe (1996), Bedford and Cooke (2001), and Bedford and Cooke (2002). Finally, the HKMN of Horváth et al. (2020) and the newly introduced VCAD will be benchmarked against some recognized anomaly detection methods.

This master's thesis is divided in two parts. Part I explores the theoretical background necessary for the implementation of the algorithms. A specific emphasis was put on graph theory, Markov graphs and Chow-Liu trees, as both HKMN and VCAD rely on graph concepts. Thus, Chapter 1 of this document will focus on describing these important concepts. Another subject of attention is copulas and copula-trees that will be used in order to model dependencies between variables. This will be discussed in Chapter 2. Then, Chapter 3 will describe theoretically HKMN and VCAD. It will also contextualize these algorithms in the anomaly detection field. Part II aims to put these algorithms into practice. In Chapter 4 and 5, the implementation of the algorithms will be presented. Our implementation of HKMN will remain as close as possible to the one presented in Horváth et al. (2020). When it will deviate from the original article, these deviations will be discussed and argued. The last chapters of the master's thesis are dedicated to the benchmarking in Chapter 6 and the performance discussion in Chapter 7. The goal of the discussion will be to assess when HKMN or VCAD should be preferred to other anomaly detection methods.

Part I

Theoretical background

Chapter 1

Chow-Liu trees

The HKMN algorithm published by Horváth et al. (2020) and the VCAD that will be introduced later in the master's thesis both rely on tree structures. The HKMN finds anomalies based on a single tree while the VCAD has several trees stacked together. These tree structures allow to work with individual features and bivariate distributions rather than with a d-dimensional distribution. To decompose the d-dimensional feature space and build the tree structure, Horváth et al. (2020) adopted the same procedure as in Chow and Liu (1968), where a similar problem has been solved for discrete joint probability distributions. The trees used in HKMN have inherited many characteristics from the Chow-Liu trees introduced in Chow and Liu (1968). Therefore, this chapter will be dedicated to Chow-Liu trees and their building. To describe the Chow-Liu trees, some basis of graph theory must first be presented.

This chapter will explore the graph theory necessary for Chow-Liu trees, HKMN and VCAD in Section 1.1. This section defines the concepts of graphs and trees. The minimum spanning tree algorithm used by Chow and Liu (1968), Horváth et al. (2020) and VCAD to build trees based on individual variables is also briefly described. In Section 1.2, Markovs graphs and their independence properties are explained. Finally, in Section 1.3, the Chow-Liu algorithm is detailed.

1.1 Undirected graph theory

In this section, we introduce the necessary concepts to understand the Chow-Liu trees and later on the copula dependence trees used in HKMN and VCAD. The concepts and notation used in this section are taken from Eisner (1997) and Hartmann and Weigt (2005).

1.1.1 Graph

The principal concept in graph theory is obviously the graph itself. Its definition is large as it may take many forms. An undirected graph $G = (V, E)$ is given by its vertices $i \in V$ and its undirected edges $\{i, j\} \in E \subset V^{(2)}$. The fundamental difference between directed and undirected graphs is that $\{i, j\}$ and $\{j, i\}$ denote the same edge for undirected graphs, while they represent different edges for a directed graph. In this master's thesis, we will only discuss undirected graphs.

The graph has some interesting properties that allow to describe it or its components. For example, the number of vertices $N = |V|$ is named the order of the graph and the number of edges $M = |E|$ is named the size. Two vertices $i, j \in V$ that share an edge, edge $\{i, j\} \in E$, are called adjacent or neighbouring. The degree deg(i) of vertex i equals the number of adjacent vertices (Hartmann and Weigt 2005).

A path is a subgraph $G' = (V', E')$ of G with a set of vertices, $V' = \{i_0, i_1, \ldots, i_l\}$ and a set of edges, $E' = \{\{i_0, i_1\}, \{i_1, i_2\}, \ldots, \{i_{l-1}, i_l\}\}$. The particularity of the path is then that its consecutive vertices are adjacent. A path is said to form a cycle if $i_0 = i_l$. A graph G is denoted as connected if all pairs $\{i, j\} \in E^{(2)}$ of vertices can be linked by a path.

1.1.2 Tree and Minimum spanning Tree (MST)

A tree is a particular occurrence of a graph where this latter is both connected and cycle-free (acyclic). In other words, all vertices of the tree can be accessed through a path and no path of this graph forms a cycle. One can easily derive from this definition that a tree of order N has a size $N - 1$.

From the definition of the tree, a forest can also be defined as a graph, all of whose connected components are trees. These trees might not be connected together. The forest is then a set of trees.

Let $G = (V, E)$ be a connected graph of order N. A spanning tree is a cycle-free sub-graph of order N having maximum size $N - 1$, i.e., the spanning tree is still connected. In other words, the spanning tree is composed of all vertices of the original graph G connected together by $N - 1$ edges, without any cycle or unconnected vertex.

If weights have been assigned to the edges of G, a minimum spanning tree is a spanning tree of G whose edges have minimal total weight (Hartmann and Weigt 2005). A common task is to find the minimum spanning tree of a graph. Among the possible applications, we note that Chow-Liu trees, HKMN and VCAD require to find the minimum spanning tree. Over the years, many algorithms have been created to resolve this problem (Eisner 1997). A classical approach to get the minimum spanning tree is to focus on its specific properties. The notion of the cut is useful as it enables to highlight some these properties.

A cut $(S, V - S)$ of an undirected graph $G = (V, E)$ is a partition of V. We say that an edge $e = \{i, j\} \in E$ crosses the cut $(S, V - S)$ if one of its end vertices is in S and the other is in $V - S$(Cormen et al. 2014).

From the notion of cut, two properties of the MST originate:

1. Strong cut property: $e \in MST \Leftrightarrow e$ is the lightest edge across some cut of G.
2. Strong cycle property: $e \notin MST \Leftrightarrow e$ is the heaviest edge on some cycle of G.

Either of these implies at once that the MST is unique. These properties are at the basis of the algorithms to determine the minimum spanning tree of a graph. The Prim's and the Kruskal's algorithms (Prim 1957; Kruskal 1956) are the two most famous algorithm to resolve the MST problem, although recently many algorithms outperform these two (Eisner 1997). The advantage of Prim's algorithm and Kruskal's one is their implementation simplicity which relies on a greedy procedure. Detailed descriptions of Prim's algorithm and of Kurskal's one are available in the Appendix.

1.2 Markov graphs

Graphs can also model the joint distribution of random variables. In this case, each vertex represents one of the random variables. Let a Markov graph $G = (V, E)$ whose vertices represent a set of random variables having joint distribution P. This Markov graph G models the conditional independence relations existing between the variables of P. For instance, in G, the absence of edges between two vertices implies that the corresponding random variables are conditionally independent given the other variables (Hastie, Tibshirani, and Friedman 2009).

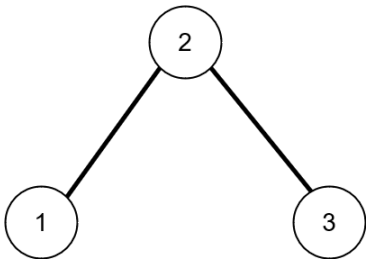


Figure 1.1: A Markov graph composed of three nodes 1, 2 and 3. These nodes represent respectively variables X_1, X_2, X_3

In Figure 1.1, the variable X_1 represented by node 1 is independent of X_3 given X_2. This is expressed with the following notation:

$$\text{No edge between nodes 1 and 3} \iff X_3 \perp X_1 \mid rest$$

where *rest* means all other variables represented in graph G. For the example in Figure 1.1, the expression of conditional independence can be denoted as:

$$X_1 \perp X_3 \mid X_2$$

This property is called *pairwise Markov property.*

The second independence property of Markov graphs is called the *global Markov property.* In Hastie, Tibshirani, and Friedman (2009), this property is described with a graph $G = (V, E)$ and three subgraphs A, B and $C \in V$. C is said to separate A and B if every path between A and B intersects a node in C. The global Markov property relies on this notion of separation between subgraphs. C actually breaks the graph into conditionally independent pieces.

$$\text{if } C \text{ separates } A \text{ and } B \text{ then } A \perp B \mid C$$

The global Markov property is interesting as it allows to divide the graphs into subsets and infer independence between the sets.

Finally, the last Markov property is the *local Markov property.* It states that any vertex will be conditionally independent of all other vertices given its neighbors. This definition is often associated with the Markov blanket of a node X which is the minimal set of nodes that isolates X from the rest of the graph. In other words, the set of nodes which are directly linked to X.

Markov trees are Markov graphs that are acyclic and connected, similarly to the trees defined in Section 1.1.2.

1.3 Chow-Liu trees

In 1968, Chow and Liu published an algorithm to approximate a **discrete** joint probability distribution P using Markov trees based on a product of pairwise conditional and marginal distributions (Chow and Liu 1968).

To support their approach, Chow and Liu prove that the resulting probability distribution P_{CL} minimizes the Kullback-Leibler divergence with respect to the original distribution P in comparison with other product approximations of pairwise conditional and marginal distributions.

The Chow-Liu algorithm relies on three steps:

1. Computation of the mutual information criteria between each pair of variables.
2. Construction of a tree using a maximum spanning tree algorithm.
3. Approximate the full joint distribution.

1.3.1 Computation of the mutual information criteria

Mutual information describes the dependence existing between two variables. Let (X, Y) be a pair of discrete random variables. Their mutual information is defined as

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p_{(X,Y)}(x,y) \log \Big(\frac{p_{(X,Y)}(x,y)}{p_X(x) p_Y(y)} \Big)$$

for discrete features.

If the variables are independent, the mutual information will be null. If it is higher than zero, the measure reflects how much knowing about the first variable reduces the uncertainty of the second one.

1.3.2 Construction of a tree using a maximum spanning tree algorithm

Using the mutual information as weight, one can build a tree with a maximum spanning tree algorithm. This algorithm is similar to the minimum spanning tree discussed in section 1.1.2 except that weights have been multiplied by -1. Kruskal's and Prim's algorithms can provide this maximum spanning tree. The resulting tree is called a maximum dependence tree.

Let $(X_i \mid i \in I = \{1, .., 4\})$ be random variables with mutual information $I(X_i, X_j)$ for $i, j \in I$. The mutual information between these variables are provided in Table 1.1.

$I(X_i, X_j)$	X_1	X_2	X_3	X_4
X_1	-	0	0.6	0.2
X_2	0	-	0.3	0.1
X_3	0.6	0.3	-	1.1
X_4	0.2	0.1	1.1	-

Table 1.1: The mutual information between X_1, X_2, X_3, X_4

With the mutual information as weight and using Prim's algorithm, one can find the maximum information tree of Figure 1.2.

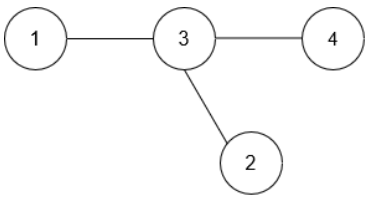


Figure 1.2: The maximum dependence tree of X_1, X_2, X_3, X_4

1.3.3 Approximate the full joint distribution

Starting from a randomly chosen vertex, one can multiply the marginal density of this first vertex with the second-order conditional probability of each of the edges of the tree. The probability distribution obtained from the tree is an optimal second order approximation of the target distribution.

Starting from the vertex 1 in figure 1.2, one obtains the following probability distribution,

$$P_{CL}(X_1, X_2, X_3, X_4) = P(X_1)P(X_3|X_1)P(X_2|X_3)P(X_4|X_3)$$

The Chow-Liu tree and the resulting probability distribution P_{CL} can be formalized. For $\mathbf{X} = (X_1, \ldots, X_d)$ a vector of random variables, the product-form probability distribution P_{CL} associated with such a Chow-Liu tree is

$$P_{CL}(\mathbf{X}) = \frac{\prod_{(i,j)\in T} P(X_i, X_j)}{\prod_{i\in V} P(X_i)^{v_i - 1}} \tag{1.1}$$

where T is a spanning tree with a set of vertices $V = \{1, \ldots, d\}$ and v_i denotes the degree of the vertex representing X_i.

In their article, the authors demonstrate that the Kullback-Leibler divergence of the second-order approximation P_{CL} of a distribution P is inversely proportional to the sum of the mutual information used to weight the dependence tree. Consequently, selecting the highest mutual information will result in minimising the Kullback-Leibler divergence. So, although P_{CL} remains an approximation of $P(\mathbf{X})$, it is the best possible one constructed this way.

The principal inconvenience of this algorithm is that it ignores higher-order dependencies. Consequently, variables whose vertices are not adjacent in the maximum dependence tree are assumed to be independent conditionally to a variable in their path. Unless there exist no higher-order dependencies, the Chow-Liu algorithm only provides an approximation of the target distribution. Moreover, it is limited to discrete distributions.

Chapter 2

Copula theory

As explained already, we want to model a joint distribution function underlying a dataset in order to identify rare occurrences as anomalies. It is evident that the complexity of this modelling increases with the dimensionality of the dataset, since each new variable interacts with the previous ones. A solution to deal with this complexity is to decompose the problem into smaller pieces. In Chapter 1, we saw the solution introduced by Chow and Liu for discrete distributions. The main approach discussed and implemented in this master's thesis is the one taken by Horváth et al. (2020). It consists in first modelling the marginal distribution functions of variables and then modelling the dependence between variables. This chapter focuses on this last point, the dependence. Fortunately, dependence modelling is an extensive research topic in the statistical literature. This literature has as foundation Sklar (1959). This paper has demonstrated the relationship between a joint cumulative distribution function and the marginal distribution functions of random variables. The relationship is based on the notion of copula. By characterizing the interactions between variables, these copulas allow to separate the modelling task in two stages; first, by focusing on the margins and then on the dependence. This method is widely used in financial mathematical modelling (Cherubini, Luciano, and Vecchiato 2004). However, the use of copulas for anomaly detection is innovative. In their article, Horváth et al. (2020) introduce this practice with copula dependence trees. The authors justify their approach by the benefits of copulas in modelling tails of distributions. There are copula families that can accommodate to both low and high tail-dependencies. This is an important criterion since the anomalies are generally located in the tails of distributions, which must be reproduced accurately.

In this chapter, dependence modelling with copulas will be explored. We will review the basic theory of copulas in Section 2.1, as well as the basis of any modelling with copulas, the Sklar's theorem, in Section 2.2. Afterwards, we will discuss in Section 2.3 the measures of dependence including Kendall's tau, which will be useful in the following chapters. Then, we will discuss the dependence modelling with copulas and Markov trees in Section 2.4. We will conclude with a more complex dependence modelling with pair copula constructions in Section 2.5.

2.1 Dependence modeling with copulas

In this section, we introduce the concepts around copulas that are used throughout the master's thesis. The main reference for this section is Segers (2016).

As copulas arise in the study of cumulative distribution functions, we first start by recalling this notion.

Definition 1. *The cumulative distribution function (cdf) of a d-dimensional random vector* $\mathbf{X} = (X_1, \ldots, X_d)$ *is the function*

$$F(\mathbf{x}) = \Pr[\mathbf{X} \leq \mathbf{x}] = \Pr[X_1 \leq x_1, \ldots, X_d \leq x_d], \quad \mathbf{x} = (x_1, \ldots, x_d) \in [-\infty, \infty]^d$$

The marginals cdfs or margins of F *are the univariate cdfs* $F_1, \ldots, F_d$ *given by*

$$F_j(x_j) = \Pr[X_j \leq x_j], \quad x_j \in \mathbb{R}$$

Copulas are a specific case of cumulative distribution functions (cdf), each of their margins being uniformly distributed on $[0, 1]$.

Definition 2. *A d-dimensional copula* $C : [0,1]^d \to [0,1]$ *is the cumulative distribution function of a random vector* $U = (U_1, \ldots, U_d)$ *whose margins are uniform on (0,1):*

$$C(\mathbf{u}) = \Pr[\mathbf{U} \leq \mathbf{u}] = \Pr[U_1 \leq u_1, \ldots, U_d \leq u_d]$$

where

$$\Pr[U_j \leq u_j] = u_j$$

for $j \in \{1, \ldots, d\}$ *and* $0 \leq u_j \leq 1$.

Copulas have probability density functions that respect the following definition.

Definition 3. *A d-dimensional copula* C *has density function* c *if*

$$C(\mathbf{u}) = \int_{[\mathbf{0},\mathbf{u}]} c(\mathbf{v}) d\mathbf{v}, \quad \mathbf{u} \in [0,1]^d.$$

In this case, C *is called absolutely continuous.*

From a cdf, a density is obtained through differentiation of the cumulative distribution function. Consequently, the same goes for the copula density which is retrieved by differentiating the copula.

In practice, during a statistical analysis, we only have a sample and not the entire population. To infer the underlying copula behind this sample, a data-driven approximation, called the empirical copula, is then performed.

To get this empirical copula, we must first transform the analyzed data as it might not have uniform margins. This operation requires to compute the empirical marginal cdf for each variable of the dataset. For a sample $\mathbf{X_1}, \ldots, \mathbf{X_n}$, these empirical marginal cdf are defined as follow:

$$\hat{F}_{n,j}(x) = \frac{1}{n+1} \sum_{i=1}^{n} 1(X_{ij} \leq x), \quad x \in \mathbb{R}$$

The ranks of X_{ij}, defined as $R_{ij}^{(n)} = \sum_{k=1}^{n} 1(X_{kj} \leq X_{ij})$, can also be derived from the empirical marginal cdf with $\hat{F}_{n,j}(X_{ij}) = \frac{1}{n+1} R_{ij}^{(n)}$. The pseudo-observations are the random vectors $\hat{\mathbf{U}}_{\mathbf{i}}^{(\mathbf{n})} = (\hat{U}_{i1}^{(n)}, \ldots, \hat{U}_{id}^{(n)})$ where $\hat{U}_{ij}^{(n)} = \hat{F}_{n,j}(x) = \frac{1}{n} R_{ij}^{(n)}$. These pseudo-observations are used in practice to compute the empirical copula.

Definition 4. *For the sample $\mathbf{X_1}, \ldots, \mathbf{X_n}$ with empirical marginal distributions $\hat{F}_{n,1}, \ldots, \hat{F}_{n,d}$, the empirical copula approximation is defined as*

$$\begin{aligned} \hat{C}(\mathbf{u}) =& \frac{1}{n+1} \sum_{i=1}^{n} 1\{\hat{F}_{n,1}(X_{i1}) \leq u_1, \ldots, \hat{F}_{n,d}(X_{id}) \leq u_d\} \\ =& \frac{1}{n+1} \sum_{i=1}^{n} 1\{\hat{U}_{i1}^{(n)} \leq u_1, \ldots, \hat{U}_{id}^{(n)} \leq u_d\}, \end{aligned}$$

where $\mathbf{u} \in [0,1]^d$.

2.2 Sklar's Theorem

Since Sklar (1959), it is known that copulas can model the dependence structure between random variables with continuous margins. This was formalized in the famous Sklar's Theorem, which is at the center of most applications of copulas in statistics.

Definition 5. *Let F be an d-dimensional distribution function with margins $F_1, F_2, \ldots, F_d$. Then there exists a d-copula C such that for all $\boldsymbol{x}$ in $\overline{\mathbb{R}}^d$,*

$$F(x_1, x_2, \ldots, x_d) = C(F_1(x_1), F_2(x_2), \ldots, F_d(x_d)).$$

If $F_1, F_2, \ldots, F_d$ are all continuous, then C is unique. Conversely, if C is an d-copula and $F_1, F_2, \ldots, F_d$ are distribution functions, then the function F is an d-dimensional distribution function with margins $F_1, F_2, \ldots, F_d$.

This theorem is particularly useful as it enables to separate the effect of the dependence between variables from the one of the margins. Therefore, modeling the d-dimensional distribution function F can be performed in sequence by first focusing on finding margins $F_1, \ldots, F_d$ then modeling the dependence function or copula, C.

By applying the chain rule, we further have for an absolutely continuous distribution function F with strictly increasing continuous margins $F_1, \ldots, F_d$ admitting densities $f_1, \ldots, f_d$ that

$$f(x_1, \ldots, x_d) = \Big[\prod_{k=1}^{d} f_k(x_k) \Big] \times c(F_1(x_1), F_2(x_2), \ldots, F_d(x_d)).$$

Here $c(\cdot)$ denotes the copula density.

2.3 Dependence and its measures

As explained in Section 2.2, copulas are useful tools to model dependence between random variables. Yet, they are complex to interpret. In this section, we explore scalar measures to evaluate dependence between variables. Kendall's tau is one of the measure presented in this section. It will play a significant role later in the master's thesis, especially for the VCAD method where it will help us choose which dependencies to model. It is presented with two other common measures: Pearson's linear correlation and Spearman's rank correlation coefficient (or Spearman's rho). The reference for this section is Nelsen (2006).

Turning now to the most familiar dependence measure, Pearson's linear correlation is an easily interpretable measure of the linear dependence existing between two random variables.

$$cor(X, Y) = \frac{cov(X, Y)}{\sqrt{var(X)var(Y)}}$$

However, it has some important drawbacks:

1. it does not exist if $\mathrm{E}[X^2] = \infty$ or $\mathrm{E}[Y^2] = \infty$.
2. for increasing f and g, in general $\mathrm{cor}(f(X), g(Y)) \neq \mathrm{cor}(X, Y)$.
3. if X and Y are perfectly associated, $\mathrm{cor}(X, Y)$ is not necessary equal to 1.

Consequently, we need to identify other measures to assess the dependence between random variables. Two well-know measures will be described below, Kendall's tau and Spearman's rho. Both Kendall's tau and Spearman's rho are dependence measures which are rank based and therefore invariant to monotonic transformations of the marginals.

More precisely, these measures are said to assess the concordance between the variables.

Definition 6. *(Concordance) Let (x_i, y_i) and (x_j, y_j) denote two observations from a random vector (X, Y) of continuous variables. (x_i, y_i) and (x_j, y_j) are said to be concordant if $x_i < x_j$ and $y_i < y_j$, or $x_i > x_j$ and $y_i > y_j$. Alternatively, a pair of observations can be discordant if $x_i > x_j$ and $y_i < y_j$, or $x_i < x_j$ and $y_i > y_j$.*

Kendall's tau is first introduced in Kendall (1938). For a population, it is defined as the difference between the probability of a pair of observations to be concordant minus the probability of the pair to be discordant.

Definition 7. *Let (X_1, Y_1) and (X_2, Y_2) be independent and identically distributed random vectors, Kendall's tau can be represented as*

$$\tau = \tau_{X,Y} = P[(X_1 - X_2)(Y_1 - Y_2) > 0] - P[(X_1 - X_2)(Y_1 - Y_2) < 0].$$

For a set of n observations, there exist $\binom{n}{2}$ distinct pairs of observations. With c being the number of concordant pairs and d the number of discordant pairs, the empirical Kendall's tau,$\hat{\tau}$, is computed as follows

$$\hat{\tau} = \frac{c-d}{c+d} = \frac{c-d}{\binom{n}{2}}.$$

Additionally, Kendall's tau can be computed by using the copulas of the random variables. Indeed, if F is the bivariate cumulative distribution function of (X, Y) and has copula C, then

$$\tau(F) = \tau(C)$$

since the ordering of X_1 and X_2 is similar as the one of $U_1 = F_1(X_1)$ and $U_2 = F_2(X_2)$, and similarly for Y_1 and Y_2.

Let X and Y be continuous random variables whose copula is C. Then the population version of Kendall's tau for X and Y is given by

$$\tau_{X,Y} = \tau_C = 4 \iint_{[0,1]^2} C(u, v) dC(u, v) - 1$$

The second measure of dependence we will explore is also based on the concordance. This is called rank correlation or Spearman's rho.

Definition 8. *(Rank correlation) The rank correlation of random variables X, Y with cumulative distribution functions F_X and F_Y is*

$$\rho_r(X, Y) = \rho(F_X(X), F_Y(Y))$$

where ρ represent pearson's linear correlation.

Let (X_1, Y_1), (X_2, Y_2) and (X_3, Y_3) be three independent random vectors with a common joint distribution function F (whose margins are again F_1 and F_2) and copula C. The Spearman's rho for the population is proportional to the probability of concordance of the vectors $(X_1, Y_1), (X_2, Y_3)$ minus their discordance.

$$\rho_{X,Y} = 3(P[(X_1 - X_2)(Y_1 - Y_3) > 0] - P[(X_1 - X_2)(Y_1 - Y_3) < 0])$$

Once again, this measures can be expressed using copulas.

$$\rho_{X,Y} = 12 \iint_{[0,1]^2} C(u,v)dudv - 3$$

2.4 Dependence modelling using copula-trees

In this section and the following one, we focus on decomposing the dependence modelling. To overcome the complexity of modelling high-dimensional distributions, we use a method called *pair copula constructions* whose goal is to decompose the complex dependence structure into bivariate dependencies as building blocks. This notion of pair copula constructions, also called regular vine copulas, has been introduced by Joe (1996). It was described further by Bedford and Cooke (2001) and Bedford and Cooke (2002). Its core concept is to model multivariate data by using a cascade of pair-copulas and the marginal distributions of variables. Firstly, in this section, we introduce the concept of copula-trees, and in the following section, pair copula constructions will be presented. The main reference for these sections is the book Cooke and Kurowicka (2006) and Czado (2019).

Let us start by defining dependence trees. As stated in Section 1.1.2, a tree is an acyclic undirected graph. Yet, when trees are used to describe dependence structures in high-dimensional distributions, statisticians call them *dependence trees*. These dependence trees can be used with copulas specifying the bivariate dependencies.

Definition 9. *(Bivariate- and Copula-tree specification)* $(F, \mathcal{T}, B)$ *is a bivariate tree specification if*

1. $F = (F_1, \ldots, F_d)$ *is a vector of one-dimensional distribution functions for random vector* $(X_1, \ldots, X_d)$ *such that* $X_i \neq X_j$ *for* $i \neq j$.
2. T *is a tree of order d elements, with a set of vertices* V *and a set of edges* E.
3. $B = \{B_{ij} \quad | \quad \{i,j\} \in E$ *and* B_{ij} *is a non-empty subset of the set of bivariate distributions with margins* $F_i, F_j\}$.

A bivariate tree specification $(F, \mathcal{T}, B)$ *is a copula-tree if* $(F_1, \ldots, F_n)$ *are continuous invertible and if* $B = \{C_{ij} \quad | i,j \in E$ *and* C_{ij} *is the copula for* $(X_i, X_j)\}$.

Copulas-trees are then special cases of bivariate trees where the bivariate distributions characterizing edges B_{ij} are bivariate copulas.

As the dependence tree is a Markov tree such as defined in Section 1.2, Markov properties apply to them. A distribution G is has Markov tree dependence when it can be expressed as a tree $\mathcal{T}$ respecting the *global Markov property* (see 1.2). The latter property allows us to introduce the following interesting theorem.

Theorem 1. *Let $(F, \mathcal{T}, B)$ be an d-dimensional bivariate tree specification that specifies the marginal densities f_i , $1 \leq i \leq d$, and the bivariate densities f_{ij}, $\{i,j\} \in E$. Then, there is a unique density g on $\mathbb{R}^d$, with margins $f_1, \ldots, f_d$, and bivariate margins f_{ij} for $\{i,j\} \in E$, such that g has Markov tree dependence for $\mathcal{T}$. The density g is given by*

$$g(x_1, \ldots, x_d) = \frac{\prod_{\{i,j\} \in E} f_{ij}(x_i, x_j)}{\prod_{i \in V} (f_i(x_i))^{deg(i)-1}}$$

where $deg(i)$ denotes the degree of node i and $f_i(x_i) > 0$, $i = 1, \ldots, d$. With copula densities c_{ij}, $\{i,j\} \in E$, this becomes

$$g(x_1, \ldots, x_n) = f_1(x_1) \ldots f_d(x_d) \prod_{\{i,j\} \in E} c_{ij}(F_i(x_i), F_j(x_j))$$

Based on Theorem 1, we can determine the density of a multivariate distribution function whose variables have been represented in a copula-tree. The drawback of dependence trees to represent a multivariate distribution function is the global Markov property. Indeed, for two non-adjacent variables in the tree, given any set of variables separating them on the path, their are conditional independent. It would be possible to enrich the copula-tree with additional information on the conditional dependence. For instance, in Figure 2.1 how could we enrich the copula-tree to depict the relation between 2 and 3 given 1? This is the starting question which led to vines.

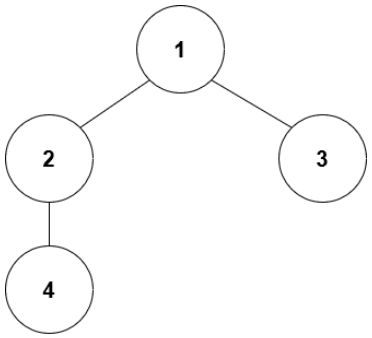


Figure 2.1: A dependence tree

2.5 Dependence modelling with regular vine copula

To make up for the conditional independence induced by Markov properties, the vines will stack several trees, each one of them representing an additional level of the dependence structure. A vine on d variables is then a nested set of trees, where the edges of the tree j are the nodes of the tree $j+1$. The higher-level trees in the set enable to encode conditional constraints and thus enrich the previous trees with more information about the conditional dependence structure.

Definition 10 (Regular vine tree sequence)**.** *The set of trees $\mathcal{V} = (T_1, \ldots, T_{d-1})$ is a regular vine tree sequence on d elements if:*

1. *Each tree $T_j = (V_j, E_j)$ is connected, i.e. for all vertices $a, b \in T_j$, $j = 1, \ldots, d-1$, there exists a path that contains both vertices a and b.*

2. *T_1 is a connected tree with vertices $V_1 = \{1, \ldots, d\}$ and edges E_1.*

3. *For $j \geq 2$, T_j is a tree with vertices set $V_j = E_{j-1}$ and edge set E_j .*

4. *For $j = 2, \ldots, d-1$ and $\{a, b\} \in E_j$ it must hold that $|a \cap b| = 1$.*

A regular vine tree sequence is often called a R-vine in the literature. The property (4) is called the *proximity condition*. It ensures that if there is an edge e connecting a and b in tree T_j, $j \geq 2$, then a and b (which are edges in T_{j-1}) must share a common node in T_{j-1}. In Figure 2.2, two vines are represented. The vine (a) is a regular (R-)vine as it respect the proximity condition while the vine (b) is non-regular.

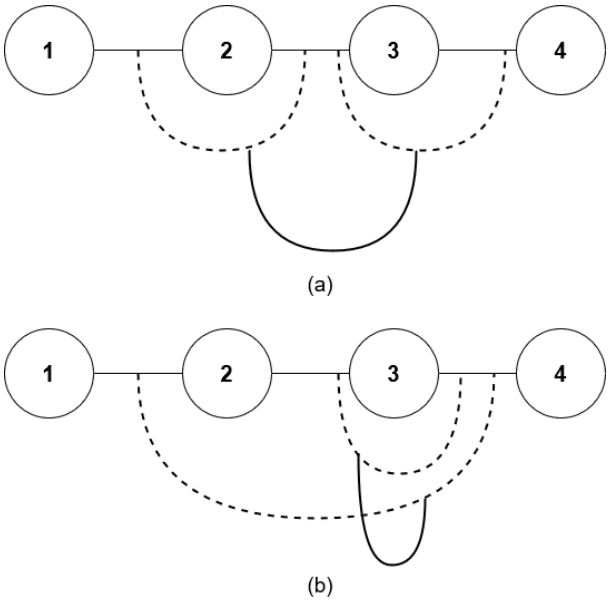


Figure 2.2: A regular vine (a) and a non-regular vine (b) on four variables.

To ease the notation of the vertices and edges of each tree of a vine, a nomenclature is introduced. This notation is based on a condition set of for each edge.

Definition 11 (Complete union and conditioned sets)**.** *For any edge $e \in E_i$, let us define the set*

$$A_e \coloneqq \Big\{ j \in V_1 | \quad \exists \quad e_1 \in E_1, \ldots, e_{i-1} \in E_{i-1} \text{ such that } j \in e_1 \in \cdots \in e_{i-1} \in e \Big\}.$$

The set A_e is called the complete union of the edge e. The conditioning set D_e of an edge $e = \{a, b\}$ is defined by

$$D_e := A_a \cap A_b$$

and the conditioned sets $\mathcal{C}_{e,a}$ *and* $\mathcal{C}_{e,b}$ *are given by*

$$\mathcal{C}_{e,a} := A_a \setminus D_e, \quad \mathcal{C}_{e,b} := A_b \setminus D_e \quad \textit{and} \quad \mathcal{C}_e := \mathcal{C}_{e,a} \cup \mathcal{C}_{e,b}.$$

We often abbreviate each edge $e = (\mathcal{C}_{e,a}, \mathcal{C}_{e,b}; D_e)$ *in the vine tree sequence by*

$$e = (e_a, e_b; D_e).$$

An example of vine using the notation of Definition 11 is illustrated in Figure 2.3. The obvious advantage of the notation is that the reader clearly notices for each edge the conditioned set and the conditioning set. Therefore, the edges that were combined to create vertices in the following tree are directly apparent.

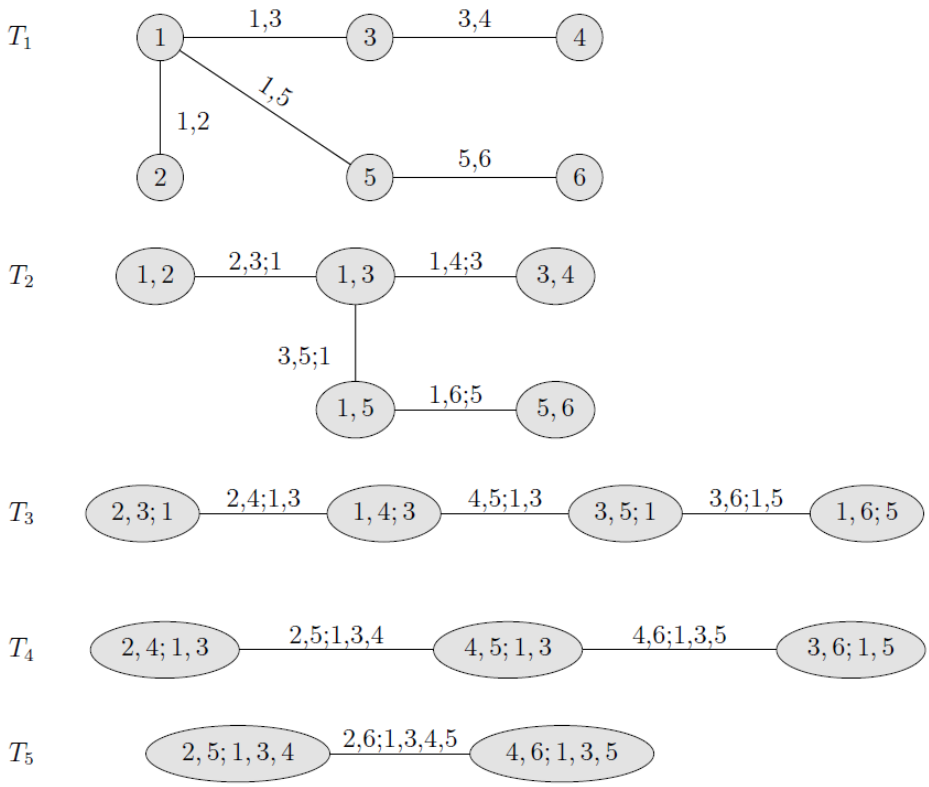


Figure 2.3: A six dimensional regular vine tree sequence. The illustration is taken from page 101 of Czado (2019).

As the purpose of nested trees in vines is to model higher dependence structures between random variables, we must define the concept of regular vine distribution.

Definition 12 (Regular vine distribution)**.** *The joint distribution* F *for the* d *dimensional random vector* $\mathbf{X} = (X_1, \ldots, X_d)$ *has a regular vine distribution, if we can specify a triplet* $(\mathcal{F}, \mathcal{V}, B)$ *such that:*

1. ***Marginal distributions***: *$\mathcal{F} = (F_1, \ldots, F_d)$ is a vector of continuous invertible marginal distribution functions, representing the marginal distribution functions of the random variable X_i, $i = 1, \ldots, d$.*

2. ***Regular vine tree sequence***: *$\mathcal{V}$ is an R-vine tree sequence on d elements.*

3. ***Bivariate copulas***: *The set $B = \{Ce| \quad e \in E_i; \quad i = 1, \ldots, d-1\}$, where C_e is a symmetric bivariate copula with density. Here E_i is the edge set of tree T_i in the R-vine tree sequence $\mathcal{V}$.*

4. ***Relationship between R-vine tree sequence $\mathcal{V}$ and the set B of bivariate copulas***: *For each $e \in E_i$, $i = 1, \ldots, d-1$, $\quad e = \{a, b\}$, C_e is the copula associated with the conditional distribution of $X_{C_{e,a}}$ and $X_{C_{e,b}}$ given $\mathbf{X}_{D_e} = \mathbf{x}_{D_e}$. Further $C_e(.,.)$ does not depend on the specific value of $\mathbf{x}_{D_e}$.*

The property (4) of Theorem 12 is called the *simplifying assumption* (Brechmann, Czado, and Aas 2012). It states that bivariate copulas $C_e(.,.)$ do not depend on the specific value of $\mathbf{x}_{D_e}$. To illustrate the above definition, let's take the edge $e_{14;3}$ of T_2 in Figure 2.3. The copula C_e for this edge equals the conditional bivariate distribution function of (X_1, X_4) given $X_3 = x_3$ and it is given by

$$F_{14|3}(x_1, x_4|x_3) = C_{14|3}\big(F_{1|3}(x_1|x_3), F_{4|3}(x_4|x_3)\big)$$

Bedford and Cooke (2002) showed that a R-vine ($\mathcal{F}$, $\mathcal{V}$, B) respecting the three first properties of definition 12 can be uniquely connected to a d-dimensional distribution F, in particular the following theorem holds.

Theorem 2 (Existence of a regular vine distribution)**.** *Assume that ($\mathcal{F}$, $\mathcal{V}$, B) satisfy the properties (1)–(3) of Definition 12, then there is a unique d-dimensional distribution F with density*

$$\begin{aligned} f_{1,\ldots,d}(x_1, \ldots, x_d) &= f_1(x_1) \ldots f_d(x_d) \\ &\prod_{i=1}^{d-1} \prod_{e \in E_i} c_{\mathcal{C}_{e,a}\mathcal{C}_{e,b};D_e}(F_{\mathcal{C}_{e,a}|D_e}(x_{\mathcal{C}_{e,a}}|\mathbf{x}_{D_e}), F_{\mathcal{C}_{e,b}|D_e}(x_{\mathcal{C}_{e,b}}|\mathbf{x}_{D_e})), \end{aligned} \tag{2.1}$$

such that for each $e \in E_i$, $i = 1, \ldots, d-1$, with $e = \{a, b\}$ we have for the distribution function of $X_{C_{e,a}}$ and $X_{C_{e,b}}$ given $\mathbf{X}_{D_e} = \mathbf{x}_{D_e}$

$$F_{\mathcal{C}_{e,a}\mathcal{C}_{e,b}|D_e}(x_{\mathcal{C}_{e,a}}, x_{\mathcal{C}_{e,b}}|\mathbf{x}_{D_e}) = C_e(F_{\mathcal{C}_{e,a}|De}(x_{\mathcal{C}_{e,a}}|x_{D_e}), F_{\mathcal{C}_{e,b}|D_e}(x_{\mathcal{C}_{e,b}}|x_{D_e})).$$

Further the one dimensional margins of F are given by $F_i(x_i), i = 1, \ldots, d$.

Definition 12 and Theorem 2 give us the density characterising the joint distribution of a random vector $\mathbf{X}$. More precisely, this density is factorized into a product of the marginals and some pair-copulas for each tree of the vines. Theorem 2 goes one step further than Theorem 1 discussed for dependence tree. Indeed, the conditional dependence structure is now modeled by the higher-level trees of the vines rather than assumed independent as in standalone Markov trees. To illustrate Theorem 2, the density associated with the 6 dimensions R-vine distribution of Figure 2.3 is decomposed below. To ease the notation, the abbreviation $f_{1,\dots,j}$ and f_j will be used to refer to $f_{1,\dots,j}(x_1,\dots,x_j)$ and $f_j(x_j)$ respectively. Regarding copulas, we take the simplifying assumption and take the following abbreviation :

$$c_{i,j;D} \coloneqq c_{i,j;D}(F_{i|D}(x_i|\mathbf{x}_D), F_{j|D}(x_j|\mathbf{x}_D))$$

The regular vine copula density becomes :

$$\begin{aligned} f_{123456} =& f_1 \cdot f_2 \cdot f_3 \cdot f_4 \cdot f_5 \cdot f_6 \cdot \\ & c_{15} \cdot c_{34} \cdot c_{13} \cdot c_{12} \cdot c_{56} \cdot \\ & c_{23;1} \cdot c_{14;3} \cdot c_{35;1} \cdot c_{16;5} \cdot \\ & c_{24;13} \cdot c_{45;13} \cdot c_{36;15} \cdot \\ & c_{25;134} \cdot c_{46;135} \cdot \\ & c_{26;1345} \end{aligned}$$

Chapter 3

Anomaly Detection with copula-trees

Anomaly detection or outlier detection has been the topic of intensive research over the years. One can already find articles on the subject as early as the 19^{th} century with, for instance, Edgeworth (1887). More recently, Grubbs (1969) gave a definition of an outlier: "An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs". The definition remains correct but the motivation to detect anomalies changed over time. At first, the purpose was to remove outlying observations from the training process as models tend to be highly sensitive to outliers. In recent years, its use has expanded to a wide range of applications. For instance, anomaly detection is used for fraud detection in the banking and financial sector, for intrusion detection in cybersecurity or for medical application in life-science.

In this chapter, we present theoretically two anomaly detection methods, HKMN and VCAD, that use the copula-trees introduced in Section 2.4. Before reviewing them in details, we cover the different types of anomaly detection algorithms in section 3.1. The aim of this section is to contextualise the algorithms in the broad field of anomaly detection and to enlighten the reader on the conditions under which these algorithms can be applied.

Later, in Section 3.2, the HKMN of Horváth et al. (2020) will be detailed. Although the concept of copula-trees is largely present in Horváth et al. (2020), the authors prefer to rely on Chow and Liu (1968) rather than on the current literature dealing with copula-trees and pair copula constructions such as Joe (1996), Bedford and Cooke (2001), and Bedford and Cooke (2002) or, more recently, Czado (2019). Consequently, the implementation of the HKMN diverges from the implementations of copula-trees and vines found in the context of financial modelling. Here, we will focus on the HKMN of Horváth et al. (2020) to present how the copula-trees are constructed as well as the way they are used to score and locate anomalies.

Finally, in section 3.3, a new algorithm for anomaly detection will be presented: the vine copula anomaly detection (VCAD). This algorithm takes into consideration the results of Section 2.5

about regular vine copulas and adapts them to anomaly detection as did Horváth et al. (2020) for a single copula-tree. The added value of this algorithm is that it will be able to identify anomalies which have their sources in the higher-order conditional dependence structure. The design of the algorithm as well as some divergences with the HKMN will be explained in that section.

3.1 Categorization of anomaly detection problems

As mentioned in the introduction of this chapter, anomaly detection can serve numerous purposes. These purposes generally come with some specificity or complexities. This is why the anomaly detection problem can take many forms. The approach will generally be determined by the characteristics of the analysed dataset and its labelling. In this section, the different forms of anomaly detection problems are briefly discussed. We also describe where the HKMN and VCAD stand in this categorization.

Anomaly detection methods vary in function of the data type they can handle, from univariate to multivariate, from discrete to continuous. There may also have some kind of relationship existing between records of a dataset (Chandola, Banerjee, and Kumar 2009). These relationships are generally categorised as follow,

- **Point data** where no relationship is assumed among the records
- **Sequence data** such as time-series where realisations are linearly ordered
- **Spatial data** where records are linked to neighbouring records
- **Graph data** where instances are represented as graphs with vertices and edges.

In the current master's thesis, we will focus on multivariate continuous point data. This means, no relationship is assumed between records.

A common categorisation between anomaly detection techniques relates to the labelling of data. Depending on the availability of such labels, three categories are differentiated:

- **Supervised anomaly detection**: Such techniques rely on labelled training and test data. Each realisation of the training data has been flagged either normal or anomalous. This problem can then be addressed as a classification where traditional pattern recognition algorithms can be applied. The slight difference with classification problem is the strongly unbalanced classes associated with anomaly detection (Goldstein and Uchida 2016).
- **Semi-Supervised anomaly detection**: These techniques are applied when only a small fraction of the data is labelled. These labelled observations are generally normal instances and will form the training set. The remaining data will be unlabelled and will therefore contain both normal and anomalous instances. The fitting phase will model a single normal

class. Based on this model, the purpose is to identify in the unlabelled set if the realisations deviate from the modelled class. The HKMN and VCAD discussed in this master's thesis both belong to this category.

- **Unsupervised anomaly detection**: The particularity of these techniques is that no labelling is needed. The basic assumption is that normal observations are much more frequent than the anomalous ones. The algorithms will then score observations based on the characteristics of the dataset, assuming that common trait of the dataset represent normality. It generally uses distances or densities to evaluate the abnormality of the observation. These techniques can however perform poorly when outliers are too frequent as their characteristics become common. It results in many False Negative (type II error), as many outliers remain unnoticed.

Semi-supervised and unsupervised algorithms have the advantage that no information about the anomalous class is necessary. Consequently, even if new types of anomalies would occur in the future, it should be detected. By contrast, supervised algorithms are models trained to spot specific classes, anomalous or normal. If a never encountered class of anomaly would appear, the model would not know how to label it.

Algorithms are also distinguished by their output. Either, for each observation, they assign an anomaly score which reflects the degree of abnormality, either it directly assigns a label to the observation (normal or anomaly). For semi-supervised or unsupervised anomaly detection, anomaly scores are more common for practical reasons (Goldstein and Uchida 2016). Indeed, these algorithms rank anomalies based on their deviation from a modelled class (*semi-supervised*) or based on their deviation from a common behaviour in the dataset (*unsupervised*). They then only report the most likely anomalies based on the rank. In this document, we will focus on Semi-Supervised anomaly detection, so we will prefer scoring observations. The scoring can then easily be used for labelling by choosing an appropriate threshold for the score. Realisations with lower scores than the threshold will be considered as normal, and those above will be considered as anomalies.

3.2 Copula-based anomaly scoring and localization by Horváth et al. (HKMN)

Horváth et al. (2020) presented an algorithm relying on copula dependence trees to detect anomalies. The algorithm is based on probabilistic modelling. The idea is to determine the joint distribution of the variables and to define an anomaly score per observation relying on their densities. The main feature of the algorithm is the ability to identify the subspace where the anomalous observations deviate from the regular observations. This characteristic allows us to determine which variables are involved in the anomalous behaviour. Localizing anomalies could be very useful and lead to practical applications. For instance, an application would be to advise an operator to perform maintenance on a specific equipment of a production line following an anomalous behavior in the features related to this equipment. In the following of the document,

this method presented by Horváth et al. (2020) will be referred to using its authors' initials as the *HKMN* method.

The HKMN method could be categorized as semi-supervised anomaly detection algorithm. In other words, it models a class of observations considered as normal during the fitting process. The purpose of the model is then to flag all future observations that differ from this class. The HKMN could also be used as an unsupervised anomaly detection algorithm.

In this section, we will review the algorithm in detail and describe its implementation. Simultaneously, we will compare the authors' choices and approach with related techniques like the Chow-Liu algorithm from Section 1.3 and the pair copula constructions from Section 2.5.

3.2.1 Decomposition of high dimensional space into two-dimensional spaces

As we already discussed in previous chapters, the main challenge of the HKMN is to model the joint high-dimensional distribution. The higher is the dimensionality, the more complex the modelling. The authors tackle this complexity by decomposing the feature space into two-dimensional probability distributions. This decomposition is inspired by the work of Chow and Liu (1968) presented in Section 1.3 and is also comparable to the approach of dependence trees used in pair copula constructions presented in Section 2.5.

To determine which pairs of variables to choose in the decomposition, Horváth et al. (2020) follow the path of Chow and Liu which consist in choosing pairs retaining as much information as possible. The approach is to build a maximum information tree T where information is measured by mutual information. To follow in the footsteps of Chow and Liu while using continuous data rather than discrete ones, Horváth et al. (2020) proceeded to partition each feature into the same number of intervals. Each interval contains the same number of observations. With the mutual information between two variables (X_i, X_j) defined as

$$I(X_i, X_j) = H(X_i) + H(X_j) - H(X_i, X_j)$$

where $H(X) = -\sum P(x_i) \log P(x_i)$ is the entropy, if intervals contain the same number of observations, the maximal mutual information only depends on $\sum_{(i,j)\in T} H(X_i, X_j)$. The authors then weigh the edges of the fully connected graph by $H(X_i, X_j)$, called the cross-entropy, and use a minimum spanning tree algorithm (MST) such as discussed in Section 1.1.2. As a result, they obtain a spanning tree whose edges and underlying bivariate distributions capture as much information as possible.

This process of obtaining a maximum information tree is also performed to obtain copula-trees in pair copula constructions. The algorithm of Dißmann et al. (2013) that constructs the vine and its components also relies on maximum information trees. Their structures are selected with MST. Regarding the weight criteria when building vine copulas, there are four common choices in the literature:

- Empirical Kendall's τ (Czado, Schepsmeier, and Min 2012)

- AIC of copulas fitted for each pair of variables (Czado 2019)
- Degrees of freedom of Student's t pair copulas fitted for each pair of variables (Mendes, Semeraro, and Leal 2010)
- p-value of a copula goodness of fit test (Czado, Hofmann, and Jeske 2013)

Therefore, using the mutual information criterion to build the dependence tree with continuous variables is not common. The authors mention that Kendall's τ could also be used to weigh edges. They note that dependence tree structures remained identical when using Kendall's τ or mutual information in all of their studied cases. They favored mutual information arguing that Kendall's τ could be complex to compute with big data. I could not find a citation in the literature to support that claim.

To summarize the approach taken by Horváth et al. (2020), the authors start by building the dependence tree structure as in the Chow-Liu algorithm. They then resume by adapting Chow-Liu algorithm to continuous features and by combining it with copula theory until they obtain the probability density function,

$$f_{CL} = \prod_{i=1}^{d} f(x_i) \prod_{e \in T} c_{j(e)k(e)}\big(F_{j(e)}(x_{j(e)}), F_{k(e)}(x_{k(e)})\big). \tag{3.1}$$

where x_i, x_j and x_k are realisations of random variables represented by the vertices $i, j, k \in V = \{1, \ldots, d\}$ and $e = \{j, k\}$ is an edge of the maximum information tree T. This result is actually the density of a single copula-tree such as in Theorem 1 of Section 2.4.

Once the dependence tree is constructed, the high-dimensional space has been reduced into a combination of two-dimensional spaces. The goal is now to infer the multivariate density function following these steps:

1. For each edge of the tree, fitting a bivariate copula
2. For each vertex of the tree, fitting a univariate marginal distribution
3. Infer the density of the bivariate distribution of each edge using the pair copula and the marginals
4. Create an anomaly score from the joint probability distribution

The different steps of HKMN are presented in Figure 3.1. This figure is taken from the original article (Horváth et al. 2020).

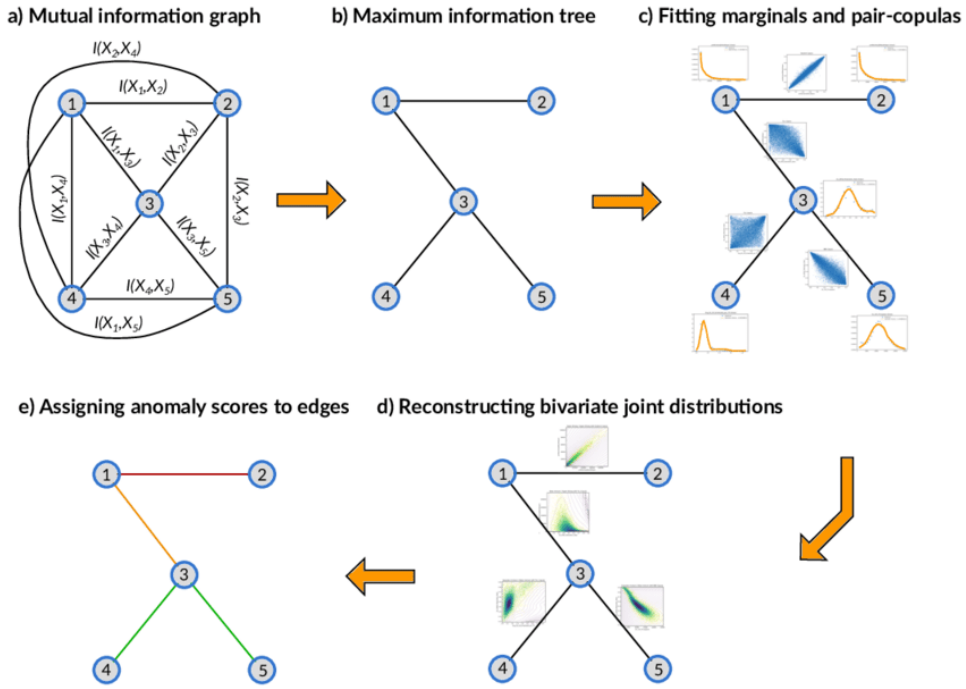


Figure 3.1: Concept of the presented anomaly scoring method

3.2.2 Fitting bivariate copulas

The empirical approximations of the bivariate copula are computed for each edge. The authors use the Semiparametric Estimation Procedure of Dependence Parameters introduced by Genest, Ghoudi, and Rivest (1995). To do so, for each edge, the random variables of the bivariate vector are transformed into pseudo-observations. In other words, observations are replaced by the normalized ranked data such as discussed in Section 2.1. This is a common practice to obtain a vector where variables are uniformly distributed and their joint values are realizations of their copula distribution.

The authors then find the best fitting copula model by repeatedly fitting different families of copulas and selecting the one with the highest likelihood. For this task, the authors relied on the vinecopulib package[1]. This package fits various copulas from the Archimedean family, Elliptical copulas and non-parametric copulas.

[1] https://github.com/vinecopulib/vinecopulib

3.2.3 Fitting univariate margins

The fitting process of marginals for each variable is approached similarly as the fitting of copulas. A set of distribution families are defined. For each variable, the authors repeatedly fitted different distributions to the observations and kept the one maximizing the likelihood. The following univariate distribution families were the candidates for fitting the marginals:

- Simple univariate distributions like the exponential, student-t, log-normal and the Pareto distributions. The fitting of these univariate distributions is easy when using the maximum likelihood estimators. The advantage is then the efficiency. However, these are generally not flexible enough to accurately fit real-world data.
- Gaussian mixture models. These models are highly flexible. There are often used to fit real datasets. Yet, Gaussian mixtures tend to perform poorly when fitting heavy tailed data.
- Hyper-Erlang distributions. The Hyper-Elang is a mixture of Erlang distributions which is a specific case of the Gamma distribution. The authors noticed far better fitting while using the Hyper-Erlang in comparison to fitting mixtures of Gamma distributions. This is a surprising statement as the algorithms used for fitting Gamma mixtures should be able to fit a specific case such as the Hyper-Erlang. As the authors did not provide further detail, this statement could not be investigated.
- Mixture of Beta distributions. These models were preferred when features only exhibit values in the interval $[0, 1]$.

3.2.4 Infer joint densities

By combining both the marginals and the pair-copula, the two-dimensional joint density function for the edge $\{X_i, X_j\}$ is computed as follows,

$$f_{X_i,X_j}(x_i, x_j) = c_{X_i,X_j}\big(F_{X_i}(x_i), F_{X_j}(x_j)\big) \cdot f_{X_i}(x_i) \cdot f_{X_j}(x_j), \tag{3.2}$$

while the d-dimensional joint density is computed by the product defined in equation (3.1).

3.2.5 Anomaly score

For multivariate distributions with multimodal density functions, defining an anomaly score is complex. The authors rely on the theory of the minimum volume set such as defined in Einmahl and Mason (1992) and Polonik (1997). The concept is to separate the feature space in two parts. The observations falling in the minimum volume set are flagged as normal and those laying outside of the minimum volume set are flagged as anomalous. The authors are not using this straightforward categorisation as they are interested in scoring observations. They will rather follow in the footsteps of Clémençon and Thomas (2018) where a Mass Volume Curve (MV curve) is defined. In their work, Clémençon and Thomas (2018) define an ordering upon which each

observation is ranked according to how rare it is. Based on these concepts, the scoring function and the level set are defined. Let us consider a d-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_d)$ taking values in $\chi \subset \mathbb{R}^d$, with a bounded probability density function denoted by $f(x)$.

Definition 13. *A scoring function is any measurable function $s : \chi \to \mathbb{R}_+$ that is integrable with respect to the Lesbesque measure.*
A level set corresponding to a scoring function s and a level t is given by $A_t = \{\mathbf{x} \in \chi | s(\mathbf{x}) \geq t\}, t \in [0, \infty]^2$.

The authors then proceed to define the most important notions which are the mass and the volume of a level set.

Definition 14. *For a scoring function s and a level t, $\alpha_s(t) = P(s(\mathbf{X}) \geq t)$ is called the mass of the level set A_t.*

Definition 15. *For a scoring function s and a level t, $\lambda_s(t) = \lambda(\mathbf{x} \in \chi | s(\mathbf{x}) \geq t)$ is called the volume corresponding to a level t , with respect to the Lesbesque measure.*

These concepts are illustrated in Figure 3.2 taken from Horváth et al. (2020) where a univariate density is represented with the volume and mass at level t.

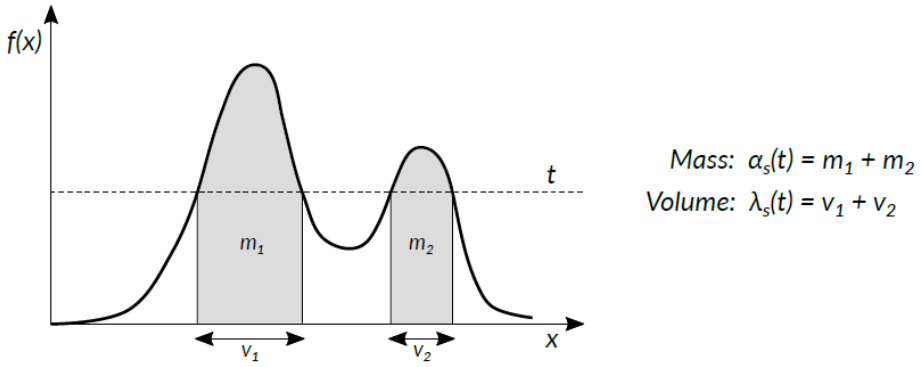


Figure 3.2: Mass and volume on a bi-modal density function

As the focus of the article is anomaly detection, the problem is addressed with the minimum volume sets defined as

$$A_\alpha^* = \underset{A \subset \chi}{\arg\min}\, \lambda(A), \text{ such that } P(\mathbf{X} \in A) \geq \alpha. \tag{3.3}$$

This represents the smallest possible volume with a mass α. Based on these notions, the authors proceed to define the theoretical anomaly score as follows.

Definition 16. *The theoretical anomaly score of a realization $\mathbf{x}$ is defined by*

$$\mathcal{A}(\mathbf{x}) = \inf\{\alpha : \mathbf{x} \in A_\alpha^*\}. \tag{3.4}$$

This score $\mathcal{A}(\mathbf{x})$ is then a value between 0 and 1. The larger the value of $\mathcal{A}(\mathbf{x})$, the rarer the observation $\mathbf{x}$ is. The problem remaining with the anomaly score is its numerical computation. To overcome this issue, Horváth et al. rather rely on an approximate anomaly score that they compute with a Monte-Carlo integration. They generate m i.i.d random samples from $f(\mathbf{x})$, denoted by $\tilde{x}^i$, $i = 1, \ldots, m$. Afterward, they compute the density $f(\tilde{x}^i)$ of each $\tilde{x}^i$. This vector of $f(\tilde{x}^i)$ is an univariate discrete random variable τ whose values have the probability $p_i = \frac{1}{m}$. The approximate anomaly score is obtained by the complementary distribution function of this random variable denoted by $\bar{G}(t) = P(\tau > t)$ at $t = f(\tilde{x}^i)$.

Definition 17. *The approximated anomaly score of any realization $\boldsymbol{x}$ is defined by*

$$\hat{\mathcal{A}}(\mathbf{x}) = \bar{G}(f(\mathbf{x})) = \frac{1}{m} \sum_{\tau > f(\mathbf{x})} 1. \tag{3.5}$$

In this case, the approximate anomaly score takes values between 0 and 1. Moreover, its value will be larger for rare events. The approximation of the anomaly score is illustrated in Figure 3.3 taken from Horváth et al. (2020).

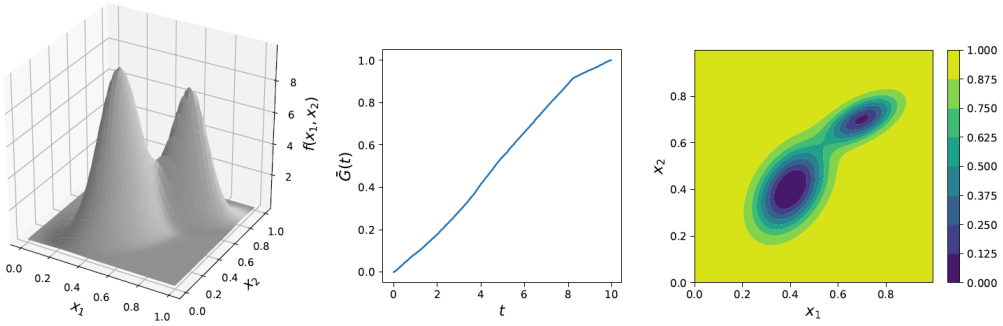


Figure 3.3: Obtaining anomaly score based on the pdf, in case of a 2-dimensional Gaussian mixture distribution. Left-most: the density, in the middle: $\bar{G}(t)$, right-most: the anomaly scores of the two-dimensional realizations.

This process to score observations can be applied either on the d-dimensional density function of equation (3.1) or on each of the bivariate density such as in equation (3.2). The authors choose to apply it on each bivariate density to be able to localize anomalous behaviour. The other advantage of this choice is that missing values for one of the variables does not prevent to detect anomalous behaviours for other pairs of features. In that case, there would be an edge of the dependence tree which would remain empty.

This anomaly score still has flaws. There is not one single score per observation. To compare the performance of their algorithm with other existing methods, the authors created an overall anomaly score. They did not use the anomaly score with the equation (3.1) but rather defined a global score,

$$\tilde{\mathcal{A}}(\mathbf{x}) = \frac{1}{d-1} \sum_{e \in T} -\log(1 - \hat{\mathcal{A}}(\mathbf{x_e})), \tag{3.6}$$

where $\mathbf{x_e}$ is the two-dimensional vector containing the components of $\mathbf{x}$ represented by the edge e of the three T. The logarithmic transformation, $l(s) = \log(1 - s)$, ensures that only rare observations will get high scores. The equation (3.6) is then the opposite of the mean over the $d - 1$ transformed edge scores.

3.3 Vine copula anomaly detection (VCAD)

In this section, we will present the vine copula anomaly detection (VCAD). In their paper, Horváth et al. (2020) focused on the first tree of a regular vine copula to score an anomaly. The obvious advantage of this method is their localization capability on a single tree. One of the disadvantages is that it assumes conditional independence of variables whose nodes are not directly adjacent, given a set of variable nodes on their path. One could deepen the HKMN method by selecting additional trees to model the missing conditional dependencies such as presented in Section 2.5. Indeed, the conditional dependence structure would then be modelled by the higher-level trees of the vines rather than assumed independent as for the HKMN standalone copula-tree. In this case, the joint density distribution of the vine would be the one presented in equation (2.1)

$$\begin{aligned} f_{1,\dots,d}(x_1, \dots, x_d) &= f_1(x_1) \dots f_d(x_d) \\ &\prod_{i=1}^{d-1} \prod_{e \in E_i} c_{\mathcal{C}_{e,a}\mathcal{C}_{e,b};D_e}(F_{\mathcal{C}_{e,a}|D_e}(x_{\mathcal{C}_{e,a}}|\mathbf{x}_{D_e}), F_{\mathcal{C}_{e,b}|D_e}(x_{\mathcal{C}_{e,b}}|\mathbf{x}_{D_e})), \end{aligned} \tag{3.7}$$

respecting the notations introduced in Theorem 11.

For the selection of the tree T_j, $2 \leq j \leq d$, the process is similar to the one of HKMN, except that the possible edges assessed by the minimum spanning tree algorithm must have their vertices respecting the proximity condition of Definition 10. This methodology for building the vine is actually known as Dißmann's algorithm (Dißmann et al. 2013) in the regular vine copula literature.

One could simplify the vine by omitting some higher-level trees, such as proposed by Brechmann, Czado, and Aas (2012). This step has the advantage to simplify the vine, which might reduce overfitting. From the resulting truncated vine copula, a single anomaly score would be computed for each observation. It differs from HKMN where observations receive anomaly scores for each edge which are then aggregated. In VCAD, a single global score is computed based on a Monte Carlo integration of the MV curve. The Monte Carlo integration of the MV Curve is similar to the one used in HKMN except that it is performed on the d-dimensional density distribution $f_{1,\dots,d}(x_1, \dots, x_d)$ of equation (3.7) rather than on each edge bivariate distribution. The global score is then similar to $\hat{\mathcal{A}}(\mathbf{x})$ in Definition 3.5 where $\mathbf{x}$ is not a 2-dimensional vector but a d-dimensional one.

In the following sections of this master's thesis, the HKMN and VCAD will be implemented and compared together. Subsequently, the performance of these two models shall be compared with those of other anomaly detection algorithms.

Part II

Implementation and discussion

In the first part of this master's thesis, we presented the theoretical background necessary to the understanding of HKMN and VCAD. In this second part, we will use this knowledge to implement HKMN and VCAD. The implementations of the algorithms will be done with the software R and will be described in Chapter 4 and Chapter 5. For each of these implementations, we will test them on a given dataset in order to verify their performance and understand their behaviour. This step will involve the analysis of graphs and various performance criteria. Later, in Chapter 6, these findings will be deepened when we will compare the efficiency and the behaviour of these two algorithms with other recognized anomaly detection methods. Finally, we will discuss the advantages and limitations of HKMN and VCAD in an attempt to answer the question: is anomaly detection with copula-trees worth it?

Chapter 4

HKMN implementation

The HKMN implementation of this master's thesis will follow the one of Horváth et al. (2020). Yet, this implementation may deviate from the one of the article. In this case, the reasons will be clearly stated. To describe the steps of the algorithm, it will be applied to the dataset *Breast Cancer Wisconsin (Diagnostic)* from UCI Machine Learning Repository (Dua and Graff 2019).

The dataset features are computed from digitized images of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the images (Dua and Graff 2019). All thirty features describing the cell nuclei are continuous and real-valued.

The dataset is divided in both training and test sets. The training set is composed of 200 observations labelled as *inliers* while the test set is composed of 369 observations, 157 inliers (43%) and 212 outliers (57%). The goal is to build the HKMN on the training set and to test its anomaly detection, scoring and localisation abilities on the test set.

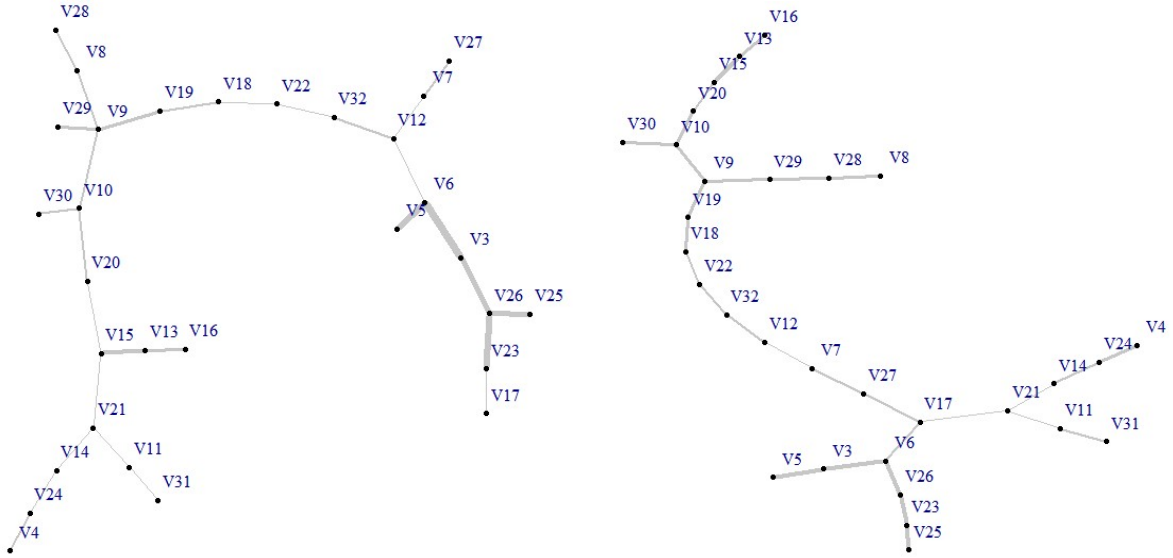
As a reminder from Section 3.2, there are five steps to HKMN. The practical implementation of these steps will be described in different sections of this chapter:

1. Building the maximum information tree is the starting point and it will described in Section 4.1.
2. In Section 4.2, the marginals will be fitted.
3. Based on the edge defined in Section 4.1, the copulas will be fitted in Section 4.3.
4. In Section 4.4, we use the marginals and copulas to construct the bivariate distributions that will be used to sample and score observations. The samples are already created and visualized at this step to ensure that they share the same shape and characteristics as the original data.
5. The final step of the algorithm is to derive the edge anomaly scores from the bivariate distributions and the samples, and to aggregate them into an overall score. This is performed in Section 4.5.

4.1 Building the maximum information tree

HKMN starts by building the maximum information tree from a full graph whose nodes represent features of the dataset. As explained in Section 3.2.1, Horváth et al. (2020) choose to partition the features data, so that partitions share a common entropy. This enables to use the cross entropy between variables as weight of edges for the minimum spanning tree algorithm.

In this implementation, edges are weighted with the mutual information rather than the cross entropy. The mutual information is computed with the package *infotheo* (Meyer 2008). As a reminder in Horváth et al. (2020), the authors explained that Kendall's tau could be used also to build the maximum information tree. They claimed that the tree structure remained identical in all studied cases using either mutual information or Kendall's tau. To verify their statement, the maximum information tree for the WDBC dataset was build with both criteria. Using Prim's minimum spanning tree algorithm with the negative of the mutual information as weight, we obtain the tree 4.1a. The tree obtained with absolute Kendall's tau is illustrated in 4.1b.



(a) Build with mutual information

(b) Built with absolute Kendall's tau

Figure 4.1: Maximum information trees for the Breast Cancer Wisconsin (Diagnostic) dataset built with the mutual information and Kendall's tau. The edges are weighted by absolute Kendall's tau or mutual information. The wider is the edge, the more dependent the vertices are.

These graphs invalidate the authors' claim that choosing mutual information or Kendall's tau does not influence the tree structure. For instance, the path $V25 - V23 - V26$ with Kendall's tau becomes $V25 - V26 - V23$ in the mutual information tree. Other variations can be noticed, such as $V17$ in 4.1b which is at the center of tree edges, but is left out with a single edge in 4.1a.

Although the structures remain mostly similar, divergences exist with both highly dependent

variables ($V25 - V26 - V23$) and almost independent variables ($V17 - V \dots$). Whether these differences impact the anomaly detection ability of the model should be explored in more depth. One can already formulate hypotheses to explain the differences. The method requires that all variables are linked together in a tree. Yet, variables that are independent of all others could make the structure highly volatile. For example, $V17$ exhibits low dependence with other variables. So, the criterion used to build the tree impacts its position greatly. An interesting question is whether an edge to $V17$ is relevant. On the other hand, for highly dependent variables such as in the path $V25 - V26 - V23$, one can question the utility of having perfectly correlated features which might indicate colinearity. Rather than questioning the relevance of the edge such as for $V17$, one can question the relevance of some vertices. For instance, should $V25$ and $V23$ be discarded to only keep $V26$? To illustrate the implementation, three edges and their vertices will be analysed in the following sections:

1. $V3 - V6$ with a mutual information of 1.53 (highest dependence)
2. $V9 - V29$ with a mutual information of 0.73 (medium dependence)
3. $V11 - V31$ with a mutual information of 0.32 (low dependence)

4.2 Fitting marginals

To obtain bivariate distributions upon which anomaly scores rely, margins are needed for each feature. There are two common options for distribution fitting; either, using parametric or nonparametric distributions. Both have pros and cons. The benefit of parametric fitting is its simplicity, as there are few parameters to optimize, and these parameters can be obtained by simple algorithms such as maximum likelihood estimation. On the other hand, nonparametric distributions are very flexible and can then easily model real-life data. However, the number of parameters is consequent. These distributions are more likely to overfit when there is few data.

In this implementation, the parametric approach was preferred such as in Horváth et al. (2020). The following distributions were fitted when the domain of the feature allows it:

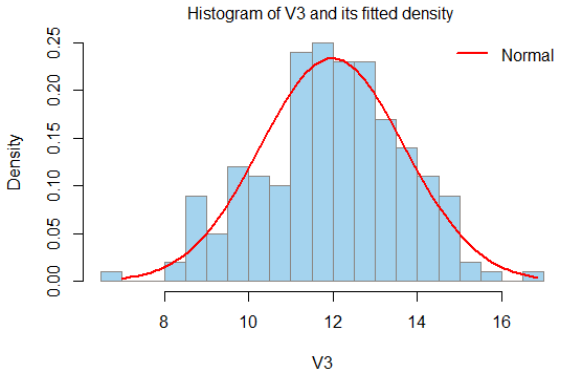
1. Beta
2. Gamma
3. Log-normal
4. Normal
5. Student
6. Weibull
7. Pareto

8. Mixtures of beta distributions
9. Mixtures of gamma distributions
10. Mixtures of normal distributions

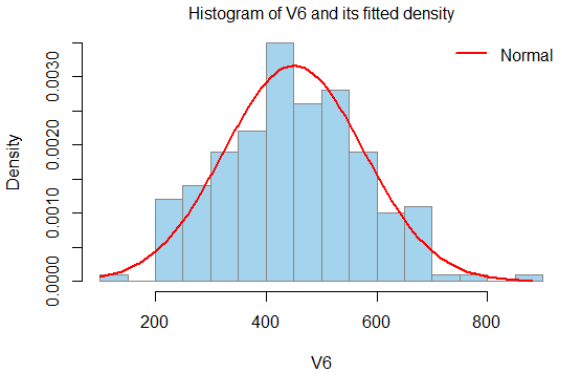
The seven first distribution families were fitted with maximum likelihood estimation with the package *fitdistrplus* (Delignette-Muller and Dutang 2015). The mixtures of distributions are composed of three components. The package *RBesT* (Weber 2020) was used to fit the mixtures through Expectation–Maximization algorithm (Dempster, Laird, and Rubin 1977). Mixture models are good compromises between parametric and nonparametric methods. They have the advantage to gain in flexibility compared to classical parametric models as they can model multimodal distributions.

Each distribution family and mixture was fitted to features. For each feature, the model with the lowest AIC was then selected (Akaike 1974). This diverges from Horváth et al. (2020) where the selection criterion was the highest likelihood. Likelihood as model selection criterion is not recommended. Indeed, it favours complex models which tend to overfit. AIC tries to balance the goodness of fit with a penalty on the number of parameters to encourage model simplicity.

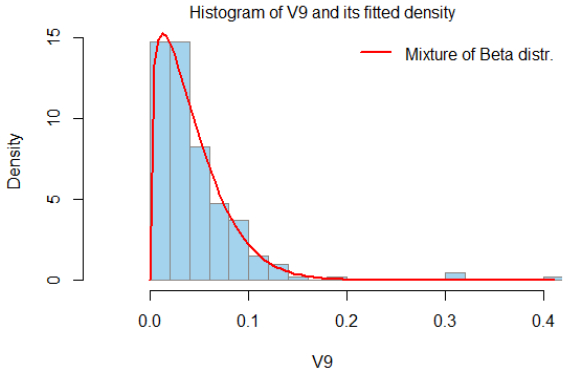
To illustrate the univariate distribution fitting process, histograms of variables $V3$, $V6$, $V9$, $V29$, $V11$ and $V31$ are displayed with their selected fitted densities in the Figure 4.2. One can notice that the fitting process is working as expected. The fitted densities reflect very well the histograms. The more complex distributions are generally modelled with mixtures such as in Figures 4.2c and 4.2e. In Figure 4.2e, the use of the mixture to compensate for the lack of flexibility of classical univariate distribution families is evident.



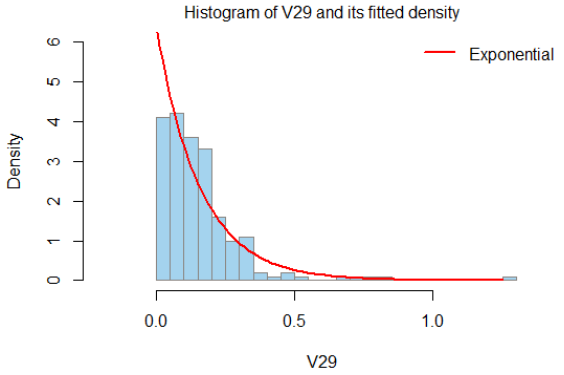
(a) Histogram of V3 with its fitted Normal density



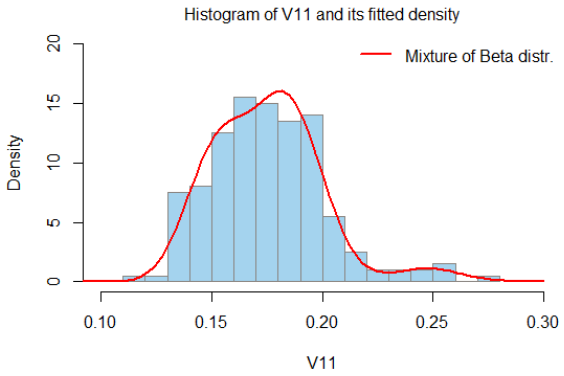
(b) Histogram of V6 with its fitted Normal density



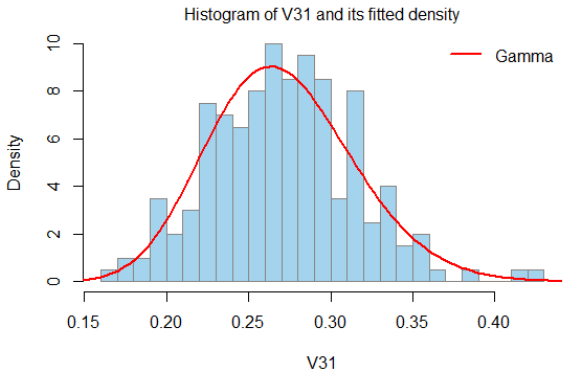
(c) Histogram of V9 with the density of its fitted Beta mixture distribution



(d) Histogram of V29 with its fitted Exponential density



(e) Histogram of V11 with the density of its fitted Beta mixture distribution



(f) Histogram of V31 with its fitted Exponential density

Figure 4.2: Histograms and fitted density distribution for features V3, V6, V9, V29, V11, V31

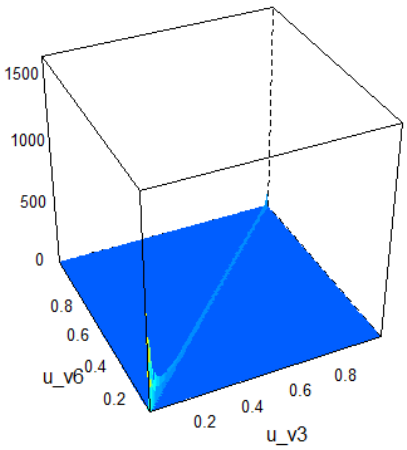
4.3 Fitting bivariate copulas

Bivariate copulas are needed to build the bivariate distributions. For each edge, based on the empirical data of the two features forming the edge, a bivariate copula is fitted. This step was performed with the package *rvinecopulib* (Thomas Nagler and Vatter 2021) such as in Horváth et al. (2020). This package fits parametric bivariate copulas (elliptical or Archimedean) using a maximum likelihood estimator, or nonparametric bivariate copulas (Transformation kernel) using local-likelihood approximations. This latter method requires to fit many parameters which is time and resource consuming. However, it enables to fit dependence relations which are too asymmetrical to be modelled by Archimedean or elliptical copulas (Geenens, Charpentier, and Paindaveine 2017).

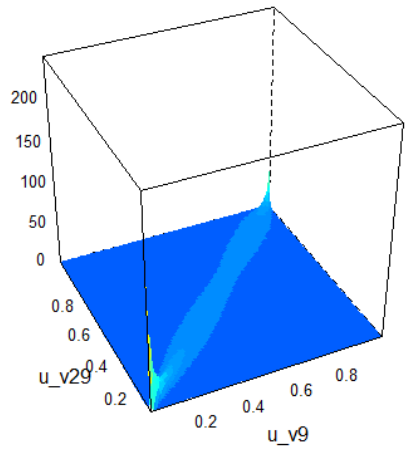
In Horváth et al. (2020), the authors selected the copula family based on maximum likelihood which means they do not penalize complex models and risk to overfit. In this implementation, the selection criterion for the copula family was AIC to palliate to this lack from the original article. For the three analyzed edges of this section, we obtained the following copulas:

- $V3 - V6$: an Archimedean copula, Clayton-Gumbel (BB1)
- $V9 - V29$: an Archimedean copula, Joe-Clayton (BB7)
- $V11 - V31$: an Archimedean copula, Gumbel

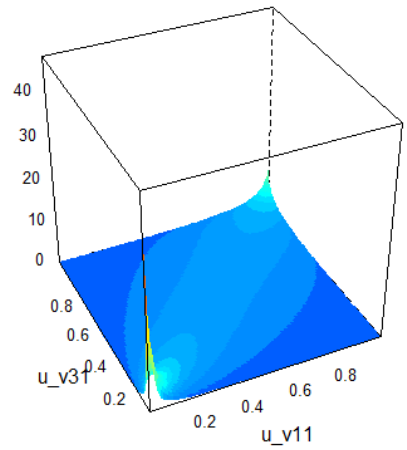
These copula densities are illustrated in Figure 4.3. As their shapes were relatively similar, another edge $V17 - V23$ is also represented. The relation between the variable $V17$ and $V23$ is modelled by a nonparametric copula. The relevance of edges containing $V17$ were discussed in Section 4.1. As the variable $V17$ exhibits almost no dependence with any other variables, its dependence is complex to model. That is probably why a nonparametric copula was needed to fit this edge.



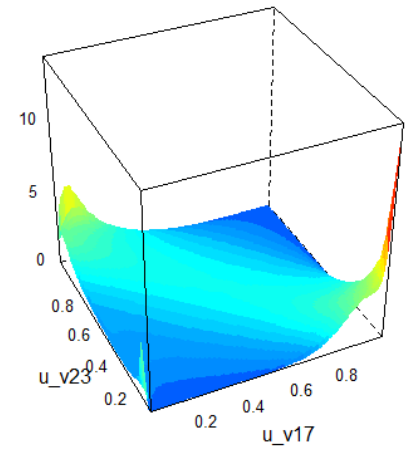
(a) The Clayton-Gumbel copula modeling the dependence between V3 and V6



(b) The Joe-Clayton copula modeling the dependence between V9 and V29



(c) The Gumbel copula modeling the dependence between V11 and V31

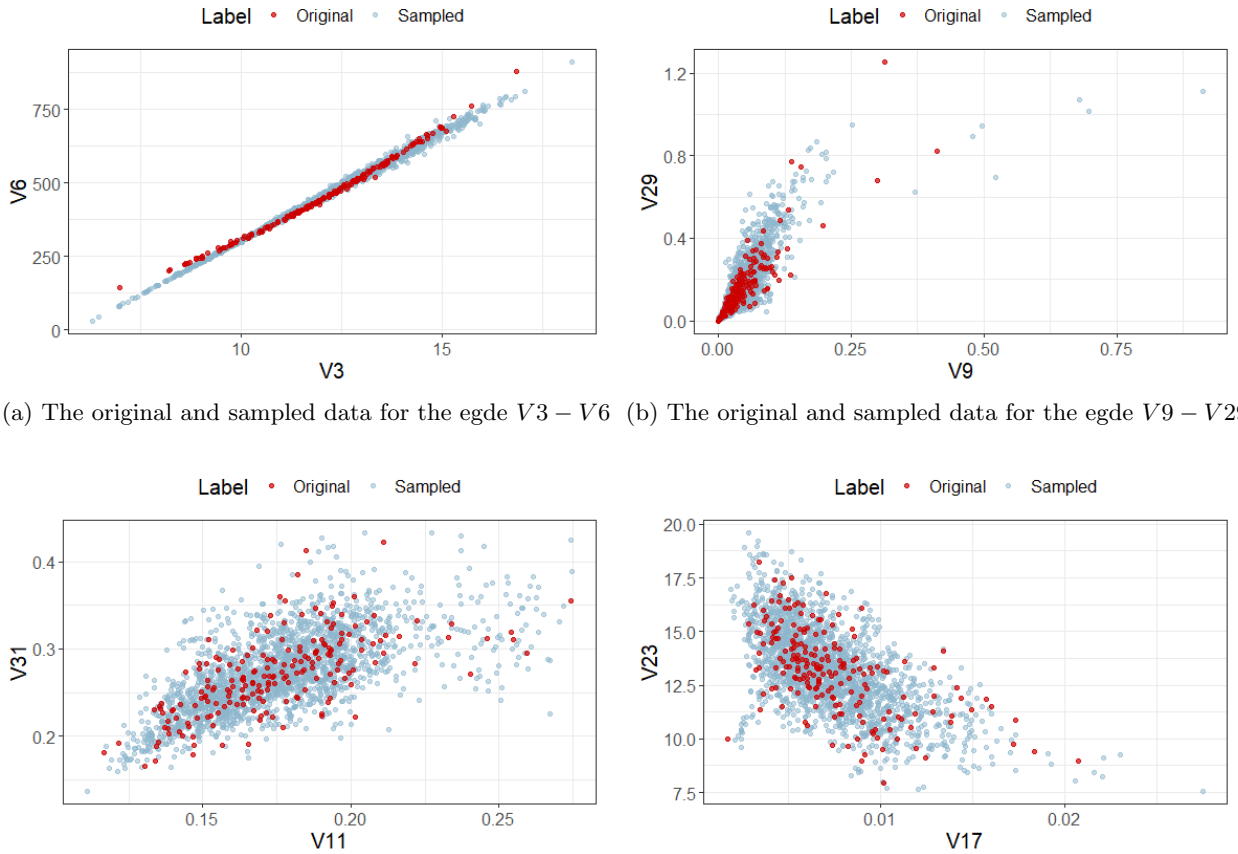


(d) The nonparametric copula modeling the dependence between V17 and V23

Figure 4.3: The fitted copula densities of four edges

4.4 Bivariate distribution and sample generation

The equation (3.1) combines the marginals from Section 4.2 and the copulas from Section 4.3 to obtain a bivariate density distribution for the edge. For the four discussed edges, samples from their bivariate distributions are depicted in Figure 4.4.



(a) The original and sampled data for the egde $V3 - V6$ (b) The original and sampled data for the egde $V9 - V29$

(c) The original and sampled data for the egde $V11 - V31$(d) The original and sampled data for the egde $V17 - V23$

Figure 4.4: Fitted copulas for four edges of the maximum information tree

In Figure 4.4, one can notice that the original data are faithfully reproduced. The samples and the real data intermingle very well and share the same shapes. From these graphs, one can assume that the modelled bivariate distributions are good approximations of the true ones. Yet, some small differences can be observed. Mainly in Figure 4.4a, the $V3 - V6$ sample seems to be perfectly linearly dependent whereas in the original data the dependence relation between $V3$ and $V6$ is slightly curved. This might lead to errors when evaluating the anomaly scores for this edge. Indeed, very high values or very low values of $(V3, V6)$ will easily be flagged as anomalies as they will diverge from the perfectly linear dependence expected by the model. This will certainly lead to some false positive cases for this edge.

4.5 Anomaly scoring

The HKMN model resulting from the previous steps will now score the test observations. As explained in Section 3.2.5, the anomaly score of an observation for a specific edge is determined by :

1. Computing the density of the observations from the bivariate distribution of the edge
2. Performing a Monte-Carlo integration to evaluate the MV curves of the edges distributions. In this implementation, Monte-Carlo integrations rely on 2000 samples for each edge.
3. Using the estimated MV curves, define the probabilities of the observation for each edge as the anomaly scores

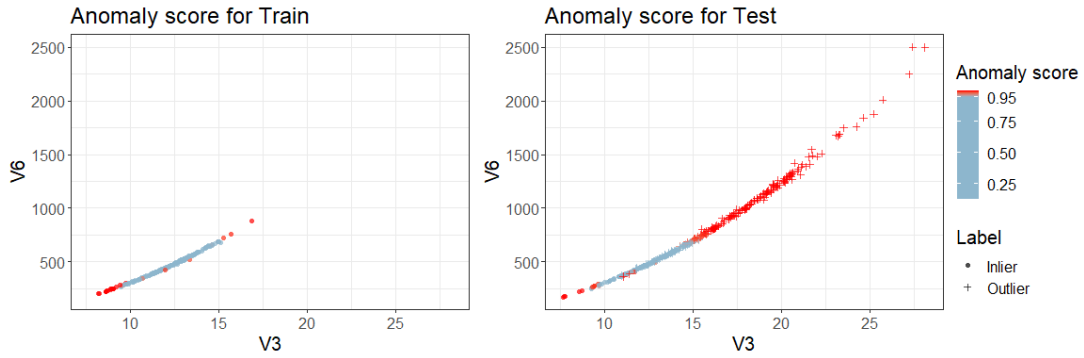
Consequently, an observation will receive many scores, one by edge of the maximum information tree, some indicating an anomaly and others pointing toward normality. To resolve the potential conflicts, Horváth et al. (2020) reduce these edge anomaly scores to a single score per observation by aggregating them (see Section 3.2.5).

The scoring step is interesting to compare the performance of HKMN with other methods. In the original article, the anomaly score of the HKMN is compared to scores from different semi-supervised methods and plotted to visualize the correlation between the HKMN scores and those of the others. Yet, this part of the article is not deepened and does not allow to properly compare the anomaly detection ability of HKMN. The only insight conveyed by the article is the strong positive correlation existing between the HKMN scores and those of the other methods. To go further, the anomalies should be clearly flagged. However, deciding which observations to flag as inliers or outliers is complicated. The threshold between normality and abnormality is unclear. In supervised learning, cross-validation could be performed to select the best threshold. Yet, HKMN is semi-supervised. In this context, the test set is not labelled and cross-validation is not possible. To address this concern, the model trainer must pick an appropriate threshold. An approach is to decide on a False Positive Rate (type I error) the model trainer is willing to accept. This is the method performed in this section, although the test set is labelled in the presented case. This set up allows to compare the algorithm performance with other methods while staying consistent with semi-supervised context. In this section, the tolerated FPR on the training set will be 5%. In the following sections, the area under the ROC curve (AUC) will also be analyzed as it is a good indicator of the anomaly detection capability and allows to compare the performance of different models without having to choose a threshold. An interesting interpretation of the AUC is given in Fawcett (2006) where it is transposed into the anomaly detection domain: The AUC is the probability that an anomaly detection algorithm assigns a randomly chosen normal instance a lower score than a randomly chosen anomalous instance.

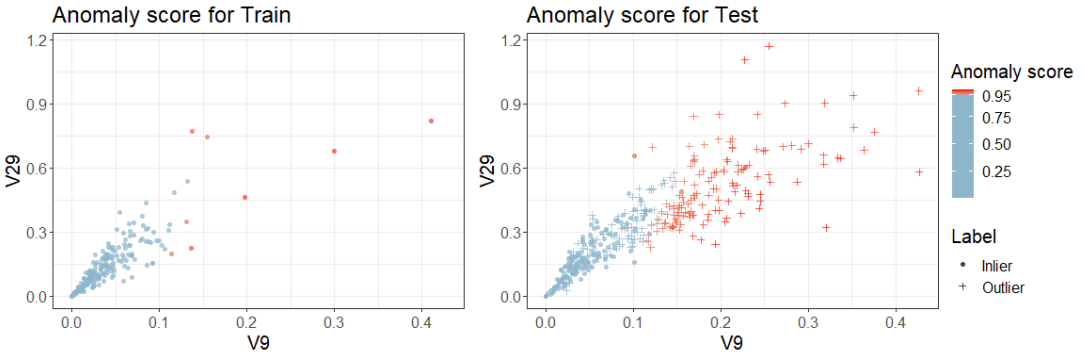
Firstly, the scores for the four investigated edges are computed. In Figure 4.5, the scores per observation are represented for both training and test set. By observing Figure 4.5, it is easily noticeable that anomalies tend to have higher anomaly scores than inliers. The edge $V17 - V23$

seems to be a very effective indicator of abnormality as most anomalies have a higher than 0.95 anomaly score, while inliers tend to be below the 0.95 threshold.

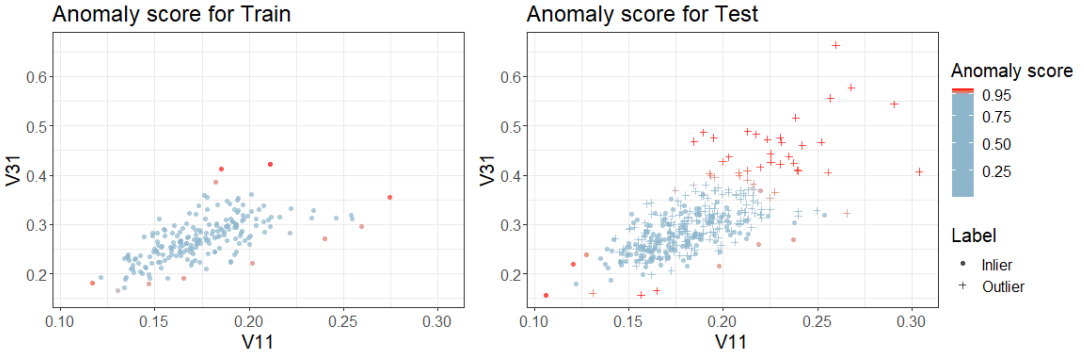
By aggregating the edge anomaly scores of the such as described in Section 3.2.5 and by defining the FPR on the training set to 5%, the threshold is 2.61. On the test set, 84% of anomalies are detected and the precision of the model is 87.8%. The confusion matrix for the test set is represented in Table 4.1. From this table, we observe that the model underestimates the number of anomalies. In Table 4.2, performance metrics describe the model classification behaviour. As suggested by the confusion matrix, the True Positive Rate (TPR or Sensitivity) is the lowest metric which confirms the underestimated number of anomalies. Yet, this metric is well compensated with a high value for the True Negative Rate (TNR or Specificity) and the precision. It is worth mentioning that these values remain subjective, as they only depend on the threshold defined by the model trainer.



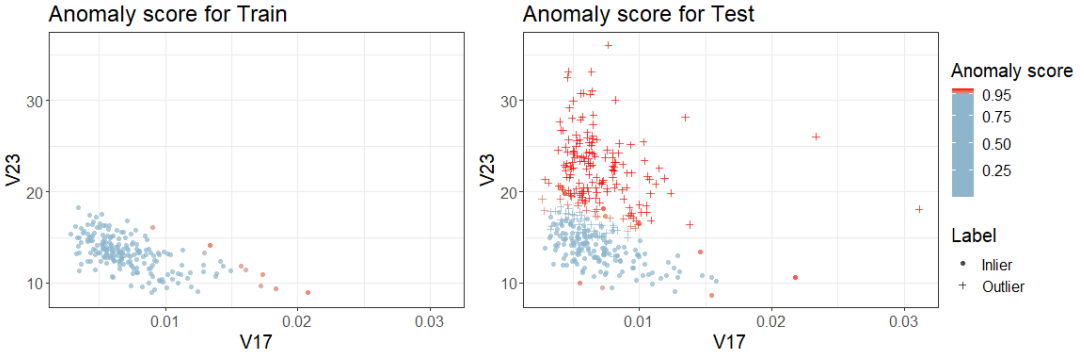
(a) Training and test scores for the edge $V3 - V6$



(b) Training and test scores for the edge $V9 - V29$



(c) Training and test scores for the edge $V11 - V31$



(d) Training and test scores for the edge $V17 - V23$

Figure 4.5: Training and Test observations for four edges and their anomaly score

	Real Positive	Real Negative	
Predicted Positive	178 (TP)	11 (FP)	189
Predicted Negative	34 (FN)	146 (TN)	180
	212	157	369

Table 4.1: Confusion matrix for the test set

	Definition	**Score on test set**
TPR, Sensitivity	$\frac{TP}{TP+FN}$	**0.840**
TNR, Specificity	$\frac{TN}{TN+FP}$	**0.930**
Precision	$\frac{TP}{TP+FP}$	**0.942**
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	**0.878**
F1 Score	$\frac{2 \times Precision \times TPR}{Precision+TPR}$	**0.888**
Youden Index	$TPR+TNR-1$	**0.770**
AUC	$\int ROC$	**0.969**

Table 4.2: Performance metrics for the test set

What would have been the optimal threshold if the False Positive and False Negative were considered equally costly? This could be answered with the Youden index, which is a summary statistic of the performance for dichotomous classification. It relies both on the TPR and on the TNR to evaluate the model performance. This is particularly useful when combined with a ROC curve to define the optimal threshold (Schisterman et al. 2005). In Figure 4.6, the optimal Youden index of 0.803 is displayed on the ROC curve. This point is associated with TPR of 0.925 and a TNR of 0.879. If this was converted to a FPR threshold on training set such as the one defined earlier, the threshold would be 12.5%. Another important metric is the AUC which in this case is 0.968. As the maximal AUC possible is 1, this model displays a very good performance.

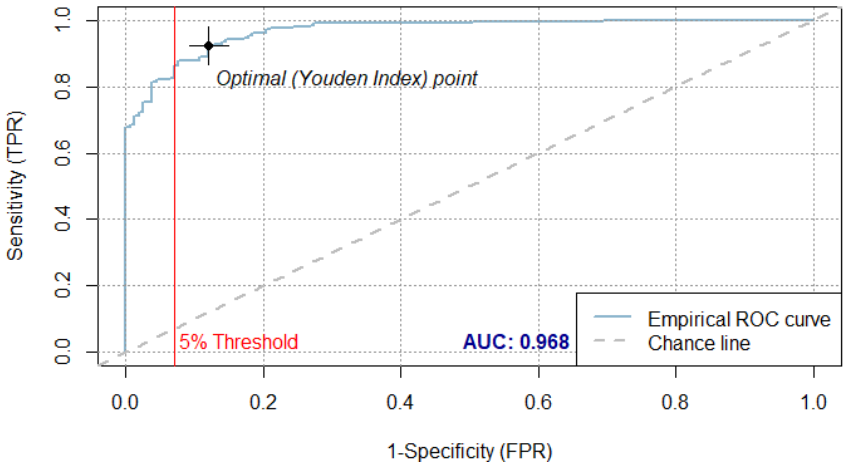


Figure 4.6: The ROC curve associated with the HKMN model fit, the 5% threshold and the optimal Youden index

Another very interesting characteristic of the algorithm is its anomaly localisation capability. Indeed, for each observation, the copula dependence tree can be illustrated with its anomalous edges highlighted, such as in Figure 4.7. These graphs could be meaningful in some domain where the localisation of the anomaly could lead to specific maintenance, for instance.

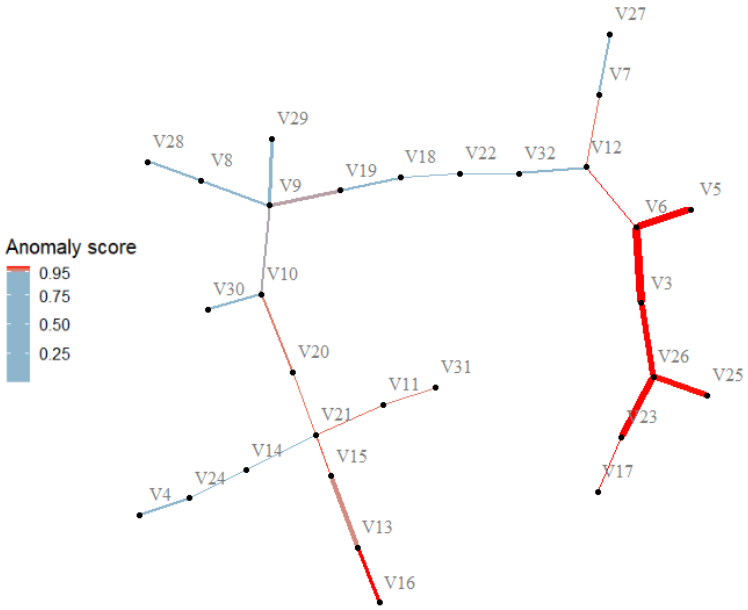


Figure 4.7: An anomalous instance illustrated with its edges colored according to their anomaly scores. The edge width represents the mutual information of the edge.

Chapter 5

VCAD implementation

In this section, the VCAD is implemented on the WDBC dataset. The purpose of this chapter is to compare the performance of this algorithm with HKMN. The hypothesis to be tested is that adding more trees to a vine would help detect anomalies. Each added tree would indeed model part of the dependence structure discarded by HKMN.

This implementation will be performed with the package *rvinecopulib* (Thomas Nagler and Vatter 2021). It has the advantage to build the full vine copula from the training dataset. There are, however, differences with the HKMN method:

1. The trees are built with Kendall's tau rather than mutual information. As demonstrated in Section 4.1, there exist differences between trees constructed with mutual information and Kendall's tau.
2. In *rvinecopulib*, the fitting of marginals is nonparametric. The consequence is that more observations are required to obtain stable and smooth fitted distributions.
3. The last difference with HKMN is the process to score anomalies. In HKMN, observations receive anomaly scores for each edge. A global score is then computed based on the edge scores. In VCAD, a single global score is computed per observation based on a Monte Carlo integration of the Mass Volume curve of the d-dimensional cumulative distribution.

The Monte Carlo integration of the MV curve is performed with 20 000 samples. For the sake of the comparison, three VCAD models will be explored:

1. a VCAD model with a fully grown vine.
2. a VCAD model with a truncated vine, the truncation level is chosen based on the Modified vine copula Bayesian information criterion (mBIC) (T. Nagler, Bumann, and Czado 2018).

3. a VCAD model with a truncated vine to a single tree. The purpose of this single tree VCAD is to compare two implementations of a single copula dependence tree anomaly detection, the HKMN and the single tree VCAD.

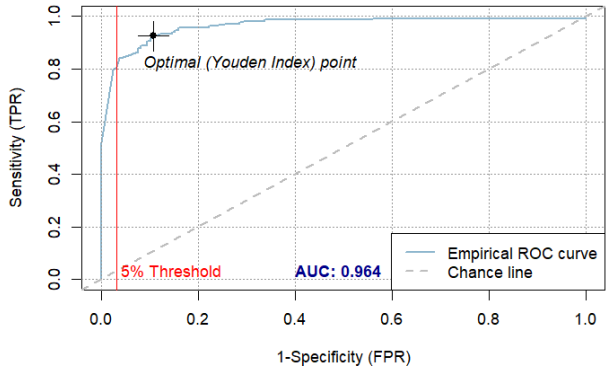
The analysis will be supported by confusion matrices and performance metrics for each model. Those are illustrated in Table 5.1 and Table 5.2. To discuss the results, we will take two approaches. The first one is to compare the models ability to differentiate normal and anomalous instances. To do so, the area under the curve (AUC), the integral of the ROC, is a good metric as explained in Section 4.5. The second approach will be to select a threshold (5%) and compare other metrics such as TPR, TNR, Precision, Accuracy, F1 score and Youden index. The purpose of this approach is to analyse more closely the decision of each model at this threshold and to understand the differences in classification behaviour between the models. The ROC curve is displayed with the 5% cut off threshold computed on the training test in Figure 5.1.

	VCAD single tree			**VCAD truncated with mBIC**			**VCAD fully grown**			**HKMN**		
	Real Positive	Real Negative	Total	Real Positive	Real Negative	Total	Real Positive	Real Negative	Total	Real Positive	Real Negative	Total
Predicted Positive	169	4	173	186	19	205	188	19	207	178	11	189
Predicted Negative	43	153	196	26	138	164	24	138	162	34	146	180
Total	212	157	369	212	157	369	212	157	369	212	157	369

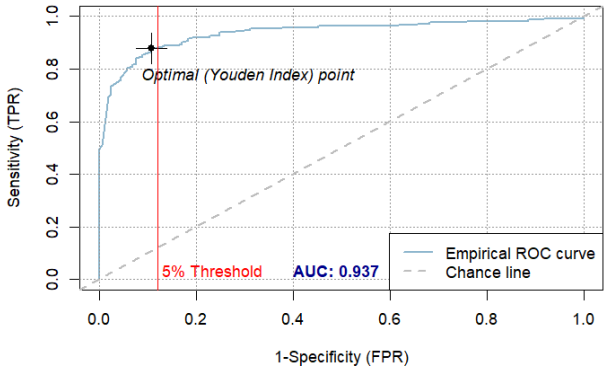
Table 5.1: Confusion matrices of three different VCAD models (1 tree, 13 trees and 30 trees) and HKMN for the WDBC dataset

	Definition	**VCAD Single tree**	**VCAD Truncated**	**VCAD Fully grown**	**HKMN**
TPR, Sensitivity	$\frac{TP}{TP+FN}$	**0.797**	**0.877**	**0.887**	**0.840**
TNR, Specificity	$\frac{TN}{TN+FP}$	**0.975**	**0.879**	**0.879**	**0.930**
Precision	$\frac{TP}{TP+FP}$	**0.977**	**0.907**	**0.908**	**0.942**
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	**0.873**	**0.878**	**0.883**	**0.878**
F1 Score	$\frac{2 \times Precision \times TPR}{Precision+TPR}$	**0.878**	**0.892**	**0.897**	**0.888**
Youden Index	$TPR+TNR-1$	**0.771**	**0.756**	**0.766**	**0.770**
AUC	$\int ROC$	**0.964**	**0.937**	**0.941**	**0.969**

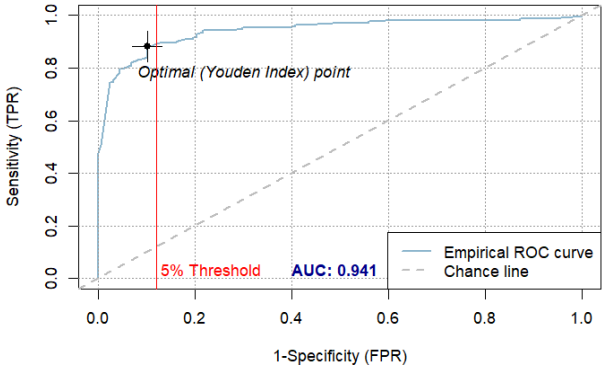
Table 5.2: Performance metrics for the test set with three different VCAD models (1 tree, 13 trees and 30 trees) and HKMN for the WDBC dataset



(a) Single tree VCAD model (1 tree)



(b) Truncated VCAD model (13 trees)



(c) Fully grown VCAD model (30 trees)

Figure 5.1: ROC curves represented with the 5% threshold for three different VCAD models

From the confusion matrices in Table 5.1, one can notice that pruning the vine from 30 trees in the fully grown model to 13 in the truncated model is insignificant. Only two observations move from the False Positive to the True Positive leading to an increase of accuracy of 0.05. A trend between models is also noticeable. The number of positive predictions increases with the number of trees. The single tree model favours negative predictions, while deeper vines favour positive predictions. Although the partition between positive and negative predictions evolves, the number of incorrect predictions remains stable, going from 47 to 43. This observation is also evident in the performance Table 5.2 where the accuracy evolves from 0.873 to 0.883.

In the ROC curves from Figure 5.1, two interesting observations can be pointed out. Firstly, the Area Under Curve (AUC) is maximal with a single tree. The single tree model is then more efficient to distinguish the two classes. Secondly, the F1 score and the accuracy at the 5% threshold are higher with the two more complex models. This is counter-intuitive seeing the AUC. Yet, this is easily explicable when looking at the optimal Youden index. The 5% threshold is closer to the optimal Youden index for those two latter models which indicates that the cut-off is close to the optimal performance of the model. In comparison, the single tree model with the 5% threshold is quite far from its optimal which shows that choosing a higher threshold would yield better results. This is a compelling example that cut-off selection in semi-supervised learning is a crucial determinant of model performance.

If we compare the single tree VCAD model with the HKMN, their performance is almost similar. The AUC of these models varies by 0.02 which shows that both models achieve similar discriminating capabilities. At the 5% threshold, the global performance metrics such as accuracy, F1 score and Younden index display negligible variations. However, the TPR, TNR and Precision vary very much. The HKMN tend to predict more positive cases than the single tree VCAD which leads to more balanced results for HKMN. The single tree VCAD finds almost all true negative cases, yielding to very high TNR and Precision. On the other hand, it is too prompt to flag observations as negative which results in poor TPR compared to its counterpart. It is worth mentioning that the single tree VCAD with the 5% cut-off is still far from its optimal Youden index (0.816) while the HKMN is closer to its optimum (0.804). This result shows that the implementation differences between HKMN and single tree VCAD are rather insignificant.

Finally, we note that adding trees to deepen the vine does not result in a higher performance in this context. Indeed, both the single tree VCAD and the HKMN perform better than the truncated vine and fully grown VCAD. There might be several explanations to this phenomenon:

1. Adding trees creates complex models that overfit. The number of observations in the WDBC dataset may be too small to attempt such complex models. If that is the case, the following attempts on bigger datasets would yield better results (see Chapter 6).

2. The data do not lend itself to such complex models. The abnormality of the dataset simply does not lie in the conditional dependency modelled by higher-order trees. Consequently, the added copula dependence trees only result in noise preventing the model to make the right decision based on the first tree. Again, if this is the explanation for the poor results, it might be refuted by more complex datasets (see Chapter 6).

In the following section, the HKMN and VCAD will be put to the test with more datasets and compared to other well-known semi-supervised anomaly detection algorithms.

Chapter 6

Benchmarking

The anomaly scoring capabilities of HKMN and VCAD have already been demonstrated. To go one step further, their performance will be compared to other semi-supervised anomaly detection methods. Five other algorithms will be presented and tested:

1. k-nearest neighbours global unsupervised anomaly detection (Knn): The principle of Knn is to find for each observation the k closest observations. The anomaly score is then the mean distance between the observation and its k-nearest neighbours (Angiulli and Pizzuti 2002).

2. Local outlier factor (LOF) (Breunig et al. 2000). LOF relies on three steps. First, it detects the k-nearest neighbours for each observation. Secondly, the local density for an observation is evaluated by computing the local reachability density (LRD):

$$LRD_k(x) = 1/\left(\frac{\sum_{o \in N_k} d_k(x,o)}{|N_k(x)|}\right)$$

   where d_k is called the reachability distance and N_k is the set of k-nearest neighbours. The last step is to compute the LOF score by comparing the LRD of a record with the LRDs of its k neighbours:

$$LOF(x) = \frac{\sum_{o \in N_k(x)} \frac{LRD_k(o)}{LRD_k(x)}}{|N_k(x)|}$$

   The LOF score could be interpreted as a ratio of local densities. The observation with a ratio close to 1 are considered normal while those with large scores are considered abnormal.

3. Histogram-based outlier score (HBOS) is a statistical method which assumes independence between features (Goldstein and Dengel 2012). The algorithm creates histograms for each feature. The anomaly score for an observation is then the product of the inverse bin heights of each of its features. The dependence between variables is then completely

discarded. Yet, the advantage of this procedure is the low computing power needed to score observations.

4. Robust principal component analysis for anomaly detection (PCA) is built from the major and minor principal components of normal instances (Shyu et al. 2003). The anomaly score can be obtained as the sum of the projected distances of a sample on all eigenvectors.

5. Cluster-Based Local Outlier Factor (CBLOF) first uses clustering to assess dense area in the feature space (He, Xu, and Deng 2003). Secondly, for each cluster identified, it performs a density estimation. The algorithm then heuristically classifies clusters either as small or large clusters. The anomaly score for large clusters observations is its distance to the centroid. For the small clusters observations, the distance to the closest large cluster is used as anomaly score.

To perform the benchmarking, seven datasets will be used. All of these are well-known anomaly detection datasets with continuous features. They have been separated in training and test sets. Their descriptions can be found in Table 6.1.

	Number of observations (training set)	Number of observations (test set)	Number of attributes	% of anomalies (test set)
Wave	1000	4000	21	83.57 %
Satellite	4500	600	36	12.50 %
Letters	337	120	617	65.83 %
Penn-global	400	409	16	22.00 %
Glass	70	144	9	20.14 %
WDBC	200	369	30	57.45 %
Spambase	2000	579	57	8.81 %

Table 6.1: Description of the datasets used for benchmarking

The purpose of the sections is to evaluate how HKMN and VCAD perform compared to well recognized anomaly detection algorithms. Two metrics will support the comparison. The first one is the AUC. As explained in Section 4.5, AUC is a good evaluation method for anomaly detection, and it is a good metric for this benchmarking. The second metric will be F1 score at the 5% threshold on the training set. This step is also interesting as the arbitrary threshold is the method users with an unlabelled test set would apply.

	KNN	LOF	HBOS	PCA	CBLOF	HKMN	VCAD single	VCAD truncated	VCAD full
Wave	0.840	0.832	0.815	0.841	0.828	0.820	0.838	**0.843**	0.837
Satellite	0.965	0.966	0.882	0.914	**0.967**	0.955	0.926	0.949	0.949
Letters	0.893	**0.916**	0.735	0.736	0.897	0.821	-	-	-
Penn-global	**0.998**	0.887	0.766	0.857	0.989	0.915	0.982	0.990	0.991
Glass	0.929	0.954	0.950	0.945	0.929	0.957	0.962	0.962	**0.963**
WDBC	**0.975**	0.966	0.916	0.971	0.969	0.968	0.964	0.937	0.941
Spambase	0.787	0.586	0.836	0.821	0.759	**0.875**	-	-	-
Mean	**0.912**	0.872	0.843	0.869	0.905	0.902	-	-	-
Std dev.	0.077	0.135	0.078	0.081	0.085	**0.064**	-	-	-
	↓ Without Spambase and Letters ↓								
Mean	**0.941**	0.921	0.866	0.906	0.936	0.923	0.934	0.936	0.936
Std dev.	0.062	0.060	0.075	**0.056**	0.064	0.061	0.058	**0.056**	0.059

Table 6.2: AUC scores of five anomaly detection algorithms, the HKMN and three VCAD (single, truncated, full) for seven different datasets.

In Table 6.2, the AUC values for VCAD models are missing with *Letters* and *Spambase*. This is due to the package *rvinecopulib* and its underlying C++ library, *boost*, which runs into errors when the dimensionality of the vine is too big. One of the temporary variables used in the fitting algorithm is initialized with only 4 bits which is not enough to save the value for these datasets. Unfortunately, we could not find a fix to overcome this inconvenience.

By observing Table 6.2, one can notice that AUC values for both HKMN and VCAD are good compared to their pairs. They may even surpass the most performing algorithm, KNN, in some cases. The average AUC values for the HKMN is 0.902 which rank third over the six algorithms passing all tests. One of HKMN characteristics standing out from these results is the stability of its performance. HKMN is the algorithm displaying the lowest variance in results.

The VCAD results although incomplete tend to be slightly better than the HKMN ones. By looking only at datasets where the VCAD algorithm converges, VCAD "truncated" and VCAD "full" rank second ex-aequo with CBLOF in average AUC. Again, the performance of these algorithms is the most stable. To be precise, the truncated VCAD has the smallest variance. From these observations, one can conclude that the results obtained with the WDBC dataset from Chapter 5 are not transposable here. Indeed, with the WDBC dataset, HKMN outperformed VCAD. Yet, these new results indicate that adding trees to an anomaly detection vine may improve performance.

	KNN	LOF	HBOS	PCA	CBLOF	HKMN	VCAD single	VCAD truncated	VCAD full
Wave	0.728	0.731	0.671	0.496	0.762	0.517	0.737	0.760	**0.764**
Satellite	0.702	0.736	0.490	**0.787**	0.762	0.726	0.678	0.688	0.681
Letters	0.760	0.730	0.547	0.736	**0.824**	0.702	-	-	-
Penn-global	**0.914**	0.756	0.185	0.531	0.894	0.712	0.874	0.837	0.837
Glass	0.111	**0.954**	0.731	0.867	0.737	0.836	0.836	0.824	0.829
WDBC	0.930	0.903	0.671	0.902	**0.933**	0.888	0.878	0.892	0.897
Spambase	0.400	0.241	**0.543**	0.400	0.364	0.494	-	-	-
Mean	0.649	0.722	0.548	0.674	**0.754**	0.696	-	-	-
Std dev.	0.295	0.231	0.182	0.197	0.187	**0.147**	-	-	-
	↓ Without Spambase and Letters ↓								
Mean	0.677	**0.816**	0.550	0.717	0.818	0.736	0.801	0.800	0.802
Std dev.	0.333	0.105	0.223	0.190	0.089	0.143	0.089	**0.078**	0.082

Table 6.3: F1 scores of five anomaly detection algorithms, the HKMN and three VCAD (single, truncated, full) for seven different datasets.

Regarding Table 6.3, the 5% threshold for the F1 score is relatively low. This explains that F1 scores are also quite low for most models. Yet, this table enables to highlight algorithms that can quickly identify anomalies with such low threshold. At this step, the HKMN and VCAD also rank third behind LOF and CBLOF. Although they kept their ranking, they only landed first for a single dataset. Regarding the stability of their prediction, they remain the ones with the smallest variance: HKMN is the most stable with seven datasets and VCAD is the most stable when removing *Letters* and *Spambase*. This indicates that with low threshold CBLOF and LOF, are likely to be more accurate. Yet, HKMN and VCAD are good challenger which will perform good no matter the dataset.

Chapter 7

Discussion

Let us now analyse the pros and cons of HKMN and VCAD. In the two previous chapters, we have proven that these algorithms are effective methods for semi-supervised anomaly detection. They compete with the best algorithms available. Yet, when should one use HKMN or VCAD compared to KNN or CBLOF which top our ranking? In this chapter, we attempt to answer this question by examining limitations of the models in Section 7.1. We also outline the benefits of HKMN in Section 7.2, a few remarks concerning the implementation in Horváth et al. (2020) in Section 7.3 and the cost for increased VCAD performance in Section 7.4. The final verdict is given in Section 7.5 in which we indicate when to use HKMN or VCAD. To conclude the chapter, we present suggestions for improvement for later research in Section 7.6.

7.1 Limitations HKMN and VCAD

It is important to start this analysis by exposing the limits of HKMN and VCAD. The main constraint is that they require features to be non-categorical and have continuous distributions. This constraint derives from the copula definition, which implies uniform margins. This requirement limits the usability of the models. Indeed, other methods such as HBOS or, with an appropriate distance measure, LOF, CBLOF and KNN can accommodate both continuous and categorical features. Another disadvantage of HKMN and VCAD is that they are really complex and heavy. They require to optimize many parameters with the building of dependence trees, the marginals fitting, the copula fitting and the MV curve estimations. This high number of parameters implies that many observations are necessary during the training process to avoid overfitting. In comparison with other methods discussed in the previous chapter, it represents a drawback since computing power may be lacking for very large datasets.

7.2 Benefits of HKMN

Now that these limits have been discussed, let us review this HKMN implementation and its benefits. First, the most interesting property of the algorithm is the anomaly localisation property. It is a powerful tool that could help in many contexts such as maintenance planning, intrusion detection, etc. Even though it may be outperformed in some case, experts might prefer HKMN to other methods. Indeed, in combination with domain expertise, HKMN might be convenient to domains which exclude black box models for legal requirements such as the banking or insurance industry.

Secondly, once the model has been fitted, computing the anomaly score for new instances is very fast compared to models such as KNN which requires to compute many distances to get the closest neighbours. Indeed, the fitting time may be long for high dimensional datasets, but once distributions and MV curve estimations have been computed, the prediction itself is very fast. This means HKMN could be used for near real-time anomaly detection. Obviously, for the VCAD, the more trees are added to the vine, the less applicable it becomes for real-time detection.

Moreover, in Chapter 6, we have showed that HKMN has a very stable performance. Once data respect the HKMN constraints (real continuous features), the algorithm's performance is very consistent no matter the dataset. In a semi-supervised context, this property is certainly desirable. Since the test set is unlabelled, there is normally no opportunity to prove that an instance is anomalous. In this case, an anomaly detection method with an uncertain performance such as depicted by HBOS in the benchmarking might be considered as a risk. On the contrary, HKMN will almost certainly deliver a satisfying performance.

Finally, HKMN can score an instance even for datasets with missing values. This specific situation was not tested in the benchmark, as most other anomaly detection methods do not share this property. Yet, if a feature exhibits missing values, HKMN will simply ignore this feature edge scores in the computation of the global anomaly score. Although the global score might be biased due to the missing edge score, this still allows to label the observation rather than excluding it from the study.

7.3 Remarks regarding the HKMN implementation in Horváth et al. (2020)

Regarding Horváth et al. (2020) implementation, there are questionable elements. One of them is the use of likelihoods to choose distribution families to fit marginals and copulas. This selection criterion is likely to lead to overfitting. Other selection criteria such as AIC or BIC should be preferred.

Besides the use of the likelihood as selection criterion, some choices are valid but debatable, such as the use of parametric fitting of the marginals while copulas could be fitted by both parametric

families or nonparametric ones. Using parametric distribution families for margins has the disadvantage to be less adapted to real-world data. Moreover, some datasets might require preprocessing such as Min-Max normalization to be used with parametric families. In this master thesis, no processing was performed before using the algorithms but it might have been necessary with other datasets. Indeed, if a dataset would contain features with only negative values, the only distribution families proposed in the article that could have fitted these features would have been the normal distribution, the gamma one or one of their mixtures. With nonparametric fitting methods, this would not have been an issue. This is actually a trade-off to be made before using HKMN knowing the dataset. The best implementation choice would be to let users choose whether they prefer a parametric or nonparametric fitting for both marginals and copulas. The choice should be determined by the amount of training data at hand, and the response and fitting time expected. The more data available, the more nonparametric fitting will be accurate. If computing resources are lacking and fitting time is a priority, parametric fitting might then be preferred due to its lower number of parameters to optimize.

Another debatable element in Horváth et al. (2020) is the choice of mutual information to build the dependence tree. This is actually the approach taken in Chow and Liu (1968), but this is unusual in recent vine copula literature where Kendall's tau is considered as an appropriate choice (Czado, Schepsmeier, and Min 2012). We have proven in Section 4.1 that the created trees differ whether one or the other is used. In Table 7.1, the AUC of HKMN with both mutual information and Kendall's tau is presented for seven different datasets. The weighting criterion did influence the structure of the tree slightly. However, the performance of these different structures does not vary enough to validate that one criterion surpasses the other to build copula-trees for anomaly detection.

	HKMN Mutual info.	HKMN Kendall's tau
Wave	0.820	**0.839**
Satellite	**0.955**	0.954
Letters	0.821	**0.827**
Penn-global	**0.915**	0.911
Glass	**0.957**	0.954
WDBC	0.968	0.968
Spambase	0.875	**0.885**
Mean	0.902	**0.905**
Std dev.	0.064	**0.057**

Table 7.1: The AUC performance of HKMN with mutual information and with Kendall's tau

7.4 The cost of VCAD performance

Regarding VCAD, many points discussed for HKMN are also valid such as its good performance and its reliability. The performance with VCAD was improved for datasets with complex dependence structures. Adding trees might then sometimes be relevant. Yet, adding trees to

create a more efficient anomaly detection algorithm is not completely successful. The cost of adding those trees is a longer computational time and the loss of the localisation capability. The slightly increased performance observable in Chapter 6 might not be sufficient to justify this cost.

It is worth mentioning that the localisation capability could be retrieved with VCAD by using edge scores such as HKMN does, rather than using the estimated vine cumulative distribution as a global anomaly function. This would considerably complicate the task compared to HKMN since higher-order trees require to integrate the distribution functions of the previous tree as input. As a reminder, in Figure 7.1, the copula associated to the edge $1,4;3$ in T_2 is

$$C_{14|3}\big(F_{1|3}(x_1|x_3), F_{4|3}(x_4|x_3)\big)$$

where $F_{1|3}(x_1|x_3)$ and $F_{4|3}(x_4|x_3)$ are obtained by $\int_{-\infty}^{x_1} \frac{f_{1,3}(t,x_3)dt}{f_3(x_3)}$ and $\int_{-\infty}^{x_4} \frac{f_{4,3}(t,x_3)dt}{f_3(x_3)}$.

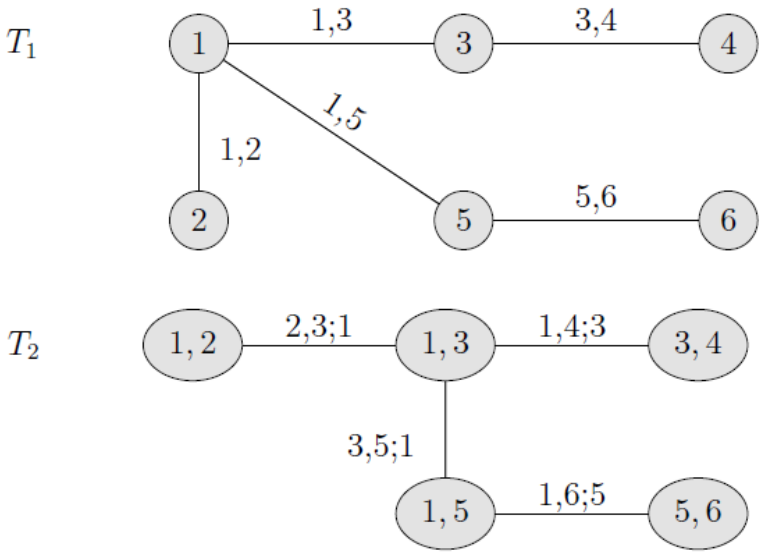


Figure 7.1: The two first trees of a six dimensional regular vine tree sequence. The illustration is taken from page 101 of Czado (2019).

For each edge of each tree, the distributions should be computed based on the edge copula and previous tree distributions. Afterward, a MV curve for each of these distributions should be estimated with Monte-Carlo integration. For a dataset with 30 features, HKMN computes 29 MV curves, one per edge, to build anomaly scores. If VCAD used similar scoring functions with a vine of three trees, this would imply 84 (29+28+27) MV curves to assess. This would again deteriorate the fitting time. Moreover, for higher-order trees, the anomaly scores for an edge become difficult to interpret.

7.5 When should the HKMN or the VCAD be used?

Given the arguments presented, the HKMN should be used mainly for its reliability and localisation capability. Indeed, its performance is good and comparable to other existing methods, but these two characteristics make it desirable.

As for the VCAD, as it loses the ability to localise, it should rather be preferred in case of complex dependence relations between variables, where it consistently provides some of the best performance. If VCAD is chosen, the truncated vine based on mBIC should perform well and is a good alternative to the fully grown vine.

7.6 Suggestions for improvement

There is obviously room for improvement with copula-trees for anomaly detection. In this section, we present some of them.

Weighted anomaly score: For HKMN, an improvement would be to give more weight in the global anomaly score to some of the edges. In this implementation, all edge scores received a similar weight in the global score. Some of them may be better indicators than others. For instance, one could wonder if weighting the edge scores by the mutual information or Kendall's tau in the global score would improve the performance.

Feature selection: Another aspect discussed briefly in Section 4.1 is whether all variables should be included in the first dependence tree. Some highly dependent variables such as $V23$, $V25$ and $V26$ in Section 4.1 may be repetitive. Some feature selection or dimension reduction might be interesting before building trees. On the other hand, should all edges be included? Some of them barely bear information and might then be discarded to only keep the marginals of the variables. This is equivalent to modelling the dependence with an independent copula. Yet, flagging this uninteresting relations early on might help to save computation resources. If some edges were removed, the structure would not satisfy the definition of a tree and the model would then resemble a Markov network with copulas modelling the dependence structure.

HKMN and VCAD for discrete variables: Lastly, a major inconvenience of these algorithms noted in Section 7.1 is the requirement to work with continuous and real-valued data. Panagiotelis, Czado, and Joe (2012) may bring a solution to this limitation as they extend pair copula constructions to discrete variables. One could adapt HKMN and VCAD accordingly.

Conclusion

In this master's thesis, the use of dependency trees with copulas for anomaly detection has been studied. In particular two algorithms were detailed: the HKMN by Horváth et al. (2020) and the VCAD introduced in this thesis. The VCAD combined the work of Horváth et al. (2020) that introduced copula-trees in anomaly detection with the pair copula constructions (Joe 1996; Bedford and Cooke 2001; Bedford and Cooke 2002). The aim of the thesis was to verify the efficiency of copula-trees in the field of anomaly detection and to investigate the impact of building vines rather than standalone copula-trees for this task.

In the first part, we started by recalling the theoretical concepts associated with these dependence trees and copulas. Previous works on dependence trees were presented, such as the Chow-Liu trees or the pair copula constructions. Then, the two studied algorithms were explained and implemented. In Part II, an evaluation of the algorithms was performed on seven datasets. The results produced in this phase were encouraging as our implementations of HKMN and VCAD matched and sometimes even outperformed the competing anomaly detection methods. One of the striking features of these implementations was the consistency of their performance. Where some competing algorithms might shine in one context and then fail in another, the HKMN and VCAD maintained a consistent performance that appeared to be independent of the specific characteristics of the dataset under study. Afterwards, we presented the limitations of HKMN and VCAD but also their advantages. An evident limitation is that both algorithms require data to be continuous and real-valued. Another limitation is the large number of parameters to optimise, which requires datasets to have a sufficient number of observations to prevent overfitting. This remark is all the more valid for VCAD where the risk of overfitting increases with the number of trees. Regarding the advantages, for the HKMN, its localisation capability is particularly valuable and could be applied in many areas. In particular, we highlight the advantage of this algorithm over other methods that are so-called black-box models. Unfortunately, this capability is lost for the VCAD. However, this does not stop VCAD from being an excellent anomaly detection algorithm that performs extremely well with datasets with complex dependency structures.

Amongst future research on the topic, the usefulness of dimension reduction or variable selection should be further investigated. It would also be interesting to analyse whether an anomaly score weighted according to a dependence measure could improve performance. Lastly, Panagiotelis, Czado, and Joe (2012) have adapted the pair copula constructions to discrete variables, so HKMN and VCAD could also be extended to such variables.

Appendix

Kruskal's algorithm

Kruskal's algorithm relies on the cut property and the cycle property of the minimum spanning tree. The intuition of the algorithm is to sort edges increasingly by their weights and, then, to decide if the weight is necessary to the spanning tree. If it is necessary, it is added to a growing forest F. Otherwise, it is discarded. The decision to discard or not is based on the cut and cycle properties. If the edge is the first one across a cut, it is the lightest as edges are considered based on their increasing weights. If an edge is the last edge of a cycle, it is the heaviest. Consequently, it is rejected from the growing forest F. This operation is repeated with every edge until the forest F is a minimum spanning tree.

Data: $G = (E, V)$ with weights w_e
Result: A minimum spanning tree F
begin
  $F \longleftarrow \emptyset$
  `sortByWeight(`E`)`
  for $e = \{i, j \mid i \in V, j \in V, i \neq j\} \in E$ *by increasing weight* w_e **do**
    if *i and j are not connected in F* **then**
      $F \longleftarrow F \cup \{e\}$
    end
  end
end

Algorithm 1: Kruskal's algorithm for minimum spanning tree

Prim's algorithm

This algorithm also takes a greedy approach. It starts by selecting randomly one vertex of the graph. All edges connecting this vertex are then considered, the one with the lowest weight is added to the new tree. At each iteration, the edges of the tree connecting to new vertices are considered, and the one with the lowest weight is added until the tree connects all vertices.

```
Data: G = (E, V) with weights w_e
Result: A minimum spanning tree F
begin
    for v ∈ V do
        cost_v ⟵ +∞
        prev_v ⟵ ∅
    end
    v_0=selectRandomly(V)
    cost_{v_0} = 0
    F ⟵ ∅
    Q ⟵ V
    while Q ≠ ∅ do
        v ⟵ selectMinWeight(Q)
        e ⟵ {v, prev_v}
        // e is the edge between v and the existing tree, if any
        F ⟵ F ∪ {e}
        Q ⟵ Q − {v}
        for {v, w | w ∈ Q} ∈ E do
            if cost_w > w_{v,z} then
                cost_w = w_{v,z}
                prev_w = v
            end

        end

    end

end
```

Algorithm 2: Prim's algorithm for minimum spanning tree

Bibliography

[1] H. Akaike. "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6 (1974), pp. 716–723. DOI: 10.1109/TAC.1974.1100705.

[2] Fabrizio Angiulli and Clara Pizzuti. "Fast Outlier Detection in High Dimensional Spaces". In: *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*. PKDD '02. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 15–26. ISBN: 3540440372.

[3] Tim Bedford and Roger Cooke. "Probability Density Decomposition for Conditionally Dependent Random Variables Modeled by Vines". In: *Ann. Math. Artif. Intell.* 32 (Aug. 2001), pp. 245–268. DOI: 10.1023/A:1016725902970.

[4] Tim Bedford and Roger Cooke. "Vines: A New Graphical Model for Dependent Random Variables". In: *The Annals of Statistics* 30.4 (2002), pp. 1031–1068. ISSN: 00905364. URL: http://www.jstor.org/stable/1558694.

[5] E. C. Brechmann, C. Czado, and K. Aas. "Truncated regular vines in high dimensions with application to financial data". In: *Canadian Journal of Statistics* 40.1 (2012), pp. 68–85. DOI: https://doi.org/10.1002/cjs.10141. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cjs.10141. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cjs.10141.

[6] Markus M. Breunig et al. "LOF: Identifying Density-Based Local Outliers". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388. URL: https://doi.org/10.1145/335191.335388.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41 (July 2009). DOI: 10.1145/1541880.1541882.

[8] U. Cherubini, E. Luciano, and W. Vecchiato. *Copula Methods in Finance*. The Wiley Finance Series. Wiley, 2004. ISBN: 9780470863459. URL: https://books.google.be/books?id=0dyagVg20XQC.

[9] C. K. Chow and C. N. Liu. "Approximating discrete probability distributions with dependence trees". In: *IEEE TRANSACTIONS ON INFORMATION THEORY* 14.3 (1968), pp. 462–467.

[10] Stephan Clémençon and Albert Thomas. *Mass Volume Curves and Anomaly Ranking*. 2018. arXiv: 1705.01305 [stat.ML].

[11] Roger Cooke and Dorota Kurowicka. *Uncertainty Analysis With High Dimensional Dependence Modelling*. May 2006. ISBN: 0-470-86306-4. DOI: 10.1002/0470863072.

[12] Thomas H. Cormen et al. *Introduction to algorithms*. MIT Press, 2014.

[13] C. Czado. *Analyzing Dependent Data with Vine Copulas: A Practical Guide With R*. 1st ed. Vol. 222. Lecture Notes in Statistics. Springer International Publishing, 2019.

[14] C. Czado, M. Hofmann, and S. Jeske. "Selection strategies for regular vine copulae". In: *Journal de la Société Française de Statistique* 154.1 (June 2013), pp. 174–191.

[15] C. Czado, U. Schepsmeier, and A. Min. "Maximum likelihood estimation of mixed C-vines with application to exchange rates". In: *Statistical Modelling* 12.3 (2012), pp. 229–255. DOI: 10.1177/1471082X1101200302. URL: https://doi.org/10.1177/1471082X1101200302.

[16] Marie Laure Delignette-Muller and Christophe Dutang. "fitdistrplus: An R Package for Fitting Distributions". In: *Journal of Statistical Software* 64.4 (2015), pp. 1–34. URL: https://www.jstatsoft.org/v64/i04/.

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977), pp. 1–38. ISSN: 00359246. URL: http://www.jstor.org/stable/2984875.

[18] J. Dißmann et al. "Selecting and estimating regular vine copulae and application to financial returns". In: *Computational Statistics Data Analysis* 59 (2013), pp. 52–69. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2012.08.010. URL: https://www.sciencedirect.com/science/article/pii/S0167947312003131.

[19] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2019. URL: http://archive.ics.uci.edu/ml.

[20] F.Y. Edgeworth. "XLI. On discordant observations". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 23.143 (1887), pp. 364–375. DOI: 10.1080/14786448708628471. eprint: https://doi.org/10.1080/14786448708628471. URL: https://doi.org/10.1080/14786448708628471.

[21] J. H.J. Einmahl and D.M. Mason. "Generalized quantile processes". English. In: *The Annals of Statistics* 20.2 (1992), pp. 1062–1078. ISSN: 0090-5364.

[22] Jason Eisner. *State-of-the-Art Algorithms for Minimum Spanning Trees – A Tutorial Discussion*. 1997.

[23] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[24] Gery Geenens, Arthur Charpentier, and Davy Paindaveine. "Probit transformation for nonparametric kernel estimation of the copula density". In: *Bernoulli* 23.3 (2017), pp. 1848–1873. DOI: 10.3150/15-BEJ798. URL: https://doi.org/10.3150/15-BEJ798.

[25] C. Genest, K. Ghoudi, and L.-P. Rivest. "A Semiparametric Estimation Procedure of Dependence Parameters in Multivariate Families of Distributions". In: *Biometrika* 82.3 (1995), pp. 543–552. ISSN: 00063444. URL: http://www.jstor.org/stable/2337532.

[26] Markus Goldstein and A. Dengel. "Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm". In: 2012.

[27] Markus Goldstein and Seiichi Uchida. "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data". In: *PLOS ONE* 11.4 (Apr. 2016), pp. 1–31. DOI: 10.1371/journal.pone.0152173. URL: https://doi.org/10.1371/journal.pone.0152173.

[28] Frank E. Grubbs. "Procedures for Detecting Outlying Observations in Samples". In: *Technometrics* 11.1 (1969), pp. 1–21. DOI: 10.1080/00401706.1969.10490657. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00401706.1969.10490657. URL: https://www.tandfonline.com/doi/abs/10.1080/00401706.1969.10490657.

[29] Alexander K. Hartmann and Martin Weigt. "Introduction to graphs". In: *Phase Transitions in Combinatorial Optimization Problems*. Wiley-VCH, 2005.

[30] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 9780387848846. URL: https://books.google.be/books?id=eBSgoAEACAAJ.

[31] Zengyou He, Xiaofei Xu, and Shengchun Deng. "Discovering cluster-based local outliers". In: *Pattern Recognition Letters* 24.9-10 (2003), pp. 1641–1650.

[32] Gábor Horváth et al. "Copula-Based Anomaly Scoring and Localization for Large-Scale, High-Dimensional Continuous Data". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 11 (2020), pp. 1–26.

[33] Harry Joe. "Families of m-Variate Distributions with Given Margins and m(m-1)/2 Bivariate Dependence Parameters". In: *Lecture Notes-Monograph Series* 28 (1996), pp. 120–141. ISSN: 07492170. URL: http://www.jstor.org/stable/4355888.

[34] M. G. Kendall. "A new measure of rank correlation". In: *Biometrika* 30.1-2 (June 1938), pp. 81–93. ISSN: 0006-3444. DOI: 10.1093/biomet/30.1-2.81. URL: https://doi.org/10.1093/biomet/30.1-2.81.

[35] Joseph B. Kruskal. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". In: *Proceedings of the American Mathematical Society* 7.1 (1956), pp. 48–50. ISSN: 00029939, 10886826. URL: http://www.jstor.org/stable/2033241.

[36] Beatriz Mendes, Mariângela Semeraro, and Ricardo Leal. "Pair-copulas modeling in finance". In: *Financial Markets and Portfolio Management* 24 (Apr. 2010), pp. 193–213. DOI: 10.1007/s11408-010-0130-1.

[37] Patrick E. Meyer. "Information-Theoretic Variable Selection and Network Inference from Microarray Data". PhD thesis. Université Libre de Bruxelles, 2008.

[38] T. Nagler, C. Bumann, and C. Czado. *Model selection in sparse high-dimensional vine copula models with application to portfolio risk*. 2018. arXiv: 1801.09739 [stat.ME].

[39] Thomas Nagler and Thibault Vatter. *rvinecopulib: High Performance Algorithms for Vine Copula Modeling*. R package version 0.5.5.1.1. 2021. URL: https://CRAN.R-project.org/package=rvinecopulib.

[40] Roger B. Nelsen. *Introduction to Copulas*. Springer New York, 2006.

[41] Anastasios Panagiotelis, C. Czado, and Harry Joe. "Pair Copula Constructions for Multivariate Discrete Data". In: *Journal of the American Statistical Association* 107.499 (2012), pp. 1063–1072. ISSN: 01621459. URL: http://www.jstor.org/stable/23427413.

[42] Wolfgang Polonik. "Minimum volume sets and generalized quantile processes". In: *Stochastic Processes and their Applications* 69.1 (1997), pp. 1–24. ISSN: 0304-4149. DOI: https://doi.org/10.1016/S0304-4149(97)00028-8. URL: https://www.sciencedirect.com/science/article/pii/S0304414997000288.

[43] R. C. Prim. "Shortest connection networks and some generalizations". In: *The Bell System Technical Journal* 36.6 (1957), pp. 1389–1401. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

[44] Enrique F. Schisterman et al. "Optimal Cut-Point and Its Corresponding Youden Index to Discriminate Individuals Using Pooled Blood Samples". In: *Epidemiology* 16.1 (2005), pp. 73–81. ISSN: 10443983. URL: http://www.jstor.org/stable/20486002.

[45] Johan Segers. *Copulas: Models and Inference*. 2016.

[46] Mei-ling Shyu et al. "A novel anomaly detection scheme based on principal component classifier". In: *in Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'03*. 2003, pp. 172–179.

[47] M. Sklar. "Fonctions de repartition a n dimensions et leurs marges". In: 1959.

[48] Sebastian Weber. *RBesT: R Bayesian Evidence Synthesis Tools*. R package version 1.6-1. 2020. URL: https://CRAN.R-project.org/package=RBesT.

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