DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

VINE COPULA BASED MULTIVARIATE MODELLING FOR CLASSIFICATION

by
Tolga YAMUT

February, 2025
İZMİR

VINE COPULA BASED MULTIVARIATE MODELLING FOR CLASSIFICATION

A Thesis Submitted to the
Graduate School of Natural And Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Statistics, Statistics Program

by
Tolga YAMUT

February, 2025
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**VINE COPULA BASED MULTIVARIATE MODELLING FOR CLASSIFICATION**" completed by **TOLGA YAMUT** under supervision of **PROF. DR. BURCU HÜDAVERDİ AKTAŞ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

.....

Prof. Dr. Burcu HÜDAVERDİ AKTAŞ

Supervisor

.....	.....
Assoc. Prof. Engin YILDIZTEPE	Prof. Dr. Femin YALÇIN KÜÇÜKBAYRAK
Thesis Committee Member	Thesis Committee Member
.....	.....
Prof. Dr. Eralp DOĞU	Assoc. Prof. Özge ELMASTAŞ GÜLTEKİN
Examining Committee Member	Examining Committee Member

Prof. Dr. Abdullah SEÇGİN
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I am sincerely grateful to my advisor, Prof. Dr. Burcu HÜDAVERDİ AKTAŞ, whose unwavering support and understanding made even the most tiring times more manageable. Her kindness and encouragement provided the motivation I needed to push forward, turning challenges into valuable learning experiences in my PhD journey.

I would like to express my gratitude to my thesis committee members, Prof. Dr. Femin YALÇIN KÜÇÜKBAYRAK and Assoc. Prof. Engin YILDIZTEPE, for their insightful feedback, and continuous support throughout this research. I am truly grateful for their time, patience, and encouragement, which have been essential in the completion of this thesis.

My friendship with Hasan KIRAN goes all the way back to high school, filled with countless funny moments and unforgettable experiences. From iconic inside jokes to epic FRP games and intense Yu-Gi-Oh! duels, we've shared so much over the years. Our adventures at punk concerts created some of the best memories, and through all of life's ups and downs, we've always been there to support and motivate each other. I'm truly grateful for his friendship and the many stories we'll keep adding to.

I am incredibly grateful to my dear friend, Yusuf CAN SEVİL, whose friendship has been one of the brightest parts of this journey. No matter how challenging things got, we could always find—or create—something to laugh about and able to create iconic moments or memes if I may. His unique talent for hilariously misunderstanding me never failed to lighten the mood, and our deep discussions about science and academia has been very insightful.

I met Didem KAVCI at dance school, and since then, we've shared many enjoyable dances at social Latin nights. Her enthusiasm and talent for planning events have brought so much energy to our friendship—she's very thoughtful and always up for something fun. We've been into amazing events and hangouts that created lasting memories and added extra motivation along the way. By the way, this year, we're taking it up a notch as bachata show partners. We'll be working on the choreography

for a global show event, no pressure, right?! Update: We aced it!

I met Özgür İLGEN at a social Latin night, and since then, our friendship has been a great mix of shared interests. From debugging code to navigating the dance floor, our conversations have always been engaging and fun. Whether hanging out after socials or playing online games, his friendship has added both motivation and enjoyment to this journey.

I would also like to express my gratitude to employees of Chaplin Coffee Shop, where I spent countless hours studying and had some memorable breakthroughs such as finally able to running the custom vine density algorithm and making thesis follow up presentation inside the shop due to a power cut problem in my neighborhood. The cozy atmosphere, complete with a black sleepy cat who always claimed the same spot and but somehow able to emit a vibrant energy.

My thanks also go to the anonymous kind and smiling people with whom I shared the dance floor, your positive energy and shared love for dance created moments I'll always cherish."

Lastly, I would like to pay special thankfulness and appreciation to my Mother and Father for their endless support, motivation and patience.

Tolga YAMUT

VINE COPULA BASED MULTIVARIATE MODELLING FOR CLASSIFICATION

ABSTRACT

In this thesis, the capabilities of statistical learning are investigated through various classification applications on simulated and biological data. The models are developed by fitting multi-dimensional data to standard and custom Bernstein copulas, as well as vine density frameworks. A convex Bernstein copula approach is proposed to address the imbalanced classification results observed with standard Bernstein copulas. Vine classifiers demonstrate strong performance in terms of classification scores and flexibility in capturing patterns within multi-dimensional feature spaces. For constructing vine classifiers, a truncation framework based on the classification performance of validation sets is proposed, which reduces the complexity of the trained vine classifier. This truncated vine classifier achieves relatively more balanced and improved results. Furthermore, a neural network framework is employed to generate the underlying distribution and density functions of gene expression data. The impact of neural networks on density-based classification is explored by connecting neural marginals to the vine density as its first tree nodes.

Keywords: Statistical learning, Bayesian method, Bernstein copula, Vine density, Neural marginal distribution

SINIFLANDIRMA İÇİN VINE KOPULAYA DAYALI ÇOK DEĞİŞKENLİ MODELLEME

ÖZ

Bu tezde, istatistiksel öğrenme yöntemlerinin kapasitesi, simüle edilmiş ve biyolojik veriler üzerindeki çeşitli sınıflandırma uygulamalarıyla incelenmiştir. Modeller, çok boyutlu verilerin standart ve özel Bernstein kopulası ile vine yoğunluk fonksiyonu yöntemleri üzerinden fit edilerek oluşturulmuştur. Standart Bernstein kopulanın dengesiz sınıflandırma sonuçlarını çözümlendirmek için konveks bir Bernstein copula yaklaşımı önerilmiştir. Vine sınıflandırıcıları, sınıflandırma skorları ve çok boyutlu öznitelik uzayındaki desenleri yakalama esnekliği açısından üstün bir performans göstermiştir. Vine sınıflandırıcılarının oluşturulması sırasında, validasyon setlerinin sınıflandırma performansına dayalı bir kırpma (truncation) yöntemi önerilmiş ve böylece eğitilmiş vine sınıflandırıcılarının karmaşık yapısı azaltılmıştır. Kırpılmış vine sınıflandırıcıyla, nispeten daha dengeli ve yüksek sonuçlar elde edilmiştir. Ayrıca, gen ekspresyonu veri setine ait temel dağılım ve yoğunluk fonksiyonlarını oluşturmak için bir sinir ağı yöntemi kullanılmıştır. Sinir ağlarının yoğunluk fonksiyonu tabanlı sınıflandırma üzerindeki etkisi, sinirsel marjinal dağılımların vine yoğunluk fonksiyonunun ilk ağaç düğümleri olarak bağlanmasıyla incelenmiştir.

Anahtar kelimeler: İstatistiksel öğrenme, Bayes yöntemi, Bernstein kopula, Vine yoğunluk fonksiyonu, Sinirsel marjinal dağılım

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1.1 Overview

Machine Learning is a research field that focuses on providing machines with the ability to recognize patterns through algorithms based on mathematical or statistical theories. The general purpose of this field is to extract meaningful results from data and to give machines the ability to distinguish objects from one another, similar to how humans do. Machine Learning is regarded as a subfield of Artificial Intelligence due to its endeavor to mimic human identification capabilities in computers.

The field of Statistical Learning can be described as one that focuses on using statistical inference methods to perform machine learning tasks. This learning approach forms the foundation of our research for the thesis. Primarily, in our applications, probability distributions are utilized in various ways to create learned models. In other words, the characteristics or patterns of the data are represented by fitted distributions, and new data instances are labeled based on these distribution models. The field utilizes supervised learning (e.g., regression and classification), (e.g., clustering and dimensionality reduction), and more advanced techniques like ensemble methods and neural networks in order to build models that can generalize from data (Hastie (2009), James (2013)). Such techniques are foundational for tasks like determining relationships between variables or making automated predictions in finance, healthcare, and marketing (James (2013)). The classification based on supervised approach is the base machine learning approach of the thesis.

Supervised classification is a subcategory of machine learning in which the classes are known prior to the classification process. A class is a categorical variable represented by a dataset consisting of features or variables. A learned model refers to a machine learning algorithm that has extracted feature information from each class and is ready to classify new data. From a statistical perspective, a learned model corresponds to a fitted probability distribution, and a feature corresponds to a random

variable.

Naive Bayes is one of the well-known supervised classification methods based on a probabilistic framework. Essentially, the method compares the probability density functions of features to determine the class of a given data point. Due to its independence assumption (hence ”naive”), the method is relatively simple to apply. However, this assumption makes it prone to failure when capturing complex dependencies among features. Complex associations between features in high-dimensional datasets are common in real-world machine learning problems.

To build an effective classification model, it is crucial to consider as many features as possible to obtain sufficient information about the classes and create a robust classification environment. However, joint distributions in high dimensions often lack familiar forms outside generalizable distribution families, such as the Gaussian distribution. To model a complex dependence structure, one can use copula distributions. Furthermore, in high dimensions, vine copulas can be employed, where bivariate copulas serve as building blocks for constructing flexible dependency models.

Copulas are flexible distributions that can model or help to model joint dependence structure. By using copula one can model joint association of random variables and their marginal distributions independent from each other. A vast choice of copula families exist to capture a bivariate dependence structure. But they do not possess obvious forms in high dimensions. In order to bypass this limitation, using bivariate copulas in a vine structure gives a very effective way to deal with the high dimensional inference. Vine copula allows using bivariate copulas in multiple steps to model multivariate structure. First ideas on vine copula is given in Joe (1996), then it is defined and systemized as a study field in Bedford & Cooke (2001), Bedford & Cooke (2002).

In the progression of vine copula construction, one needs to determine multiple bivariate copula distributions. This problem is getting more complex as dimension

increases. Misspecification of copula families causes increase in error of a vine model. In order to minimize error potential of a vine model, modifications are integrated into the vine construction. The adopted vine models are tested based on the classification metrics. Applying a full nonparametric or semiparametric procedure on vine copula construction is a promising study topic. Estimating bivariate copula distributions with the help of non-parametric procedures, error propagation in the vine process can be avoided. There are several nonparametric candidate instead of copula families. For example, Bernstein copulas have appropriate properties and flexibility to use when the distribution is unknown. Furthermore hybrid methods can be used to obtain more flexibility where vine nodes can have parametric or non-parametric copula. In high dimensions building a full vine structure is time consuming and could be redundant to obtain an optimal vine. Modified vines called truncated vine is a simplifcation strategy which avoids unnecessary errors and it is based on replacing certain tree nodes with independence copula after a cut point. For more details address Kurowicka (2010) and Brechmann et al. (2012).

The last application of the research focuses on building artificial neural networks to adopt cumulative distribution function and probability density function. The framework in the Zeng & Wang (2022) is used in order to construct cdf and pdf generating neural networks. According to the results on the benchmark datasets, the trained Neural copula and its marginals are closer to the true distribution for both tranining and test cases than classical models; t-copula, Frank copula and Gaussian copula. In our application Neural copula framework is used to build marginal distributions and copula distributions for each class labeled data of gene expression benchmark dataset. The classification performance of neural copula is investigated in the binary classification context.

1.2 Literature Review

The usage of statistical methods is prominent in the area of machine learning whether they are used as performance or interpretability enhancer, or as a pattern

learning method. Okoli (2023) explored the integration of statistical inference techniques, such as bootstrapping and confidence intervals, within the framework of ALE, which quantifies feature importance in machine learning models. The methods include adapting bootstrap algorithms to account for non-parametric data distributions. This approach enhances interpretability in models like decision trees and random forests by calculating more robust confidence intervals for ALE values across multiple bootstrap samples. In the work of Lemaire et al. (2024) naive Bayes classifiers are used for counterfactual optimization in binary classification tasks. It modifies naive Bayes by iteratively adjusting feature values to maximize the likelihood of a sample belonging to a target class. These adjustments provide insight into decision boundaries and facilitate interpretability improvements. In Kovács et al. (2024), generalized naive Bayes(GNB) method is used to extend the classical naive Bayes. The GNB introduces new algorithms that are specifically constructed to find a better-fitting probability distribution. These algorithms include a greedy algorithm and an optimal distribution algorithm based on Kullback-Leibler divergence, aiming to maximize information content and minimize redundancy during the data-fitting process. Experimental results indicate that the GNB algorithms often outperform classical Naive Bayes and other related algorithms in terms of accuracy. In the work of Gohari et al. (2023), Bayesian Latent Class Analysis (BLCA) is introduced to define latent variable that acts as a parent of model's parent attributes. This approach helps in capturing complex dependencies among the attributes, which is a limitation of the traditional naive Bayes (NB) classifier. The model is applied to real-world data involving 976 Gastric Cancer (GC) patients and 1189 Non-ulcer dyspepsia (NUD) patients. These results collectively indicate that the NB-BLCA model not only enhances classification accuracy but also provides a robust framework for dealing with real-world data complexities, particularly in the classification of gastric cancer patients.

Applying machine learning methods for diagnostics, has an important role as a study area. Diagnosis from an objective source allows for a thorough examination of any doubts and contributes to establishing decision rules with minimal error.

Classification methods are very beneficial to control this process. In the work of Torres-Roca et al. (2005), tumor cells are classified according to their radiation sensitivity to enhance treatment protocol for cancer patients. McKinney et al. (2020), developed an A.I. system to help radiologists to spot breast cancer. Findings of the preceding study is that, radiologists can spot breast cancer more accurately with the help of machine learning. Thus, machine learning approach in health care, has a big potential to become a standard.

Vine copula is used in many studies which involves high dimensional modeling. There are many examples in financial area some of them are; Dissmann et al. (2013), So & Yeung (2014), Loaiza Maya et al. (2015), Aloui & Aïssa (2016), Scheffer & Weiß (2017), Kraus & Czado (2017). As a different approach, Gräler (2014) used spatial vine copula to model emergency routine dataset and used spatial interpolation. Soto et al. (2012) used vine copula in molecular docking problem and found that vine copula based algorithms have relatively good performance. Vine distribution in the context of machine learning is relatively limited study area. Carrera et al. (2016) used D-vine classifiers for mind reading dataset. Chen (2016) also used D-vine classifiers to benchmark machine learning datasets and compare the method with classical methods including naive Bayes. Carrera et al. (2019), allowed flexibility by using R-vine classifiers and consider all kinds of association between variables in the model. Preceding study made a classification on Mars dunes. On the other hand, Sun et al. (2019) proposed vector and reinforcement learning representation of vine structure problem to find best possible vine model. Şahin & Joe (2024), also used vine classifiers in their study. Each class in the dataset, univariate distributions are fitted alongside a vine copula. This approach enables the calculation of posterior probabilities for each class, which can then be utilized for discriminant analysis. The results indicate that the vine copula-based classifier outperforms traditional discriminant analysis methods and random forests, especially in scenarios where features exhibit different dependent structures across classes. Vines also can be used as a booster for the data management phase of a machine learning task as it can be seen in the work of Konstantelos et al. (2018). The paper uses vine copulas primarily

for data preparation in machine learning tasks. Specifically, the vine copulas model the dependencies between variables to generate synthetic datasets representative of real-world system states. These datasets are then used to train machine learning models, such as classifiers for power system security assessment. To reduce the complexity of vine structure for the large data scalibility, only the Gaussian copula is feed to the vine. The approach outperforms traditional sampling methods, producing datasets that improve the performance of machine learning classifiers used in power system security assessment.

1.3 Research Scope and Aims

In statistical learning, challenges often arise when fitting data labeled with classes, including:

- Detecting appropriate fit for the non-obvious data patterns.
- Maintaining multivariable distributions to represent high dimensional feature space.

To be able to overcome such problems, modifications on the distribution based classifiers are proposed and tested on the simulated and experimental real life datasets alongside with the classical approaches. The modifications are made particularly for the Bernstein copula and vine density classifiers. To address such objectives, a flexible distributional classifier algorithm was developed in R and progressively modified.

First, as a non-parametric approach the multivariate Bernstein copula classifier is applied to learn the data patterns. Performance and limitations are investigated in the context of classification.

Then vine density classifiers are employed for the machine learning tasks. Vines are structured based on three different tecniques namely, parametric, non-parametric

and hybrid. In parametric case, the classic R-vine is fitted by feeding well-known bivariate copula families into our algorithm. In non-parametric case the vine is structured only by calibrating Bernstein copulas. Lastly for hybrid case, the vine is structured by feeding both copula families and Bernstein copula into the algorithm. Hence, the node distributions of the vine are chosen from parametric families and the Bernstein copula.

CHAPTER TWO
METHODOLOGY

2.1 Copula

Let $X_1, \ldots, X_d$ be the random variables with joint distribution function $H(x_1, \ldots, x_d)$ with marginals $F_1(x_1) = P(X_1 \leq x_1), \ldots, F_d(x_d) = P(X_d \leq x_d)$. Sklar's theorem plays a crucial role in the theory of copula. The relation between the copula and the joint distribution function of these random variables can be defined as

$$H(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)) = C(u_1, \ldots, u_d).$$

where $U_1 = F_1(x_1), ..., U_d = F_d(x_d)$, and $C : [0,1]^d \to [0,1]$ is a d-variate copula. The copula function is unique if the marginals are continuous. Note that, this study is restricted for the continuous case.

Copulas establish a link between a multivariate distribution and its individual marginals. When the marginal distributions are integrated into the copula, the multivariate distribution can be derived. Through copula calibration, the original random variables are transformed from their initial marginal distributions into uniform variates.In this manner, through copula inference, the relationship between variables can be explored from the perspective of marginal distributions, leading to a more comprehensive understanding of their associations. The copula distributions are especially useful for the inferences involving the bivariate random vectors because the bivariate copulas have various different forms to represent wide range of dependence characteristics. Overall, their application in data analysis is beneficial in terms of capturing non-obvious distribution structures in high dimensional data sets. For more details, see Nelsen (2006).

2.2 Bernstein Copula Classifier Framework

2.2.1 Bernstein Copula

The Bernstein polynomial with degree m is given as

$$B_{m,k}(z) = \binom{m}{k} z^k (1-z)^{m-k}, \tag{2.1}$$

$$\text{where } \; k = 0, ...m \in \mathbb{N} \; \text{ and } \; 0 \leq z \leq 1.$$

Let $\mathbf{U} = (U_1, U_2, \ldots, U_d)$ be a discrete random vector with uniform margins over the set $S_i = \{0, 1, ..., m_{i-1}\}$ where m_i denotes the grid size. Let the grid size of each argument equal to each other i.e. $m = m_1 = m_2 = \cdots = m_d$, then Bernstein copula can be defined as

$$C(u_1, u_2, \ldots, u_d) = \sum_{k_1=0}^{m} \sum_{k_2=0}^{m} \cdots \sum_{k_d=0}^{m} \alpha(k_1/m, k_2/m, \ldots, k_d/m) \prod_{i=1}^{d} B_{m,k_i}(u_i), \tag{2.2}$$

where,

$$\alpha(k_1, k_2, \ldots, k_d) = P\Big(\bigcap_{i=1}^{d} (U_i \leq k_i/m)\Big),$$

$$\{u_i\}_1^d \in [0,1]^d, \;\; \{k_i\}_1^m \in S^d,$$

where S^d is the d-dimensional set for k.

The equation (2.2) can be interpreted as Bernstein copula density is induced by $\mathbf{U}$. The constant function α can be any copula function to center the Bernstein copula. If the unknown copula function is approximated, the equation (2.2) formally is known as the empirical Bernstein copula. Hence, α refers to the counts of original realizations, fall in equally spaced cells within the d-dimensional hypercube. Thus, firstly contingency matrix with size of m^d is obtained from the α function computations. Secondly, the smoothing effect of Bernstein polynomials is applied on the empirical copula to complete the evaluation of Bernstein copula.

Numerous polynomial forms are available for representing continuous functions, and Bernstein polynomials share certain properties with copula distribution functions. Consequently, Bernstein polynomials can be applied in the context of copula theory. Additionally, Bernstein polynomials exhibit properties such as closure under differentiation and lower variance compared to common nonparametric estimators. For more details address, Sancetta (2004).

The implemented Bernstein distribution algorithm consists of three primary phases. These are *Grid Fit*, *Uniform optimization* and *Bernstein Density Computations* which are given with detail in the next section. Since α is an empirical copula within induced in d-dimensional canvas, to be a valid copula approximation for each marginal component of α should be distributed uniformly. This validation is not certain during the inference of Bernstein copula. In order to cover this condition, Pfeifer et al. (2020) proposed contingency table transformation. Let a_{st} be the original entries of grid fit, and b_{st} be the entries of new contingency table, then optimization problem can be stated as the following

$$\textit{Minimize} \sum_{s=1}^{m}\sum_{t=1}^{m}(a_{st}-b_{st})^2, \tag{2.3}$$

$$\text{subject to,}$$

$$\sum_{s=1}^{m} b_{sj} = \sum_{t=1}^{m} b_{it} = \frac{1}{m} \quad \text{and} \quad b_{ij} \geq 0,$$

$$\text{where} \quad i,j = 1,...,m.$$

Optimization problem is carried out by using *quadprog* library in R. To utilize the library for this particular problem, a function is designed to dynamically implement optimization for any contingency table derived from raw frequencies within grid cells. If non-negativity condition is neglected, this problem also can be immediately solved as a Lagrange problem. In this case, the general solution is

$$b_{st} = a_{st} - \frac{a_{.t}}{m} - \frac{a_{s.}}{m} + \frac{2}{m}, \tag{2.4}$$
$$\text{where} \;\; s, j = 1, ..., m,$$

and a_{st} matrix is the computed raw frequencies of the observations that fall into each cell of md dimensional hypercube. To recover the negative entries in contingency table, additive correction method can be used. Below, the final form of the contingency table c_{st} is defined.

$$b := -min\{b_{st} | 1 \leq b, \; s \leq m\},$$
$$c_{st} = \frac{b_{st} + b}{1 + m^2 \, a} \tag{2.5}$$

where a is the original entry of the grid fit. The matrix c_{st} provides initial or suboptimal solution for the defined minimization problem. Both the description of optimization problem and its initial solution can be easily enhanced to arbitrary dimensions. For details refer to Pfeifer et al. (2020).

It's important to note that both global and initial solutions are employed in real data applications for the purpose of comparison.

2.2.2 Setting Bernstein Copula Classifier

In machine learning applications, a Bayes rule framework is implemented, leveraging the joint Bernstein copula density. The Bernstein copula serves as an approximation to the true copula function. Adjusting the degree of polynomials allows for fine-tuning the smoothing effect in the Bernstein copula, ultimately leading to a more accurate approximation. Moreover, higher grid sizes enhance the smoothing and the capability to capture dependence patterns. When a sample is

approaching to the independence, the lower grid sizes would be more appropriate to capture a symmetrical pattern.Indeed, as the correlation weakens and approaches independence, the copula becomes increasingly similar to the independence copula, resulting in a data scatter that exhibits a more symmetrical pattern. Hence, using low grid sizes can indeed be more fitting for robustly capturing symmetrical patterns in such cases. See, Pfeifer et al. (2020), Rose (2015). Each training set is learned with the copula density which yields

$$c(u_1, u_2, \ldots, u_d) = \sum_{k_1=0}^{m-1} \sum_{k_2=0}^{m-1} \cdots \sum_{k_d=0}^{m-1} p(k_1/m, k_2/m, \ldots, k_d/m) \prod_{i=1}^{d} mB_{m-1,k_i}(u_i), \tag{2.6}$$

where,

$$p(k_1, k_2, \ldots, k_d) = p\Big(\bigcap_{i=1}^{d}(U_i = k_i/m)\Big).$$

In real data application, additionaly 2 component convex Bernstein density is used to improve the discrimination ability. A Convex Bernstein density can be defined as

$$c_{CB} = wc_{m_1} + (1-w)c_{m_2}, \tag{2.7}$$

where $w \in [0, 1]$.

The equation (2.7) indicates a weighted average of two different Bernstein densities estimated through sizes of m_1 and m_2, respectively. Cross-validation processes involve the initiation of various grid sizes to conduct a comprehensive analysis. The selected grid size is then applied to the current training sets for each category, resulting in contingency matrices. Subsequently, the test set is computed using the Bernstein copula density, applying each categorywise pre-determined contingency matrices.Following that, each observation in the test set is assigned to the category with the higher density. This setup is analogous to Gaussian Naive Bayes, with the key difference being that we abandon the independence assumption and employ the joint Bernstein copula for modeling dependence.

More formally, let there are two classes with class index l, and d-variate realized

random vector pseudo-observations; $\mathbf{u}_j = u_{1j}, u_{2j}, \ldots, u_{dj}, \quad j = 1, \ldots, n.$ After the calibration of Bernstein densities, $c_l(\mathbf{u}_j|l)$, the unlabeled data point $\mathbf{u}_j$ is assigned to the class l according to the decision rule,

$$\underset{l=1,2}{argmax} \pi_l(c_l(\mathbf{u}_j|l)) \tag{2.8}$$

In this study, we focus on estimating each density function $c_l(\mathbf{u}_j|l)$ and prior probabilities are set equals so $\pi_l = P(l) = \frac{1}{2}$ since 2-classes are considered.

The pseudo code for the classification process is presented in Figure 2.1, Figure 2.2 and Figure 2.3.

Step 1: Grid Fit

########
$gsize$: grid size
$glist$: grid list
$gpoints$: grid points
CM: contingency matrix
########
function JointGridFit(data, gsize)
d = number of data column (dimension)
n = number of data row (size)
$glist$ = for $1 : d$ do $vector(1, 2, ..., gsize)$
$gpoints = \times_d glist$
Initialize contingency matrix:
CM = ZeroMatrix(row = $gsize^d$, column = 1)
for ($i\ in\ 1 : n$) for ($j\ in\ 1 : m^d$) (if ($data_i \leq gpoints_j$) $(CM_j + 1)/n$ break)

Figure 2.1 Grid Fit step that is used in Bernstein copula module

Step 2: Uniform Optimization

if optimization == TRUE
Run $CM =$ *function GridOptimizer(CM)*
GridOptimizer() function designed to establish the coefficients of quadratic problem and constraints for the grid optimization problem in matrix forms then feed them into *solve.QP* from *quadprog* library.
return CM

Figure 2.2 Grid Optimization step that is used in Bernstein copula module

Step 3: Bernstein Density Computations

########
$sholder$: summation holder
c: counter
########
function BernDensity(CMobject = CM)
Initialize CMobject elements: $d, gpoints, CM, gsize$
n = number of of data row
nn = number of CM row
$for(1:d)$ index = find (data, gpoints == gsize)
gpoints = discard index from gpoints
CM = discard index from CM
# Initialize: $c = 0$, $sholder = Matrix(row = nn, column = 1)$, $density = Matrix(row = n, column = 1)$
(for $(j\ in\ 1:n)$
(for $(i\ in\ 1:nn)$
$sholder_c = gsize^d * CM_i * (apply\ product\ for\ 1 : d(Binomial(gpoints_{i,\,d}, gsize - 1, data_{j,\,d})))$
)
$Density_j$ = summing up $sholder$
$c = 0$
)
return $Density$

Figure 2.3 Density computation step that is used in Bernstein copula module

2.3 Vine Classifier Framework

2.3.1 Vine Distributions

Vine distribution process was developed through the works of Joe (1996), Bedford and Cooke (2001, 2002) to adopt non-obvious joint copulas. Originally, the process is known as the pair copula construction (PCC) and then embedded into the multiple directed tree representation (hence the name; vine) and also become a graphical tool to visualize distribution in arbitrary dimensions. The construction process of vines is based on the exchanging the parts of conditional distributions with bivariate conditional copula arguments. The chain rule, which refers to disintegrating a joint distribution into conditional parts is given as

$$f(x_1, x_2, \ldots, x_n) = f(x_1)\ f(x_2|x_1)\ \ldots\ f(x_n|x_1, x_2,\ \ldots,\ x_{n-1}). \tag{2.9}$$

The main conditional term for the exchange is given below,

$$f(x_i|x_{\mathbf{v}}) = c_{i,\ j|\mathbf{v}_{-j}}(F(x_i|x_{\mathbf{v}_{-j}}),\ F(x_j|x_{\mathbf{v}_{-j}}))\ f(x_i|x_{\mathbf{v}_{-j}}), \tag{2.10}$$

where $\mathbf{v}_{-j}$ is the vector which jth element is excluded.

The conditional arguments of copulas in the form of Eq. (2.10) can be expressed as,

$$h =_{i,\mathbf{v}} F(x_i|x_{\mathbf{v}}) = \frac{\partial C_{i,\ j|\mathbf{v}_{-j}}(F(x_i|x_{\mathbf{v}_{-j}}), F(x_j|x_{\mathbf{v}_{-j}}))}{\partial F(x_j|x_{\mathbf{v}_{-j}})}. \tag{2.11}$$

The marginals $F(x_i|x_{\mathbf{v}_{-j}})$ and $F(x_j|x_{\mathbf{v}_{-j}}))$ in Eq. (2.11) can be calculated recursively by plugging conditional copula term that calculated beforehand. The initial form of h function yields

$$F(x_i|x_j) = \frac{\partial C_{i,\ j}(F(x_i), F(x_j))}{\partial F(x_j)}. \tag{2.12}$$

In order to carry out h-function recursions for Bernstein vine, conditional distribution form of the Bernstein copula must be integrated into the recursions. The conditional Bernstein copula distribution in bivariate case can be defined with partial derivative as below. (Janssen et al., 2016)

$$\frac{\partial C(u_1, u_2)}{\partial u_2} = m \sum_{k_1=0}^{m} \sum_{k_2=0}^{m-1} p^{(2)}(k_1, k_2) B_{m,k_1}(u_1) B_{m-1,k_2}(u_2) \tag{2.13}$$

where $p^{(2)}(k_1, k_2)$ refers that contingency matrix is completed by fixing the second argument.

By using directed tree terminology, vine density can be defined in a compact fashion. Let E be the edge set and $e \in E_i$ ($i = 2, ..., n-2$), $e = j(e), k(e)|D(e)$, where $\{j(e), k(e)\}$ is the conditioned set of bivariate elements and $D(e)$ is the conditioning set. The constraint set U^c that corresponds to edge e is defined as a complete union shown as

$$U^c = \{j(e), k(e), D(e)\}. \tag{2.14}$$

See Figure 2.4 which presents a five-dimensional vine density to see the calibration scheme in detail.

A unique density function f for a d-variate random vector $\mathbf{X}$ can be obtained by transforming Eq. 2.9 in terms of copula functions recursively. Hence a d-variate distribution can be expressed in terms of the product of bivariate (conditional) copulas as,

$$f(x_1, \ldots, x_n) = \Big[\prod_{k=1}^{d} f_k(x_k)\Big] \times \Big[\prod_{i=1}^{d-1} \prod_{e \in E_i} c_{j(e),k(e)|D(e)}(F_{j(e)|D(e)}, F_{k(e)|D(e)})\Big]. \tag{2.15}$$

Note that, if the left hand side density in Eq. 2.15 is expressed with its copula, this equation becomes a vine copula with the simplification of right hand side marginals. In this arrangement, T_1 has marginals as nodes, then for $T_{\geq 2}$ each edge $e \in E_i$ has distributions with the general form,

$$c_{j(e),k(e)|D(e)}(F_{j(e)|D(e)}, F_{k(e)|D(e)}). \tag{2.16}$$

As the root nodes are fundamental for ordering and calibration, selecting a vine's root can influence the vine's shape and distribution of nodes.

2.3.2 Structuring A Valid Vine Distribution

Dissman et al. (2013) proposed an algorithm which connects highly correlated pairs in each tree by the widely known measure Kendall's tau. Prioritizing the

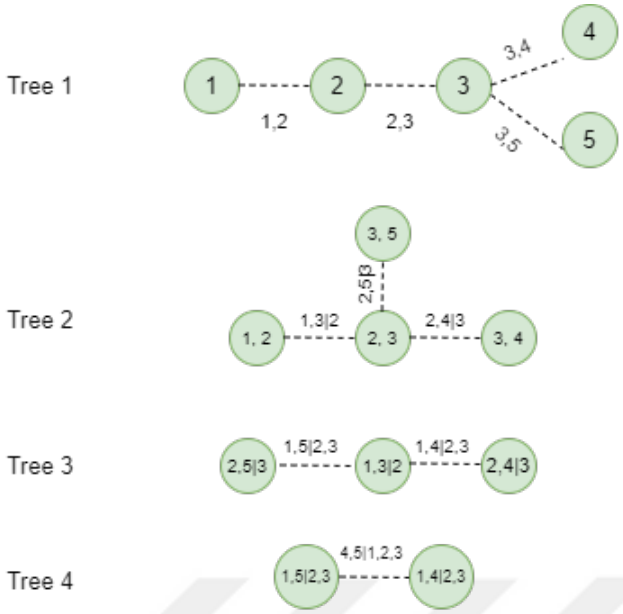

Figure 2.4 A five-dimensional R-vine graphic:Tree 4 has node set: $N_4 = \{(1,5|23),(1,4|23)\}$ and edge set: $E_4 = \{(4,5|123)\}$. The edge $(4,5|123)$ can be inspected as contraint elements: $j(e) = 4$, $j(e) = 5$, $D(e) = \{1,2,3\}$.

appropriate calibration of correlated variables in the early tree segments is important since copulas are generally successful at independent cases. For Kendall's tau $\tau_{i,j}$, the objective function of absolute value summation yields,

$$max \sum_{edges\ e\ in\ spanning\ tree} |\tau_{i,j}|, \tag{2.17}$$

where $-1 \leq \tau_{i,j} \leq 1$, $i \neq j$ denotes the edge elements of the spanning tree. In the main algorithm, this optimization is achieved by the NetworkToolbox library of R. To adopt a valid vine distribution, two constraints should be satisfied; connectedness of tree and proximity condition. Therefore, while maximizing a particular spanning tree, the constraints should be checked through the vine structure setup. Let V be $n-$ variate vine, then connected tree and proximity conditions are defined as,

- **Connected tree:** Every node is reachable from any starting node only with total of $n-1$ edges.
- **Proximity Condition:** If $\{x_i, x_j\} \in E$(any edge set of a spanning tree), then

$|x_i \triangle x_j| = 2$, for $i \neq j$.

2.3.3 Machine Learning Framework: Vine Bayes

For the machine learning applications, vine densities are constructed as discriminant functions which represent the training data for each class labels. Then the fitted vine models are compared to determine the class membership of the current test data point. In this study, we work on supervised data with two categories hence the decision rule is defined accordingly. Let $(\mathbf{x}|l) = ((x_{1i}, x_{2i}, \ldots, x_{di})|l)$ be realized d-variate random vector for each categories $l = 1, 2$ and sample size $i = 1, 2, \ldots, n$. In the learning process, two vectors are fitted to the class labelled vine densities $f(x|l)$ as in Eq. 2.15. And then, the decision rule for test data point $\mathbf{x}_t$ that is, assign $\mathbf{x}_t$ to class L where,

$$L = \underset{l=1,2}{argmax}(f(\mathbf{x}_t|l)). \tag{2.18}$$

Classification procedure is applied with 3 different vine approaches; parametric vine, Bernstein vine and hybrid vine. The posterior probability is set as equal i.e. $f(\mathbf{x}_t|l) = 0.5$. Parametric vine density involving independence, Gumbel, Clayton and Gaussian bivariate copulas is used as a benchmark model. Bernstein vine is the vine model fully integrated with Bernstein density and conditional Bernstein distribution. And finally the hybrid vine model is referred to a semi-parametric vine process that chooses between the parametric copulas and Bernstein copula and also assigns the adequate one to the current vine node. The procedure of choosing among a semi-parametric distribution candidates is given in the following section in detail.

2.3.4 Truncated Vine Model

Truncated vine distributions are the vines structured until a determined tree level k. After the k^{th} level, structuring does not continue and rest of the node densities are evaluated as independence copula which is 1. The truncation level is implemented

inside of the vine distribution module, allowing establishment of vine until given level, then the algorithm exits with the information of partial vine structure. Also, the density algorithm of a pre-determined vine structure is adjusted in order to evaluate the density (density product of each node) according to the limited structure.

A sample structure can be visualized in Figure 2.5. In earlier studies, Brechman et al. (2012) used the truncated vine distributions based on Bayesian Information Criterion (BIC) metric. They mainly examined the issue of whether, following a specific tree, R-vine copulas can be pairwise reduced or, alternatively, simplified using Gaussian pair-copulas. They used the simplification of a canonical vine copula using a multivariate copula as previously treated by Heinen and Valdesogo (2009) and Valdesogo (2009). Additionally, Carrera, D. (2019) employed BIC to create truncated vines for a vine-based categorization technique. According to this strategy, a selected metric is computed for each tree level of vine and if the metric of tree T_{k+1} is smaller than the previous tree T_k then the vine is truncated after the tree level k and the independence copula assigns to all trees. In our work, we establish the truncation level using this process in a machine learning validation. The truncation level is chosen based on the validation set's highest accuracy percentage. As a result, the machine learning grid search technique for the cut-off parameter yields an optimal level. By applying the truncated vine distributions, it is aimed to prevent possible overfitting caused by the complex vine structures. As getting more copula densities included in the vine, the model could suffer from complexity. In addition, correlations are getting weak as trees advance, hence it will be more appropriate to use independence copula after some tree level. To use the truncated vine in machine learning, the validation set convention is used along with the training set. After training the model with vine calibration, the model is tried on the validation set for all of the cut points. And the cut point k which gives the highest accuracy is selected. Hence, the decision metric is determined by the classification accuracy achieved on the validation set.

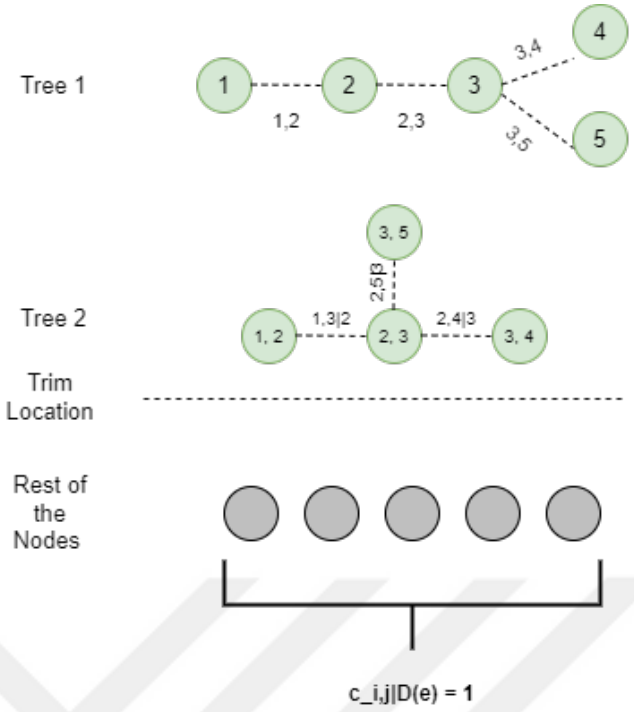

Figure 2.5 A five-dimensional truncated R-vine graphic. Vine is cut from tree level 3. Nodes(densities) after the level 2 are taken as the independence copula.

2.3.5 Hybrid Vine Model

The process of selecting a copula from the candidate set of parametric copulas and Bernstein copula for structuring a hybrid vine model has the following steps given below. Note that, since the Bernstein copulas with the largest grid size always have the higher score in our experiments, we use only one Bernstein copula with a moderate grid size,

Step 1 Fit current data to parametric copulas and generate number of K simulations for each parametric model.

Step 2 Compute mean Akaike score for each model then select the copula which has the minimum score.

Step 3 Use the selected parametric copula and compute density values. This computed values will be the *baseline density*. Then generate number of K simulations based on this copula.

Step 4 Compute the density values for number of K simulations by using both the selected parametric copula and Bernstein copula. (*candidate densities*)

Step 5 Determine the mean relative distance between the baseline density and candidate densities.

Step 6 Continue with the originally selected parametric copula if its distance is smaller or change the selected parametric copula with the Bernstein copula if its distance is smaller and assign it to the current vine node.

Akaike Information Criterion (AIC) score is obtained by the expression,

$$2k - 2L(\theta|u, v), \tag{2.19}$$

where k is the number of parameters,

and L is the copula log likelihood.

And then, relative distance is calculated by the distance formula below as in the study of Rose (2015),

$$RD_{N,p} = \frac{||c_\theta - c_e||_{N,p}}{||c_e||_{N,p}}. \tag{2.20}$$

where N is the sample size and p is the distance order. In the distance calculcations p set as 2 so, the numerator is equivalent to the mean squared error (MSE).

Pseudo-code for the selection process is also given in Algorithm 1. Note that, the inner functions are only expressed with symbolic names to point out their role, and they are either customly written or used from base functions of RStudio (2013) along with the VineCopula package functions.

```
function CopulaSelector(data, paramList, nonparamList, trialNumber, relativeDistance())
    ▷ Initialize the vector that holds the average akaike score that corresponds to the ith copu
    aic ← vector()
    for i in paramList do
        Initialize: aicDummy ← vector()
        copPar ← function fitCopula()
        for j in trialNumber do
            set jth random number seed
            copSim ← Simulated observations for data based on copPar
            aicDummy[j] ← Akaike score based on the jth simulation
        end for
        aic ← average of aicDummy
    end for
  ▷ Initialize the following variables for the second part of the algorithm,
    paramCopCand ← minimum of aicDummy
    baseCopPar ← function fitCopula(data)
    baseCopPdf ← function copulaPdf(data, baseCopPar)
    hybridCopList ← string(paramCopCand, nonparamList)
▷ Vector that holds the average of relative distance that corresponds to ith copula.
    rd ← vector()

    for i in hybridCopList do
        Initialize: rdDummy ← relative distance of the current trial
        for j in trialNumber do
            set jth random number seed
      ▷ Simulate a bootstrap sample with the ith copula and make needed calculations.
            copSim ← function simulateCopula()
            bootstrapCopPar ← function fitCopula(copSim)
            bootstrapCopPdf ← function copulaPdf(copSim, bootstrapCopPar)
            rdDummy[j] ← function relativeDistance(bootstrapCopPdf, baseCopPdf)
        end for
        rd[i] ← average of rdDummy
    end forreturn chosenCop ← copula that matches the minimum value index of rd
end function
```

Figure 2.6 Select a copula in a semi-parametric environment.

2.4 Neural Copula

Neural networks can operate in parallel to learn and estimate cumulative distribution functions (CDFs) through the integration of specialized loss functions.

These loss functions enforce the necessary conditions for a model to satisfy the properties of both CDFs and probability density functions (PDFs). To ensure the validity of the probabilistic model, penalties for violating these conditions are incorporated into the system. According to the work of Zeng & Wang (2022) Zeng (2022) such networks have the capability of reaching a convergence. The study is inspired from the applications on the constructing Archimedes copulas based on neural networks Ling et al. (2020). In general, neural based distributions offer a generalized distribution representation combined with pattern learning power of neural networks. After calibrating a distribution as a tensor, necessity of fitting a data into known distribution families can be bypassed.

2.4.1 Neural Copula General Framework

To construct a copula model with the network, the marginal model is first learned using neural networks. As the initial step, marginal distributions are estimated through several fully connected neural networks. These networks represent the arguments of the copula and collectively form the marginal model. By estimating the CDF using the marginal model, it is integrated into the copula network, resulting in the construction of the copula's CDF. TensorFlow's automatic differentiation system enables efficient computation of gradients for each argument, allowing the joint PDF to be estimated as the gradient of the network with respect to its input layer.

Given the neural network's flexibility in gradient computation, it is practical to first estimate the CDF. Estimating the PDF directly with a neural network would require a chain integration system, which is less flexible.

2.4.1.1 Constructing Marginal Networks

The fully connected marginal CDF which represented by fully connected neural network can be expessed as:

Input Layer:

$$\mathbf{h}_m^0 = x \in \mathbf{\Omega} \subseteq \mathbb{R} \tag{2.21}$$

where $\mathbf{\Omega}$ is the domain of x.

Hidden layer:

$$\mathbf{h}_m^{j+1} = \tanh\left(\mathbf{w}_m^j \mathbf{h}_m^j + \mathbf{b}_m^j\right) \quad j \in \{0, 1, \ldots, l_m - 1\} \tag{2.22}$$

Output layer:

$$\hat{F}_m = Sigmoid\left(\mathbf{w}^{l_m}\mathbf{u}^{l_m} + \mathbf{b}^{l_m}\right) \in [0, 1] \tag{2.23}$$

The marginal model is denoted as the $\hat{F}_m(x, \boldsymbol{\theta}_m)$, where $\boldsymbol{\theta}_m$ is the corresponding network weights that is $\boldsymbol{\theta}_m = \{\mathbf{w}_m^j\} \cup \{\mathbf{b}_m^j\}$ By conducting the gradient operation on the whole network the PDF can be obtained as

$$\hat{f}_m(x, \boldsymbol{\theta}_m) = \frac{d\hat{F}_m(x, \boldsymbol{\theta}_m)}{dx} \tag{2.24}$$

2.4.1.2 Constructing Copula Network

The network that represens a copula is expressed as below.

Input Layer:

$$\mathbf{h}_c^0 = \mathbf{u} = [u_1, \ldots, u_d]^T \in [0, 1]^d \tag{2.25}$$

Hidden layer:

$$\mathbf{h}_c^{j+1} = \tanh\left(\mathbf{w}_c^j \mathbf{h}_c^j + \mathbf{b}_c^j\right) \quad j \in \{0, 1, \ldots, l_c - 1\} \tag{2.26}$$

Output layer:

$$\hat{C}_m = Sigmoid\left(\mathbf{w}^{l_c}\mathbf{u}^{l_c} + \mathbf{b}^{l_c}\right) \in [0, 1] \tag{2.27}$$

- d denotes the dimension of the u vector.
- The copula model has total of $l_c - 1$ hidden layers.
- The j^{th} layer has the weight: $\mathbf{w}_c^j$ and bias: $\mathbf{b}_c^j$
- For the estimated copula function $\hat{C}_m(\mathbf{u}, \boldsymbol{\theta}_c)$ set of all weights can be shown as $\boldsymbol{\theta}_c = \{\mathbf{w}_c^j\} \cup \{\mathbf{b}_c^j\}$
- By the networks gradient mechanism the PDF of copula can be obtained as:

$$\hat{c}(\mathbf{u}, \boldsymbol{\theta}_c) = \frac{\partial^d \hat{C}_m(x, \boldsymbol{\theta}_m)}{\partial \mathbf{u}}$$

By the estimated copula distribution, the PDF of the sampled data can be expressed as

$$\hat{f}\left(\mathbf{x}, \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c\right) = \hat{c}\left(\hat{F}_1\left(x_1, \boldsymbol{\theta}_m^1\right), \hat{F}_2\left(x_2, \boldsymbol{\theta}_m^2\right), \ldots, \hat{F}_d\left(x_d, \boldsymbol{\theta}_m^d\right), \boldsymbol{\theta}_c\right) \prod_{i=1}^{d} f_i\left(x_i, \boldsymbol{\theta}_m^i\right) \tag{2.28}$$

The CDF of the sampled data can be expressed as

$$\hat{F}\left(\mathbf{x}, \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c\right) = \hat{C}\left(\hat{F}_1\left(x_1, \boldsymbol{\theta}_m^1\right), \hat{F}_2\left(x_2, \boldsymbol{\theta}_m^2\right), \ldots, \hat{F}_d\left(x_d, \boldsymbol{\theta}_m^d\right), \{\theta_m^i\}, \boldsymbol{\theta}_c\right) \tag{2.29}$$

2.4.2 Loss Functions of the Marginal Model

2.4.2.1 Log-loss

The fitness of $\hat{f}_m(x, \boldsymbol{\theta}_m)$ can be integrated in the loss function as maximum likelihood:

$$L_m^{(1)}(\theta_m) = -\frac{1}{n_m^l} \sum_{i=1}^{n_m^l} \log \hat{f}_m(x_i, \boldsymbol{\theta}_m) \tag{2.30}$$

The loss $L_m^{(1)}$ involves the training samples: $D_m^1 = \{x_i\}$ where $i = 1, \ldots, n_m^1$

2.4.2.2 Non-negativeness

The violation of $\hat{f}_m(x, \boldsymbol{\theta}_m) > 0$ is added as penalty coming from the negative part:

$$L_m^{(2)}(\boldsymbol{\theta}_m) = \int_{x_m \in \Omega_m} \text{relu}(-\hat{f}_m(x, \boldsymbol{\theta}_m)) dx_m \tag{2.31}$$

The sample version can be calculated as

$$L_m^{(2)}(\theta_m) \approx \frac{1}{n_m^l} \sum_{i=1}^{n_m^l} \text{relu}(-\hat{f}_m(x_i, \boldsymbol{\theta}_m)) \tag{2.32}$$

The loss $L_m^{(2)}$ involves the training samples: $D_m^2 = \{x_i\}$ where $i = 1, \ldots, n_m^2$

2.4.2.3 PDF Summation Over Domain Leads to 1

Penalty is added if the summation of $\hat{f}_m(x, \boldsymbol{\theta}_m)$ over the domain is not 1 as

$$L_m^{(3)}(\boldsymbol{\theta}_m) = \left| 1 - \int_{x_m \in \Omega_m} \hat{f}_m(x, \boldsymbol{\theta}_m) dx_m \right| \tag{2.33}$$

The sample version can be calculated as:

$$L_m^{(3)}(\boldsymbol{\theta}_m) \approx \left| 1 - \frac{1}{n_m^l} \sum_{i=1}^{n_m^l} \hat{f}_m(x_i, \boldsymbol{\theta}_m) \lambda_m \right| \tag{2.34}$$

The loss $L_m^{(3)}$ involves the training samples: $D_m^3 = \{x_i\}$ where $i = 1, \ldots, n_m^3$

2.4.2.4 CDF function constraints

$\hat{F}_m(x, \boldsymbol{\theta}_m)$ must satisfy the following conditions:

$$\begin{cases} \hat{F}_m(0, \boldsymbol{\theta}_m) = 0 \\ \hat{F}_m(1, \boldsymbol{\theta}_m) = 1 \end{cases} \tag{2.35}$$

The loss function for the given constraint yields:

$$L_m^{(4)}(\theta_m) = \hat{F}_m(0, \boldsymbol{\theta}_m) + \left| 1 - \hat{F}_m(1, \boldsymbol{\theta}_m) \right| \tag{2.36}$$

The loss $L_m^{(4)}$ involves the training samples: $D_m^4 = \{0, 1\}$

2.4.3 Loss Functions of the Copula Model

2.4.3.1 Log-loss

The fitness of $\hat{f}_c(\mathbf{x}_i; \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c)$ used as penalty in the loss:

$$L_c^{(1)}(\{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c) = -\frac{1}{n_c^1} \sum_{i=1}^{n_c^l} \log \hat{f}_c(\mathbf{x}_i; \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c) \tag{2.37}$$

The loss $L_c^{(1)}$ involves the training samples: $D_c^1 = \{\mathbf{x}_i\}$ where $i = 1, \ldots, n_c^1$

2.4.3.2 Non-negativeness

The violation of $\hat{f}_c(\mathbf{x}_i; \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c) > 0$ is added as penalty coming from the negative part:

$$L_c^{(2)}(\{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c) = \int_{x_1 \in \Omega_1} \cdots \int_{x_d \in \Omega_d} \text{relu}(-\hat{f}_c(\mathbf{x}_i; \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c)) dx_1 \cdots dx_d \tag{2.38}$$

With numerical approximation the calculation can be simply made as:

$$L_c^{(2)}(\{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c) \approx \frac{1}{n_c^2} \sum_{i=1}^{n_c^2} \text{relu}\left(-\hat{f}_c\left(\mathbf{x}_i, \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c\right)\right) \tag{2.39}$$

The loss $L_c^{(2)}$ involves the training samples: $D_c^2 = \{\mathbf{x}_i\}$ where $i = 1, \ldots, n_c^2$

2.4.3.3 PDF Summation Over Domain Leads to 1

Similarly if the sum of $\hat{f}_c\left(\mathbf{x}_i, \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c\right)$ over the domain is not 1 penalty is added:

$$L_c^{(3)}(\{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c) = \left| 1 - \int_{x_1 \in \Omega_1} \int_{x_2 \in \Omega_2} \cdots \int_{x_d \in \Omega_d} \hat{f}_c \left(\mathbf{x}_i, \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c \right) d\mathbf{x} \right| \tag{2.40}$$

Calculation can be approximated as

$$L_c^{(3)}(\{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c) \approx \left| 1 - \frac{1}{n_c^3} \sum_{i=1}^{n_c^3} \hat{f}_c \left(\mathbf{x}_i, \{\boldsymbol{\theta}_m^i\}, \boldsymbol{\theta}_c \right) \prod_{j=1}^{d} \Delta_j \right| \tag{2.41}$$

The loss $L_c^{(3)}$ involves the training samples: $D_c^3 = \{\mathbf{x}_i\}$ where $i = 1, \ldots, n_c^3$

2.4.3.4 CDF function constraints

$\hat{C}(\boldsymbol{u}, \boldsymbol{\theta}_c)$ must satisfy Copula's definition:

$$\begin{cases} \hat{C}(\underline{\mathbf{u}}_1, \boldsymbol{\theta}_c) = \hat{C}(0, u_2, \ldots, u_d, \boldsymbol{\theta}_c) = 0 \\ \hat{C}(\underline{\mathbf{u}}_2, \boldsymbol{\theta}_c) = \hat{C}(u_1, 0, u_3, \ldots, u_d, \boldsymbol{\theta}_c) = 0 \\ \vdots \\ \hat{C}(\underline{\mathbf{u}}_d, \boldsymbol{\theta}_c) = \hat{C}(u_1, u_2, \ldots, 0, \boldsymbol{\theta}_c) = 0 \\ \hat{C}(\bar{\mathbf{u}}_1, \boldsymbol{\theta}_c) = \hat{C}(u_1, 1, 1, \ldots, 1, \boldsymbol{\theta}_c) = u_1 \\ \hat{C}(\bar{\mathbf{u}}_2, \boldsymbol{\theta}_c) = \hat{C}(1, u_2, 1, \ldots, 1, \boldsymbol{\theta}_c) = u_2 \\ \vdots \\ \hat{C}(\bar{\mathbf{u}}_d, \boldsymbol{\theta}_c) = \hat{C}(1, 1, \ldots, u_d, \boldsymbol{\theta}_c) = u_d \end{cases} \tag{2.42}$$

Corresponding loss function can be expressed as

$$L_c^{(4)}(\{\boldsymbol{\theta}_m^i\},\boldsymbol{\theta}_c)=\sum_{i=1}^{d}\sum_{j=1}^{n_c^4}\hat{C}(\underline{\mathbf{u}}_i^j;\boldsymbol{\theta}_c)+\sum_{i=1}^{d}\sum_{j=1}^{n_c^4}\left|\hat{C}(\bar{\mathbf{u}}_i^j;\boldsymbol{\theta}_c)-u_i^j\right| \tag{2.43}$$

The loss $L_c^{(4)}$ involves the training samples:

$$D_c^4=\bigcup_{i=1}^{d}\left(\left\{\underline{\boldsymbol{u}}_i^j\right\}_{j=1,2,\ldots,n_c^4}\bigcup\left\{\bar{\boldsymbol{u}}_i^j\right\}_{j=1,2,\ldots,n_c^4}\right) \tag{2.44}$$

2.4.3.5 Uniqueness of CDF

If we only have information about the distribution of the derivatives of the cumulative distribution function (CDF) and the boundary values of the CDF in higher dimensions ($d \geq 3$), it is impossible to definitively ascertain the shape of the CDF. To preserve the the uniquness of CDF one must integrate a loss function to the network. Firstly, we know that at a particular point the $\hat{F}_c(\mathbf{x}_i;\{\boldsymbol{\theta}_m^i\},\boldsymbol{\theta}_c)$ satisfy

$$\hat{F}_c\left(\boldsymbol{x},\{\boldsymbol{\theta}_m^i\},\boldsymbol{\theta}_c\right)=\int_{-\infty}^{x_d}\cdots\int_{-\infty}^{x_2}\int_{-\infty}^{x_1}\hat{f}_c\left(\boldsymbol{y},\{\boldsymbol{\theta}_m^i\},\boldsymbol{\theta}_c\right)dy_1dy_2\cdots dy_d \tag{2.45}$$

Approximation yields

$$\hat{F}_c\left(\boldsymbol{x},\{\boldsymbol{\theta}_m^i\},\boldsymbol{\theta}_c\right)\approx\frac{1}{n_c^1}\sum_{\boldsymbol{y}\in D_c^1}\text{flag}(\boldsymbol{x},\boldsymbol{y}) \tag{2.46}$$

where

$$\text{flag}(\boldsymbol{x},\boldsymbol{y})=\begin{cases}1 & \forall j\leq d,\ y_j<x_j\\ 0 & \text{otherwise}\end{cases} \tag{2.47}$$

According to the constraints the loss function can be defined as

$$L_c^5\left(\{\boldsymbol{\theta}_m^i\},\boldsymbol{\theta}_c\right)=\frac{1}{n_c^1 n_c^5}\sum_{i=1}^{n_c^5}\left|\hat{F}_c\left(\boldsymbol{x}_i,\{\boldsymbol{\theta}_m^i\},\boldsymbol{\theta}_c\right)-\sum_{y\in D_c^1}\text{flag}(\boldsymbol{x}_i,\boldsymbol{y})\right| \tag{2.48}$$

The loss $L_c^{(5)}$ involves the training samples: $D_c^5=\{\mathbf{x}_i\}$ where $i=1,\ldots,n_c^5$

2.4.4 Integration of Main Loss Function

The loss functions of the marginal and copula networks can be defined respectively as the linear combination of individual loss functions

$$L_m\left(\boldsymbol{\theta}_m\right)=\sum_{k=1}^{4}\lambda_k L_k\left(\boldsymbol{\theta}_m\right) \tag{2.49}$$

$$L_c\left(\left\{\boldsymbol{\theta}_m^i\right\},\boldsymbol{\theta}_c\right)=\sum_{k=1}^{5}\lambda_k L_k\left(\left\{\boldsymbol{\theta}_m^i\right\},\boldsymbol{\theta}_c\right) \tag{2.50}$$

2.5 Graphical Summary

A graphical summary from the work of Zeng (2022) is also added to visualize the combined network structures neatly.

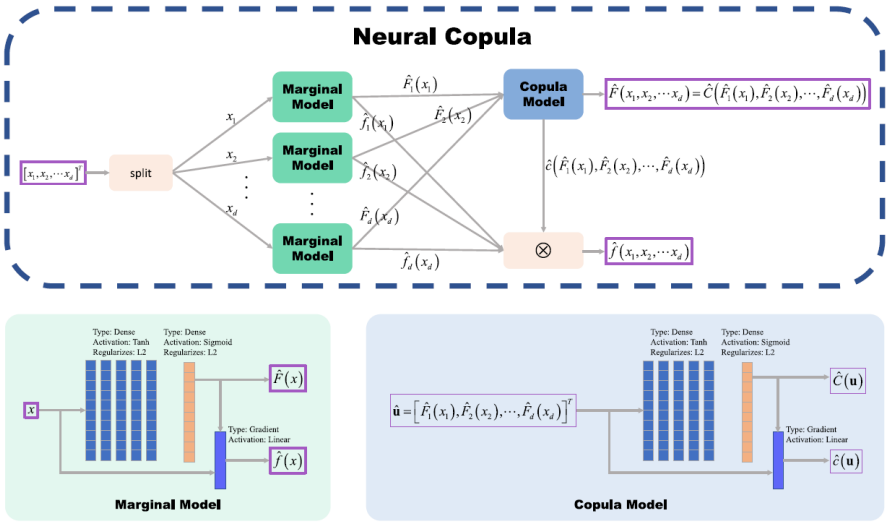

Figure 2.7 General framework of Neural Copula and its networks: Marginal and Copula Model

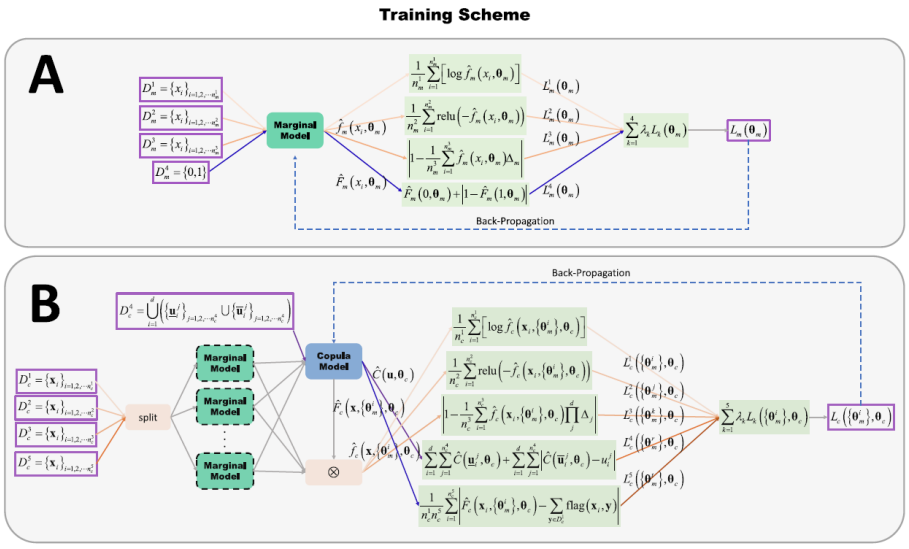

Figure 2.8 Training of Neural Copula

CHAPTER THREE
APPLICATION

3.1 Bernstein Copula Classifier

In this section, first Bernstein copula and well-known copula distribution classifiers are compared in terms of classification performance on simulated data. Then a classification performance analysis is conducted on benchmark biological data through convex Bernstein copula classifier.

3.1.1 Simulation Study

In the simulation study, the focus is on classifying the generated copula random variables. Two well-known copulas, the Gumbel and Clayton copulas, are designated as two distinct classes, and observations are generated to represent various dependence structures. Total of 100 observations are generated for each copula family. We preferred small sample size for the simulation design since the used real data in this study has small sample size (which is a frequent case in biomarker data) and we wanted to obtain an equivalent observation. These two Archimedean families of copulas exhibit different dependence characteristics: the Gumbel copula displays right-tailed dependence, while the Clayton copula showcases left-tailed dependence. Our interest lies in assessing how effectively the Bernstein copula classifier can capture the patterns inherent in the selected parametric copulas. Furthermore, observations generated from a Gaussian copula with varying dependence structures are also classified.

In the machine learning process involving the Bernstein copula, a grid search is performed, and only the best results are presented alongside the optimal grid sizes. To prevent oversmoothing and expedite computation, grid size permutations are employed. The maximum grid size for a bivariate Bernstein copula is set at 20, while for a convex Bernstein copula, it is set at 6.

Classification using the Gaussian distribution is performed within a Naive Bayes framework. Naive Bayes is a valuable method for density-based machine learning because it assumes independence between random variables, which greatly simplifies the classification process. As a result, Gaussian Naive Bayes (GNB) is utilized as a benchmark for the machine learning simulation. 10-fold cross validation with repetition is used for evaluation criteria.Each 10-fold cross-validation is repeated 20 times. This repetition allows for the training and testing data to be drawn from various parts of the dataset, enabling a comprehensive analysis of the performance of the learned models. Simulations consist of weak-weak, strong-strong and strong-weak associations in terms of Kendall's tau correlations. In our evaluation, we employ accuracy as a common performance measure in machine learning to assess the classification model's effectiveness.

In Figure 3.1, each figure illustrates the decision regions with actual test data points. The first column displays the performance of the Bernstein copula classifier for the fold in which the maximum accuracy is achieved. For the parametric copula and Gaussian Naive Bayes classification, the figures are generated using the same fold as the Bernstein copula. The GCC classifiers exhibit symmetrical patterns with curved regions. In the case of a strong-weak structure, the classifier for the higher Kendall's tau tends to create its discriminating region in the middle due to the symmetry of the associated data. In the $0.7 - 0.7$ case, GCC manages to capture more of the structure, while in the $0.2 - 0.2$ case, it struggles to determine adequate regions. Interestingly, Gaussian Naive Bayes (GNB) categorizes regions as linear, curved, and circular, depending on the best possible outcome for classification. In comparison to BC and GCC, GNB fails to capture tail locations and focuses more on central and non-central regions, especially when there are different associations among learned models. Unlike GCC and GNB, BC exhibits asymmetric localizations with irregular patterns.

Nevertheless, the patterns observed in BC are relatively close to GCC, especially in cases involving strong-weak dependence structures. Overall, BC exhibits behavior that is similar to GCC.

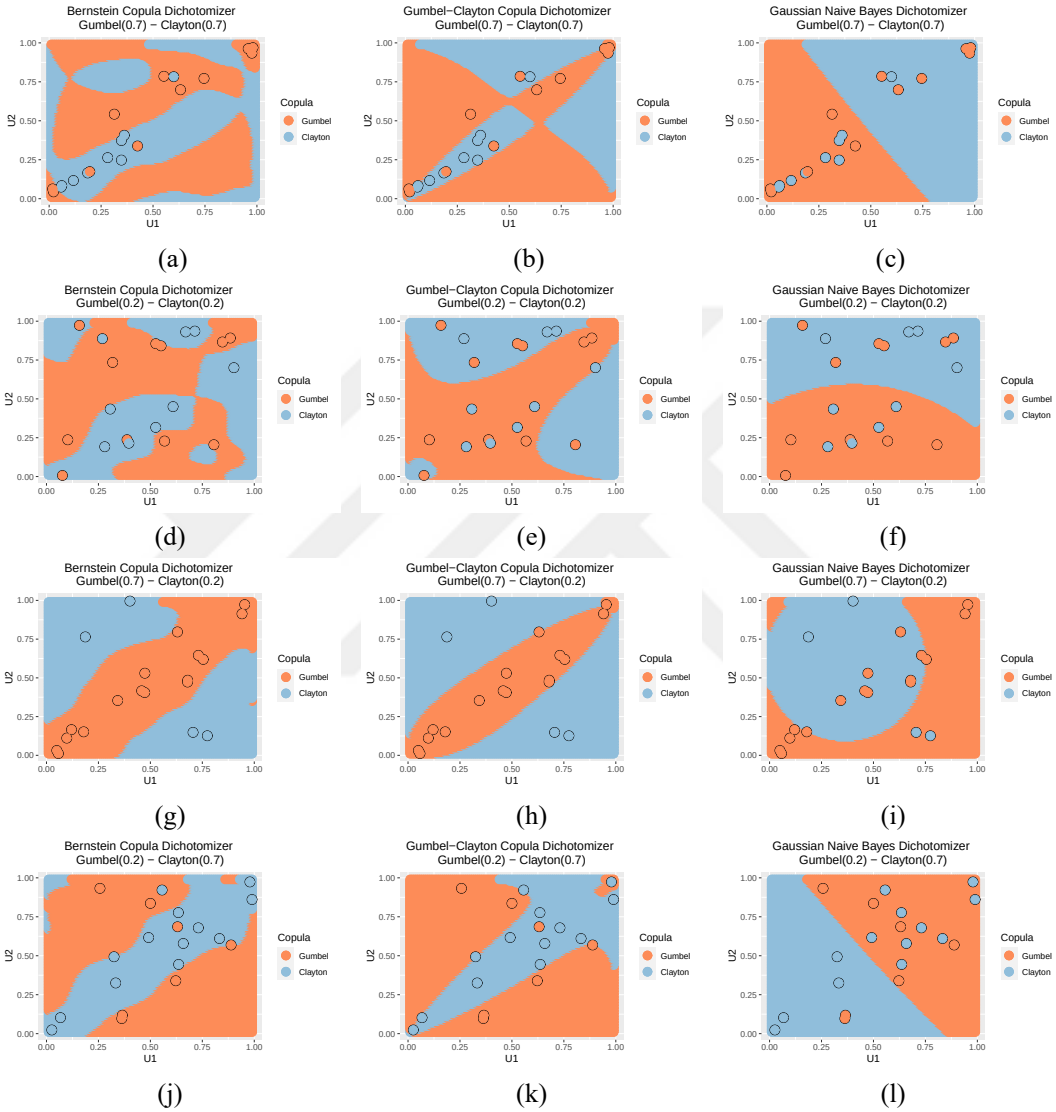

Figure 3.1 Decision regions with actual test data points. **First column shows the Bernstein copula classifier's performance for the fold with maximum performance. For other columns, the displayed results are obtained by using the same fold as BC in order to compare.

3.1.2 Real Life Data:Coimbra Breast Cancer

Early detection and treatment plans are essential for the notorious diseases such as breast cancer. Machine learning proves to be a valuable supportive tool in medical area being an objective decision maker for the assessment of the diseases. To assess the potential of the Bernstein copula classifier in this context, we establish a machine learning framework for the Coimbra Breast Cancer dataset.The data is available on UCI machine learning repository. The data consists of several features namely; Glucose, Insulin, HOMA, Leptin, Adinopectin, Resistin, MCP-1, Age and Body Mass Index (BMI). Those biomarkers were collected through consultation and blood analysis for modeling of the obesity-associated breast cancer since they are valid indicators K. et al. (2013). The data has total of 116 subjects; 52 of them are in the control group and 64 subject shows presence of the breast cancer. In the simulation analysis, the classification results obtained using the Bernstein copula are promising, particularly when the density function is unknown. In further analysis, we extend our investigation to real data classification within a high-dimensional framework to study the behavior of the joint Bernstein copula density when there are more than two variables involved. First, we adopt feature importance order from the study Patrício et al. (2018). The order reflects the individual impact of biomarkers on breast cancer models by using the Gini's coefficient. In decreasing fashion, the order is; Glucose, Resistin, Age, BMI, HOMA, Leptin, Insulin, Adinopectin, MCP-1. For classification purposes, the first four biomarkers, ordered by their importance, are selected as the feature space. As it can be seen in the Kendall's tau values in Figure 3.2, there are relatively low correlations between biomarkers, falling within the [-0.25, 0.25] range. The highest correlation value is observed in the pair (Glucose, Age) with a Kendall's tau-value of 0.13. Therefore, there is a need for a copula model that can capture data with low dependence. In this case, the Bernstein copula with relatively lower grid sizes would be more suitable, especially considering the symmetrical nature of the dependence. To classify the presence of breast cancer, we establish learned models using cross-validation. However, due to the limitations of the joint Bernstein copula in handling high-dimensional data, we establish four-dimensional models. In addition

to the Bernstein copula, we also implement the convex Bernstein copula as a classifier to capture weighted density information from various calibrated Bernstein densities. As discussed in Section 2.2.1, achieving copula validity requires marginal transformation to uniformity, which is accomplished by solving the described optimization problem. During cross-validation, each fold is learned through marginal optimization. However, it's worth noting that this process gradually slows down the running time of the machine learning algorithm as the number of variables increases, along with the grid size and dimension. The initial solution described in Section 2.2.1 is also employed as a support for the cases where global optimization is unattainable. In situations where optimization cannot be achieved, the problematic folds are excluded from the analysis, and the results are compared accordingly. In the context of supervised learning, both 10-fold cross-validation with 20 repetitions and leave-one-out cross-validation are applied.

In the first step, the grid search is implemented for each pair from $\{3,4,5,6\} \times \{3,4,5,6\}$. The grid search is configured in this specific manner because, when the grid size exceeds 6, non-convergent results become increasingly common, in addition to significantly slowing down the computation time. Table 3.1 displays the classification performance of BC for each grid. The ordered grid pairs of 6-4, 6-5, 6-6 provide relatively better performance. Overall, there are unbalanced class-wise results observed as the grid size increases.

In the second step, the relatively better cases are selected, and the classification is once again carried out, this time utilizing the convex Bernstein density as the discriminant function. The weighted averages of the two Bernstein copulas are integrated into the machine learning algorithm, as defined in the equation (2.7). The copula is also can be referred as two component convex Bernstein. In Table 3.2, only the Bernstein with a 4-6 grid order is displayed as it yields the best results among the other options. It is evident that the convex Bernstein manages to address the class-wise imbalance issue. The weights ranged among the permutations of $(0.2, 1-0.2)$ incremented by 0.05 and the pairs which have the best classification result are displayed. The optimal results are achieved by assigning closer weights. In

the table, the upper weight represents the weight of Bernstein estimated with the greater grid size. For instance, if the upper weight is 0.4 for the grid pair 4-6, the weight of c_4 is 0.6, and the weight of c_6 is 0.4. In the final results, global accuracy increases along with balanced class-wise accuracies when weighting by 0.49, which provides relatively optimal accuracy.

In Table 3.3, the comparison between mixed and global optimization is conducted with equal weights, and leave-one-out cross-validation is included in addition to the 10-fold cross-validation with 20 repetitions (10F20R). For the 10F20R approach, the mean accuracy slightly increases when global accuracy is considered, even with 22 missing folds.On the other hand, the mean accuracy shows a slight improvement when 5 missing folds are addressed with mixed optimization. Additionally, there is a relatively larger change in mean class 2 accuracy following the same pattern as the mean accuracy. As the training sample size decreases (for 10F20R, training samples consist of data with sizes of 104 or 105), the number of unsolved cases increases, and the neglected folds can have a negative impact on accuracy, as seen in the case of 10F20R. In the case of leave-one-out, where there are only 5 missing folds, including them by using the initial solution even leads to a slight increase in accuracy. Therefore, especially when there are many missing folds, optimizing with the initial solution may result in a decrease in accuracy. In Table 3.4, In this case, we select the fold with the best-balanced performance, achieving an accuracy of 0.83, a class 1 accuracy (specificity) of 0.8, and a class 2 accuracy (sensitivity) of 0.8571. According to the confusion matrix for this specific fold, our algorithm makes only one error case in both false positive and false negative classifications.

Lastly, several widely known supervised learning algorithms are applied on the breast cancer data along with the two component convex Bernstein. Results for the analysis given in Table 3.5 and the included classifiers are; linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic linear regression (LLR), logistic linear regression involving all of the interactions (LLR inter.) and convex Bernstein (CB). The classification is conducted using the same four-dimensional feature set, as performance deteriorated when more features were

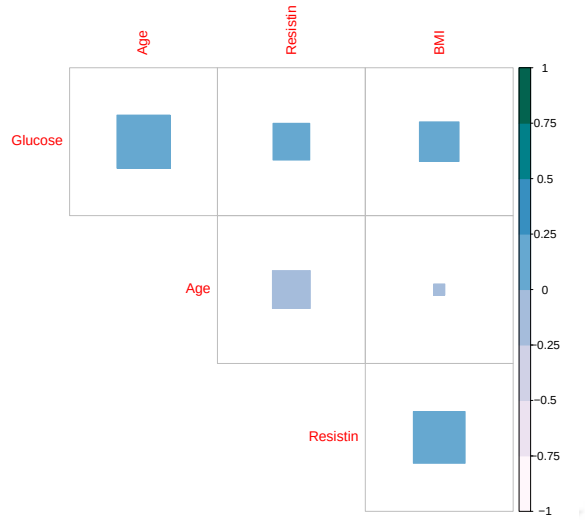

Figure 3.2 Kendall's tau values for features for Coimbra Breast cancer data.

added. Only leave-one-out cross-validation is performed. Additionally, for CB (Convex Bernstein), only mixed optimization is utilized, and a more detailed parameter search is conducted to explore whether improved classification results can be achieved. The implementation for CB is chosen with a grid configuration of $4, 6 - 4, 6$ and weight values of $0.4, 0.6 - 0.55, 0.45$. As a result, there is a slight increase of 2% in Class 1 accuracy but a slight decrease of 1% in Class 2 accuracy. Based on the results, classifiers, except for LLR inter. (Logistic Regression with Interaction Terms), exhibited lower overall accuracy compared to CB. While CB is slightly better than LLR for the given feature set, the LLR inter. model performed better when 6 additional features were added to the 4 biomarkers, resulting in superior performance in the 10-dimensional classification task. However, it is important to note that 10-dimensional classification could not be performed with CB due to computational difficulties, so a direct comparison in that context is not available. Also, it is naturally expected to obtain a better performance by including feature variations in the training data.

Table 3.1 Four-variate Bernstein classifier results for the chosen grid sizes.
Class 1: Control, Class 2: Breast Cancer.
Number of no solutions shows number of non-convergent optimizations 200 models.

Accuracy	Class 1 Accuracy	Class 2 Accuracy	Grid Size 1	Grid Size 2	No Sol. Cases
0.5942	0.7322	0.4767	3	3	0
0.5728	0.858	0.3404	4	3	0
0.5079	0.9516	0.1455	5	3	0
0.5907	0.9774	0.2806	6	3	20
0.5872	0.414	0.7335	3	4	0
0.6021	0.6912	0.5305	4	4	0
0.5694	0.7449	0.4217	5	4	0
0.6725	0.8916	0.5009	6	4	20
0.5279	0.0815	0.8913	3	5	1
0.5774	0.356	0.7574	4	5	1
0.5862	0.5928	0.5881	5	5	1
0.682	0.7818	0.6109	6	5	21
0.5177	0.0539	0.8962	3	6	9
0.5653	0.2181	0.844	4	6	9
0.5663	0.403	0.7	5	6	9
0.6813	0.6796	0.687	6	6	29

Table 3.2 Results for the 4-dimensional convex Bernstein models (c_{CB}) adopted with grid sizes of $(4, 6-4, 6)$.
Class 1: Control, Class 2: Breast Cancer.
Number of no solutions shows number of non-convergent optimizations (200 models).

Accuracy	Class 1 Accuracy	Class 2 Accuracy	Model 1 Weight	Model 2 Weight	No Soln. Cases
0.7266	0.7492	0.7116	0.48, 0.52	0.48, 0.52	24
0.7275	0.7503	0.7121	0.49, 0.51	0.49, 0.51	24
0.727	0.7495	0.7123	0.50, 0.50	0.50, 0.50	24
0.7265	0.746	0.7142	0.51, 0.49	0.51, 0.49	24
0.726	0.7447	0.714	0.52, 0.48	0.52, 0.48	24
0.726	0.7445	0.7135	0.53, 0.47	0.53, 0.47	24

Table 3.3 Convex Bernstein classifier (c_{CB}) results for the 4-dimensional models where trained models adopted with both mixed optimization and global optimization with missing folds. For both classification models, Bernstein components have aggregated grid sizes of $(4, 6 - 4, 6)$, weighted equally i.e., $(0.5, 0.5 - 0.5, 0.5)$.

	Accuracy	Class 1 Accuracy	Class 2 Accuracy
10F20R	0.7267	0.7263	0.7229
Leave-one-out	0.7500	0.7885	0.7188
Optimization: Mixed			
10F20R - 22 Missing Case	0.7205	0.7423	0.7013
Leave-one-out - 5 Missing Case	0.7477	0.7885	0.7119
Optimization: Global			

Table 3.4 Confusion matrix of convex Bernstein applied fold that shows real and predicted counts of subjects.

		Predicted Class	
Actual Class		Class 1	Class 2
	Class 1	4	1
	Class 2	1	6

Table 3.5 Comparison between well-known machine learning algorithms. Convex Bernstein trained according to a detailed parameter search is included.

Classifier	Accuracy	Class 1 Accuracy	Class 2 Accuracy
LDA	0.6983	0.6923	0.7031
QDA	0.7241	0.8077	0.6562
LLR	0.7414	0.7500	0.7344
LLR inter.	0.8276	0.8654	0.7969
CB	0.7500	0.8077	0.7031

3.2 Discussion

Probability distributions serve as powerful tools in machine learning, offering great potential for improvement.Embracing a distributional approach when learning from training datasets can result in a precise data fit. Nonetheless, accurately pinpointing the underlying distribution structure, especially in real-life scenarios, remains a significant challenge. Consequently, nonparametric approaches offer promising opportunities for enhancement. In this study, a Bernstein copula implemented as a classifier is used in cross validation procedure. The algorithm for the classification is designed to search the optimal grid size that is capable of recognizing the test data optimally. In the simulation study, the classes are defined based on two well-known copulas: Gumbel and Clayton, each characterized by distinct tail dependencies. The Gumbel copula represents a right-tail dependence structure, while the Clayton copula represents a left-tail dependence structure. The Bernstein copula approach has exhibited superior performance compared to the benchmark classification method, Gaussian Naive Bayes. Moreover, the Bernstein copula approach demonstrates a significant resemblance to the Gumbel-Clayton classifier, especially in cases characterized by weak dependence. Particularly, when dealing with structures that exhibit mixed correlations and tail dependencies, Gaussian Naive Bayes (GNB) fails

to provide satisfactory accuracies.The figures from the best-case machine learning simulations demonstrate that GNB is ineffective at capturing the tails of the data and struggles with high levels of dependence. In a real data application, the Coimbra breast cancer data is analyzed, and classification is performed using a four-variate Bernstein copula. This approach results in relatively higher accuracies, although there are unbalanced class-wise accuracies. To address the poor performance, the convex Bernstein copula is employed with grid pairs that exhibit higher accuracies. This approach effectively resolves the issue of class-wise imbalance. A weight search for the discriminant functions leads to satisfactory results in comparison to the previous analysis. Therefore, the integration of distributional information proves to be a valuable tool, and there is potential for further enhancements in this approach.

Optimizing the contingency table before evaluating the Bernstein density is perhaps the most prominent limitation. Transforming discrete variables to be equal to m^d can pose challenges in this context. Therefore, it is advisable to use the Bernstein copula for classification problems involving small feature spaces that have been validated to explain the target variable. Based on the findings, as the number of unknowns in the optimization algorithm approaches approximately 1000, the optimization process becomes slower, and non-convergent results become more apparent. Empirical observations suggest that using initial solutions in optimization can lead to a slight decrease in accuracies, especially when there are many unsolvable folds. However, in general, for efficiency, moderate dimensions are needed along with the initial solution framework.

3.3 Vine Density Classifier

3.3.1 Simulation Study

In this section, we examine the efficacy of vine density machine learning frameworks, comparing their performance to that of random forest and support vector machines. The simulation is conducted using two classes of Gaussian distributions

characterized by distinct dependence structures.

In simulation design, 500 observations are generated with 10 trials and the average accuracy results are displayed. The first class is generated with a Gaussian copula using $\tau = 0.7$, while the second class is generated with a Gaussian copula using $\tau = 0.2$. This is achieved by leveraging the vine copula simulation module of the VineCopula package in R. The dataset consists of 5-dimensional data with 10 nodes, where 4 nodes have Gaussian copula, and 6 nodes have independence copula (i.e., noise). The classification is based on a 10% test set and a 10% validation set, which are used for trained model optimization. For evaluation process, we use accuracy measure as a common performance measure in machine learning to assess the classification model's effectiveness. Accuracy is the ratio of correct predictions obtained by the model to the total number of predictions. The accuracy is calculated as:

Accuracy = (Number of correct predictions) / (Total number of predictions).

$C1$ and $C2$ are considered as Class 1 and Class 2 , respectively. Classes are simulations of 5 dimensional vine density with 15 nodes where 10 of them are copula distribution.Class 1 indicates the data generation process for vine that consists of empirical marginals in the first tree, only Gaussian (0.7) copula calibration in the second tree (4 nodes) and the independence copula for the rest of the trees (noise on 6 nodes). Class 2 has the same data generation process only difference is that the second tree is built by using Gaussian (0.2) copula.

In Table 3.6 overall results are displayed. The full-parametric vine has the highest accuracy, mean C2 accuracy, F1 score and AUC score while truncated-parametric vine is relatively close in terms of all metrics excluding mean C2 accurracy. In terms of accuracy, random forest and support vector machines fall behind the full-parametric and truncated-parametric vine frameworks. The parametric frameworks along with the truncated hybrid vine exhibit relatively more balanced class-wise accuracies. Notably, the hybrid methods achieve higher class 1-based accuracy compared to other vine frameworks. The performance of Bernstein vine is

relatively low for the simulated data. In summary, it can be inferred that achieving high accuracy performances may not necessitate calibrating a full vine, as the performance of the truncated-parametric model closely approximates that of the full-parametric model.

In Table 3.7 and Table 3.8, the trim numbers are higher for the first class, as the Gaussian distribution with a correlation parameter of 0.7 leads to more correlated tree instances. Calibrating these trees provides valuable information for the training data. Conversely, for second class trainings, either the first tree calibration is applied, or no vine calibrations are used at all, as zero values indicate that the discriminant function is a product of marginal distributions, corresponding to the naive Bayes method.

In Table 3.10 and Table 3.9, the simulation-wise performances, along with the hyperparameters, are displayed for Support Vector Machines (SVM) and Random Forest (RF), respectively. In the simulation scenario, the training model capacity is optimized by selecting parameters that yield the highest accuracy from the validation set.

Table 3.6 Class 1: 4 Gaussian(0.7) - 6 Noise; Class 2: 4 Gaussian(0.2) - 6 Noise; 500 observations; 10 simulations

Method	Mean Accuracy	Mean C1 Acc.	Mean C2 Acc.	F1 Score	AUC Score
full-parametric	0.877	0.8826	0.87224	0.88387	0.93574
full-bernstein	0.685	0.84129	0.55186	0.65205	0.74283
full-hybrid	0.775	0.93695	0.63703	0.74991	0.82907
truncated-parametric	0.863	0.88695	0.8426	0.86816	0.9281
truncated-bernstein	0.699	0.71305	0.68705	0.70636	0.77141
truncated-hybrid	0.852	0.90652	0.80555	0.85285	0.91984
svm	0.855	0.93478	0.78703	0.85296	0.9333
random forest	0.835	0.86087	0.81297	0.8407	0.90936

3.3.2 Gene Expression Data Classification Application

For the machine learning model evaluation, Monte Carlo cross validation method is used such that dataset is sampled 20 times for the train-test split. Classification with truncated vine also has validation set in the same manner. Performance metrics are obtained by aggregating 20 classification tasks for each data. Note that, for some splits

Table 3.7 Truncated parametric vine model simulations

Accuracy	C1 Acc.	C2 Acc.	F1 Score	AUC Score	Trim Number C1	Trim Number C2
0.82	0.8913	0.7593	0.8200	0.9066	3	1
0.85	0.9130	0.7963	0.8515	0.9046	2	1
0.86	0.8913	0.8333	0.8654	0.9114	1	0
0.81	0.8696	0.7593	0.8119	0.8917	4	1
0.90	0.8913	0.9074	0.9074	0.9444	2	1
0.87	0.9348	0.8148	0.8713	0.9505	1	0
0.90	0.8261	0.9630	0.9123	0.9565	1	1
0.84	0.8043	0.8704	0.8546	0.9179	2	1
0.91	0.9130	0.9074	0.9159	0.9638	1	0
0.87	0.9348	0.8148	0.8713	0.9336	1	1

Table 3.8 Truncated hybrid vine model simulations

Accuracy	C1 Acc.	C2 Acc.	F1 Score	AUC Score	Trim Number C1	Trim Number C2
0.81	0.8913	0.7407	0.8081	0.8969	2	0
0.82	0.9348	0.7222	0.8125	0.9026	2	0
0.86	0.8913	0.8333	0.8654	0.9114	1	0
0.77	0.8913	0.6667	0.7579	0.8482	2	1
0.88	0.8913	0.8704	0.8868	0.9400	1	0
0.87	0.9348	0.8148	0.8713	0.9505	1	0
0.89	0.8696	0.9074	0.8991	0.9481	4	0
0.85	0.8913	0.8148	0.8544	0.9114	4	0
0.91	0.9130	0.9074	0.9159	0.9638	1	0
0.86	0.9565	0.7778	0.8571	0.9255	2	0

Table 3.9 SVM model with polynomial kernel of third degree simulations

Accuracy	C1 Acc.	C2 Acc.	F1 Score	AUC Score	svm_coef	svm_gamma
0.81	0.913	0.7222	0.8041	0.8965	1	1
0.84	0.8913	0.7963	0.8431	0.9054	2	1
0.88	0.9348	0.8333	0.8823	0.9416	2	0.1
0.82	0.9348	0.7222	0.8125	0.9106	1	1
0.89	0.9565	0.8333	0.891	0.9461	3	0.1
0.89	0.9565	0.8333	0.891	0.9654	1	0.2
0.86	0.9348	0.7963	0.86	0.9513	3	0.1
0.77	0.8913	0.6667	0.7579	0.9042	1	0.1
0.94	0.9783	0.9074	0.9423	0.9791	1	0.2
0.85	0.9565	0.7593	0.8454	0.9328	2	0.1

vine structures are failed to be optimized, those cases are omitted.

Table 3.10 Random Forest model simulations

Accuracy	C1 Acc.	C2 Acc.	F1 Score	AUC Score	rf_nest	rf_mdep	rf_ms_split	rf_ms_leaf
0.81	0.8696	0.7593	0.8119	0.8907	50	10	1	1
0.84	0.9348	0.7593	0.8368	0.877	50	5	1	1
0.86	0.8261	0.8889	0.8727	0.9235	50	5	1	1
0.83	0.8913	0.7778	0.8317	0.9052	150	5	1	2
0.89	0.913	0.8704	0.8953	0.9422	50	5	1	2
0.85	0.8913	0.8148	0.8544	0.9308	50	5	5	2
0.81	0.7609	0.8519	0.8288	0.9261	100	5	5	2
0.76	0.8043	0.7222	0.7647	0.8529	50	5	5	1
0.9	0.8478	0.9444	0.9107	0.9541	100	10	5	2
0.8	0.8696	0.7407	0.8	0.8911	150	10	5	2

3.3.2.1 GSE2109 Benchmark Data

The gene expression data series referenced by GSE2109 in NCBI repository are used as a benchmark data for vine classification. The data originally created as expression project for oncology (expO). The project expO seeks to manage cancer patients clinically. The expression data holds a strong profile of malignant tumor samples to establish distinguishing analysis among tissues. Data is reorganized and standardized for various cancers in the OSF data portal. The classification analysis is made for three different expression data namely; GSE2109 Kidney, GSE2109 Lung, GSE2109 Uterus. Each subject has the prior class information on cancer whether he/she is in early stage or late stage. Note that in the original curation, the first class is labeled as the combination of 1 and 2 stages and the second class is labeled as the combination of 3 and 4 stages. We address them early and late stages, respectively. The curation techniques and detailed information are given in Golightly et al (2018). The study aims to help researchers to make use of benchmark comparisons on machine learning algorithms. Various machine learning methods were applied on the data in the study Piccolo et al. (2021).

3.3.2.2 Classification of GSE2109 Benchmark Data

For each data genes are scored by the mutual information criteria and top five expressed genes are selected. Mutual information is used for feature selection that measures how much information one variable provides about other. Using this

method, genes are ranked according to the degree to which their expression is linked to the classification goal. The top five genes are chosen based on their scores. The most useful characteristics for classification are these genes. An overview for the top gene annotations are given based on the analysis reports in Human Protein Atlas Portal. For the kidney tumor data, the gene with ID ENSG00000159899 came first. The gene encodes natriuretic peptide receptor B (NPR2), a membrane receptor for natriuretic peptides, primarily C-type natriuretic peptide (CNP), which activates guanylyl cyclase upon ligand binding. Based on the findings, the expression levels are altered in the case of kidney cancer and the expression levels show a significant association with patient survival, where high expression refers to a relatively higher survival chance. (Given in Table 3.11)

For the lung tumor data, the gene ENSG00000011426 is on top. It encodes an actin-binding protein that is involved in cell growth and migration, and cytokinesis. The gene has a significant association with patient survival, where lower expression refers to a lower survival rate. Those two mentioned genes have relatively low expression activity for overall cancer types and have no specific enrichment observation for any cancer type. (Given in Table 3.12)

Lastly the gene ENSG00000124939 which involves in androgen receptor signaling pathway and located in extracellular space has the most score in uterus data. The gene shows no significant association with survival rate. However, this gene is more actively transcribed in the RNA for the cancer types namely; Ovary Serous Cystadenocarcinoma, Uterine Corpus Endometrial Carcinoma. In another words the gene is group enriched for the mentioned types where uterus is also involved. According to the RNA expression overview, unlike the ENSG00000159899 and ENSG00000011426, the expression activity is almost nonexistent for the other types of cancer and is high for the given cancer types; hence, the term group enriched is used.(Given in Table 3.13) The examined expression activity for the chosen uterus data genes here also in parallel with its the classification results.

Classification based on R-vine is carried with test percentages of 10%, 15% and 20%. For truncated vines, different percentages are used for pairs of test and validation. In Table 3.14, there is a slight increase in truncated vine especially when models are built with parametric vine. The accuracy results are in favor of the later stage which is lesser class. The results in kidney data may suffer the imbalanced situtation of the classes. The most balanced results are adopted for the second row of non-parametric truncated vine(0.714; 0.704; 0.738). In Table 3.15, there is also improvement in truncated vine and the last row of parametric truncated vine (0.718; 0.688; 0.764) has the most balanced performance and there exists a promising increase in later stage accuracy. Lastly, the results for the uterus dataset is in Table 3.16. The cancer stage distinguishment can be achieved with relatively high performance for this data. With truncation, better results are obtained. Accuracy-wise, the first row of parametric truncated vine (0.875; 0.936; 0.758) has the highest performance. However, the first row of the non-parametric truncated vine (0.87; 0.901; 0.801) has the most balanced result. In this case, the mean $C2$ accuracies are also kept relatively higher.

Lastly, hybrid vine models which have semi-parametric vine setup are used to investigate existence of a boosting effect. Overall, the results are slightly differ from the earlier ones. For this reason, only the results for the uterus dataset is given in Table 3.17. Results are particularly closer to the parametric ones for all of the benchmark datasets. It can be deduced that, semi-parametric algorithm for vine tends to choose parametric copulas more. According to the results, there is no proof that hybrid distribution setup does not have a boosting effect that goes beyond already calculated parametric and non-parametric results. Accuracies of different vine setups are displayed in the Figure 3.3, Figure 3.4, Figure 3.5 for all of the datasets.

In Figure 3.6, the cut points of the truncated vine models are displayed for each fold of the uterus data. According to the findings one can see that, there is no need to fit a vine distribution with full tree and nodes calibration to achieve a higher classification performance. In fact, since zeros are the most appearent, just marginals will be enough for a good performance in uterus data. However, the most balanced result achieved

in the non-parametric case uses various cut points. For example, the model 1 uses cut point 1, 7 seven times and cut point 2, 1 time which refers the model needed at least one level tree generation in many cases. Overall, the models needed to fit vine fully in just one instance which is the model 2 of non-parametric modeling.

Table 3.11 Top 5 selected genes for Kidney data

Gene Annotation	MI Score
ENSG00000159899	0.1310481
ENSG00000198719	0.1260605
ENSG00000174640	0.1258081
ENSG00000187840	0.1242762
ENSG00000100234	0.1224906

Table 3.12 Top 5 selected genes for Lung data

Gene Annotation	MI Score
ENSG00000011426	0.2340990
ENSG00000111665	0.2177974
ENSG00000117650	0.2156703
ENSG00000088325	0.2153979
ENSG00000123485	0.2062613

Table 3.13 Top 5 selected genes for Uterus data

Gene Annotation	MI Score
ENSG00000124939	0.2882380
ENSG00000109794	0.2806210
ENSG00000082175	0.2795868
ENSG00000156049	0.2674682
ENSG00000129003	0.2657077

3.3.3 Discussion

Vine copulas model the distribution of high-dimensional random vectors by capturing the relationships between pairs of conditional random variables. They offer the flexibility to represent various dependence structures between pairs of random variables. Moreover, the vine model fulfills generalization of copula in arbitrary dimensions. This property makes vine copula highly flexible models for computationally tractable estimation and model selection. We use vine densities as

Table 3.14 Vine classifier performances on kidney tumor data. There are 132 early stage and 63 late stage cancer subjects.

R-vine classifier	Test Percent	Validation Percent	Mean Accuracy	Mean C1 Accuracy	Mean C2 Accuracy	F1 Score	AUC Score
Parametric	0.1	-	0.6775	0.637015	0.74474	0.53479	0.7261
	0.15	-	0.66725	0.629235	0.75237	0.596665	0.752615
	0.2	-	0.650005	0.617205	0.71968	0.575635	0.73271
Non-Parametric	0.1	-	0.605	0.48722	0.868905	0.569975	0.74199
	0.15	-	0.62931	0.53198	0.843035	0.60246	0.73752
	0.2	-	0.61922	0.52997	0.80126	0.59774	0.74951
Truncated R-vine classifier	**Test Percent**	**Validation Percent**	**Mean Accuracy**	**Mean C1 Accuracy**	**Mean C2 Accuracy**	**F1 Score**	**AUC Score**
Parametric	0.1	0.1	0.69	0.633	0.794	0.53	0.741
	0.2	0.1	0.696	0.671	0.739	0.581	0.772
	0.1	0.2	0.7	0.66	0.791	0.55	0.727
	0.2	0.2	0.691	0.679	0.717	0.62	0.77
Non-Parametric	0.1	0.1	0.703	0.633	0.844	0.461	0.687
	0.2	0.1	0.714	0.704	0.738	0.543	0.723
	0.1	0.2	0.695	0.641	0.809	0.487	0.655
	0.2	0.2	0.7	0.684	0.738	0.538	0.7

Table 3.15 Vine classifier performances on lung tumor data. There are 53 early stage and 43 late stage cancer subjects.

R-vine classifier	Test Percent	Validation Percent	Mean Accuracy	Mean C1 Accuracy	Mean C2 Accuracy	F1 Score	AUC Score
Parametric	0.1	-	0.655	0.634	0.710	0.541	0.725
	0.15	-	0.643	0.641	0.664	0.555	0.708
	0.2	-	0.653	0.702	0.608	0.556	0.699
Non-Parametric	0.1	-	0.575	0.648	0.528	0.633	0.746
	0.15	-	0.618	0.695	0.554	0.651	0.765
	0.2	-	0.634	0.716	0.563	0.658	0.788
Truncated R-vine classifier	**Test Percent**	**Validation Percent**	**Mean Accuracy**	**Mean C1 Accuracy**	**Mean C2 Accuracy**	**F1 Score**	**AUC Score**
Parametric	0.1	0.1	0.655	0.641	0.686	-	0.674
	0.2	0.1	0.708	0.698	0.715	0.65	0.734
	0.1	0.2	0.675	0.683	0.685	0.566	0.708
	0.2	0.2	0.718	0.688	0.764	0.617	0.722
Non-Parametric	0.1	0.1	0.645	0.627	0.674	0.654	0.715
	0.2	0.1	0.708	0.688	0.738	0.679	0.740
	0.1	0.2	0.625	0.606	0.662	0.645	0.721
	0.2	0.2	0.697	0.684	0.72	0.657	0.705

discrimination functions. Through the machine learning process in simulation study, the densities represent the training data for each class label. Subsequently, class membership of the test data is determined based on the vine model with the highest density value. Truncated vine model approaches are proposed for the classification procedure based on the vine Bayes context. Some hybrid vine structures are also constructed by semi-parametric vine modeling which makes the selection between the parametric copulas and Bernstein copulas with different grid sizes based on the relative distance given in equation 2.20. Based on our observations, it always tends to select the higher grid size to apply more smoothing effect on the distribution. Hence, we fix the Bernstein copula grid size to 10 as the degree of polynomial should be enough to capture the dependence pattern. If not, it will be practical to engage a

Table 3.16 Vine classifier performances on uterus tumor data. There are 64 early stage and 41 late stage cancer subjects.

R-vine classifier	**Test Percent**	**Validation Percent**	**Mean Accuracy**	**Mean C1 Accuracy**	**Mean C2 Accuracy**	**F1 Score**	**AUC Score**
Parametric	0.1	-	0.825	0.86125	0.78001	0.704	0.886
	0.15	-	0.790625	0.854605	0.70137	0.7	0.859
	0.2	-	0.804755	0.867015	0.70621	0.706	0.864
Non-Parametric	0.1	-	0.795	0.76028	0.87334	0.787	0.889
	0.15	-	0.7875	0.75752	0.86119	0.744	0.876
	0.2	-	0.77142	0.736775	0.850315	0.73	0.888
Truncated R-vine classifier	**Test Percent**	**Validation Percent**	**Mean Accuracy**	Mean C1 **Accuracy**	**Mean C2 Accuracy**	**F1 Score**	**AUC Score**
Parametric	0.1	0.1	0.875	0.936	0.758	0.76	0.892
	0.2	0.1	0.84	0.906	0.719	0.716	0.889
	0.1	0.2	0.845	0.878	0.774	0.753	0.899
	0.2	0.2	0.84	0.903	0.733	0.706	0.873
Non-Parametric	0.2	0.1	0.843	0.896	0.754	0.716	0.888
	0.1	0.2	0.86	0.885	0.797	0.749	0.901
	0.2	0.2	0.843	0.896	0.756	0.715	0.872

Table 3.17 Truncated R-vine classifier performance with hybrid setup on uterus tumor data.

	Test Percent	Validation Percent	Mean Accuracy	Mean C1 Accuracy	Mean C2 Accuracy	F1 Score	AUC Score
Semi-Parametric	0.1	0.1	0.875	0.927	0.778	0.758	0.9
	0.2	0.1	0.84	0.903	0.725	0.723	0.891
	0.1	0.2	0.85	0.895	0.762	0.757	0.875
	0.2	0.2	0.838	0.908	0.708	0.724	0.874

copula family instead since bivariate parametric copulas can be more sensitive to capture solid dependence structures. Also high grid sizes for Bernstein copula does not guarantee better performance and reduce the computational performance as can be seen in the study Yamut & Hudaverdi (2023). The performance of the proposed procedures is compared on the class membership of the current data. Some well-known machine learning methods, such as Support Vector Machines (SVM) and Random Forest, are employed to compare the performance of the proposed method. It can be concluded that high accuracy performances can be achieved without fully calibrating the vine copula structure. The performance of the truncated-parametric copula closely approximates that of the full-parametric model. Moreover, when compared with other techniques, the truncated-parametric vine model demonstrates superior performance.

Furthermore, for the proposed vine type classification, dependence structures among gene features are constructed using the pair copulas with Bernstein vine, hybrid vine, and parametric vine modeling. Monte Carlo cross-validation is utilized for the gene expression dataset and it is sampled 20 times for the train-test split. Performance metrics are obtained by aggregating results from 20 classification tasks

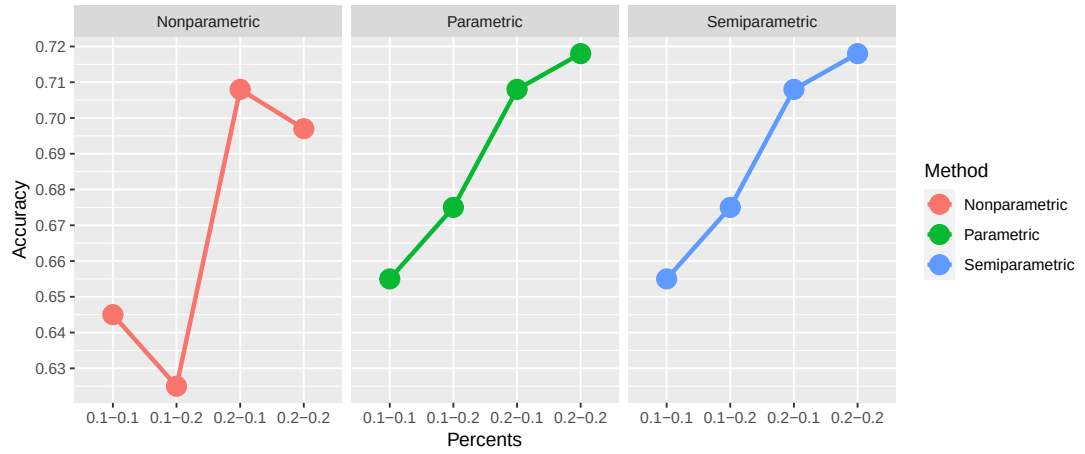

Figure 3.3 Accuracy versus test-validation percent line graphs for lung data with respect to the different truncated vine approaches.

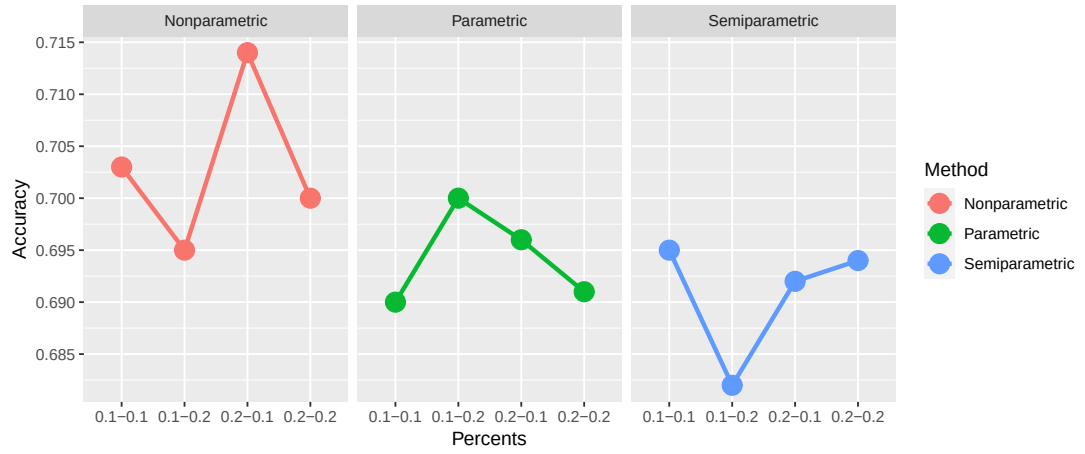

Figure 3.4 Accuracy versus test-validation percent line graphs for kidney data with respect to the different truncated vine approaches.

for each dataset. This approach represents a selection copula process in a semiparametric environment. In vine application, we observe as the tree levels progressed and conditional observations accumulate, the dependence measure Kendall's tau tends to getting weaker. This suggests, co-movements of prominent variables getting weaker as they are conditioned with more and more variables.

In general, for the gene data, there is a slight increase in accuracy observed in truncated vine models when models are constructed using both parametric and nonparametric vines. However, semiparametric designs do not exhibit the desired boosting effect, as they closely resemble the results obtained from parametric models.

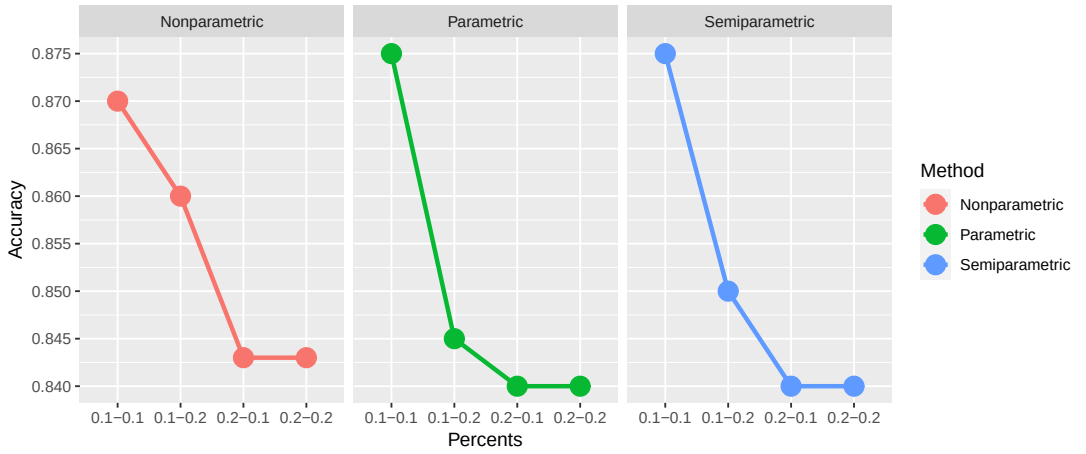

Figure 3.5 Accuracy versus test-validation percent line graphs for uterus data with respect to the different truncated vine approaches.

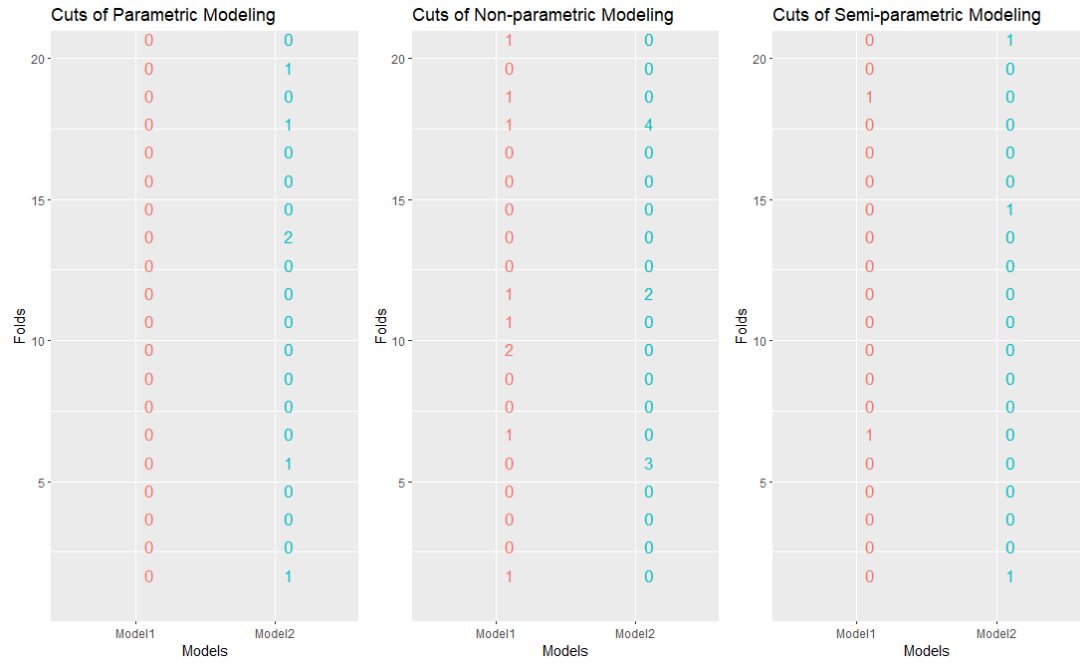

Figure 3.6 Cut points of truncated vines for each model and fold.

Conversely, the nonparametric vine with truncation design yields the most impactful result, demonstrating the most balanced classification performance in the Uterus gene expression dataset. The F1 scores are relatively low in the results. Notably, the highest F1 score is achieved in the full non-parametric case and the highest AUC score is achieved in the truncated non-parametric case of the uterus data classification. Based on the cut-off results on the uterus data cross validation, the parametric approach relatively benefit more from the truncated vine as it tends to cut

vine off earlier levels than other methods. From this observation it can be deduced that the bivariate parametric families captured most of the co-movement information in earlier tree levels as they are flexible to detect existing dependence patterns. Also, we can interpret the classification performances and cut point figures as

- Full calibration of the vine is unnecessary to achieve satisfactory classification results.
- In certain datasets, relying solely on the naive approach, where the product of marginals is used, may be insufficient, particularly when utilizing truncated vine models. In such cases, various cross-validation strategies involve constructing vine trees to enhance performance and achieve superior results.

Owing to the decomposable structure of pair copulas, researchers can analyze gene-by-gene dependencies to better understand networks. Because pair-copula constructions (PCCs) effectively depict the joint distribution of features, they can enhance the effectiveness of predictive models.

3.4 Neural Density Based Classifier

In this section, the predictive power of neural network based distribution is examined in machine learning classification task. Benchmark gene expression survival data is used and the survival outcome profiles are classified for the task. In the marginal distribution part, 5 layers are used along with the 5 hidden layer neurons. In the copula part number of 5 hidden layers are used along with the 10 hidden layer neurons. The networks are compiled with Nadam optimizer with learning rate of 0.01. The general loss function is set as mean absolute error.

3.4.1 GSE62564 Benchmark Survival Gene Expression Data

The GSE62564 dataset was curated to evaluate the predictive potential of RNA sequencing compared to microarray data, with a focus on overall survival. Survival outcomes were modeled as a binary classification problem, where a gene profile labeled as 0 indicates subjects with favorable survival outcomes, and 1 represents unfavorable outcomes. Cox regression was employed to model survival times and assess the relationship between gene expression profiles and survival. Based on this modeling approach, the curators demonstrated that subjects in the favorable survival group had significantly longer survival periods compared to those in the unfavorable group. GSE62564 benchmark data consist of 499 subjects and 43827 gene profiles. The class labeled as favorable survival outcome has 393 subjects and the one labeled as unfavorable survival outcome has 105 subjects.

3.4.2 Data Preprocessing

The RNA-sequence data is originally curated as RPM type data with log2 transformation. The data is extracted through GEO dataset module of Python. By using the meta data the gene profiles are properly labeled with correspondent survival outcomes. Gene selection conducted with the mutual information criteria and top genes within the top five is gathered as the classification data. Then the data is standardized by using the min-max scaling. In Figure ,one can deduce the correlation between the chosen genes are relatively high Kendall's tau. Hence for proper classification modeling the dependence structure of genes are crucial. Due to the imbalanced class random undersampling procedure is used in the the train/test split. The size of unfavorable survival outcome is referenced as it is the class with lesser subjects. 105 subjects are randomly chosen for both classes. %20 of the equalized dataset is determined as the test data and the class distribution of the train and test is preserved by using stratification. In the end, training data with 168 subjects obtained where each class labeled data has 84 subjects along with the test data with 42 subjects.

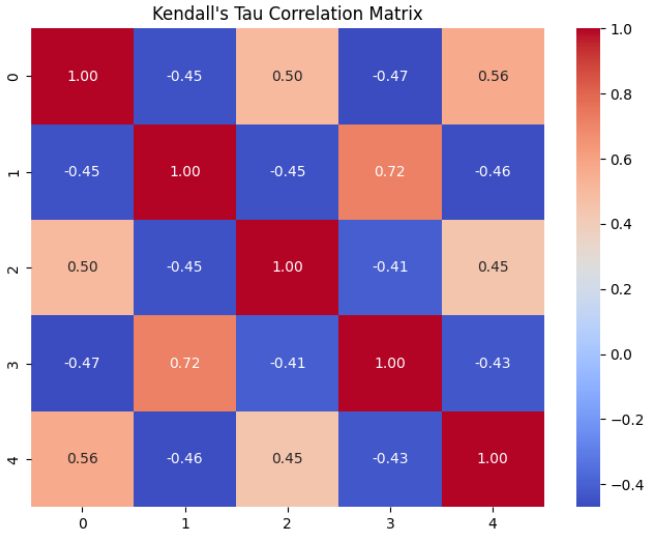

Figure 3.7 Kendall's tau correlation matrix of top five survival genes

3.4.3 Classification of Gene Expression Based Survival Data

3.4.3.1 Neural Copula Classification

Since the management of neural network which conditioned to build a copula is difficult in arbitrary dimensions the classification of the survival data is made by two genes. Hence, the top two gene profiles are extracted for the classification task. The marginal neural networks and copula network are obtained after 2000 epochs. In Figure 3.8 and Figure 3.9 the marginal distributions of class labeled data that generated by neural networks can be displayed. In copula distributions that generated by neural network can be displayed. According to the results, the networks provided an adequate fit close to the true distributions. In Table 3.18 the classification results on the survival data can be examined. Overall results are not poor however the recall score is relatively low causing also lower F1 score. The recall score is especially important since identifying at-risk patients is crucial for intervention. Predicting at-risk patients influence clinical decision-making, like adjusting treatment plans or monitoring high-risk patients. On the other hand, the AUC score is based unfavorable survival class probabilities and indicates how well your model predicts patients who are more likely to have an unfavorable survival outcome. Hence, AUC score is not satisfying for identifying unfavorable survival characteristics of the patients.

Nonetheless, model used two genes (features) for the problem and lacks sufficient information.

3.4.3.2 Neural Marginal Based Vine Distribution

As the second step of the classification task vine density function is calibrated with top five genes. The employed procedure is the full vine calibration with parametric, Bernstein (non-parametric) and hybrid vine as in the previous section. As mentioned in the previous application the root tree of the vine has a crucial role which means calibration of the marginal distributions has a crucial effect on modeling of the vine density. The difference is that the marginal distributions are generated through the neural network procedure. Hence the CDF and PDF information of the marginal distributions are fed to the vine density function by marginal neural network computations. The classification performance of the neural marginal based vine density compared to the vine density with parametric marginals. The parametric marginal distributions are determined by Akaike criterion. The chosen marginal distributions for the genes of the favorable survival class are; Weibull, Gaussian, logistic, Weibull and Weibull. For the unfavorable survival group the chosen marginals are; Gaussian, Weibull, Weibull, logistic and logistic. Neural marginal distributions are generated with 2000 and 50000 epochs to investigate the effect of the training capacity. These models are called as neural marginal based vine (NMBV). In Figure 3.12, Figure 3.13 marginal distributions trained with 2000 epochs and in Figure 3.14, Figure 3.15 marginal distributions trained with 50000 epochs can be visualized. Overall an adequate fit can be observed for each gene in terms of CDF and PDF. In the marginals trained with 50000 epochs, vertices of PDFs are relatively more captured than the marginals trained with 2000 epochs. Nevertheless, capturing the PDFs in more detail is not necessarily sign of a good trained model as this may cause overfitting. In Table 3.19 the classification performance vine classifiers are given. Naturally, by using more genes to train the models, the overall performance is increased. The R-vine classifier with Bernstein approach has the best result among the classifiers. On the other hand, NMBV with Bernstein approach performed close to

the best results when trained with 2000 epochs. When trained with 50000 epochs the NMBV performed poor results which shows a sign of overfitting in the model. The parametric and hybrid approaches did not present a result that is particularly better compared to the Bernstein approach.

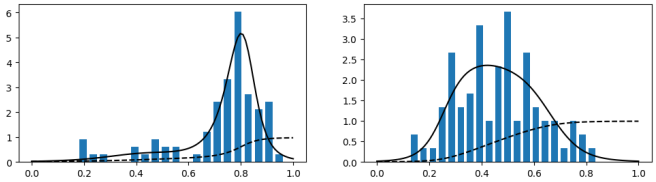

Figure 3.8 CDF and PDF plots of bivariate favorable survival data after 2000 epochs

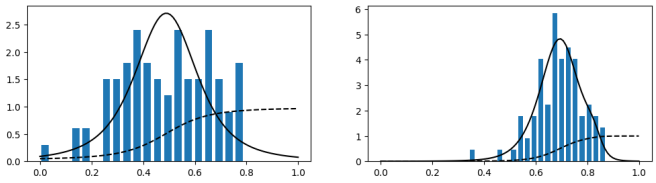

Figure 3.9 CDF and PDF plots of bivariate unfavorable survival data after 2000 epochs

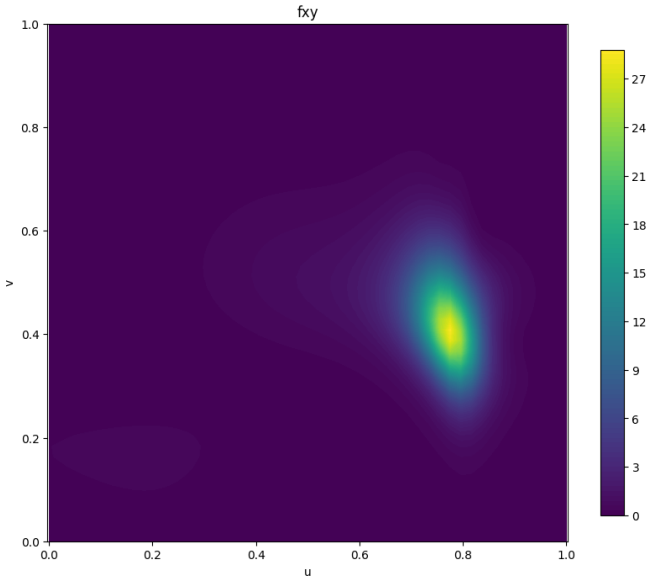

Figure 3.10 CDF and PDF plots of bivariate favorable survival data after 2000 epochs

3.4.3.3 Discussion

In this section, applications were carried out on how artificial neural networks can be used in statistical learning problems and what results can be obtained. Additionally, it

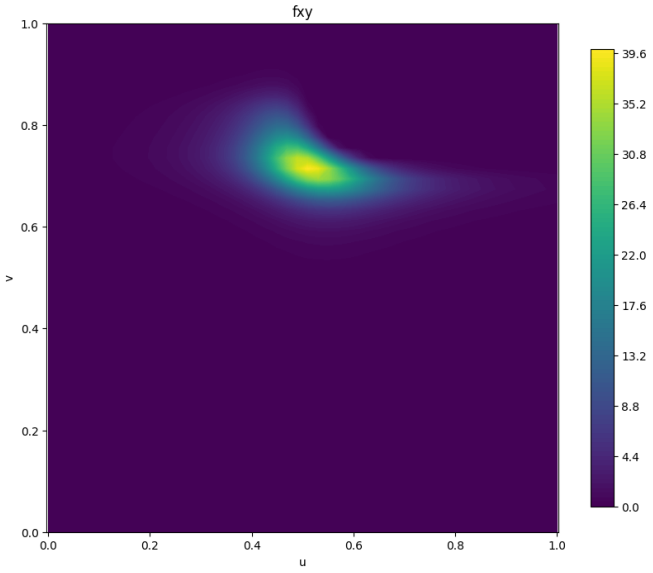

Figure 3.11 CDF and PDF plots of bivariate unfavorable survival data after 2000 epochs

Table 3.18 Classification performance of neural copula.

Accuracy	Precision	Recall	F1 Score	AUC Score
0.81	0.93	0.67	0.78	0.82

was emphasized that vine densities can incorporate their marginal distributions as trees. Such procedure is given as an alternative to generating multi dimensional distributions using artificial neural networks as it is a difficult process.

It was observed that the vine distribution constructed with parametric marginals and the Bernstein copula achieved better classification results, while the vine distribution created with neural marginals and the Bernstein copula yielded results close to it. Based on these findings, it can be said that parametric distribution families provide a better fit for this dataset. However, it was also demonstrated that the vine distribution calibrated with neural marginals could be a critical candidate for scenarios where parametric families fall short. Overall, it was shown that the dependency structure observed in the Kendall tau correlation matrix could be more accurately calibrated with the smoothing effect of the Bernstein copula.

In cases trained with a high number of epochs, overfitting was observed for this small dataset, resulting in performance even lower than two-dimensional classification

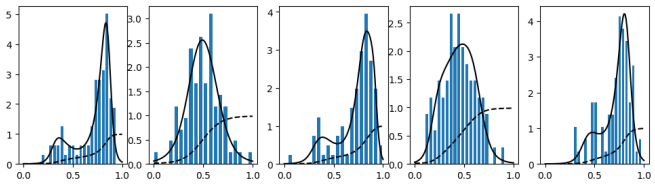

Figure 3.12 CDF and PDF plots of the five favorable survival data genes after 2000 epochs

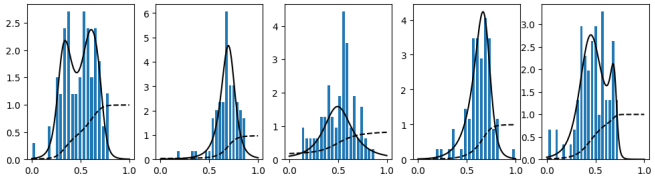

Figure 3.13 CDF and PDF plots of the five unfavorable survival data genes after 2000 epochs

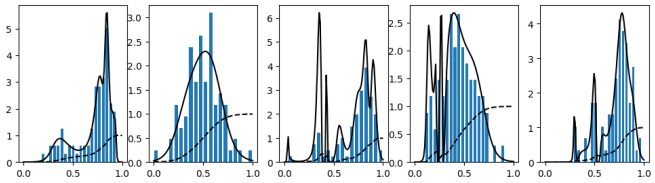

Figure 3.14 CDF and PDF plots of the five favorable survival data genes after 50000 epochs

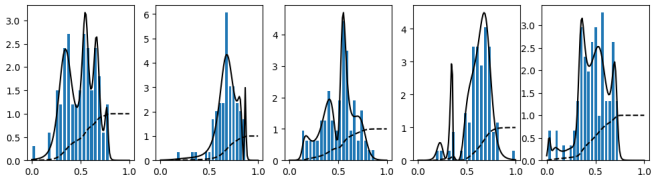

Figure 3.15 CDF and PDF plots of the five unfavorable survival data genes after 50000 epochs

for some cases.

In this application, where survival status of patients were classified, it was demonstrated that, despite the small size of the genetic dataset, good results could be obtained using the five dimensional vine approach with top five genes.

Table 3.19 Classification results on survival data by depicted vine classifiers.

R-vine classifier	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC Score**
Parametric	0.8095	0.8095	0.8095	0.8095	0.8571
Non-Parametric	0.881	0.8333	0.9521	0.8889	0.8639
Hybrid	0.8333	0.8182	0.8571	0.8372	0.8549
NMBV (2K epochs)	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC Score**
Parametric	0.7619	0.7619	0.7619	0.7619	0.8254
Non-Parametric	0.8571	0.8261	0.9048	0.8637	0.8286
Hybrid	0.7857	0.7727	0.8095	0.7907	0.8345
NMBV (50K epochs)	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**AUC Score**
Parametric	0.6667	0.6667	0.6667	0.6667	0.7415
Non-Parametric	0.7619	0.7619	0.7619	0.7619	0.7475
Hybrid	0.7143	0.6957	0.7619	0.7273	0.7551

CHAPTER FOUR
CONCLUSION

In this thesis, multivariate pattern capturing capacity of statistical learning through Bayes procedure is investigated through various classification applications. Also, frameworks on building learned models are proposed to achieve more flexible distributional classifiers. The statistical learning through Bayes method is conducted based on three main applications.

The first application is based on Bernstein copula classifiers. A binary simulation study is conducted and it is shown that Bernstein copula classifier are able to capture the patterns of actual model that is Gumbel - Clayton copula characteristics and exhibited superior performance compared to the Gaussian naive Bayes as the generated data has correlation. In classification of Coimbra breast cancer data, four-variate convex Bernstein classifier manage to bypass the imbalanced classification results of standard Bernstein copula classifier. The experimented convex Bernstein classifier also showed relatively better results compared to the conventional machine learning methods. Moreover it is also shown that, the optimization of Bernstein copula's contingency table are prominent step in the analysis. The computation scale difficulty and non-convergent results of contingency table optimization is overcame by initial solution approach with affordable decrease in the performance.

The next application consists of using vine density approach in modeling patterns of features as connected bivariate copulas. The class membership of the novel data is determined based on the highest vine density among models. Vines are calibrated fully by using well-known copula families, Bernstein copula. Also, a hybrid method is investigated where vine trees are created by both parametric copulas or Bernstein copula. In order to obtain less complex and more generalized vine classifier a truncation process is conducted by determining a cut level for vine trees. The cut point is determined as the one which gives the maximum accuracy on the validation set. In the simulation study, high accuracy can be achieved without fully calibrating

the vine copula structure. The truncated-parametric copula performs similarly to the full-parametric model and outperforms other techniques. According to the classification study on gene expression data the it has shown that vine classifiers are in general able to classify a small size gene expression data. Notably, the full parametric vine reached the highest F1 score and non-parametric truncated vine reached highest AUC score among the other vine classifiers. On the other hand, the hybrid method Based on the appropriate performance of the truncated vine models it is deduced that full parametrization of vine is not a necessity to obtain a generalized vine classifier and solely product of marginals also might not be enough for better classification. In such situations cross-validation approach on truncated vine can enhance the performance.

As last application, the power of neural networks combined with distributional classifiers and the classification performances put to test for benchmark survival data. In more detail, the neural marginals arer combined with the vine density as its first tree nodes to create a neural marginal based vine density. The vine distribution with parametric marginals and the Bernstein copula achieved the best classification results, while neural marginals performed comparably. This suggests parametric distributions fit the dataset better. However, neural marginals proved valuable when parametric families were insufficient. In general, for this dataset the Bernstein copula's smoothing effect effectively calibrated the dependency structure examined in the Kendall tau correlation matrix.

REFERENCES

Aloui, R., & Aïssa, M. S. B. (2016). Relationship between oil, stock prices and exchange rates: A vine copula based garch method. *The North American Journal of Economics and Finance*, *37*, 458–471.

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

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

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

Carrera, D., Bandeira, L., Santana, R., & Lozano, J. A. (2019). Detection of sand dunes on mars using a regular vine-based classification approach. *Knowledge-Based Systems*, *163*, 858–874.

Carrera, D., Santana, R., & Lozano, J. A. (2016). Vine copula classifiers for the mind reading problem. *Progress in Artificial Intelligence*, *5*(4), 289–305.

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

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

Gohari, K., Kazemnejad, A., Mohammadi, M., Eskandari, F., Saberi, S., Esmaieli, M., & Sheidaei, A. (2023). A bayesian latent class extension of naive bayesian classifier and its application to the classification of gastric cancer patients. *BMC Medical Research Methodology*, *23*(1), 190.

Gräler, B. (2014). Modelling skewed spatial random fields through the spatial vine copula. *Spatial Statistics*, *10*, 87–102.

Hastie, T. (2009). The elements of statistical learning: data mining, inference, and prediction.

James, G. (2013). An introduction to statistical learning.

Janssen, P., Swanepoel, J., & Veraverbeke, N. (2016). Bernstein estimation for a copula derivative with application to conditional distribution and regression functionals. *Test*, *25*, 351–374.

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

K., C., H.J., H., & L., W. (2013). Breast cancer biomarker measurements and standards. *PROTEOMICS–Clinical Applications*, *7*(1-2), 17–29.

Konstantelos, I., Sun, M., Tindemans, S. H., Issad, S., Panciatici, P., & Strbac, G. (2018). Using vine copulas to generate representative system states for machine learning. *IEEE Transactions on Power Systems*, *34*(1), 225–235.

Kovács, E. A., Ország, A., Pfeifer, D., & Benczúr, A. (2024). Generalized naive bayes. *arXiv preprint arXiv:2408.15923*.

Kraus, D., & Czado, C. (2017). D-vine copula based quantile regression. *Computational Statistics & Data Analysis*, *110*, 1–18.

Kurowicka, D. (2010). Optimal truncation of vines. In *Dependence Modeling: Vine Copula Handbook* (233–247). World Scientific.

Lemaire, V., Le Boudec, N., Guyomard, V., & Fessant, F. (2024). Viewing the process of generating counterfactuals as a source of knowledge: a new approach for explaining classifiers. In *2024 International Joint Conference on Neural Networks (IJCNN)*, (1–8). IEEE.

Ling, C. K., Fang, F., & Kolter, J. Z. (2020). Deep archimedean copulas. *Advances in Neural Information Processing Systems*, *33*, 1535–1545.

Loaiza Maya, R. A., Gomez-Gonzalez, J. E., & Melo Velandia, L. F. (2015). Latin american exchange rate dependencies: A regular vine copula approach. *Contemporary Economic Policy*, *33*(3), 535–549.

McKinney, S. M., Sieniek, M., Godbole, V., Godwin, J., Antropova, N., Ashrafian, H., Back, T., Chesus, M., Corrado, G. C., Darzi, A. et al. (2020). International evaluation of an ai system for breast cancer screening. *Nature*, *577*(7788), 89–94.

Nelsen, R. B. (2006). An introduction to copulas. *Journal of Statistical Planning and Inference*.

Okoli, C. (2023). Statistical inference using machine learning and classical techniques based on accumulated local effects (ale). *arXiv preprint arXiv:2310.09877*.

Patrício, M., Pereira, J., Crisóstomo, J., Matafome, P., Gomes, M., Seiça, R., & Caramelo, F. (2018). Using resistin, glucose, age and bmi to predict the presence of breast cancer. *BMC cancer*, *18*, 1–8.

Pfeifer, D., Strassburger, D., & Philipps, J. (2020). Modelling and simulation of dependence structures in nonlife insurance with bernstein copulas. *arXiv preprint arXiv:2010.15709*.

Rose, D. (2015). *Modeling and estimating multivariate dependence structures with the Bernstein copula*. PhD thesis, lmu.

Şahin, Ö., & Joe, H. (2024). Vine copula-based classifiers with applications. *Journal of Classification*, 1–29.

Sancetta, A. (2004). The bernstein copula and its applications to modeling and approximations of multivariate distributions. *Econometric Theory*, *20*(3), 535–562.

Scheffer, M., & Weiß, G. N. (2017). Smooth nonparametric bernstein vine copulas. *Quantitative Finance*, *17*(1), 139–156.

So, M. K., & Yeung, C. Y. (2014). Vine-copula garch model with dynamic conditional dependence. *Computational Statistics & Data Analysis*, *76*, 655–671.

Soto, M., Ochoa, A., González-Fernández, Y., Milanés, Y., Álvarez, A., Carrera, D., & Moreno, E. (2012). Vine estimation of distribution algorithms with application to molecular docking. In *Markov Networks in Evolutionary Computation* (209–225). Springer.

Sun, Y., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Learning vine copula models for synthetic data generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, (5049–5057).

Torres-Roca, J. F., Eschrich, S., Zhao, H., Bloom, G., Sung, J., McCarthy, S., Cantor, A. B., Scuto, A., Li, C., Zhang, S. et al. (2005). Prediction of radiation sensitivity using a gene expression classifier. *Cancer research*, *65*(16), 7169–7176.

Yamut, T., & Hudaverdi, B. (2023). Classification with bernstein copula as discrimination function. *Communications in Statistics-Simulation and Computation*, 1–17.

Zeng, Z., & Wang, T. (2022). Neural copula: A unified framework for estimating generic high-dimensional copula functions. *arXiv preprint arXiv:2205.15031*.

APPENDICES

A.1

In the appendix, a summarized outline of created R functions which used in the applications are provided below. Also, the custom vine structure function is compared with official R package.

A.1.1 R Modules of Statistical Learning Frameworks

CustomVineStructure function deploys a valid vine stucture corresponds to the d-variate probability density function. The function contains custom handler functions to carry out generating conditional observations, h-function recursions and on choosing the spanning tree that has optimal correlation sum. The spanning tree optimization is provided by NetworkToolbox library for maximum objective and for minimum objective it is provided by the igraph library.

```
CustomVineStructure(Dataset, SpanningTreeRule = "max",
calibration_process, parametricMarginals = T,
trimVine = NULL, vineObject = NULL, preMarginals = NULL)
```

- **Dataset:** d-variate copula observations object.
- **SpanningTreeRule:** Decising spanning tree rule with respect to minimum or maximum of the total of Kendall's tau. Values: 'min', 'max'.
- **calibration_process:** Calibration handler function for bivariate copulas of the vine.
- **parametricMarginals:** Flag for using parametric or empirical marginals.
- **trimVine:** Tree cut point for truncated vine.
- **vineObject:** Pre-structured vine object to make calibration continue from the last determined tree.

- **preMarginals:** Pre-calculated marginal values to integrate in the first tree of the vine.

CustomVineDensity function computes density of d-variate data object with respect to pre-determined vine order and estimated parameters.

```
CustomVineDensity(vineobject, Dataset, preDist = NULL,
preDensity = NULL)
```

- **vineobject:** Pre-structured vine object to compute vine density values.
- **Dataset:** d-variate copula observations object.
- **preDist:** Pre-calculated marginal distribution values.
- **preDensity:** Pre-calculated marginal density values.

```
CopulaComputer(BiData, dist_type = "dens",
onlypar = F, calibration_process = NULL)
```

- **BiData:** Given bivariate data to be evaluated.

- **dist_type:** The form of the copula function. Options are; ”dens” and ”cond”.

- **calibration_process:** This argument is a handler function, which can be used with CopulaSelectorSetup() or CopulaManualSetup() and it allows specification of candidate copulas, selection criterias, trial numbers and extra distribution options.

- **onlypars:** If true, return the estimated parameter of selected copula.

```
calibration_process = CopulaSelectorSetup(
select_param = c("Gumbel", "Clayton"),
select_nonparam = c("Bernstein5", "Bernstein7"),
criteria = selectioncriteria(measure = "rd", porder = 2),
dist_options = nonpara_options(IntSoln=TRUE)
)
############################################################
calibration_process = CopulaSelectorSetup(
select_param = c("Gumbel", "Clayton", "Gaussian"),
trial = 20,
)
############################################################
calibration_process = CopulaManualSetup("Gumbel", 2.2)
############################################################
calibration_process = CopulaManualSetup("Bernstein10",
dist_options = nonpara_options(IntSoln=TRUE))
```

- **select_param:** Parametric copula candidates.

- **select_nonparam:** Bernstein copula candidates.

- **criteria:** Selection criteria for choosing from candidate copulas.

- **dist_options:** Details on calibrating bivariate copulas.

- **trial:** Number of times that the criteria will be computed in order to compute final weighted criteria.

```
selectioncriteria = function(measure, porder)(testValues,
benchValues)
```

- **measure:** Selection criteria function. Options are; 'rd' (relative distance), 'mse' (mean squared error)

- **porder:** Order of the criteria function.

- **testValues:** True values that fitted to candidate distributions.

- **benchValues:** Simulated values from a candidate distribution.

GridJointDensityFit function computes the raw density values on $n \times n$ contingency matrix (grid) then carries out optimization on the Bernstein copula arguments to make them satisfy the uniformity condition of the copula.

```
GridJointDensityFit = function(X, gridsize = 5, GridOptim = TRUE,
IntSoln = FALSE)
```

- **X:** Data object that will be fitted to the Bernstein copula

- **gridsize:** The degree of Bernstein polynomial.

- **GridOptim:** Flag of choice of using optimization on contingency matrix.

- **IntSoln:** Flag of using initial solution approach on contingency matrix optimization (Lagrange).

BernJointDensity computes the density values of Bernstein copula for given dataset with and the provided contingency matrix. **BernPartial2** computes the values of conditional distribution of Bernstein copula with respect to the second for given dataset with and the provided contingency matrix.

```
BernJointDensity = function(X, CMobject)


BernPartial2 = function(X, CMobject)
```

- **X:** Referenced data object for computation.

- **CMobject:** Fitted contingency matrix.

A.1.2 Custom Vine and R's VineCopula Codes Cross Validation

Our custom vine structure function is compared with R's VineCopula package with a standard vine procedure in order to test the validity of our approach. In Figure A.1, it can be seen that our algorithm chosen the same structure as the default framework of the package for the first and last trees of simulated 10 dimensional data. Only the Kendall's tau is slightly differ. It is suspected that the library uses parametric calculation instead of the sample Kendall's tau as our algorithm does. Still, the structure is identical with its proportional correlations.

```
> myvine$TREE_STRUCTURE
[[1]]
  NewNodeIndex DistIndex      Corr
1            1       1,5 0.5688889
2            2       2,5 0.6016162
3            3       3,5 0.4953535
4            4       4,5 0.5725253
5            5       4,8 0.5919192
6            6       4,9 0.5555556
7            7      4,10 0.5587879
8            8       5,7 0.5951515
9            9       6,8 0.5579798
```

Tree 1 of custom algorithm

```
[[9]]
  NewNodeIndex      DistIndex       Corr
1            1 9,10|63182475 0.06020202
```

Tree 9 of custom algorithm

```
R-vine copula with the following pair-copulas:
Tree 1:
 4,9  Gumbel (par = 2.09, tau = 0.52)
 5,3  Gumbel (par = 1.97, tau = 0.49)
 5,1  Gumbel (par = 2.08, tau = 0.52)
 5,2  Gumbel (par = 2.29, tau = 0.56)
 5,7  Gumbel (par = 2.18, tau = 0.54)
 4,5  Gumbel (par = 2.21, tau = 0.55)
 8,6  Gumbel (par = 2.06, tau = 0.51)
 4,8  Gumbel (par = 2.32, tau = 0.57)
10,4  Gumbel (par = 2.31, tau = 0.57)
```

Tree 1 of VineCopula package

```
Tree 9:
10,9;6,1,2,7,8,3,5,4  Gumbel (par = 1.06, tau = 0.05)
```

Tree 9 of VineCopula package

Figure A.1 Output of the vine structure for simulated observations. The first column shows the first tree, and the second column shows the last one.