D-vine copula based mean regression and a comparison with gradient boosting

Master’s Thesis

by

Georg Huber

Supervisor: Prof. Claudia Czado, PhD
Advisor: Prof. Claudia Czado, PhD
Nicole Barthel
Submission date: September 30, 2018
I hereby declare that this thesis is my own work and that no other sources have been used except those clearly indicated and referenced.

Munich, September 30, 2018
Acknowledgments

First of all, I would like to express my gratitude to Prof. Claudia Czado for making this Master’s thesis possible. The frequent meetings throughout the last six months with the many helpful discussions and hints enriched the progress of my work a lot. Moreover, I want to thank my advisor Nicole Barthel for all the guidance and explanations, especially during the final phase of my thesis. I also want to thank Thomas Nagler for the support concerning the implementations and the simulations that ran on the cluster.
Abstract

Over the last years, vine copulas have gained more and more popularity in the statistical community as they allow for very flexible modelling of complex dependence structures. Using so-called pair copula constructions we can construct high-dimensional copulas by using bivariate building blocks.

In this thesis we use the subclass of D-vine copulas in order to develop a new method for mean regression. We present five different methods for conditional mean estimation using the properties of D-vine copulas. These methods allow us to construct predictive models for a response variable given some observed covariates. Within this framework the copulas can either be estimated parametrically or nonparametrically. Moreover, we discuss a workaround for discrete data, since in this case the classical copula theory cannot be applied.

Another currently hot topic are machine learning methods and in the context of mean regression especially gradient boosting is a very frequently used tool. However, often it is seen as a black-box and most of the literature lacks detailed mathematical explanations. Therefore, we will give an extensive exposition on gradient boosting for regression trees, including an introduction to regression trees and simple examples in order to clarify the methodology.

Afterwards, we will show in an exhaustive simulation study that our newly developed D-vine copula based mean regression can achieve very good results in many different scenarios and can even outperform gradient boosting in some setups. Finally, we will also apply the presented methods to a real-world data set.
Zusammenfassung


In dieser Arbeit benutzen wir die Unterklassen der D-vine Copulas, um eine neue Mittelwert-Regressions-Methode zu entwickeln. Wir erläutern fünf verschiedene Methoden zur Schätzung des bedingten Erwartungswertes, indem wir die Eigenschaften von D-vine Copulas benutzen. Diese Methoden können anschließend verwendet werden, um Prognosemodelle zu entwickeln. Im Rahmen dieser Methodik können wir Copulas sowohl parametrisch als auch nicht-parametrisch schätzen. Darüber hinaus erläutern wir eine Methode, die auch die Benutzung von diskreten Variablen erlaubt, da auf diese die klassische Copula-Theorie nicht angewendet werden kann.


Im Anschluss zeigen wir in einer ausführlichen Simulationsstudie, dass die entwickelten D-vine Copula basierten Regressionsmethoden in vielen verschiedenen Szenarien sehr gute Ergebnisse liefern und darüber hinaus in einigen Fällen sogar besser als Gradienten-Boosting sind. Zum Schluss wenden wir die erläuterten Methoden auf einen echten Datensatz an.
## Contents

1 Introduction  

2 Background  
   2.1 Important distributions ........................................ 3  
   2.2 Numerical integration in R .................................... 4  
   2.3 Inverse transform method ...................................... 6  
   2.4 Kernel density estimation ...................................... 6  
   2.5 Generalized linear model (GLM) ................................. 7  
   2.6 Performance measures .......................................... 8  
   2.7 Dummy variable encoding ...................................... 9  
   2.8 Cross-validation (CV) .......................................... 10  

3 Introduction to copulas  
   3.1 Definitions and basic properties .............................. 13  
   3.2 Dependence measures .......................................... 14  
   3.3 Important copula families .................................... 15  
   3.4 D-vine copulas ................................................ 18  
   3.5 R-vine copulas ................................................ 22  
   3.6 D-vine quantile regression .................................... 25  
   3.7 Jittering for discrete variables ............................... 27  

4 D-vine mean regression  
   4.1 Preparations .................................................. 29  
   4.2 Estimation of the conditional mean .................................. 30  
   4.3 Remarks on the implementation .................................. 33  

5 Gradient boosting with regression trees  
   5.1 Introduction to regression trees .................................. 35  
   5.2 Main idea of gradient boosting .................................. 38  
   5.3 Loss minimization via function optimization ................... 42  
   5.4 Tuning parameters in gradient boosting ......................... 47  

6 Setup of simulation studies  
   6.1 Remarks on general procedure .................................. 51  
   6.2 Studied data structures ....................................... 51
6.3 Explanation of different models ........................................... 58
6.4 Explanation of scenarios ..................................................... 60

7 Selection of MC sample size ................................................. 63
7.1 Scenario MCS.1 ............................................................... 63
7.2 Scenario MCS.2 ............................................................... 65
7.3 Scenario MCS.3 ............................................................... 65
7.4 Scenario MCS.4 ............................................................... 68

8 Number of quantiles in Method SQI ........................................ 71

9 Analyzing the effect of data set size ....................................... 73

10 Comparison for three covariates .......................................... 77
10.1 Results of Scenario SL.3 ................................................... 77
10.2 Results of Scenario QL.3 .................................................. 80
10.3 Results of Scenario GR.3 .................................................. 83
10.4 Results of Scenario DF.4 ................................................... 83
10.5 Results of Scenario DR.4 ................................................... 86
10.6 Results of Scenario RR.4 ................................................... 86
10.7 Analyzing estimated pair-copulas ....................................... 89

11 Comparison for ten covariates ............................................. 91
11.1 Results of Scenario SL.10 .................................................. 91
11.2 Results of Scenario QL.10 .................................................. 93
11.3 Results of Scenario GR.10 .................................................. 93
11.4 Results of Scenario DF.11 .................................................. 96
11.5 Conclusion of simulation studies ....................................... 96

12 Application to abalone data set .......................................... 99
12.1 Introduction to data set .................................................... 99
12.2 Exploratory data analysis ................................................. 100
12.3 Model selection and comparison ....................................... 102

13 Conclusion and outlook ..................................................... 105

List of Figures ................................................................. 107

List of Tables ................................................................. 109

Bibliography ................................................................. 113

A Implementation of D-vine mean regression ............................. 117
A.1 Integration over quantiles ............................................... 117
A.2 Calculating the estimating equation ................................... 118
A.3 Monte Carlo simulation .................................................... 120
A.4 Integration over (conditional) copula density ....................... 121
A.5  Simplified quantile integration ............................ 123
A.6  Calculate conditional vine-copula density ................. 123

B  Further results of simulation studies .......................... 127
B.1  Selection of parameters ........................................ 127
B.2  Comparison for three covariates .............................. 139
B.3  Comparison for ten covariates ................................. 142
Chapter 1

Introduction

The need for good statistical models is more and more increasing as the amount of available data is growing. These days, in all fields of daily life huge amounts of data are collected and data scientists try to extract as much information as possible from the data. The tool-set that is used for these analyses has grown over the last years and especially machine learning methods have gained a lot of popularity. However, when trying to model dependencies between different variables the importance of copulas has rapidly grown as they allow to model the marginal distributions and the dependence structure separately, using the fundamental theorem of Sklar \citep{Sklar1959}. Moreover, so-called pair-copula constructions (PCCs) enable us to express high-dimensional multivariate copulas by using bivariate building blocks, yielding very flexible models that can handle complex dependence structures. Joe \citep{Joe1996} was the first to introduce by continuous mixing a special class of PCCs. These models were further developed and named vine copulas by Bedford and Cooke \citep{BedfordCooke2002} and have become the most important class of PCCs. Kraus and Czado \citep{KrausCzado2017} used D-vine copulas in order to come up with a new way for quantile regression that overcomes many disadvantages of classical quantile regression. Based on their approach we build a novel approach for D-vine mean regression. Therefore, we present five different methods for conditional mean estimation based on the theory of D-vine copulas.

In the remainder of this thesis we first of all introduce in Chapter 2 the theoretical background and methods that are used throughout the thesis. An introduction on copulas and their basic properties is then given in Chapter 3. The special class of D-vine copulas is further explained in Section 3.4 and the approach of Kraus and Czado \citep{KrausCzado2017} is presented in Section 3.6. Afterwards, we extend their results to D-vine mean regression in Chapter 4. Moreover, in Chapter 5 we introduce a very popular method from machine learning, namely gradient boosting with regression trees. We will then compare our newly developed method for D-vine mean regression with the prediction performance of gradient boosting models in an extensive simulation study. Therefore, we first of all describe the setup of the simulation studies in Chapter 6 and find optimal parameters for the different methods to perform D-vine mean regression in Chapters 7 to 9. Afterwards, we compare the models from D-vine mean regression and gradient boosting with the fit of ordinary least square models or the fit of a generalized linear model. These comparisons are made in Chapter 10 for setups with
three covariates and in Chapter 11 for scenarios with ten covariates. The simulation studies are followed by a real-data application in Chapter 12. We conclude the thesis with a summary and an outlook for further possible topics of research in Chapter 13.
Chapter 2

Background

In this chapter we introduce important mathematical concepts and definitions which will be used throughout the thesis.

2.1 Important distributions

We start with the definitions and properties of the probability distributions that we use in the thesis.

2.1.1 Normal distribution

The normal distribution is one of the most famous distributions. In the univariate case, it is a continuous probability distribution and has two parameters \( \mu \in \mathbb{R} \) and \( \sigma > 0 \). It is denoted as \( \mathcal{N}(\mu, \sigma^2) \) with density function given by

\[
\varphi(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \text{ for all } x \in \mathbb{R}.
\]

Moreover, for a normal distributed random variable \( X \sim \mathcal{N}(\mu, \sigma^2) \) the mean and variance are given by

\[
\mathbb{E}[X] = \mu \quad \text{and} \quad \text{Var}(X) = \sigma^2.
\]

The 2-dimensional normal distribution is defined as \( \mathcal{N}_2(\mu, \Sigma) \), where

\[
\mu = (\mu_1, \mu_2) \in \mathbb{R}^2 \quad \text{and} \quad \Sigma = \begin{pmatrix} \sigma_1^2 & \rho \sigma_1 \sigma_2 \\ \rho \sigma_1 \sigma_2 & \sigma_2^2 \end{pmatrix} \in \mathbb{R}^{2 \times 2}
\]

are the two-dimensional mean vector and the 2\times2 covariance matrix. The parameter \( \rho \) denotes the correlation between the margins and \( \sigma_1 > 0 \) and \( \sigma_2 > 0 \) holds. Its density is given by

\[
\phi(x|\mu, \Sigma) = \frac{1}{2\pi \sqrt{\det(\Sigma)}} \cdot \exp\left\{ -\frac{1}{2} (x - \mu)^\top \Sigma^{-1} (x - \mu) \right\} \text{ for all } x \in \mathbb{R}^2.
\]
We will also use the property of the bivariate normal distribution that the conditional distribution of one margin given the other is again a normal distribution, this means for \( \mathbf{X} = (X_1, X_2) \sim \mathcal{N}_2(\mathbf{\mu}, \mathbf{\Sigma}) \) it holds

\[
X_1 | X_2 = x_2 \sim \mathcal{N} \left( \mu_1 + \frac{\sigma_1}{\sigma_2} \rho(x_2 - \mu_2), (1 - \rho^2)\sigma_1^2 \right).
\]

(2.1)

### 2.1.2 Gamma distribution

The gamma distribution is also a continuous two-parametric distribution and we will use two different parametrizations of it. We start with the shape-rate parametrization which is specified by the shape \( a > 0 \) and the rate \( b > 0 \) and density function given by

\[
g(y|a,b) = \frac{b^a}{\Gamma(a)} y^{a-1} e^{-by} \quad \text{for all } y > 0.
\]

Then, for a random variable \( Y \sim \text{Gamma}(a,b) \) the mean and variance are given by

\[
E[Y] = \frac{a}{b} \quad \text{and} \quad \text{Var}(Y) = \frac{a}{b^2}.
\]

However, in this thesis we will mainly use the so-called mean-parametrization which is given by the density

\[
g(y|\mu,\nu) = \frac{1}{\Gamma(\nu)} \left( \frac{\nu y}{\mu} \right)^{\nu} e^{-\frac{\nu y}{\mu}} \quad \text{for all } y > 0, \mu > 0 \text{ and } \nu > 0.
\]

(2.2)

The mean and variance of \( Y \sim \text{Gamma}(\mu,\nu) \) are then given as

\[
E[Y] = \mu \quad \text{and} \quad \text{Var}(Y) = \frac{\mu^2}{\nu}.
\]

### 2.2 Numerical integration in R

In this section we briefly explain how numerical integration is done in the statistical programming software R (R Core Team, 2018) which we will use for our implementations later on. All the explanations are based on Piessens et al. (1983) and correspond to the R-function \texttt{integrate()}.

Suppose we are given a one-dimensional, real-valued function

\[
f : \mathbb{R} \rightarrow \mathbb{R}, \quad x \mapsto f(x)
\]

and integration boundaries \(-\infty < a < b < \infty\). Then, the goal of the numerical integration is to find an approximation of

\[
I := \int_{a}^{b} f(x) \, dx
\]
within a requested absolute and/or relative accuracy denoted by $\varepsilon_a$ and $\varepsilon_r$, respectively. The approximation is attained through a so-called quadrature sum given by

$$Q_n := \sum_{i=1}^{n} w_i \cdot f(x_i) \approx I,$$

where $w_1, \ldots, w_n$ are weights and $x_1, \ldots, x_n$ are $n$ distinct evaluation points which have to lie in the interval $(a, b)$. We can then define the error as

$$E_n := Q_n - I.$$

Now, we need to find a procedure that gives weights $w_1, \ldots, w_n$ and evaluation points $x_1, \ldots, x_n$ such that

$$E_n \leq \max\{\varepsilon_a, \varepsilon_r\}$$

holds. A natural choice would be to set

$$x_i := a + \frac{i(b - a)}{n + 1} \quad \text{and} \quad w_i = \frac{1}{n}, \quad \text{for } i \in \{1 \ldots, n\}$$

to obtain $n$ equidistant points in the interval $(a, b)$ with the same weights for all of them. However, in general this does not yield a good approximation if the function $f$ changes more rapidly in some sub-intervals $(x_j, x_{j+1})$ where $j \in D \subset \{1, \ldots, n\}$. Therefore, Piessens et al. (1983) present a method which chooses the evaluation points and weights such that regions with more rapid changes of $f(x)$ include more evaluation points and regions where $f(x)$ is slowly changing include less evaluation points. The exact procedure is beyond the scope of this thesis and will hence not be discussed.

So far we assumed that $-\infty < a < b < \infty$. Now, assume that $a$ is still finite but $b$ is infinite. Then, the integral can be transformed to

$$\int_{a}^{\infty} f(x) \, dx = -\int_{0}^{1} f\left(a + \frac{1-t}{t}\right) t^{-2} \, dt.$$

Similarly we get

$$\int_{-\infty}^{b} f(x) \, dx = \int_{0}^{1} f\left(b - \frac{1-t}{t}\right) t^{-2} \, dt,$$

for the case where $a$ is infinite and $b$ is finite. Moreover, if both integration limits are infinite we get

$$\int_{-\infty}^{\infty} f(x) \, dx = \int_{-\infty}^{0} f(x) \, dx + \int_{0}^{\infty} f(x) \, dx = \int_{0}^{1} \left(f\left(\frac{1-t}{t}\right)f\left(\frac{t-1}{t}\right)\right) t^{-2} \, dt.$$

After this transformation the integrals with infinite limits can be approximated in the same way as described before.
CHAPTER 2. BACKGROUND

2.3 Inverse transform method

During D-vine mean regression, which will be introduced in Chapter 4, we will need to be able to generate random numbers from arbitrary distributions. Therefore, we will introduce a popular method for random number generation which is called inverse transform method. We will base our explanations on Rizzo (2007).

We start by introducing the probability integral transform (PIT) which says that for any continuous random variable $X$ with cumulative distribution function (cdf) $F_X(x)$ the random variable $U := F_X(X)$ is uniformly distributed on $(0, 1)$. Moreover, we define the inverse of $F_X$ as

$$F_X^{-1} := \inf\{x : F_X(x) = u\} \quad \text{for } 0 < u < 1.$$  

Now for $U \sim Unif(0, 1)$ we get for all $x \in \mathbb{R}$

$$\mathbb{P}(F_X^{-1}(U) \leq x) = \mathbb{P}(\inf\{t : F_X(t) = U\} \leq x) = \mathbb{P}(U \leq F_X(x)) = F_U(F_X(x)) = F_X(x).$$

This means that $F_X^{-1}(U)$ has the same distribution as $X$. Hence, we can generate a random observation of $X$ by first generating a random observation $u$ from the $Unif(0, 1)$ distribution and then calculating the value of $F_X^{-1}(u)$. This method is always easy to apply if we are able to calculate $F_X^{-1}$.

2.4 Kernel density estimation

In many statistical applications one needs to find estimates of the cumulative distribution function $F$ of a random variable $X$. One frequently used approach is kernel density estimation (kde) as it yields a nonparametric, continuous and invertible estimate $\hat{F}$ of the cdf $F$. We will use the kernel density estimator introduced by Parzen (1962). For an observed sample $x_1, \ldots, x_n$ from $X$ and $h > 0$ a bandwidth parameter it is defined as

$$\hat{F}(x) := \frac{1}{n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \quad \text{for all } x \in \mathbb{R},$$

where $K(x) := \int_{-\infty}^{x} k(t)dt$ with $k(\cdot)$ being a symmetric probability density function. Throughout this thesis we will always use the Gaussian kernel which is given by

$$k(x) = \varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

Moreover, for the selection of the optimal bandwidth we use the direct plug-in methodology of Sheather and Jones (1991).
2.5 Generalized linear model (GLM)

During our simulation studies in Chapter 6 we will use data that follow the assumptions of a gamma regression which is a special case of a so-called **generalized linear model (GLM)**. Therefore, we will first of all explain the general setup of a GLM in 2.5.1 and then give details for gamma regression in 2.5.2.

2.5.1 Short introduction to GLMs

GLMs are a widely used generalization of the ordinary least square regression method and have been introduced by Nelder and Wedderburn (1972). We assume that a set of independent observations \( y_1, \ldots, y_n \) comes from a random variable \( Y \) whose distribution is a member of the so-called **exponential family** which is given by

\[
f(y, \theta, \phi) = \exp \left( \frac{y \theta - b(\theta)}{a(\phi)} + c(y, \phi) \right),
\]

where \( \theta \) is called the **canonical parameter**, \( \phi \) is a **dispersion parameter** and \( a(\cdot), b(\cdot) \) and \( c(\cdot) \) are known functions. The mean and variance for a random variable \( Y \) coming from the exponential family are given by

\[
E[Y] = b'(\theta) \quad \text{and} \quad Var(Y) = b''(\theta)a(\phi),
\]

respectively,

where \( b'(\cdot) \) and \( b''(\cdot) \) denote the first and second derivative of \( b(\cdot) \).

Furthermore, for given covariates \( x_i := (x_{i1}, \ldots, x_{id}) \) the **linear predictor** for \( i = 1, \ldots, n \) is defined as

\[
\eta_i(\beta) := x_i^\top \beta = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_d x_{id},
\]

where \( \beta = (\beta_0, \ldots, \beta_d) \) are \( d \) unknown **regression parameters**. Finally, we use a **link function**

\[
g(\mu_i) = \eta_i(\beta) = x_i^\top \beta
\]

to define the relationship between the mean \( \mu_i \) of \( Y_i \) and the linear predictor \( \eta_i \).

2.5.2 GLM for gamma regression

In the special case of gamma regression we assume \( Y_i \sim \text{Gamma}(\mu_i, \nu) \) to follow a gamma distribution using the mean parametrization as defined in (2.2). First of all, we note that the gamma distribution parametrized by mean \( \mu \) and shape \( \nu \) is a
member of the exponential family given in (2.3) with
\[
\theta = \frac{-1}{\mu}, \\
\phi = \frac{1}{\nu}, \\
a(\phi) = \phi, \\
b(\theta) = \ln(\mu) = -\ln(-\theta), \\
b'(\theta) = \frac{-1}{\theta} = \mu, \\
b''(\theta) = \frac{1}{\theta^2} = \mu^2 \quad \text{and} \\
c(y, \phi) = \frac{1 - \phi}{\phi} \ln(y) - \frac{\ln(\phi)}{\phi} - \ln\left(\Gamma\left(\frac{1}{\phi}\right)\right).
\]
A detailed derivation of the parameters and functions can be found in Czado (2017). Moreover, we will use the \textbf{log link}
\[g(\mu) = \ln(\mu),\]
implying that the expectation and variance of \(Y\) are given by
\[E[Y] = \mu = \exp(x_i^\top \beta) \quad \text{and} \quad Var(Y) = \frac{\mu^2}{\nu}.
\]

2.6 Performance measures

In this section we define some performance measures which we will later use in our simulation studies and the data application for the comparison of different models.

\textbf{Definition 2.1 (RMSE, rRMSE, oRMSE):}

Let \(y_1, \ldots, y_n\) be some observations coming from a random variable \(Y\) and \(\hat{y}_1, \ldots, \hat{y}_n\) be the corresponding predictions coming from some estimator. Then, the \textbf{root mean squared error} of this estimator is defined as
\[RMSE := \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}.
\]
Moreover, we define the \textbf{relative root mean squared error} as
\[rRMSE := \frac{RMSE}{\bar{y}},\]
where \(\bar{y}\) is the mean of the observed values \(y_1, \ldots, y_n\). If we know the true expectation \(\tilde{y}_1, \ldots, \tilde{y}_n\) of \(Y|\mathbf{x}_i\) where \(\mathbf{x}_i\) are some covariates observed for sample unit \(i = 1, \ldots, n\), we can define the \textbf{oracle root mean squared error} as
\[oRMSE := \sqrt{\frac{\sum_{i=1}^{n} (\tilde{y}_i - \hat{y}_i)^2}{n}}.
\]
2.7 Dummy variable encoding

Many real world regression problems contain categorical data taking values without natural ordering. Typical examples are the variable “gender” which can take levels “male” and “female” or the variable “country” which might take levels “Germany”, “USA” and “France”. In order to build a regression model using such a categorical variable as covariate one usually has to firstly transform it to some numeric variable. One popular way to do this is **dummy variable encoding** which replaces a categorical variable with $K$ levels by $K - 1$ binary variables. Therefore, the first level of the categorical variable is chosen to be the reference value. Then, for all other levels a new binary dummy variable is created and is set to 1 if the data point falls in this category and 0 otherwise. This means that the reference level corresponds to the case when all dummy variables take the value 0. The original variable is removed from the data set. We illustrate this in a small example.

**Example 2.2:**
We consider a small data set which contains the continuous variables “weight” and “age” and the categorical variables “gender” and “country”. Table 2.1 shows the original data. Now, we apply dummy variable encoding to the categorical variables.

<table>
<thead>
<tr>
<th>weight</th>
<th>gender</th>
<th>country</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>56</td>
<td>Female</td>
<td>France</td>
</tr>
<tr>
<td>2</td>
<td>83</td>
<td>Male</td>
<td>Germany</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>Female</td>
<td>UK</td>
</tr>
<tr>
<td>4</td>
<td>72</td>
<td>Male</td>
<td>USA</td>
</tr>
<tr>
<td>5</td>
<td>54</td>
<td>Female</td>
<td>France</td>
</tr>
<tr>
<td>6</td>
<td>102</td>
<td>Male</td>
<td>Germany</td>
</tr>
<tr>
<td>7</td>
<td>69</td>
<td>Female</td>
<td>UK</td>
</tr>
<tr>
<td>8</td>
<td>93</td>
<td>Male</td>
<td>USA</td>
</tr>
</tbody>
</table>

Table 2.2 shows the result of the encoding. We see that for the variable “gender” the level “female” is chosen to be the reference level and for the variable “country” the level “France” is taken as reference.

If the data set contains many categorical variables or categorical variables with many different levels the dummy encoded data contain many zero-entries. Therefore, such data sets are then called **sparse** and the corresponding matrices are called **sparse matrices**. In contrast, data sets which do not contain many zero-entries are called **dense** and the corresponding matrices **dense matrices**.
Table 2.2: Exemplary data containing continuous variables weight and age and dummy variables for the former categorical variables gender and country after application of dummy variable encoding.

<table>
<thead>
<tr>
<th>weight</th>
<th>genderMale</th>
<th>countryGermany</th>
<th>countryUK</th>
<th>countryUSA</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>70</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>59</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>67</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>53</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
</tr>
</tbody>
</table>

2.8 Cross-validation (CV)

In this section we introduce a frequently used method for model selection. Our explanations are based on the ones given in Hastie et al. (2001). Consider a regression setup where we can build a range of different models by the selection of covariates or the selection of tuning parameters, for example tree size and learning rate in gradient boosting (see Section 5.4). Then, the question arises which model is the best in terms of fitting the observed data but also in having predictive power for future observations. A model that fits the observed data too closely and therefore, does not only extract the general structure of the data but also the noise, usually loses predictive power for future observations. This phenomenon is called overfitting and an example is shown in Figure 2.1.

A popular method in order to avoid overfitting and to select a model that on the one hand fits the observed data quite well but also has predictive power is K-fold cross-validation (CV). Suppose we observe data \( y_i \) and \( x_i := (x_{i1}, \ldots, x_{id}) \) for \( i \in \{1, \ldots, n\} \) and the number of covariates \( d \geq 1 \). Now, we divide the observed data into \( K \) roughly equal-sized distinct parts and fit \( K \) models \( \hat{H}^{-k}(x, \alpha) \) for \( k \in \{1, \ldots, K\} \), where \( \hat{H}^{-k}(x, \alpha) \) denotes the model which was built containing all data except from part \( k \) and using parameters \( \alpha \). Now, let

\[
\kappa : \{1, \ldots, n\} \mapsto \{1, \ldots, K\}
\]

be an indexing function that indicates for each observation \( i \) to which partition it is allocated, this means \( \kappa(i) = k \) for all observations \( i \) that fall into partition \( k \). We define

\[
CV(\hat{H}, \alpha) := \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{H}^{-\kappa(i)}(x_i, \alpha))
\]  

(2.4)

to be the cross-validation estimate of prediction error, where \( L \) is a pre-specified loss function which will in this thesis usually be the quadratic loss defined by

\[
L : \mathbb{R}^2 \to \mathbb{R}, \quad L(y, x) := \frac{1}{2}(y - x)^2.
\]
Figure 2.1: Example for a highly overfitted polynomial model (red) and good fitting least square model (green). The polynomial model fits the observed data perfectly, however, for future observation it will very likely yield no good predictions.

The goal is to find the tuning parameter $\hat{\alpha}$ that minimizes (2.4) and we then choose our final model $H(x, \hat{\alpha})$ which we fit to all data. Typical values for the number of distinct parts of the data set are $K = 5$ or 10.
Chapter 3

Introduction to copulas

In this chapter we introduce copulas, a popular tool for analyzing multivariate distributions. They allow us to split the multivariate distribution into two parts: The univariate marginal distributions and the dependence structure. We will refer to Nelsen (2006) and Joe and Kurowicka (2011) if not stated otherwise.

3.1 Definitions and basic properties

Definition 3.1 (Copula):
A function $C : [0,1]^d \to [0,1]$ is called a $d$-dimensional copula, if it is a multivariate distribution function with uniformly distributed marginals. The corresponding copula density $c$ is obtained by partial differentiation with respect to its $d$ components

$$c(u_1, \ldots, u_d) = \frac{\partial^d}{\partial u_1 \ldots \partial u_d} C(u_1, \ldots, u_d) \quad \text{for all } u \in [0,1]^d.$$ 

Sklar (1959) proved the fundamental theorem of copula theory which allows to represent multivariate distributions in terms of their marginals and a corresponding copula.

Theorem 3.2 (Sklar’s Theorem):
Let $X = (X_1, \ldots, X_d) \in \mathbb{R}^d$ be a $d$-dimensional random vector with joint distribution function given by $F : \mathbb{R}^d \to [0,1]$ and marginal distribution functions $F_1, \ldots, F_d : \mathbb{R} \to [0,1]$. Then, there exists a copula $C$ such that for the joint distribution function it holds

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

Moreover, for the corresponding density function we get

$$f(x_1, \ldots, x_d) = c(F_1(x_1), \ldots, F_d(x_d)) \cdot f_1(x_1) \cdot \ldots \cdot f_d(x_d),$$

where $c$ is the copula density of $C$ and $f_1, \ldots, f_d$ are the marginal density functions of $X_1, \ldots, X_d$. If $F$ is absolutely continuous this copula $C$ is unique.

We can also invert this statement as

$$C(u_1, \ldots, u_d) = F(F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d)).$$
CHAPTER 3. INTRODUCTION TO COPULAS

for the copula and as
\[ c(u_1, \ldots, u_d) = \frac{f(F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d))}{f_1(F_1^{-1}(u_1)) \cdots f_d(F_d^{-1}(u_d))} \]
for the corresponding copula density.

**Theorem 3.3** (Invariance of copulas):

Let \( H_j, j \in \{1, \ldots, d\}, \) be invertible monotone functions and \( X = (X_1, \ldots, X_d) \) a random vector. Then for \( Y = (Y_1, \ldots, Y_d) \) with \( Y_j = H_j (X_j) \) the copula \( C_Y \) agrees with the copula \( C_X \).

**Proof.** Proofs of both theorems can be found in Nelsen (2006). \( \square \)

If we are only interested in the dependence structure we can also consider the so-called **u-scale** (or copula scale) which is reached by the application of the probability integral transform (PIT) – as introduced in Section 2.3 – to the marginals, this means
\[ U_j := F_j (X_j) \quad \text{for all} \quad j = 1, \ldots, d. \]
The resulting \( U_j \) are then uniformly distributed on \((0, 1)\) and according to Theorem 3.3 their joint distribution function \( F_U \) is the copula \( C_X \) which is associated with \( X \).

### 3.2 Dependence measures

In this section we introduce the most commonly used measures for the strength of dependence between two random variables and also present methods for the empirical estimation of these measures based on observed data. We start with the **Pearson product-moment correlation**, a measure for linear dependence between two random variables

**Definition 3.4** (Pearson product-moment correlation):

For two random variables \( X_1 \) and \( X_2 \) we define the coefficient of Pearson’s product moment correlation as
\[ \rho := \rho(X_1, X_2) := \text{corr}(X_1, X_2) = \frac{\text{Cov}(X_1, X_2)}{\sqrt{\text{Var}(X_1)} \sqrt{\text{Var}(X_2)}} \in [0, 1]. \]

Moreover, if we observe a random sample \((x_{11}, x_{12}), \ldots, (x_{n1}, x_{n2})\) from \((X_1, X_2)\) with sample means \( \bar{x}_1 := \frac{1}{n} \sum_{i=1}^{n} x_{i1} \) and \( \bar{x}_2 := \frac{1}{n} \sum_{i=1}^{n} x_{i2} \), we can get an empirical estimate of the Pearson correlation by
\[ \hat{\rho} := \hat{\rho}(X_1, X_2) := \frac{\sum_{i=1}^{n} (x_{i1} - \bar{x}_1)(x_{i2} - \bar{x}_2)}{\sqrt{\sum_{i=1}^{n} (x_{i1} - \bar{x}_1)^2 \sum_{i=1}^{n} (x_{i2} - \bar{x}_2)^2}}. \]

However, Pearson’s product moment correlation has a big drawback: It can only capture linear dependencies and as a result zero correlation does not imply independence. A common example is to let \( X_1 \) be a random variable with \( \mathbb{E}[X_1] = \mathbb{E}[X_1^3] = 0 \)
and to define \( X_2 := X_1^2 \). Then, \( \rho(X_1, X_2) = 0 \) but we can clearly see that \( X_1 \) and \( X_2 \) are dependent. Therefore, we introduce another dependence measure which is frequently used especially in copula modelling since it is a function of the copula and thus free of marginal effects.

**Definition 3.5** (Kendall’s \( \tau \)):

For independent and identically distributed copies \((X_{11}, X_{12})\) and \((X_{21}, X_{22})\) of the random vector \((X_1, X_2)\) Kendall’s \( \tau \) is defined as

\[
\tau(X_1, X_2) = \mathbb{P}((X_{11} - X_{21})(X_{12} - X_{22}) > 0) - \mathbb{P}((X_{11} - X_{21})(X_{12} - X_{22}) < 0).
\]

In order to calculate an empirical estimate of Kendall’s \( \tau \) given an observed random sample \((x_{11}, x_{12}), \ldots, (x_{n1}, x_{n2})\) of \((X_1, X_2)\) we first of all need the following definition.

**Definition 3.6** (Concordant and discordant pairs):

For \( x_i := (x_{i1}, x_{i2}) \) and \( x_j := (x_{j1}, x_{j2}) \) the pair \((x_i, x_j)\) is called

- **concordant** if \( x_{i1} < x_{j1} \) and \( x_{i2} < x_{j2} \) holds or \( x_{i1} > x_{j1} \) and \( x_{i2} > x_{j2} \) holds,
- **discordant** if \( x_{i1} < x_{j1} \) and \( x_{i2} > x_{j2} \) holds or \( x_{i1} > x_{j1} \) and \( x_{i2} < x_{j2} \) holds.

Now, let \( N_c \) be the number of concordant pairs and \( N_d \) the number of discordant pairs, then an estimate of Kendall’s \( \tau \) is given by

\[
\hat{\tau}(X_1, X_2) := \frac{N_c - N_d}{\binom{n}{2}} = \frac{N_c - N_d}{N_c + N_d}
\]

if there are no ties present.

### 3.3 Important copula families

After having introduced the general definition of copulas, we now show some frequently used examples of copulas which will also be used later on.

**Bivariate Gaussian copula**

The **bivariate Gaussian copula** is defined as

\[
C(u_1, u_2; \rho) := \Phi_2(\Phi^{-1}(u_1), \Phi^{-1}(u_2); \rho),
\]

where \( \Phi(\cdot) \) is the distribution function of a standard normal \( \mathcal{N}(0, 1) \) distribution and \( \Phi_2(\cdot, \cdot; \rho) \) is the bivariate normal distribution function with zero means, unit variances and coefficient of correlation \( \rho \). For the corresponding copula density we get

\[
c(u_1, u_2; \rho) := \frac{1}{\phi(x_1)\phi(x_2)\sqrt{1-\rho^2}} \exp\left(-\frac{\rho^2(x_1^2 + x_2^2) - 2\rho x_1 x_2}{2(1-\rho^2)}\right),
\]

where \( x_1 := \Phi^{-1}(u_1) \) and \( x_2 := \Phi^{-1}(u_2) \).
Moreover, we can give a relation between $\rho$ and Kendall’s $\tau$ by
\[
\tau = \frac{2}{\pi} \arcsin(\rho).
\]

A three-dimensional plot of the copula density and a normalized contour plot of a bivariate Gaussian copula can be found in the left panel of Figures 3.1 and 3.2, respectively.

**Bivariate Gumbel copula**
The bivariate Gumbel copula is given as
\[
C(u_1, u_2; \delta) := \exp \left( - \left( -\ln u_1^\delta + (-\ln u_2)^\delta \right)^{\frac{1}{\delta}} \right),
\]
with parameter of dependence $\delta \geq 1$. For $\delta = 1$ we have independence and $\delta \to \infty$ corresponds to full dependence. For the bivariate Gumbel copula we can express the parameter $\delta$ in terms of Kendall’s $\tau$ by
\[
\delta = \frac{1}{1 - \tau}.
\]
The density and normalized contour plot of a bivariate Gumbel copula are illustrated in the middle panel of Figures 3.1 and 3.2, respectively.

**Bivariate Clayton copula**
For the degree of dependence $0 < \delta < \infty$ the bivariate Clayton copula is given as
\[
C(u_1, u_2; \delta) := (u_1^{-\delta} + u_2^{-\delta} - 1)^{-\frac{1}{\delta}}.
\]
For $\delta = 0$ we have independence and $\delta \to \infty$ corresponds to full dependence. For the bivariate Clayton copula the relationship between the parameter $\delta$ and Kendall’s $\tau$ is given by
\[
\delta = \frac{2\tau}{1 - \tau}.
\]
A three-dimensional density plot and a normalized contour plot for a bivariate Clayton copula is shown in the right panel of Figures 3.1 and 3.2, respectively.

In order to extend the range of dependence one can also define new copulas by counterclockwise rotations of copula densities. This means we get the rotation of

- 90 degree by $c_{90}(u_1, u_2) := c(1 - u_1, u_2)$,
- 180 degree by $c_{180}(u_1, u_2) := c(1 - u_1, 1 - u_2)$ and
- 270 degree by $c_{270}(u_1, u_2) := c(u_1, 1 - u_2)$. 
3.3. IMPORTANT COPULA FAMILIES

Figure 3.1: Bivariate copula densities of Gaussian (left), Gumbel (middle) and Clayton (right) copula each which Kendall’s $\tau$ of 0.6.

Figure 3.2: Normalized contour plots of Gaussian (left), Gumbel (middle) and Clayton (right) copula each which Kendall’s $\tau$ of 0.6.

Another important class of copulas was introduced in Joe (1997) and more properties for this class – which will be introduced in the following – can be found in Joe (2015).

**Bivariate BB1 copula**

For parameters $\theta > 0$ and $\delta \geq 1$ the **bivariate BB1 copula** is defined as

$$C(u, v; \theta, \delta) = \left(1 + \left((u^{-\theta} - 1)^\delta + (v^{-\theta} - 1)^\delta\right)^{\frac{1}{\delta}}\right)^{-\frac{1}{\theta}} = \varphi\left(\varphi^{-1}(u) + \varphi^{-1}(v)\right),$$

where $\varphi(s) = \varphi_{\theta,\delta}(s) = (1 + s^{\frac{1}{\theta}})^{-\frac{1}{\delta}}$. Moreover, for $\theta \to 0^+$ this converges to the Gumbel family.
Bivariate BB6 copula
For parameters $\theta \geq 1$ and $\delta \geq 1$ the bivariate BB6 copula is defined as

\[
C(u, v; \theta, \delta) = 1 - \left( 1 - \left( (-\log(1 - \bar{u}^\theta))^\delta + (-\log(1 - \bar{v}^\theta))^\delta \right)^{1/\theta} \right)^{1/\delta},
\]

where $\varphi(s) = \varphi_{\theta, \delta}(s) = 1 - (1 - \exp(-s^\delta))^\theta$, $\bar{u} = 1 - u$ and $\bar{v} = 1 - v$. Moreover, for $\theta = 1$ we obtain the Gumbel family.

Bivariate BB8 copula
For parameters $\theta \geq 1$ and $0 < \delta \leq 1$ the bivariate BB8 copula is defined as

\[
C(u, v; \theta, \delta) = \delta^{-1} \left( 1 - \left( 1 - \eta^{-1} \left( 1 - (1 - \delta u)^\theta \right) \left( 1 - (1 - \delta v)^\theta \right) \right)^{1/\delta} \right),
\]

where $\eta = 1 - (1 - \delta)^\theta$. For this copula there is no tail dependence except for $\delta = 1$. In Figure 3.3 we can see some normalized contour plots of a BB8 copula for different values of Kendall’s $\tau$. For small values of Kendall’s $\tau$ the shape looks very similar to the shape of a Gaussian copula.

![Normalized contour plots of BB8 copula with Kendall’s $\tau$ of 0 (left), 0.22 (middle) and 0.68 (right).](image)

Figure 3.3: Normalized contour plots of BB8 copula with Kendall’s $\tau$ of 0 (left), 0.22 (middle) and 0.68 (right).

### 3.4 D-vine copulas
After we have introduced the general concept of copulas we now want to have a closer look on a special class of copulas, the so-called D-vine copulas. Here, the basic idea is to split a multivariate copula density with large dimension $d > 2$ into a product of bivariate copula densities. This approach enables us to estimate very complex and high dimensional dependence structures.
In the following we assume $X = (X_1, \ldots, X_d) \in \mathbb{R}^d$ to be a $d$-dimensional continuously distributed random variable with joint distribution and density function $F(x)$ and $f(x)$, respectively, and marginal distribution and density functions given by $F_1(x_1), \ldots, F_d(x_d)$ and $f_1(x_1), \ldots, f_d(x_d)$, respectively.

Remark 3.7:
For $D \subset \{1, \ldots, d\}$ and $i, j \in \{1, \ldots, d\} \setminus D$ we use the following notation:

- $C_{X_i,X_j;X_D} (\cdot, \cdot; x_D)$ denotes the copula corresponding to the conditional distribution of $(X_i, X_j)$ given $X_D = x_D$. We will use $C_{i,j;D} (\cdot, \cdot; x_D)$ as an abbreviation for the corresponding copula density.

- $F_{X_i;X_D} (\cdot|x_D)$ denotes the conditional distribution of $X_i$ given $X_D = x_D$. We will use $F_{i;D} (\cdot|x_D)$ as an abbreviation for the corresponding conditional density.

- $C_{U_i;U_D} (\cdot|u_D)$ denotes the conditional distribution of $U_i$ given $U_D = u_D$, where $U_i$ is the PIT random variable of $X_i$. Again we will use $C_{i;D} (\cdot|u_D)$ for ease of notation and $c_{i;D} (\cdot|u_D)$ for the corresponding density.

- We use the abbreviations $i : j := i, \ldots, j$ and $x_{i:j} := (x_i, \ldots, x_j)$.

In the following we present so-called pair-copula constructions (PCC), which specify a multivariate distribution only in terms of bivariate building blocks (pair-copulas), yielding a very flexible class of dependence models. In order to keep the models tractable, we will use the commonly used simplifying assumption, that copulas of conditional distributions do not depend on the values of the variables which they are conditioned on, this means $c_{i,j;D} (\cdot, \cdot; x_D) \equiv c_{i,j;D} (\cdot, \cdot)$.

Lemma 3.8 (PCC of D-vine copulas):
Using the notation from above, for example Czado (2010) has shown that one can decompose the joint density $f$ of $X$ into a product of the marginal densities $f_1, \ldots, f_d$ and (conditional) bivariate copula densities as follows

$$f(x_1, \ldots, x_d) = \prod_{k=1}^{d} f_k(x_k) \prod_{i=1}^{d-1} \prod_{j=i+1}^{d} c_{i,j;i+1:j-1}\left(F_{i|i+1:j-1}(x_j|x_{i+1:j-1}), F_{j|i+1:j-1}(x_j|x_{i+1:j-1})\right).$$

This pair-copula construction is called D-vine density with order $X_1 - X_2 - \ldots - X_d$. Moreover, if all margins are uniform we call it a D-vine copula.

Bedford and Cooke (2002) introduced a graph theoretic representation of pair-copula constructions, where each edge of the graph corresponds to a pair-copula. In Figure 3.4 we illustrate this representation for a 5-dimensional ordered D-vine copula.
Example 3.9:
The density function of $X = (X_1, \ldots, X_5)$ which corresponds to Figure 3.4 is given by

$$f(x_1, x_2, x_3, x_4, x_5) = f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_4) \cdot f_5(x_5) \cdot c_{1,2} \cdot c_{2,3} \cdot c_{3,4} \cdot c_{4,5} \cdot c_{1,3,2} \cdot c_{2,4,3} \cdot c_{3,5,4} \cdot c_{1,4,2,3} \cdot c_{2,5,3,4} \cdot c_{1,5,2,3,4}.$$ 

For simplicity we omitted the arguments of the bivariate (conditional) copula densities.

Now, let $U_1 := F_1(X_1), \ldots, U_d := F_d(X_d)$ and $u_1 := F_1(x_1), \ldots, u_d := F_d(x_d)$ be the PIT random variables. Then, using (3.1) we get for the conditional density of $U_1$ given $U_{2:d} = u_{2:d}$ as

$$c_{U_1|U_{2:d}}(u_1|u_{2:d}) = \frac{c_{U_1,\ldots,U_d}(u_1,\ldots,u_d)}{c_{U_{2:d}}(u_2,\ldots,u_d)} \tag{3.6}$$

Using this and (3.1) we get an expression for the conditional density of $X_1|X_{2:d} = x_{2:d}$. It is given by

$$f_{X_1|X_{2:d}}(x_1|x_{2:d}) = \frac{f_{X_1,\ldots,X_d}(x_1,\ldots,x_d)}{f_{X_{2:d}}(x_2,\ldots,x_d)} \tag{3.7}$$

where for simplicity we again omitted the arguments of the bivariate (conditional) copula densities in the second step. We call $c_{X_1|X_{2:d}}$ the conditional copula density of $X_1$ given $X_{2:d}$.
3.4. D-VINE COPULAS

Figure 3.4: Graph theoretic representation of a 5-dimensional D-vine copula with order $X_1 - X_2 - X_3 - X_4 - X_5$. The pair-copulas of the PCC are written above each edge.

Example 3.9 (continued):

In case of our example we get

$$f_{X_1|X_2,\ldots,X_5}(x_1|x_2,\ldots,x_5) = f_1(x_1) \cdot c_{1,2} \cdot c_{1,3,2} \cdot c_{1,4,2,3} \cdot c_{1,5,2,3,4}$$

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

$$\cdot c_{1,3,2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2))$$

$$\cdot c_{1,4,2,3}(F_{1|2,3}(x_1|x_2,x_3), F_{4|2,3}(x_4|x_2,x_3))$$

$$\cdot c_{1,5,2,3,4}(F_{1|2,3,4}(x_1|x_2,x_3,x_4), F_{5|2,3,4}(x_5|x_2,x_3,x_4)).$$

In order to be able to calculate the pair-copulas above, we need to find expressions for the conditional distributions $F_{i|D}(x_i|x_D)$, appearing in (3.1), using only the pair-copulas specified for the D-vine from lower trees.

Lemma 3.11:

For $l \in D$ and $D_{-l} := D \setminus \{l\}$ it holds

$$F_{i|D}(x_i|x_D) = h_{i|D_{-l}}(F_{i|D_{-l}}(x_i|x_{D_{-l}})|F_{l|D_{-l}}(x_l|x_{D_{-l}})).$$
Proof. A proof of the lemma using the chain rule of differentiation can be found in Joe (1997).

Example 3.9 (continued):
Using this, the first argument of $c_{1,4;2,3}$ in our example can be rewritten as

$$F_{1|2,3}(x_1|x_2,x_3) = h_{1|3;2}\left(F_{1|2}(x_1|x_2)|F_{3|2}(x_3|x_2)\right)$$

$$= h_{1|3;2}\left(h_{1|2}(F_1(x_1)|F_2(x_2))|h_{3|2}(F_3(x_3)|F_2(x_2))\right).$$

3.5 R-vine copulas

In the previous section we introduced D-vine copulas. Now, we generalize the construction principle that was used for D-vines in order to get a broader class of pair-copula constructions the so-called regular vines (R-vines). Therefore, we need some background from graph theory which we take from Diestel (2006). Our expositions are based on Scherer and Mai (2012), where one can also find more detailed explanations and examples.

Definition 3.12 (Graph, node, edge, degree):
A graph $G = (N,E)$ is defined as a pair of sets where $E \subseteq \{\{x,y\} : x,y \in N\}$. Moreover, the elements of $E$ are called edges of the graph $G$ and $N$ are the nodes of the graph. The number of neighbors of a node $v \in N$ is called the degree of $v$ and will be denoted by $\delta(v)$.

Definition 3.13 (Path, cycle):
A path is a graph $P = (N,E)$ with $N = \{v_0,v_1,\ldots,v_k\}$ for the node set and edges $E = \{\{v_0,v_1\},\{v_1,v_2\},\ldots,\{v_{k-1},v_k\}\}$. If $v_0 = v_k$ holds we call $P$ a cycle.

Now, we can give a characterization of trees the most important graphs for pair-copula constructions. Trees are connected graphs that do not contain any cycles. Let $G \pm e$ denote a graph where edge $e$ is additional/removed.

Theorem 3.14 (Characterization of trees):
For a graph $T = (N,E)$ the following statements are equivalent:

(i) $T$ is a tree.

(ii) Any two nodes of $T$ are linked by a unique path in $T$.

(iii) $T$ is minimally connected, this means $T$ is connected but $T - e$ is disconnected for every edge $e \in E$.

(iv) $T$ is maximally acyclic, this means $T$ contains no cycle but $T + \{x,y\}$ does for any two non-adjacent nodes $x, y \in N$.

Using this notation we can now define R-vine tree sequences.
Definition 3.15 (Regular vine (R-vine) tree sequence):
We call a set $\mathcal{V} = (T_1, \ldots, T_{d-1})$ a **regular vine tree sequence** on $d$ elements if:

1. $T_1$ is a tree with node set $N_1 = \{1, \ldots, d\}$ and edge set $E_1$.

2. For $j \geq 1$, $T_j$ is a tree with node set $N_j = E_{j-1}$ and edge set $E_j$.

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

Property (3) is called **proximity condition**. For an edge $e$ connecting $a$ and $b$ in tree $T_j$, $j \geq 2$, it ensures that $a$ and $b$ must share a common node in tree $T_{j-1}$.

An example of an R-vine tree sequence on five elements is given in Figure 3.5

![Figure 3.5: Exemplary R-vine tree sequence on five elements.](image)

Remark 3.16:
Using this general definition we call $\mathcal{V}$ a

- **D-vine tree sequence** if for each node $n \in N_i$ we have $|\{e \in E_i|n \in e\}| \leq 2$.

- **C-vine tree sequence** if in each tree $T_i$ there is one node $n \in N$ that is connected to all other nodes, this means $\delta(n) = d - i$.

So far, we were only considering graph theoretic objects that do not include any stochastic component. The following definition will now connect these graph theoretic objects to probability theory.
Definition 3.17 (Regular vine distribution):
For a d-dimensional random vector $X = (X_1, \ldots, X_d)$ we call the joint distribution $F$ a **regular vine distribution** if we can find a triplet $(F, V, B)$ such that:

1.) **Marginal distributions:** $F = (F_1, \ldots, F_d)$ is a vector of continuous invertible marginal distribution functions which represent the marginal distribution functions of $X_1, \ldots, X_d$.

2.) **Regular vine tree sequence:** $V$ is an R-vine tree sequence on $d$ elements.

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

4.) **Connection between tree sequence and bivariate (conditional) copulas:** For each $e \in E_i$, $i = 1, \ldots, d - 1$, $e = \{a, b\}$, $C_e$ is the corresponding copula to the conditional distribution of $X_{e,a}$ and $X_{e,b}$ given $X_{D,e} = x_{D,e}$. Moreover, $C_e(.,.)$ does not depend on the value of $x_{D,e}$.

In the special case where all margins are uniformly distributed on $[0, 1]$ we call $F$ a **regular vine copula**.

Another way to represent regular vines is via a matrix notation as introduced by Kurowicka (2009).

Definition 3.18 (Regular vine matrix):
Let $M$ be an upper triangular matrix with entries $m_{i,j}$, $i < j$, where for each entry we allow $m_{i,j} \in \{1, \ldots, d\}$. We call $M$ a **regular vine matrix** if it fulfills:

1.) $\{m_{1,i}, \ldots, m_{i-1,i}\} \subset \{m_{1,j}, \ldots, m_{j,j}\}$ for $1 \leq i \leq j \leq d$. This means, the entries of a selected column are also contained in all columns right of that column.

2.) $m_{i,i} \notin \{m_{1,i-1}, \ldots, m_{i-1,i-1}\}$. This means, the diagonal entry of a column is not contained in any column further to the left.

3.) For $i = 3, \ldots, d$ and $k = 1, \ldots, i-1$ there exist $(j, \ell)$ with $j < i$ and $\ell < j$ such that

$$\begin{align*}
\{m_{k,i}, \{m_{1,i}, \ldots, m_{k-1,i}\}\} &= \{m_{j,j}, \{m_{1,j}, \ldots, m_{\ell,j}\}\} \quad \text{or} \\
\{m_{k,i}, \{m_{1,i}, \ldots, m_{k-1,i}\}\} &= \{m_{\ell,j}, \{m_{1,j}, \ldots, m_{\ell-1,j}, m_{j,j}\}\}.
\end{align*}$$

After having defined R-vine copulas we will now present a procedure which allows us to sample from R-vine copulas. Therefore, we first of all note that in order to obtain a random sample $(x_1, \ldots, x_d)$ from the joint distribution function $F_{1:d}$ with
marginal distribution functions $F_1, \ldots, F_d$ we can do the following:

**First:** Sample $u_1, \ldots, u_d \sim \text{Unif}[0; 1]$.

**Then:**

\[
\begin{align*}
    x_1 &:= F_1^{-1}(u_1), \\
x_2 &:= F_{2|1}^{-1}(u_2|x_1), \\
    &\vdots \\
x_d &:= F_{d|d-1,\ldots,1}(u_d|x_{d-1}, \ldots, x_1).
\end{align*}
\]

Dißmann (2010) was the first to adapt this general methodology in order to be able to sample from general R-vines, using the recursion from Lemma 3.11 and the definition of h-functions in (3.8) and (3.9). A detailed algorithm using the matrix notation from Definition 3.18 can be found in Scherer and Mai (2012).

### 3.6 D-vine quantile regression

So far, we were always considering $d$-dimensional random vectors $X = (X_1, \ldots, X_d)$. Now, as we move to a regression setup we use $(d+1)$-dimensional random vectors where the first component is our response denoted by $Y$, this means we are considering random vectors $(Y, X) = (Y, X_1, \ldots, X_d)$. Moreover, when transforming to the copula scale we will use $(V, U) = (V, U_1, \ldots, U_d)$. This means, that when we refer to equations from the previous sections one has to adopt for the notation. In detail this means one has to consider the $(d+1)$-dimensional case and use the transformation

\[
\begin{align*}
    X_1 &\to Y, \quad X_2 \to X_1, \ldots, X_{d+1} \to X_d \quad \text{and} \\
    U_1 &\to V, \quad U_2 \to U_1, \ldots, U_{d+1} \to U_d.
\end{align*}
\]

In this section we present the concept of D-vine quantile regression which was introduced by Kraus and Czado (2017). This also includes a method for the estimation of a multivariate copula $C_{V,U_1,\ldots,U_d}$ by a flexible model which is able to capture asymmetric dependencies, heavy tails and tail dependencies between the variables. The results will be used in Chapter 4 where we introduce a new method for mean regression based on D-vine copulas.

The goal of quantile regression in general is to estimate the **conditional quantile function**

\[
q_\alpha(x_1, \ldots, x_d) := F_{Y|X_1,\ldots,X_d}(\alpha|x_1, \ldots, x_d),
\]

where $Y$ is a response variable and $X_1, \ldots, X_d$ are covariates.

For D-vine quantile regression Kraus and Czado (2017) assumed that $Y \sim F_Y$ and $X_j \sim F_j$ for $j \in \{1, \ldots, d\}$. In addition, they used the probability integral transform $V := F_Y(Y)$ and $U_j := F_j(X_j)$ with corresponding PIT values $v := F_Y(y)$ and $u_j := F_j(x_j)$. Using this notation an estimator of the conditional quantile function is given by

\[
\hat{q}_\alpha(x_1, \ldots, x_d) := \hat{F}_Y^{-1}\left(\hat{C}_{V|U_1,\ldots,U_d}^{-1}(\alpha|\hat{u}_1, \ldots, \hat{u}_d)\right),
\]

(3.10)
where $\hat{F}_Y^{-1}$ is an estimate of the inverse marginal distribution function of $Y$, $\hat{C}_{V|U_1,\ldots,U_d}^{-1}$ is an estimate of the conditional copula quantile function and $\hat{u}_j := \hat{F}_j(x_j)$ are the estimated PIT values of $x_j$, the so-called pseudo copula data.

For the estimation of $\hat{F}_Y$ and the $\hat{F}_j$ there are many well established methods like for example kernel density estimation (cf. Section 2.4). Therefore, Kraus and Czado (2017) focused on the question of how to estimate the multivariate copula $C_{V|U_1,\ldots,U_d}$ and the conditional quantile function $\hat{C}_{V|U_1,\ldots,U_d}^{-1}(\alpha|\hat{u}_1,\ldots,\hat{u}_d)$. They suggested to fit a D-vine copula to the multivariate random vector $(V, U_1, \ldots, U_d)$ such that $V$ is the first node in the first tree and the order of the D-vine is given by $V - U_{\pi(1)} - \ldots - U_{\pi(d')}$, where $\{\pi(1), \ldots, \pi(d')\}$ is a permutation of $\{1, \ldots, d\}$ chosen by an automatic forward covariate selection algorithm and $d' \leq d$ is the number of used covariates. This approach results in a flexible D-vine copula as each of the bivariate copulas in the pair-copula construction can be modeled separately.

For the estimation of $q_\alpha(x)$ a two step estimation procedure is presented which follows the inference for margins method of Joe and Xu (1996). Assume that we are given $n$ independent identically distributed observations of the random vector $(Y, X_1, \ldots, X_d)$. Then, $q_\alpha(x_1, \ldots, x_d)$ can be estimated in the following two steps:

### Step one: estimation of the marginals

Kraus and Czado (2017) decided to fit the marginals nonparametrically as Noh et al. (2013) point out that a misspecified parametric estimator might be biased and inconsistent. As we need the inverse of the estimated marginals for the quantile prediction, using the discrete valued empirical cumulative distribution function would not be a good choice. Hence, a kernel density estimator, as defined in Section 2.4, is used. This yields estimators $\hat{F}_Y$ and $\hat{F}_j$ for the marginal distributions which are then applied to the observed data in order to get pseudo copula data $\hat{v}_i := \hat{F}_Y(y_i)$ and $\hat{u}_{ij} := \hat{F}_j(x_{ij})$ for $j \in \{1, \ldots, d\}$ and $i \in \{1, \ldots, n\}$. As a result, we have an approximately i.i.d. sample $\hat{v} = (\hat{v}_i)_{i=1,\ldots,n}$ and $\hat{u} = (u_{i1}, \ldots, u_{id})_{i=1,\ldots,n}$ from the PIT random vector $(V, U_1, \ldots, U_d)$ which can now be used to estimate the D-vine copula in the second step.

### Step two: estimation of the D-vine

In order to be able to calculate $\hat{C}_{V|U_1,\ldots,U_d}^{-1}(\alpha|\hat{u}_1,\ldots,\hat{u}_d)$ Kraus and Czado (2017) present an algorithm which fits a D-vine of order $V - U_{\pi(1)} - \ldots - U_{\pi(d')}$ to the pseudo copula data, where $d' \leq d$. In general the ordering $\pi = \{\pi(1), \ldots, \pi(d')\}$ can be chosen arbitrarily. However, the ordering influences the explanatory power of the resulting model. As comparing all $d!$ possible orderings is infeasible in high-dimensional setups, Kraus and Czado (2017) proposed a new algorithm that chooses the order of the D-vine via an automatic forward covariate selection such that the explanatory power of the model is maximized. This is measured by the conditional
3.7. JITTERING FOR DISCRETE VARIABLES

log-likelihood defined as

\[ \text{cll}(\pi; \hat{v}, \hat{u}) := \sum_{i=1}^{n} \ln [c_{V|U}(\hat{v}_i|\hat{u}_i; \pi)], \tag{3.11} \]

where \( \pi \) is the ordering of the D-vine, \( c_{V|U} \) the conditional copula density as defined in (3.6) and \((\hat{v}, \hat{u})\) are the observed pseudo copula data.

We start with a model using zero covariates and then the algorithm adds in each step the covariate to the model that improves the models conditional log-likelihood the most. In the beginning of step \( k \) of the algorithm the current optimal D-vine contains \( k - 1 \) predictors. Now, for each of the covariates \( U_j \) that haven’t been chosen so far, we fit all pair-copulas that are needed to extend the D-vine with order \( V - U_{\pi(1)} - \ldots - U_{\pi(k-1)} \) to a D-vine of order \( V - U_{\pi(1)} - \ldots - U_{\pi(k-1)} - U_j \). The pair-copulas needed for this extension are

\[ c_{U_{\pi(1)}, U_{\pi(2)}, \ldots, U_{\pi(k-1)}}, \ldots, c_{U_{\pi(1)}, U_{\pi(2)}, \ldots, U_{\pi(k-1)}}, \text{ and } c_{V, U_j | U_{\pi(1)}, \ldots, U_{\pi(k-1)}}. \]

Then, the covariate that improves the conditional log-likelihood the most, is added to the current model. A detailed explanation of the algorithm can be found in Appendix A of Kraus and Czado (2017).

Moreover, if one is interested in more parsimonious models one can use the AIC-corrected conditional log-likelihood given by

\[ \text{cll}^{\text{AIC}}(\pi; \hat{v}, \hat{u}) := -2\text{cll}(\pi; \hat{v}, \hat{u}) + 2|\hat{\Theta}|, \tag{3.12} \]

where \( |\hat{\Theta}| \) denotes the number of parameters which were used for the construction of the D-vine. This means in a parametric setup it is just the number of parameters of all bivariate pair-copulas. In case the pair-copulas are estimated nonparametrically \( |\hat{\Theta}| \) is the so-called effective degree of freedom (EDF). A detailed explanation on how to calculate the EDF is given in Section 7.6 of Hastie et al. (2001).

D-vine quantile regression has been implemented for the programming language R (R Core Team, 2018) in the package `vinereg` (Nagler, 2018b) for both the cll and cll^{AIC}. According to Kraus and Czado (2017) using the AIC-corrected conditional log-likelihood yields better results in the sense of disregarding unimportant variables and including influential variables in the model.

### 3.7 Jittering for discrete variables

So far, all the theory on copulas was based on the assumption of only having continuous random variables. As in many regression problems also discrete variables taking only integer values are involved, we can use the jittering approach presented in Nagler (2018a) in order to treat discrete data. Therefore, suppose that any discrete
random variable $Z$ is supported on a set $\Omega_Z \subseteq \mathbb{Z}$. Now, assume that we have a discrete random vector $\mathbf{Z}$ with joint density function $f_{\mathbf{Z}}(\mathbf{z}) = \mathbb{P}(\mathbf{Z} = \mathbf{z})$ and a continuous random vector $\mathbf{X}$ with joint density function $f_{\mathbf{X}}$. Then, the random vector $(\mathbf{Z}, \mathbf{X}) \in \mathbb{Z}^p \times \mathbb{R}^q$ of mixed types has joint density
\[
f_{\mathbf{Z}, \mathbf{X}} = \frac{\partial^n}{\partial x_1 \ldots \partial x_q} \mathbb{P}(\mathbf{Z} = \mathbf{z}, \mathbf{X} \leq \mathbf{x}).
\]

Now, suppose that $\varepsilon$ is a continuous random variable with density function $f_\varepsilon$ and satisfying assumptions

(A1.) $f_\varepsilon(0) = 1$ and

(A2.) there exists $\gamma \in (0, 1)$ such that $f_\varepsilon(x) = 0$ for all $x \in \mathbb{R}^p \setminus [-\gamma, \gamma]^p$.

Then, we can define the discrete-continuous convolution of the density $f_{\mathbf{Z}, \mathbf{X}}$ as
\[
f_{\mathbf{Z}+\varepsilon, \mathbf{X}}(\mathbf{z}, \mathbf{x}) := \sum_{\mathbf{z}' \in \mathbb{Z}^p} f_{\mathbf{Z}, \mathbf{X}}(\mathbf{z}', \mathbf{x}) f_\varepsilon(\mathbf{z} - \mathbf{z}') \quad \text{for almost all } (\mathbf{z}, \mathbf{x}) \in \mathbb{R}^{p+q}.
\]

Nagler (2018a) showed that for any random variable $\varepsilon$ fulfilling (A1.) and (A2.) it holds
\[
f_{\mathbf{Z}+\varepsilon, \mathbf{X}}(\mathbf{z}, \mathbf{x}) = f_{\mathbf{Z}, \mathbf{X}}(\mathbf{z}, \mathbf{x}) \quad \text{for all } (\mathbf{z}, \mathbf{x}) \in \mathbb{Z}^p \times \mathbb{R}^q
\]
and we call this method jittering.

**Example 3.19:**
The $p$-dimensional uniform distribution on $[0.5, 0.5]^p$, this means
\[
f_\varepsilon(x) := \begin{cases} 1, & \text{for } x \in [0.5, 0.5]^p, \\ 0, & \text{else}, \end{cases}
\]
fulfills conditions (A1.) and (A2.).

Moreover, the **jittering kernel density estimator** of $f_{\mathbf{Z}, \mathbf{X}}$ is given by
\[
\tilde{f}(\mathbf{z}, \mathbf{x}) = \frac{1}{nb_n} \sum_{i=1}^n K\left\{ \frac{(\mathbf{Z}_i + \varepsilon_i, \mathbf{X}_i) - (\mathbf{z}, \mathbf{x})}{b_n} \right\},
\]
where $b_n > 0$ and $K$ is a symmetric, multivariate density function. If $\varepsilon_i = 0$ for all $i \in \{1, \ldots, n\}$ we get the classical kernel density estimator of Parzen (1962) as introduced in Section 2.4. These results allow us to also use discrete variables in copula modelling and in particular in D-vine quantile regression. The jittering kernel density estimator using the density given in Example 3.19 is implemented in the R-package `kde1d` (Nagler and Vatter, 2018a).

**Remark 3.20:**
When using jittered versions of discrete random variables in copula modelling, one has to use a nonparametric approach for the estimation of copulas as parametric modelling does not work in this case.
Chapter 4

D-vine mean regression

Building a good regression model requires a careful selection of covariates, analysis of dependencies between the covariates and the consideration of interactions. This can in general be a very tough task if many different covariates with unknown dependence structure are involved. Therefore, in this chapter we introduce a new method for mean regression which is based on D-vine copulas and uses the automatic forward covariate selection algorithm introduced by Kraus and Czado (2017) which has been presented in Section 3.6. This yields a new flexible way of building a mean regression model.

4.1 Preparations

We start by introducing some notation. Throughout this chapter $Y$ is a one-dimensional random variable serving as response in our regression setup and $X = (X_1, \ldots, X_d)$, $d \geq 1$, is the vector of covariates and $x = (x_1, \ldots, x_d)$ is a realization of $X$. Now, given $n$ independent identically distributed observations $(y_i, x_{i1}, \ldots, x_{id})_{i \in \{1, \ldots, n\}}$ of the random vector $(Y, X_1, \ldots, X_d)$ we want to find estimates of the joint copula density $c_{Y, X}$ of $Y$ and $X$ and the conditional copula density $c_{Y \mid X}$ of $Y$ given $X$. Therefore, we first apply the PIT $V = F_Y(Y)$, $U_j = F_j(X_j)$, $v_i = F_Y(y_i)$ and $u_{ij} = F_i(x_{ij})$, for $j \in \{1, \ldots, d\}$ and $i \in \{1, \ldots, n\}$. Afterwards, we use the automatic forward covariate selection algorithm presented in Section 3.6 which yields the D-vine copula that maximizes the stepwise conditional likelihood given in (3.11). As a result, we get an estimate of the copula $C_{V, U}$ including estimates for the joint copula density of $V$ and $U$ and the conditional copula density of $V \mid U = u$ which are denoted by

$$\hat{c}_{V, U}(v, u_1, \ldots, u_d) \quad \text{and} \quad \hat{c}_{V \mid U}(v \mid u_1, \ldots, u_d),$$

respectively.

Then, using Theorem 3.3 we get

$$\hat{c}_{Y, X}(\hat{F}_Y(y), \hat{F}_1(x_1), \ldots, \hat{F}_d(x_d)) = \hat{c}_{V, U}(\hat{v}, \hat{u}_1, \ldots, \hat{u}_d) \quad \text{and} \quad (4.1)$$

$$\hat{c}_{Y \mid X}(\hat{F}_Y(y) \mid \hat{F}_1(x_1), \ldots, \hat{F}_d(x_d)) = \hat{c}_{V \mid U}(\hat{v} \mid \hat{u}_1, \ldots, \hat{u}_d) \quad (4.2)$$

as estimates for the joint copula density of $Y$ and $X$ and the conditional copula density of $V \mid U = u$, respectively.
CHAPTER 4. D-VINE MEAN REGRESSION

4.2 Estimation of the conditional mean $\mathbb{E}[Y|\mathbf{x}]$

As in any mean regression problem, the goal is to find an estimate of the conditional expectation $\mathbb{E}[Y|\mathbf{x}]$. In the following we will present five different methods for the estimation of this conditional expectation based on a D-vine copula, using the estimates for the joint copula density $c_{Y,X}$ and the conditional copula density $c_{Y|X}$ as given in (4.1) and (4.2), respectively.

4.2.1 Integration over quantiles (IOQ)

We know that we can write the conditional expectation of $Y$ given $X = \mathbf{x}$ as the integral over the conditional quantiles $q_\alpha(x_1,\ldots,x_d) := F_{Y|X=\mathbf{x}}^{-1}(\alpha|x_1,\ldots,x_d)$, this means

$$\mathbb{E}[Y|\mathbf{x}] = \int_0^1 q_\alpha(x_1,\ldots,x_d)d\alpha.$$ 

In order to get an estimate of this based on a D-vine copula, we use D-vine quantile regression as presented in Section 3.6 and replace the conditional quantile function $q_\alpha(x_1,\ldots,x_d)$ by its estimator $\hat{q}_\alpha(x_1,\ldots,x_d)$ given in (3.10), yielding

$$\hat{\mathbb{E}}_{IOQ}[Y|\mathbf{x}] := \int_0^1 \hat{q}_\alpha(x_1,\ldots,x_d)d\alpha \quad (4.3)$$

as an estimator for the conditional mean. The calculation of the integral is done via numerical integration as explained in Section 2.2.

This approach is implemented in the R-function ‘predict.mean.IOQ’ which can be found in Appendix A.1.

4.2.2 Calculating the estimating equation (CEE)

The second approach is based on the usage of an estimating equation which has been derived by Nagler and Vatter (2018b). They developed a very general class of estimating equations, where in the special case of a mean regression an estimator of the conditional mean is given by

$$\hat{\mathbb{E}}[Y|\mathbf{x}] = \frac{\sum_{i=1}^n y_i \cdot c_{Y,X}(F_Y(y_i), F_1(x_1), \ldots, F_d(x_d))}{\sum_{i=1}^n c_{Y,X}(F_Y(y_i), F_1(x_1), \ldots, F_d(x_d))}.$$ 

Plugging in the observed values $y_1,\ldots,y_n$ and the estimator given in (4.1) we end up with an estimator for the conditional mean given by

$$\hat{\mathbb{E}}_{CEE}[Y|\mathbf{x}] := \frac{\sum_{i=1}^n y_i \cdot \hat{c}_{Y,X}(\hat{F}_Y(y_i), \hat{F}_1(x_1), \ldots, \hat{F}_d(x_d))}{\sum_{i=1}^n \hat{c}_{Y,X}(F_Y(y_i), F_1(x_1), \ldots, F_d(x_d))}. \quad (4.4)$$

We implemented this estimator in the R-function ‘predict.mean.CEE’ which can be found in Appendix A.2.
4.2. ESTIMATION OF THE CONDITIONAL MEAN

4.2.3 Monte Carlo simulation (MCS)

As a third method we estimate the conditional mean $E[Y|X]$ via a Monte Carlo simulation. Therefore, first of all we rewrite the conditional expectation in terms of an unconditional expectation using (3.7). The conditional expectation of $Y$ given $x$ is defined as

$$E[Y|X] = \int_{\mathbb{R}} y \cdot f_Y(y|x_1, \ldots, x_d) dy.$$  \hspace{1cm} (4.5)

Moreover, in (3.7) we have seen that we can rewrite the conditional density function $f_{Y|X}$ of $Y$ given $X$ as

$$f_{Y|X}(y|x_1, \ldots, x_d) = f_Y(y) \cdot c_{Y|X}(F_Y(y)|F_1(x_1), \ldots, F_d(x_d)).$$

Plugging this into (4.5) we get

$$E[Y|X] = \int_{\mathbb{R}} y \cdot c_{Y|X}(F_Y(y)|F_1(x_1), \ldots, F_d(x_d)) \cdot f_Y(y) dy$$

$$= E\left[ Y \cdot c_{Y|X}(F_Y(Y)|F_1(x_1), \ldots, F_d(x_d)) \right].$$  \hspace{1cm} (4.6)

Now, this expectation can be estimated using Monte Carlo simulation. For this we need to draw a random sample of $Y$ and evaluate the conditional copula density $c_{Y|X}$. An estimate of the latter is given in (4.2). In order to get a random sample of $Y$ we first use kernel density estimation to get an estimator $\hat{F}_Y(Y)$ of $F_Y(Y)$. Afterwards, we can draw a random sample of $Y$ by applying the inversion method for sampling as described in Section 2.3.

Combining all these steps, we end up with an estimate for the conditional mean which is calculated as follows:

1. Use kernel density estimation to get estimates $\hat{F}_Y, \hat{F}_{X_1}, \ldots, \hat{F}_{X_d}$ of $F_Y, F_{X_1}, \ldots, F_{X_d}$, respectively.
2. Draw an i.i.d. sample $(y^{(1)}, \ldots, y^{(m)})$ from $\hat{F}_Y$.
3. $\hat{E}_{MCS}[Y|X] := \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \cdot c_{Y|X}(\hat{F}_Y(y^{(i)})|\hat{F}_1(x_1), \ldots, \hat{F}_d(x_d)).$  \hspace{1cm} (4.7)

Due to the fact that the quality of this estimator highly depends on the size $m$ of the Monte Carlo sample, we use it as an input parameter in our implementation of this method which is given by the R-function ‘predict.mean.MCS’ in Appendix A.3. In the simulation studies in Chapter 7 we will assess how large the sample size $m$ should be in order to get good results.
4.2.4 Integration over conditional (copula) density (ICD)

A fourth possibility to estimate $\mathbb{E}[Y|x]$ is via numerical integration. As we have already seen in (4.6) we can write the conditional expectation of $Y$ given $x$ as the integral

$$
\mathbb{E}[Y|x] = \int_{\mathbb{R}} y \cdot f_Y(y) \cdot c_{Y|X_1,...,X_d}(F_Y(y)|F_1(x_1),\ldots,F_d(x_d)) \, dy. \tag{4.8}
$$

As in Method MCS we use the estimator of the conditional copula density $c_{Y|X}$ given in (4.2). Moreover, the density function $f_Y(y)$ is estimated using kernel density estimation giving a pointwise estimate $\hat{f}_Y(y)$. Plugging this into (4.8) we get

$$
\hat{\mathbb{E}}_{ICD}[Y|x] := \int_{\mathbb{R}} y \cdot \hat{f}_Y(y) \cdot \hat{c}_{Y|X}(\hat{F}_Y(y)|\hat{F}_1(x_1),\ldots,\hat{F}_d(x_d)) \, dy. \tag{4.9}
$$

We use numerical integration as introduced in 2.2 in order to calculate this integral. The approach is implemented in the R-function ‘predict.mean.ICD’ which can be found in Appendix A.4.

4.2.5 Simplified quantile integration (SQI)

The fifth and last method is a simplification of Method IOQ, where we use the approximation

$$
\int_0^1 \hat{q}_\alpha(x_1,\ldots,x_d) \, d\alpha \approx \frac{1}{n_{quant}} \sum_{i=1}^{n_{quant}} \hat{q}_{\alpha_i}(x_1,\ldots,x_d),
$$

for some $n_{quant} \in \mathbb{N}$ and a sequence of equidistant points $\alpha_1,\ldots,\alpha_{n_{quant}}$ on $[0,1]$ which is given by

$$
\alpha_i := \frac{i}{n_{quant}} + 1 \quad \text{for } i \in \{1,\ldots, n_{quant}\}.
$$

Here, $\hat{q}_\alpha(x_1,\ldots,x_d)$ is the estimate of the conditional quantile function coming from D-vine quantile regression as given in (3.10). We can then define an estimator of the conditional mean as

$$
\hat{\mathbb{E}}_{SQI}[Y|x] := \frac{1}{n_{quant}} \sum_{i=1}^{n_{quant}} \hat{q}_{\alpha_i}(x_1,\ldots,x_d). \tag{4.10}
$$

For $n_{quant} = 10$ this approach is already implemented in the R-package vinereg (Nagler, 2018b). In our R-function ’predict.mean.SQI’ which can be found in Appendix A.5 we take $n_{quant}$ as an input parameter and can therefore allow for any value $n_{quant} \in \mathbb{N}$. In our simulation studies in Chapter 8 we will compare the quality of the estimator for different values of $n_{quant}$. 
4.3 Remarks on the implementation

All our implementations, which can be found in Appendix A, are based on the R-package \texttt{vinereg} (Nagler, 2018b) which implements the approach of Kraus and Czado (2017). In contrast to the original work of Kraus and Czado (2017) where only parametric copula families were considered for the estimation of the bivariate pair-copulas in the pair-copula construction, Nagler (2018b) also implemented a nonparametric version. This means, in the nonparametric version all pair-copulas are estimated nonparametrically. Moreover, for the estimation of kernel densities we used the package \texttt{kde1d} (Nagler and Vatter, 2018a) which also implements the jittering kernel density estimator which was introduced in Section 3.7.

We will assess the quality of the Methods IOQ, CEE, MCS, ICD and SQI for both, the parametric and the nonparametric versions in the simulation studies from Chapter 6 to Chapter 11.
Chapter 5

Gradient boosting with regression trees

In the previous chapter we have introduced a novel method for fitting a mean regression model based on D-vine copulas. Now, we present gradient boosting with regression trees, a popular regression method which yields very good results in many different setups. Then, in Chapters 6 to 11 we will compare both methods in a couple of scenarios in order to assess the quality of our new developed regression method. Our explanations in this chapter are based on Hastie et al. (2001) if not stated otherwise.

5.1 Introduction to regression trees

Before we start with gradient boosting, we will first introduce regression trees according to the CART framework of Breiman et al. (1984) as they will be needed in the gradient boosting methodology. Therefore, we consider a regression problem where we have a continuous response $Y$ and continuous covariates $X = (X_1, \ldots, X_d)$ which are elements of the covariate space $\mathcal{X} \subseteq \mathbb{R}^d$. In order to model $Y$ we want to partition the space of covariates $\mathcal{X}$ into $K$ disjoint regions

$$R_1, \ldots, R_K,$$

where

$$\bigcup_{k=1}^{K} R_k = \mathcal{X} \quad \text{and} \quad R_i \cap R_j = \emptyset \quad \text{for all} \quad i \neq j.$$

The number $K$ of disjoint regions is also referred to as the size of the tree. Within each of the $K$ regions an estimate of the conditional expectation $\mathbb{E}[Y|X]$ is given by a constant value $c_k$ for all $x \in R_k$. The partitioning starts by splitting the space into two regions $\{X \in \mathcal{X}|X_i \leq t_i\}$ and $\{X \in \mathcal{X}|X_i > t_i\}$ for some $i \in \{1, \ldots, d\}$ and $t_i \in \mathbb{R}$, where $X_i$ is the so-called splitting variable and $t_i$ is the split point. Afterwards, we split one or both of these two regions into two more regions and then continue this process until some stopping criterion is reached. For the ease of notation, we will in the following use $\{X_i \leq t_i\}$ instead of $\{X \in \mathcal{X}|X_i \leq t_i\}$ for the specification of the regions.
Now, we want to create an algorithm that automatically grows the tree by choosing the splitting variables and the split points. Therefore, we assume to be given a set of data \((y_i, \mathbf{x}_i)\) for \(i \in \{1, \ldots, n\}\), with \(\mathbf{x}_i = (x_{i1}, \ldots, x_{id})\).

Given a partition of \(X\) into \(K\) disjoint regions \(R_1, \ldots, R_k\) an estimate of \(y_i\) is given by

\[
\hat{y}_i := T(\mathbf{x}_i; \mathbf{R}, \mathbf{c}) := \sum_{k=1}^{K} \hat{c}_k \mathbb{1}\{\mathbf{x}_i \in R_k\} \quad \text{for} \quad i \in \{1, \ldots, n\},
\]

where \(\mathbf{R} := \{R_1, \ldots, R_K\}\) is the set of regions and \(\mathbf{c} := (\hat{c}_1, \ldots, \hat{c}_K)\) is the vector of constant estimates within each region. The goal of the algorithm is to find the solution of

\[
\min_{\mathbf{R}, \mathbf{c}} \sum_{i=1}^{n} \left[ y_i - T(\mathbf{x}_i; \mathbf{R}, \mathbf{c}) \right]^2 = \min_{\mathbf{R}, \mathbf{c}} \sum_{i=1}^{n} \left[ y_i - \sum_{k=1}^{K} \hat{c}_k \mathbb{1}\{\mathbf{x}_i \in R_k\} \right]^2.
\]

If we define \(N_k := |\{\mathbf{x}_i \in R_k\}|\) to be the number of data points in region \(R_k\), then for given regions \(R_1, \ldots, R_K\) it is easy to see that the optimal estimates \(\hat{c}_1, \ldots, \hat{c}_K\) are given by the average of \(y_i\) in each region, this means

\[
\hat{c}_k := \min_{\mathbf{c}} \sum_{\mathbf{x}_i \in R_k} (y_i - c)^2 = \frac{1}{N_k} \sum_{i=1}^{n} y_i \mathbb{1}\{\mathbf{x}_i \in R_k\} \quad \text{for} \quad k \in \{1, \ldots, K\}.
\]

In contrast, finding the optimal partitioning \(R_1, \ldots, R_K\) is in general computationally infeasible. Therefore, Hastie et al. (2001) present a greedy algorithm that starts with all the available data and defines for a splitting variable \(j\) and a split point \(t\) the pair of half-planes

\[
R_1(j, t) := \{X_j \leq t\} \quad \text{and} \quad R_2(j, t) := \{X_j > t\}.
\]

Then, we want to find the splitting variable \(j\) and the split point \(t\) that solve

\[
\min_{j, t} \left[ \min_{c_1} \sum_{\mathbf{x}_i \in R_1(j, t)} (y_i - c_1)^2 + \min_{c_2} \sum_{\mathbf{x}_i \in R_2(j, t)} (y_i - c_2)^2 \right].
\]

As already stated before, it is easy to see that for any \(j\) and \(t\)

\[
\hat{c}_1 = \frac{\sum_{i=1}^{n} y_i \mathbb{1}\{\mathbf{x}_i \in R_1(j, t)\}}{N_1(j, t)} \quad \text{and} \quad \hat{c}_2 = \frac{\sum_{i=1}^{n} y_i \mathbb{1}\{\mathbf{x}_i \in R_2(j, t)\}}{N_2(j, t)}
\]

solve the inner minimization. Moreover, for each splitting variable \(j\), the split point \(t\) can be found quite fast by trying all \(n - 1\) possibilities to split the data into two groups. Hence, by comparing all \(d\) possible split variables we can find the best pair \((j, t)\) which splits the data into two regions. Now, at beginning of step \(k\) of the algorithm we have \(k\) disjoint regions and for each region we try to find the best split \((j, t)\) in the same way as described for the first split. After having found the optimal split in each of the \(k\) regions, the algorithm chooses only the best split and continues with step \(k + 1\).
5.1. INTRODUCTION TO REGRESSION TREES

Obviously, the question arises how large we should grow the tree. Hastie et al. (2001) present several methods for the selection of the optimal tree size $K$, however, for us this will be a tuning parameter during the model selection process, so that we do not need to further consider this question.

The step of finding the optimal split point $t$ for a given splitting variable $j$ can be accelerated by the usage of an approximate algorithm which only allows the split point $t$ to be at one of the empirical percentiles of the splitting variable, this means $t \in \{ \hat{q}_{0.01}(X_j), \ldots, \hat{q}_{0.99}(X_j) \}$, where $\hat{q}_\alpha(X_j)$ is the empirical $\alpha$-quantile of $X_j$. An example of such an algorithm is presented in Chen and Guestrin (2016).

So far, we assumed that all our covariates $X = (X_1, \ldots, X_d)$ are continuous. Now, let us suppose that covariate $X_i$ is discrete for some $i \in \{1, \ldots, d\}$, this means it only takes integer values. Then, the split finding described above works exactly the same way, as we can still divide the data set according to the rule $X_i \leq t$ and $X_i > t$. For categorical covariates we first have to apply dummy variable encoding as described in Section 2.7 and then for each of the resulting binary dummy variables the algorithm can be applied. If the used data set contains many categorical variables the application of dummy variable encoding yields a sparse matrix and Chen and Guestrin (2016) present an algorithm that fits a regression tree a lot faster for this type of data, compared to the procedure presented above.

In order to illustrate how a regression tree grows, we show a short example which is also given in Hastie et al. (2001) and visualized in Figure 5.1.

**Example 5.1:**
Consider two covariates $X_1$ and $X_2$ which both take values on the unit interval, this means the covariate space is given by $\mathcal{X} = [0,1]^2$. Now assume the algorithm chooses the first split to take place at $X_1 = t_1$ for some $t_1 \in (0,1)$. This yields the regions \{ $X_1 \leq t_1$ $\}$ and \{ $X_1 > t_1$ $\}$ . In the next step, the algorithm finds that splitting region \{ $X_1 \leq t_1$ $\}$ at $X_2 = t_2$ with $t_2 \in (0,1)$ improves the estimate the most so that we get the regions \{ $X_1 \leq t_1, X_2 \leq t_2$ $\}$ and \{ $X_1 \leq t_1, X_2 > t_2$ $\}$. Afterwards, the region \{ $X_1 > t_1$ $\}$ is split at $X_1 = t_3$ for some $t_1 < t_3 < 1$, giving us regions \{ $t_1 < X_1 \leq t_3$ $\}$ and \{ $X_1 > t_1, X_1 > t_3$ $\} = \{ X_1 > t_3$ $\}$ as this yields again the highest improvement. In a final step, the algorithm decides to split region \{ $X_1 > t_3$ $\}$ at $X_2 = t_4$ with $t_4 \in (0,1)$, resulting in regions \{ $X_1 > t_3, X_2 \leq t_4$ $\}$ and \{ $X_1 > t_3, X_2 > t_4$ $\}$ . So we end up with the regions

\[
R_1 = \{ X_1 \leq t_1, X_2 \leq t_2 \}, \\
R_2 = \{ X_1 \leq t_1, X_2 > t_2 \}, \\
R_3 = \{ t_1 < X_1 \leq t_3 \}, \\
R_4 = \{ X_1 > t_3, X_2 \leq t_4 \} \text{ and} \\
R_5 = \{ X_1 > t_3, X_2 > t_4 \}. 
\]
Then, for all $x \in [0, 1]^2$ we can get an estimate of $Y$ as

$$\hat{E}[Y|x] := \sum_{k=1}^{5} \hat{c}_k 1\{x \in R_k\},$$

where $\hat{c}_1, \ldots, \hat{c}_5 \in \mathbb{R}$ are the constant estimates given by

$$\hat{c}_k = \frac{1}{N_k} \sum_{i=1}^{n} y_i 1\{x_i \in R_k\} \text{ for } k \in \{1, \ldots, 5\}.$$

We conclude this section by noting that given the set of regions $R := \{R_1, \ldots, R_K\}$ and the corresponding constant estimates $\hat{c} := \{\hat{c}_1, \ldots, \hat{c}_K\}$ our regression tree is uniquely specified by

$$T(x; R, \hat{c}) := \sum_{k=1}^{K} \hat{c}_k 1\{x \in R_k\} \text{ for all } x \in \mathcal{X}.$$ 

In the following we will write $T(x; \Theta)$ instead of $T(x; R, \hat{c})$, this means $\Theta := \{R, \hat{c}\}$.

### 5.2 Main idea of gradient boosting

Now that we have seen the concept of regression trees and how to fit them, we first explain the main idea of gradient boosting with regression trees. Afterwards, we will explain the mathematical details.

The basic idea of gradient boosting is to fit a strong predictive classification or regression model by combining several weak models, so-called base learners. It was
first introduced by Schapire (1990) and a first algorithm was given in Freund and Schapire (1997). In the context of classification, weak models are those which are only slightly better than random guessing, which means they classify more than 50% of the data correctly. In contrast a strong model should classify almost all data correctly, for instance 99%. When talking about regression one cannot just distinguish between correctly and wrongly classified data but rather has to assess the goodness of fit in order to specify weak and strong learners. In the context of mean regression, the goodness of fit is usually measured by the quadratic loss $\sum_{i=1}^{n} \frac{1}{2}(y_i - \hat{y}_i)^2$, where $\hat{y}_i$ is an estimate of the conditional mean $\mathbb{E}[Y|x_i], i = 1, \ldots, n$.

When using gradient boosting for regression trees, we start by fitting an initial weak model $H_0$ which returns the constant value that minimizes the quadratic loss, this means $H_0(x) = \bar{y}$ is the average over all observations. Afterwards, this model is iteratively improved by fitting in each step $m \in \{1, \ldots, M\}$ of the algorithm a regression tree $T(x; \Theta_m)$ to the residuals $r_{mi} := y_i - H_{m-1}(x_i)$ of the current model and then adding the estimates of this tree for the residuals to the current model. This means $H_m(x) = H_{m-1}(x) + T(x; \Theta_m)$. The number of iterations $M$ is a pre-specified tuning parameter. Moreover, within this framework each of the regression trees might only contain a part of the available covariates and data which makes the approach very robust against overfitting.

Before going to the mathematical details, we give a short example that clarifies the procedure.

**Example 5.2:**

*Let us consider a small data set with response variable $Y = \text{‘age’}$ and two binary covariates $X_1 = \text{‘Likes video games’}$ and $X_2 = \text{‘Likes classic music’}. The data set is given in Table 5.1.*

**Table 5.1:** Exemplary data set for Example 5.2 on gradient boosting with regression trees.

<table>
<thead>
<tr>
<th>i</th>
<th>$y_i$=Age</th>
<th>$x_{i1}$=Likes video games</th>
<th>$x_{i2}$=Likes classic music</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>38</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>53</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>64</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Now, we fit an initial model \( H_0(x) = \bar{y} \), yielding a constant estimate for all data points and corresponding residuals \( r_{1i} = y_i - H_0(x_i) = y_i - \bar{y} \) (cf. Table 5.2), where

\[
\bar{y} = \frac{12 + 15 + 20 + 20 + 24 + 38 + 42 + 53 + 64}{9} = 32.
\]

Table 5.2: Initialization of boosting algorithm with residual sum of squares
\( SSR = 2642 \).

<table>
<thead>
<tr>
<th>i</th>
<th>( y_i )=Age</th>
<th>( x_{i1} )=Likes video games</th>
<th>( x_{i2} )=Likes classic music</th>
<th>( H_0(x_i) = \bar{y} )</th>
<th>( r_{1i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-20</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>Yes</td>
<td>Yes</td>
<td>32</td>
<td>-17</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>No</td>
<td>No</td>
<td>32</td>
<td>-12</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-12</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-8</td>
</tr>
<tr>
<td>6</td>
<td>38</td>
<td>No</td>
<td>Yes</td>
<td>32</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>No</td>
<td>Yes</td>
<td>32</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>53</td>
<td>Yes</td>
<td>Yes</td>
<td>32</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>64</td>
<td>No</td>
<td>Yes</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

The next step is fitting a first regression tree \( T(x; \Theta_1) \) to the residuals \( r_{1i} \). Suppose, this tree decides to only split the data according to variable \( X_1 = \text{‘Likes video games’} \). Then, we get

\[
T(x_i; \Theta_1) = c_{11} \cdot 1 \{ x_{i1} = \text{‘Yes’} \} + c_{12} \cdot 1 \{ x_{i1} = \text{‘No’} \}, \quad \text{where}
\]

\[
c_{11} = \frac{-20 - 17 - 12 - 8 + 21}{5} = -7.2 \quad \text{and} \quad c_{12} = \frac{-12 + 6 + 10 + 32}{4} = 9.
\]

So we can now update our model to

\[
H_1(x) = H_0(x) + T(x; \Theta_1)
\]

\[
= \bar{y} + c_{11} \cdot 1 \{ x_{i1} = \text{‘Yes’} \} + c_{12} \cdot 1 \{ x_{i1} = \text{‘No’} \}
\]

\[
= 32 - 7.2 \cdot 1 \{ x_{i1} = \text{‘Yes’} \} + 9 \cdot 1 \{ x_{i1} = \text{‘No’} \},
\]

yielding new residuals \( r_{2i} = y_i - H_1(x_i) \) (cf. Table 5.3).
5.2. MAIN IDEA OF GRADIENT BOOSTING

Table 5.3: Step one of boosting algorithm, decreasing residual sum of squares to $SSR = 2058.8$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$y_i$</th>
<th>$x_{i1}$</th>
<th>$x_{i2}$</th>
<th>$H_0(x_i) = \bar{y}$</th>
<th>$r_{1i}$</th>
<th>$T(x_i; \Theta_1)$</th>
<th>$H_1(x_i) = H_0(x_i) + T(x_i; \Theta_1)$</th>
<th>$r_{2i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-20</td>
<td>-7.20</td>
<td>24.80</td>
<td>-12.80</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>Yes</td>
<td>Yes</td>
<td>32</td>
<td>-17</td>
<td>-7.20</td>
<td>24.80</td>
<td>-9.80</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>No</td>
<td>No</td>
<td>32</td>
<td>-12</td>
<td>9.00</td>
<td>41.00</td>
<td>21.00</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-12</td>
<td>-7.20</td>
<td>24.80</td>
<td>-4.80</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-8</td>
<td>-7.20</td>
<td>24.80</td>
<td>-0.80</td>
</tr>
<tr>
<td>6</td>
<td>38</td>
<td>No</td>
<td>Yes</td>
<td>32</td>
<td>6</td>
<td>9.00</td>
<td>41.00</td>
<td>3.00</td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>No</td>
<td>No</td>
<td>32</td>
<td>10</td>
<td>9.00</td>
<td>41.00</td>
<td>1.00</td>
</tr>
<tr>
<td>8</td>
<td>53</td>
<td>Yes</td>
<td>Yes</td>
<td>32</td>
<td>21</td>
<td>-7.20</td>
<td>24.80</td>
<td>28.20</td>
</tr>
<tr>
<td>9</td>
<td>64</td>
<td>No</td>
<td>Yes</td>
<td>32</td>
<td>32</td>
<td>9.00</td>
<td>41.00</td>
<td>23.00</td>
</tr>
</tbody>
</table>

In the second and last step of this example we fit a regression tree $T(x; \Theta_2)$ to the residuals $r_{2i}$. Assume, that now the data is split according to variable $X_2 = \text{‘Likes classic music’}$. This, gives us

$$T(x_i; \Theta_2) = c_{21} \cdot 1\{x_{i2} = \text{"Yes"}\} + c_{22} \cdot 1\{x_{i2} = \text{"No"}\}, \text{ where}$$

$$c_{21} = \frac{-9.8 - 3 + 28.2 + 23}{4} = 9.6 \text{ and } c_{22} = \frac{-12.8 - 21 - 4.8 - 0.8 + 1}{5} = -7.68.$$

Hence, our model is updated to

$$H_2(x) = H_1(x) + T(x; \Theta_2)$$

$$H_2(x) = H_0(x) + T(x; \Theta_1) + T(x; \Theta_2)$$

$$= \bar{y} + c_{11} \cdot 1\{x_{i1} = \text{"Yes"}\} + c_{12} \cdot 1\{x_{i1} = \text{"No"}\} +$$

$$c_{21} \cdot 1\{x_{i2} = \text{"Yes"}\} + c_{22} \cdot 1\{x_{i2} = \text{"No"}\}$$

$$= 32 - 7.2 \cdot 1\{x_{i1} = \text{"Yes"}\} + 9 \cdot 1\{x_{i1} = \text{"No"}\} +$$

$$9.6 \cdot 1\{x_{i2} = \text{"Yes"}\} - 7.68 \cdot 1\{x_{i2} = \text{"No"}\},$$

yielding final estimates $\hat{y}_i = H_2(x_i)$ and residuals $r_{3i} = y_i - H_2(x_i)$ (cf. Table 5.4).

Table 5.4: Step two of boosting algorithm, yielding residual sum of squares $SSR = 1395.25$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$y_i$</th>
<th>$x_{i1}$</th>
<th>$x_{i2}$</th>
<th>$H_0(x_i)$</th>
<th>$r_{1i}$</th>
<th>$T(x_i; \Theta_1)$</th>
<th>$H_1(x_i)$</th>
<th>$r_{2i}$</th>
<th>$T(x_i; \Theta_2)$</th>
<th>$H_2(x_i)$</th>
<th>$r_{3i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-20</td>
<td>-7.20</td>
<td>24.80</td>
<td>-12.80</td>
<td>17.12</td>
<td>-5.12</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>Yes</td>
<td>Yes</td>
<td>32</td>
<td>-17</td>
<td>-7.20</td>
<td>24.80</td>
<td>-9.80</td>
<td>34.40</td>
<td>-19.40</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>No</td>
<td>No</td>
<td>32</td>
<td>-12</td>
<td>9.00</td>
<td>41.00</td>
<td>-21.00</td>
<td>33.32</td>
<td>-13.32</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-12</td>
<td>-7.20</td>
<td>24.80</td>
<td>-4.80</td>
<td>17.12</td>
<td>2.88</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>Yes</td>
<td>No</td>
<td>32</td>
<td>-8</td>
<td>-7.20</td>
<td>24.80</td>
<td>-0.80</td>
<td>17.12</td>
<td>6.88</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>38</td>
<td>No</td>
<td>Yes</td>
<td>32</td>
<td>6</td>
<td>9.00</td>
<td>41.00</td>
<td>-3.00</td>
<td>50.60</td>
<td>-12.60</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>42</td>
<td>No</td>
<td>No</td>
<td>32</td>
<td>10</td>
<td>9.00</td>
<td>41.00</td>
<td>1.00</td>
<td>33.32</td>
<td>8.68</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>53</td>
<td>Yes</td>
<td>Yes</td>
<td>32</td>
<td>21</td>
<td>-7.20</td>
<td>24.80</td>
<td>28.20</td>
<td>34.40</td>
<td>18.60</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>64</td>
<td>No</td>
<td>Yes</td>
<td>32</td>
<td>32</td>
<td>9.00</td>
<td>41.00</td>
<td>23.00</td>
<td>50.60</td>
<td>13.40</td>
<td></td>
</tr>
</tbody>
</table>
5.3 Loss minimization via function optimization

Let us now assume that we are given a data set with response $y = (y_i)_{i \in \{1, \ldots, n\}}$ and covariates $x_i = (x_{i1}, \ldots, x_{id})$ for $i \in \{1, \ldots, n\}$, where $d$ is the number of covariates. We define the quadratic loss $L(H)$ of a function $H := H(x)$ as

$$L(H) := \sum_{i=1}^{n} L(y_i, H(x_i)) := \sum_{i=1}^{n} \frac{1}{2} [y_i - H(x_i)]^2.$$ 

Now, the goal is to find an approximation of the function

$$\hat{H} := \arg \min_H L(H) = \arg \min_H \sum_{i=1}^{n} \frac{1}{2} [y_i - H(x_i)]^2,$$ 

which minimizes the quadratic loss. In this thesis we will follow the approach introduced by Friedman (2001) which restricts $H(x)$ to be a member of the class of parametrized functions given by

$$H(x; \{\rho_m, \Theta_m\}_{m=1}^{M}) = \sum_{m=0}^{M} \rho_m \cdot T(x; \Theta_m) \quad \text{for all } x \in \mathbb{R}^d,$$

where $T(x; \Theta_m)$ represent the base learners which are characterized by the values of the parameter $\Theta_m$. Additionally, $\rho_m \in \mathbb{R}$ are scaling parameters and $M$ is the pre-specified number of base learners which should be involved in the model. In the context of this thesis we will choose $T(x; \Theta_m)$ to be a regression tree as described in Section 5.1. Therefore, for each tree $\Theta_m$ specifies the regions $R_{m1}, \ldots, R_{mK_m}$ and the corresponding estimates $\hat{c}_{m1}, \ldots, \hat{c}_{mK_m}$. So the minimization problem in (5.1) changes to the problem of parameter optimization

$$\left\{ \begin{array}{c}
(\hat{\rho}, \hat{\Theta}) = \arg \min_{\rho, \Theta} \sum_{i=1}^{n} \frac{1}{2} \left[ y_i - \sum_{m=1}^{M} \rho_m \cdot T(x; \Theta_m) \right]^2, \\
\hat{H}(x) = \sum_{m=0}^{M} \hat{\rho}_m \cdot T(x; \hat{\Theta}_m) \quad \text{for all } x \in \mathbb{R}^d,
\end{array} \right.$$ 

where $\rho = (\rho_1, \ldots, \rho_M)$ is a vector and $\Theta = \{\Theta_1, \ldots, \Theta_M\}$ contains the parameters of the regression trees.

In order to solve (5.1), Friedman (2001) suggests to firstly ignore the constraint that $H$ should be a sum of regression trees and instead consider (5.1) as a numerical optimization problem

$$\hat{H} = \arg \min_H \mathcal{L}(H),$$ 

where $H \in \mathbb{R}^n$ defines the vector of the approximating function $H$ evaluated at the $n$ data points $x_1, \ldots, x_n$, this means

$$H := (H(x_1), \ldots, H(x_n)).$$
Now, we can solve (5.3) as a sum of component vectors

\[ H_M = \sum_{m=0}^{M} v_m, \quad v_m \in \mathbb{R}^n, \]

where \( H_0 = v_0 \) is an initial guess, which in the case of mean regression is just the mean over all data points, this means \( H_0 \equiv \bar{y} \). Then, \( H_m \) is calculated recursively based on \( H_{m-1} \) using so-called steepest descent. Therefore, we define \( g_m \in \mathbb{R}^n \) to be the gradient of \( L(H) \) evaluated at \( H = H_{m-1} \) with components

\[ g_{mi}(y_i, x_i) := \left[ \frac{\partial L(y_i, H(x_i))}{\partial H(x)} \right]_{H(x)=H_{m-1}(x_i)} \]

for \( i = 1, \ldots, n \).

When using the quadratic loss, \( L(y, x) = \frac{1}{2} (y - x)^2 \), each component of the gradient at step \( m \) is just the residual between the data \( y_i \) and the current approximation \( H_{m-1}(x_i) \), as

\[ g_{mi}(y_i, x_i) = \left[ \frac{\partial L(y_i, H(x_i))}{\partial H(x)} \right]_{H(x)=H_{m-1}(x)} = \left[ \frac{\partial \left( \frac{1}{2} [y_i - H(x_i)]^2 \right)}{\partial H(x_i)} \right]_{H(x)=H_{m-1}(x)} = -(y_i - H_{m-1}(x_i)), \quad \text{for } i = 1, \ldots, n. \]

So we can define the residuals \( r_m = (r_{m1}, \ldots, r_{mn}) \) as

\[ r_{mi} := -g_{mi}(y_i, x_i) = y_i - H_{m-1}(x_i) \quad \text{for } i = 1, \ldots, n. \]

Steepest descent then chooses the boost \( v_m = \rho_m \cdot r_m \), where \( \rho_m \) is the solution of

\[ \rho_m := \arg \min_{\rho} \mathcal{L}(H_{m-1} + \rho \cdot r_m), \]

the so-called line search. Afterwards, we update the current solution to

\[ H_m = H_{m-1} + \rho_m \cdot r_m. \quad (5.4) \]

\textit{Hastie et al. (2001)} point out that this procedure is a very greedy strategy, as \( r_m = -g_m \) is the most rapidly decreasing local direction of \( \mathcal{L}(H_{m-1}) \). Nevertheless, this numerical approach does not yet yield a general solution of our optimization problem (5.2) as the gradient and the residuals are only defined at the observed data points \( x_i \). A solution for this problem is to use at each step \( m \) a regression tree \( T(x; \Theta_m) \), which minimizes the squared error between its predictions and the current residuals, instead of using the residuals \( r_m \) themselves. More precisely, in each step \( m \) we look for the solution of

\[ \Theta_m = \arg \min_{\Theta} \sum_{i=1}^{n} \left[ r_{mi} - T(x_i; \Theta) \right]^2. \]
This yields a parameter \( \Theta_m \) which contains the optimal regions \( R_{m1}, \ldots, R_{mK_m} \) and the corresponding estimates \( \hat{c}_{m1}, \ldots, \hat{c}_{mK_m} \) which are given by

\[
\hat{c}_{mk} = \frac{1}{N_{mk}} \sum_{i=1}^{n} r_{mi} \mathbb{1}\{x_i \in R_{mk}\} \quad \text{for } k \in \{1, \ldots, K_m\},
\]

where \( N_{mk} \) is the number of data points in region \( R_{mk} \). So we can rewrite (5.4) as

\[
H_m(x) = H_{m-1}(x) + \rho_m \cdot T(x; \Theta_m)
= H_{m-1}(x) + \rho_m \cdot \sum_{k=1}^{K_m} \hat{c}_{mk} \mathbb{1}\{x \in R_{mk}\}
= H_{m-1}(x) + \sum_{k=1}^{K_m} \hat{\gamma}_{mk} \mathbb{1}\{x \in R_{mk}\},
\]

(5.6)

where we used in (5.5) that the regression trees have an additive structure themselves and defined in the last step \( \hat{\gamma}_{mk} := \rho_m \hat{c}_{mk} \). So from (5.5) to (5.6) we moved from adding a single additive function to adding \( K \) separate basis functions. Friedman (2001) points out that one can now further improve this result by finding the optimal coefficient for each of these basis functions. They are given as the solution of

\[
\{\hat{\gamma}_{mk}\}_{k=1}^{K_m} = \arg \min_{\gamma} \sum_{i=1}^{n} L(y_i, H_{m-1}(x_i) + \sum_{k=1}^{K_m} \gamma_k \mathbb{1}\{x_i \in R_{mk}\}).
\]

Due to the fact, that the regions \( R_{m1}, \ldots, R_{mK} \) in the regression trees are disjoint and we are using quadratic loss this reduces to

\[
\hat{\gamma}_{mk} = \arg \min_{\gamma} \sum_{x_i \in R_{mk}} L(y_i, H_{m-1}(x_i) + \gamma)
= \arg \min_{\gamma} \sum_{x_i \in R_{mk}} \frac{1}{2} [y_i - H_{m-1}(x_i) - \gamma]^2
= \arg \min_{\gamma} \sum_{x_i \in R_{mk}} \frac{1}{2} [r_{mi} - \gamma]^2 \quad \text{for } k \in \{1, \ldots, K_m\},
\]

which yields

\[
\hat{\gamma}_{mk} = \frac{1}{N_{mk}} \sum_{i=1}^{n} r_{mi} \mathbb{1}\{x_i \in R_{mk}\} =: \hat{c}_{mk},
\]

as tree \( T(x, \Theta_m) \) was built on the residuals \( r_{mi} \). So we can update our model to

\[
H_m(x) = H_{m-1}(x) + T(x; \Theta_m).
\]

(5.7)

This means our final model is given by

\[
H_M(x) = \bar{y} + \sum_{m=0}^{M} T(x; \Theta_m)
= \bar{y} + \sum_{m=1}^{M} \sum_{k=1}^{K_m} \hat{c}_{mk} \mathbb{1}\{x \in R_{mk}\} \quad \text{for all } x \in \mathbb{R}^d,
\]
yielding estimates

\[ \hat{y}_i = H_M(x_i) = \bar{y} + \sum_{m=1}^{M} T(x_i; \Theta_m) \]

\[ = \bar{y} + \sum_{m=1}^{M} \sum_{k=1}^{K_m} \hat{c}_{mk} 1 \{ x_i \in R_{mk} \}. \]

If we combine all these results we end up with Algorithm 5.1 which is also visualized in Figure 5.2.

**Algorithm 5.1** Gradient tree boosting algorithm under quadratic loss

**Input:** \( y \in \mathbb{R}^n, x \in \mathbb{R}^{n \times d}, M \in \mathbb{N} \)

1. Fit initial model as mean of observations:

   \[ H_0(x) = \bar{y} \quad \text{for all} \quad x \in \mathbb{R}^d. \]

2. For \( m = 1 \) to \( M \):

   (a) Calculate the residuals of the current model:

   \[ r_{mi} = y_i - H_{m-1}(x_i) \quad \text{for} \quad i \in \{1, \ldots, n\}. \]

   (b) Fit a regression tree to the residuals:

   \[ \Theta_m = \arg \min_{\Theta} \sum_{i=1}^{n} \left[ r_{mi} - T(x_i; \Theta) \right]^2. \]

   (c) Update current model:

   \[ H_m(x) = H_{m-1}(x) + T(x; \Theta_m) \]

   \[ = H_{m-1}(x) + \sum_{k=1}^{K_m} \hat{c}_{mk} 1 \{ x_i \in R_{mk} \} \quad \text{for all} \quad x \in \mathbb{R}^d. \]

3. Return final model:

   \[ \hat{H}(x) = H_M(x). \]
Input:
Data \((y_i, x_{i1}, \ldots, x_{id})\)
for \(i \in \{1, \ldots, n\}\).
Number of iterations \(M\). 

Initialize model:
\(H_0(x) \equiv \bar{y}\).

Calculate residuals of current model:
\(r_{mi} = y_i - H_{m-1}(x_i)\).

Fit regression tree to residuals:
\(\Theta_m = \arg \min_{\Theta} \sum_{i=1}^{n} [r_{mi} - T(x_i; \Theta)]^2\).

Update current model:
\(H_m(x) = H_{m-1}(x) + T(x; \Theta_m), \quad m \rightarrow m + 1\).

Is \(m = M\)?

Return final model:
\(\hat{H}(x) = H_M(x)\).

Figure 5.2: Illustration of algorithm for gradient boosting with regression trees under quadratic loss.
5.4 Tuning parameters in gradient boosting

In Algorithm 5.1 we have not put any restrictions on the regression trees so far and in particular we have not bounded their size $K$. As stated in the section on regression trees, we treat the maximal tree size as a tuning parameter in gradient boosting which is why we will explain some further details on the effect of different tree sizes in the following subsection. Additionally, we introduce some further tuning parameters which can improve the results of gradient boosting and make it more robust against overfitting.

5.4.1 Tree size

The size of the regression trees determines the complexity of the model: The bigger the trees the more complex the model becomes. We denote the tree size with $K$ and follow Hastie et al. (2001) in their explanations on the effect of tree size. When setting $K = 2$ each of the trees only consists of a single split and can therefore only cover the main effect of a single covariate. Letting $K = 3$ then allows for interactions between two variables and in general for any $K$ the model can contain interactions of up to $K - 1$ variables. According to Hastie et al. (2001) in many applications $K = 2$ is insufficient, but it will also be rarely required to use $K > 10$. In most of the cases $4 \leq K \leq 8$ yields good results for gradient tree boosting. We will in the following denote a regression tree of size $K$ by $T(x; \Theta, K)$.

5.4.2 Number of iterations

Friedman (2001) states that increasing the number of iterations $M$ can make the loss on the training data set arbitrarily small. However, this might cause overfitting which reduces the predictive power for future observations (cf. Figure 2.1). Therefore, one should monitor the loss of training and a test data set as a function of $M$ in order to find the optimal number of iterations. The optimal size of $M$ is highly influenced by the tuning parameter shrinkage which is described in the next subsection.

5.4.3 Shrinkage

The shrinkage or learning rate, which we will denote with $\eta$, is another tuning parameter in gradient tree boosting and can significantly improve the results. It is a constant that we use to scale each increment of our model during the updating process of the model. This means (5.7) changes to

$$H_m(x) = H_{m-1}(x) + \eta \cdot T(x; \Theta_m), \quad 0 < \eta \leq 1.$$  

As already explained before the choice of $\eta$ highly influences the optimal size of $M$. The smaller $\eta$ the more iterations are needed in general. Friedman (2001) has found that smaller values of $\eta$ result in lower test errors and that setting $\eta$ very small ($\eta < 0.1$) and then choosing $M$ appears to be the best strategy. However, the
improvement of the results comes at the price of higher computational effort, as the 
computation time grows proportional to the number of iterations $M$.

### 5.4.4 Subsampling of data

In order to further improve the results of gradient boosting, Friedman (2002) introduced a tuning parameter which specifies the proportion of data which should be used in each step of the algorithm, this means in each step of the algorithm we fit a regression tree only to a part of the data which will be randomly sampled without replacement from all the data. We call this parameter $\nu_n \in (0, 1]$ and define $\tilde{n} := \lceil \nu_n \cdot n \rceil$. So in step $m$ of the algorithm we now do the following:

(a) Create random permutation $\{\pi_n(1), \ldots, \pi_n(n)\}$ of $\{1, \ldots, n\}$.

(b) Draw subsample of data

$$\tilde{y}, \tilde{x} = ((\tilde{y}_{\pi_n(1)}, \tilde{x}_{\pi_n(1)}), \ldots, (\tilde{y}_{\pi_n(\tilde{n})}, \tilde{x}_{\pi_n(\tilde{n})})),$$

$$\tilde{n} = \lceil \nu_n \cdot n \rceil.$$ 

(c) Calculate the residuals of the current model for the drawn sample:

$$\tilde{r}_{mi} = H_{m-1}(\tilde{x}_i) - \tilde{y}_i, \text{ for } i \in \{1, \ldots, \tilde{n}\}.$$ 

(d) Fit a regression tree to the residuals:

$$\Theta_m = \arg \min_{\Theta} \sum_{i=1}^{\tilde{n}} [\tilde{r}_{mi} - T(\tilde{x}_i; \Theta)]^2.$$ 

(e) Update current model: $H_m(x) = H_{m-1}(x) + T(x; \Theta_m)$.

Friedman (2002) has shown that this approach reduces computation times and makes the model more robust. A typical choice would be $\nu_n = \frac{1}{2}$, however, for large $n$, $\nu_n$ can be notably smaller.

### 5.4.5 Column subsampling

Chen and Guestrin (2016) implemented in their R-package xgboost also the possibility of column subsampling. This means that in each step $m$ of the algorithm only a part of the available covariates is used in order to fit a regression tree to the residuals. We define this parameter as $\nu_d \in (0, 1]$ and summarize all results of this section in Algorithm 5.2.

| Table 5.5: Overview of all tuning parameters in gradient boosting with regression trees. |
|---|---|---|---|
| Symbol | Domain | Name | Description |
| $K$ | $\mathbb{N} \setminus \{1\}$ | Tree size | Number of regions in regression trees |
| $M$ | $\mathbb{N}$ | Number of iterations | Number of base learners |
| $\eta$ | $(0, 1]$ | Shrinkage | Weight of each base learner |
| $\nu_n$ | $(0, 1]$ | Data subsample | Portion of included data in each step |
| $\nu_d$ | $(0, 1]$ | Column subsample | Portion of included columns in each step |
Algorithm 5.2 Tuned gradient tree boosting algorithm under quadratic loss

**Input:** $y \in \mathbb{R}^n$, $x \in \mathbb{R}^{n \times d}$, $M \in \mathbb{N}$, $K \in \mathbb{N}$, $\eta \in (0, 1]$, $\nu_n \in (0, 1]$, $\nu_d \in (0, 1]$

1. Fit initial model as mean of observations:
   $$H_0(x) = \bar{y} \quad \text{for all} \quad x \in \mathbb{R}^d.$$

2. For $m = 1$ to $M$:
   (a) Create random permutation $\{\pi_n(1), \ldots, \pi_n(n)\}$ of $\{1, \ldots, n\}$.
   (b) Draw subsample of data
   $$\tilde{(y, x)} = (\tilde{y}_{\pi_n(1)}, \tilde{x}_{\pi_n(1)}), \ldots, (\tilde{y}_{\pi_n(\tilde{n})}, \tilde{x}_{\pi_n(\tilde{n})}), \quad \tilde{n} = \lfloor \nu_n \cdot n \rfloor.$$
   (c) Calculate the residuals of the current model for the drawn sample:
   $$\tilde{r}_{mi} = \tilde{y}_i - H_{m-1}(\tilde{x}_i), \quad i \in \{1, \ldots, \tilde{n}\}.$$
   (d) Create random permutation $\{\pi_d(1), \ldots, \pi_d(d)\}$ of $\{1, \ldots, d\}$.
   (e) Create subsample of covariates
   $$\tilde{x} = (\tilde{x}_{\pi_d(1)}, \ldots, \tilde{x}_{\pi_d(d)}), \quad \tilde{d} = \lfloor \nu_d \cdot d \rfloor.$$
   (f) Fit a regression tree to the residuals:
   $$\Theta_m = \arg \min_{\Theta} \frac{1}{\tilde{n}} \sum_{i=1}^{\tilde{n}} [\tilde{r}_{mi} - T(\tilde{x}_i; \Theta, K)]^2.$$
   (g) Update current model:
   $$H_m(x) = H_{m-1}(x) + \eta \cdot T(x; \Theta_m, K).$$

3. Return final model:
   $$\hat{H}(x) = H_M(x).$$
Chapter 6

Setup of simulation studies

After having introduced D-vine mean regression in Chapter 4, we now want to assess the performance of this approach in different scenarios and compare it with other well established methods like ordinary least square estimation, generalized linear models and gradient boosting as introduced in Chapter 5. Therefore, we will use the root mean squared error (RMSE), the oracle root mean squared error (oRMSE) and the relative root mean squared error (rRMSE) as defined in 2.6 for the comparison of in-sample and out-of-sample performance. Moreover, we will compare the methods according to numerical stability and computational speed.

In the remainder of this chapter we describe the setup of our simulation studies including explanations on the general procedure, the used models and the used data. Afterwards in Chapters 7 to 11, we will present the results of the different scenarios.

6.1 Remarks on general procedure

All our simulation studies follow the same general procedure. We start by simulating a data set of size $n$ according to one of the structures that will be explained in the next section. Then, we split this data set into a training data set of size $n_{\text{train}} = \frac{2}{3}n$ and a test data set of size $n_{\text{test}} = \frac{1}{3}n$. We use the training data set to fit our models and produce predictions for both the training and the test data based on these models. Afterwards, we calculate the RMSE, rRMSE and oRMSE for the training and the test set. We repeat this procedure for 100 different seeds yielding 100 values for each of the used performance measures and methods. Based on these values we will then compare the different models according to their prediction performance.

6.2 Studied data structures

We will use a variety of different data structures, evolving from very plain ones only containing main effects and three covariates, to more complex ones with interactions and dependencies and containing ten covariates. Moreover, they can be separated into classical regression scenarios and copula based data.
6.2.1 Regression based data structures

We start by explaining the classical regression scenarios, this means the data follows a typical regression structure, where the response \( Y \) is given as a weighted sum or product of the covariates \( X \).

**Simple linear regression with three standard normal covariates (SL3)**

\( Y \) is given as a weighted sum of three standard normal distributed covariates and a standard normal error term. This means for \( i = 1, \ldots, n \) we have

- \( X_{i,1}, X_{i,2}, X_{i,3} \overset{iid}{\sim} \mathcal{N}(0, 1) \),
- \( \varepsilon_i \sim \mathcal{N}(0, 1) \),
- \( Y_i = \beta_0 + \beta_1 \cdot X_{i,1} + \beta_2 \cdot X_{i,2} + \beta_3 \cdot X_{i,3} + \varepsilon_i \) and
- \( \beta = (\beta_0, \ldots, \beta_3) = (2, 1.75, 1.5, 1.25) \).

**Quadratic linear regression with three standard normal covariates (QL3)**

\( Y \) is given as a weighted sum of three standard normal distributed covariates, one quadratic term and a standard normal error. This means for \( i = 1, \ldots, n \) we have

- \( X_{i,1}, X_{i,2}, X_{i,3} \overset{iid}{\sim} \mathcal{N}(0, 1) \),
- \( \varepsilon_i \sim \mathcal{N}(0, 1) \),
- \( Y_i = \beta_0 + \beta_1 \cdot X_{i,1} + \beta_2 \cdot X_{i,2} + \beta_3 \cdot X_{i,3} + \beta_4 \cdot X_{i,1}^2 + \varepsilon_i \) and
- \( \beta = (\beta_0, \ldots, \beta_4) = (2, 1.75, 1.5, 1.25, 1) \).

**Gamma regression with three standard normal covariates (GR3)**

\( Y \) fulfills the assumptions that are used in gamma regression with the natural logarithm as link function as defined in \( \text{2.5.2} \). This means for \( i = 1, \ldots, n \) we have

- \( X_{i,1}, X_{i,2}, X_{i,3} \overset{iid}{\sim} \mathcal{N}(0, 1) \),
- \( \nu = \frac{1}{\sigma^2} = \text{const} \) with \( \sigma = 0.5 \),
- \( \mu_i = \exp(\beta_0 + \beta_1 \cdot X_{i,1} + \beta_2 \cdot X_{i,2} + \beta_3 \cdot X_{i,3}) \),
- \( Y_i \sim \text{Gamma}(\mu_i, \nu) \) with \( \mathbb{E}[Y_i] = \mu_i \) and \( \text{Var}(Y_i) = \frac{\mu_i^2}{\nu} = \mu_i^2 \cdot \sigma^2 \) and
- \( \beta = (\beta_0, \ldots, \beta_3) = (0.50, 0.35, 0.25, 0.15) \).
Simple linear regression with ten standard normal covariates (SL10)

$Y$ is given as a weighted sum of eight standard normal distributed covariates and a standard normal error term. Moreover, the data set contains two unused standard normal variables which are used to examine whether the models can detect unimportant variables. This means for $i = 1, \ldots, n$ we have

- $X_{i,1}, \ldots, X_{i,10} \overset{iid}{\sim} \mathcal{N}(0, 1)$,
- $\varepsilon_i \sim \mathcal{N}(0, 1)$,
- $Y_i = \beta_0 + \beta_1 \cdot X_{i,1} + \ldots + \beta_8 \cdot X_{i,8} + \varepsilon_i$ and
- $\beta = (\beta_0, \ldots, \beta_8) = (2.000, 1.875, 1.750, 1.625, 1.500, 1.375, 1.250, 1.125, 1.000)$.

Quadratic linear regression with ten standard normal covariates (QL10)

$Y$ is given as the square of the weighted sum of eight standard normal distributed covariates and a standard normal error term. Moreover, the data set contains two unused standard normal variables which are used to examine whether the models can detect unimportant variables. This means for $i = 1, \ldots, n$ we have

- $X_{i,1}, \ldots, X_{i,10} \overset{iid}{\sim} \mathcal{N}(0, 1)$,
- $\varepsilon_i \sim \mathcal{N}(0, 1)$,
- $Y_i = (\beta_0 + \beta_1 \cdot X_{i,1} + \ldots + \beta_8 \cdot X_{i,8})^2 + \varepsilon_i$ and
- $\beta = (\beta_0, \ldots, \beta_8) = (2.000, 1.875, 1.750, 1.625, 1.500, 1.375, 1.250, 1.125, 1.000)$.

Gamma regression with ten standard normal covariates (GR10)

$Y$ fulfills the assumptions that are used in gamma regression with the natural logarithm as link function as defined in 2.5.2. This means for $i = 1, \ldots, n$ we have

- $X_{i,1}, \ldots, X_{i,10} \overset{iid}{\sim} \mathcal{N}(0, 1)$,
- $\nu = \frac{1}{\sigma^2} = const$ with $\sigma = 0.5$,
- $\mu_i = \exp(\beta_0 + \beta_1 \cdot X_{i,1} + \ldots + \beta_8 \cdot X_{i,8})$,
- $Y_i \sim \text{Gamma}(\mu_i, \nu)$ with $E[Y_i] = \mu_i$ and $Var(Y_i) = \frac{\mu_i^2}{\nu} = \mu_i^2 \cdot \sigma^2$ and
- $\beta = (\beta_0, \ldots, \beta_8) = (0.300, 0.200, 0.175, 0.150, 0.125, 0.100, 0.075, 0.050, 0.025)$.

Moreover, the data set contains two unused standard normal variables which are used to examine whether the models can detect unimportant variables.
6.2.2 Copula based data structures

Besides the data structures following typical regression structures, we also investigate data which come from vine copulas.

Four-dimensional D-vine copula with fixed family (DF4)

\((v_i, u_{i,1}, \ldots, u_{i,3})_{i \in \{1, \ldots, n\}}\) is a sample of a four-dimensional D-vine copula where all pair-copulas \(C_{k,k+j;k+1:k+j-1}\) are forced to be Gumbel copulas with Kendall’s \(\tau\) fixed to 0.6\(^i\) for \(k \in \{1, 2, 3\}\) and \(j \in \{1, \ldots, 4 - k\}\). We then apply the inverse PIT to get \(y_i = \Phi^{-1}(v_i)\) and \(x_{ik} = \Phi^{-1}(u_{ik})\) for \(k \in \{1, 2, 3\}\) and \(i \in \{1, \ldots, n\}\), where \(\Phi^{-1}\) is the inverse of the standard normal cumulative distribution function.

Eleven-dimensional D-vine copula with fixed family (DF11)

\((v_i, u_{i,1}, \ldots, u_{i,10})_{i \in \{1, \ldots, n\}}\) is a sample of an eleven-dimensional D-vine copula where all pair-copulas \(C_{k,k+j;k+1:k+j-1}\) are forced to be Gumbel copulas with Kendall’s \(\tau\) fixed to 0.6\(^j\) for \(k \in \{1, \ldots, 10\}\) and \(j \in \{1, \ldots, 11 - k\}\). We then apply the inverse PIT to get \(y_i = \Phi^{-1}(v_i)\) and \(x_{ik} = \Phi^{-1}(u_{ik})\) for \(k \in \{1, \ldots, 10\}\) and \(i \in \{1, \ldots, n\}\), where \(\Phi^{-1}\) is the inverse of the standard normal cumulative distribution function.

Four-dimensional D-vine copula with random family (DR4)

\((v_i, u_{i,1}, \ldots, u_{i,3})_{i \in \{1, \ldots, n\}}\) is a sample of a four-dimensional D-vine copula where for \(k \in \{1, 2, 3\}\) and \(j \in \{1, \ldots, 4 - k\}\) for each of the pair-copulas \(C_{k,k+j;k+1:k+j-1}\) the family is chosen randomly from one of the copula families in Table 6.1. Each copula family has probability \(\frac{1}{4}\). Moreover, Kendall’s \(\tau\) is chosen randomly with probability \(\frac{1}{3}\) for each of the values in Table 6.1, where \(j\) denotes the tree the pair-copula belongs to. We then apply the inverse PIT to get \(y_i = \Phi^{-1}(v_i)\) and \(x_{ik} = \Phi^{-1}(u_{ik})\) for \(k \in \{1, 2, 3\}\) and \(i \in \{1, \ldots, n\}\), where \(\Phi^{-1}\) is the inverse of the standard normal cumulative distribution function.

Four-dimensional R-vine copula with random family (RR4)

\((v_i, u_{i,1}, \ldots, u_{i,3})_{i \in \{1, \ldots, n\}}\) is a sample of a four-dimensional R-vine copula (c.f. 3.5) where for each of the pair-copulas the family is chosen randomly from one of the copula families in Table 6.1, each with probability \(\frac{1}{4}\). Moreover, Kendall’s \(\tau\) is chosen randomly with probability \(\frac{1}{3}\) for each of the values in Table 6.1, where \(j\) denotes the tree the pair-copula belongs to. We then apply the inverse PIT to get \(y_i = \Phi^{-1}(v_i)\) and \(x_{ik} = \Phi^{-1}(u_{ik})\) for \(k \in \{1, 2, 3\}\) and \(i \in \{1, \ldots, n\}\), where \(\Phi^{-1}\) is the inverse of the standard normal cumulative distribution function. For sampling from an R-vine we used the procedure explained in Section 3.5. Note that as the dimension of the R-vine copula is four we will only get C-vines and D-vines as general R-vines only occur for dimensions \(d \geq 5\).

An overview of all the used data structures containing the most important information on the marginal distributions and the connection between the response and the covariates can be found in Table 6.2.
We know that in this case.

\[ Y, X \]

We are now interested in the joint distribution of \((Y, X)\). Assume we are given a simple linear regression setup

\[ Y = \beta X + \varepsilon, \quad \text{with} \quad X \sim N(0, 1) \quad \text{and} \quad \varepsilon \sim N(0, 1). \]  

(6.1)

We know that in this case

\[ Y | X = x \sim N(\beta X, 1) \quad \text{and} \quad \beta = \frac{\sigma_{YX}}{\sigma_X^2} = \frac{\sigma_Y}{\sigma_X} = \sigma_{Y, \rho}. \]  

(6.2)

We are now interested in the joint distribution of \((Y, X)\). Therefore, we consider

\[ f(y, x) = f(y | x) \cdot f(x) = \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2} (y - \beta x)^2 \right) \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2} x^2 \right) \]

\[ = \frac{1}{2\pi} \exp\left( -\frac{1}{2} (y^2 - 2yx\beta + (1 + \beta^2)x^2) \right). \]  

(6.3)

\[ \tau \]

Table 6.1: Allowed copula families and values of Kendall’s \(\tau\) for pair-copulas in tree \(j\) in data setup (DR4) and (RR4).

<table>
<thead>
<tr>
<th>Copula Family</th>
<th>Kendall’s (\tau) for tree (j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian copula</td>
<td>0.3(^2), 0.6(^2), 0.8(^2)</td>
</tr>
<tr>
<td>Student t copula</td>
<td>0.3(^2), 0.6(^2), 0.8(^2)</td>
</tr>
<tr>
<td>Gumbel copula</td>
<td>0.3(^2), 0.6(^2), 0.8(^2)</td>
</tr>
<tr>
<td>Frank copula</td>
<td>0.3(^2), 0.6(^2), 0.8(^2)</td>
</tr>
<tr>
<td>Rotated Gumbel copula (180 degrees; “survival Gumbel”)</td>
<td>0.3(^2), 0.6(^2), 0.8(^2)</td>
</tr>
<tr>
<td>Rotated Gumbel copula (90 degrees)</td>
<td>0.3(^2), 0.6(^2), 0.8(^2)</td>
</tr>
<tr>
<td>Rotated Gumbel copula (270 degrees)</td>
<td>0.3(^2), 0.6(^2), 0.8(^2)</td>
</tr>
</tbody>
</table>

Table 6.2: Overview of data structures which are used throughout the simulation studies.

<table>
<thead>
<tr>
<th>Data</th>
<th>#Covariates</th>
<th>Marginals</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL3</td>
<td>3</td>
<td>(X_1 \sim N(0,1))</td>
<td>(E[Y_i</td>
</tr>
<tr>
<td>QL3</td>
<td>3</td>
<td>(X_1 \sim N(0,1))</td>
<td>(E[Y_i</td>
</tr>
<tr>
<td>GR3</td>
<td>3</td>
<td>(X_1 \sim N(0,1))</td>
<td>(E[Y_i</td>
</tr>
<tr>
<td>SL10</td>
<td>10</td>
<td>(X_1 \sim N(0,1))</td>
<td>(E[Y_i</td>
</tr>
<tr>
<td>QL10</td>
<td>10</td>
<td>(X_1 \sim N(0,1))</td>
<td>(E[Y_i</td>
</tr>
<tr>
<td>GR10</td>
<td>10</td>
<td>(X_1 \sim N(0,1))</td>
<td>(E[Y_i</td>
</tr>
</tbody>
</table>

6.2.3 Connection between regression and copula structures

After having seen different setups for regression and copula data, we now want to show that both are connected in the sense that also in regression setups we can find the underlying copula. Assume we are given a simple linear regression setup

\[ Y = \beta X + \varepsilon, \quad \text{with} \quad X \sim N(0, 1) \quad \text{and} \quad \varepsilon \sim N(0, 1). \]  

(6.1)

We know that in this case

\[ Y | X = x \sim N(\beta X, 1) \quad \text{and} \quad \beta = \frac{\sigma_{YX}}{\sigma_X^2} = \frac{\sigma_Y}{\sigma_X} = \sigma_{Y, \rho}. \]  

(6.2)
From (6.1) we can also conclude that $Y$ is normal distributed with
\[ \mathbb{E}[Y] = \mathbb{E}[\beta X + \varepsilon] = 0 \quad \text{and} \quad \text{Var}(Y) = \text{Var}(\beta X + \varepsilon) = \beta^2 + 1 = \sigma^2_Y \rho^2 + 1 = \sigma^2_Y \Rightarrow \sigma^2_Y = \frac{1}{1 - \rho^2}. \]

Suppose $(Y, X)$ follows a bivariate normal distribution with mean vector and covariance matrix given by
\[ \mu = (\mu_Y, \mu_X) = (0, 0) \quad \text{and} \quad \Sigma = \begin{pmatrix} \sigma^2_Y & \rho \sigma_Y \sigma_X \\ \rho \sigma_Y \sigma_X & \sigma^2_X \end{pmatrix} = \begin{pmatrix} 1 & \frac{\rho}{\sqrt{1 - \rho^2}} \\ \frac{\rho}{\sqrt{1 - \rho^2}} & 1 \end{pmatrix}. \]

Then, we would have
\[ f(y, x) = \frac{1}{2\pi \sqrt{\det(\Sigma)}} \exp\left( -\frac{1}{2} (y \ x) \Sigma^{-1} (y \ x) \right). \]

It is easy to see that $\det(\Sigma) = 1$. Therefore, we get
\[ \Sigma^{-1} = \begin{pmatrix} \frac{1}{1 - \rho^2} & -\frac{\rho}{\sqrt{1 - \rho^2}} \\ -\frac{\rho}{\sqrt{1 - \rho^2}} & \frac{1}{1 - \rho^2} \end{pmatrix}, \]

yielding
\[ f(y, x) = \frac{1}{2\pi} \exp\left( -\frac{1}{2} \frac{y^2 - 2yx \rho}{\sqrt{1 - \rho^2}} + x^2 \frac{1}{1 - \rho^2} - \frac{\rho}{\sqrt{1 - \rho^2}} \right). \]

Now, using (6.2) to get
\[ \beta = \sigma_Y \rho = \frac{\rho}{\sqrt{1 - \rho^2}} \quad \text{and} \quad \rho^2 = \frac{\beta^2}{1 + \beta^2} \]
we end up exactly at (6.3), as
\[ \frac{1}{1 - \rho^2} = \frac{1}{1 - \frac{\beta^2}{1 + \beta^2}} = \frac{1}{\frac{1 + \beta^2}{1 + \beta^2} - \frac{\beta^2}{1 + \beta^2}} = 1 + \beta^2. \]

So we have shown that $(Y, X)$ follows a bivariate normal distribution with mean vector, covariance matrix and correlation matrix given by
\[ \mu = (\mu_Y, \mu_X) = (0, 0), \quad \Sigma = \begin{pmatrix} \frac{1}{1 - \rho^2} & \frac{\rho}{\sqrt{1 - \rho^2}} \\ \frac{\rho}{\sqrt{1 - \rho^2}} & 1 \end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}. \]
6.2. STUDIED DATA STRUCTURES

We can also verify this result by using the property of the bivariate normal given in (2.1) which states

\[ Y \mid X = x \sim \mathcal{N}\left( \mu_Y + \frac{\sigma_Y}{\sigma_X} \rho(x - \mu_X), (1 - \rho^2)\sigma_Y^2 \right) \]

\[ \sim \mathcal{N}\left( \sigma_Y \rho x, (1 - \rho^2) \frac{1}{\sqrt{1 - \rho^2}} \right) \]

\[ \sim \mathcal{N}\left( \beta x, 1 \right). \]

Generalization to \( d \)-dimensional case

The previous result can also be generalized to the \( d \)-dimensional case for uncorrelated \( X \). Therefore, let

\[ Y = X^\top \beta + \varepsilon = \beta_1 X_1 + \ldots + \beta_d X_d + \varepsilon, \quad \text{with} \quad X \sim \mathcal{N}_d(0_d, I_d) \quad \text{and} \quad \varepsilon \sim \mathcal{N}(0, 1) \]

be a linear regression setup. From this we know that the distributions of \( Y \) and \( Y \mid X = x \) are given by

\[ Y \sim \mathcal{N}(0, \beta_2 + \ldots + \beta_d^2 + 1) \quad \text{and} \quad Y \mid X = x \sim \mathcal{N}(X^\top \beta, 1), \]

respectively.

Moreover, let the covariance matrix of \( Y \) and \( X \) be given by

\[ \Sigma = \begin{pmatrix} \sigma_Y^2 & \Sigma_{YX} \\ \Sigma_{X} & \Sigma_{XX} \end{pmatrix}, \]

where \( \Sigma_{YX} \) is the vector of covariances between \( Y \) and \( X \) and \( \Sigma_{XX} \) is the covariance matrix of \( X \). Using this notation, we know that it holds

\[ \beta = \Sigma_{XX}^{-1} \Sigma_{YX} = I_d \begin{pmatrix} Cov(Y, X_1) \\ \vdots \\ Cov(Y, X_d) \end{pmatrix} = \begin{pmatrix} \sigma_Y^2 \rho_{11} \\ \vdots \\ \sigma_Y^2 \rho_{dd} \end{pmatrix} = \begin{pmatrix} \sigma_Y \rho_1 \\ \vdots \\ \sigma_Y \rho_d \end{pmatrix}. \]

Furthermore, we get

\[ Var(Y) = \sigma_Y^2 = \beta_1^2 + \ldots + \beta_d^2 + 1 = \sigma_Y^2 \rho_1^2 + \ldots + \sigma_Y^2 \rho_d^2 + 1, \quad \text{yielding} \]

\[ \sigma_Y^2 = \frac{1}{1 - \rho_1^2 - \ldots - \rho_d^2} \quad \text{and} \quad \beta_j = \sigma_Y \rho_j = \frac{\rho_j}{\sqrt{1 - \rho_1^2 - \ldots - \rho_d^2}} \quad \text{for} \quad j \in \{1, \ldots, d\}. \quad (6.4) \]

As in the 2-dimensional case we are now interested in the joint density of \( (Y, X) \):

\[ f(y, x) = f(y \mid x) \cdot f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(y - x^\top \beta)^2\right) \frac{1}{(\sqrt{2\pi})^d} \exp\left(-\frac{1}{2} \sum_{j=1}^d x_j^2\right) \]

\[ = \frac{1}{(\sqrt{2\pi})^{d+1}} \exp\left(-\frac{1}{2}((y - x^\top \beta)^2 + \sum_{j=1}^d x_j^2)\right). \]
Plugging in (6.4) for the $\beta$’s and rearranging all the terms yields exactly the density of a multivariate normal distribution of dimension $d + 1$ with mean vector $\mu = 0_{d+1}$ and covariance matrix $\Sigma$ and correlation matrix $R$ given by

$$
\Sigma = \begin{pmatrix}
\frac{1}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} & \frac{\rho_1}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} & \cdots & \frac{\rho_d}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} \\
\frac{\rho_1}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} & \frac{1}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} & \cdots & \frac{\rho_d}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\rho_d}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} & \frac{\rho_d}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}} & \cdots & \frac{1}{\sqrt{1-\rho_1^2-\cdots-\rho_d^2}}
\end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix}
1 & \rho_1 & \cdots & \rho_d \\
\rho_1 & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots \\
\rho_d & \cdots & \cdots & 1
\end{pmatrix}.
$$

Moreover, if the regression coefficients are known we get

$$
\rho_j^2 = \frac{\beta_j^2}{1 + \beta_1^2 + \cdots + \beta_d^2}.
$$

So we have shown that for a linear regression with uncorrelated standard-normal covariates the joint distribution of the response $Y$ and the covariates $X$ is multivariate normal. This also induces that the underlying dependence structure is given by a Gaussian copula. Furthermore, Theorem 3.3 states that the copula does not change under monotonic transformations of the marginals. This implies that we still have the Gaussian copula as associated copula if we add an intercept $\beta_0$ to the model and use uncorrelated normal distributed covariates $X$ with arbitrary mean vector $\mu \in \mathbb{R}^d$ and arbitrary covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$, such that $\Sigma$ fulfills the properties of a covariance matrix. Moreover, if we apply the transformation $y \mapsto \exp(y)$ to our response we also do not lose the Gaussian copula as underlying dependence structure. This means in our setup the data structures SL3, GR3, SL10 and GR10 have a Gaussian copula as associated copula of the joint distribution of $(Y, X)$.

### 6.3 Explanation of different models

During the simulation studies we compare the fit of the five different Methods IOQ, CEE, MCS, ICD and SQI for D-vine mean regression, which were introduced in Section 4.2, with an ordinary least square fit (in the following abbreviated with OLS) and the fit of two gradient boosting models (in the following abbreviated with XGB) as introduced in Chapter 5. Note that in some of the scenarios we will use a generalized linear model (GLM) (cf. 2.5) instead of the OLS model for better comparison. As already explained in Section 4.3, for each of the D-vine mean regression methods we can estimate two different models, depending on the estimation of the pair-copulas which can either be done parametrically or nonparametrically, yielding together with the five methods for conditional mean estimation 10 different D-vine mean regression models. Moreover, we also use two different gradient boosting models, one with default parameter settings and one with tuned parameters reached from 5-fold cross-validation as explained in Section 2.8. The default XGB models were fitted using the R-package xgboost (Chen et al., 2018) and for the tuned models we used the package caret (Kuhn et al., 2018). We summarize the notation that will be used for the different models throughout the simulation studies in Table 6.3.
Table 6.3: Overview over all the different methods/models that are used throughout the simulation studies. Column *Abbreviation* shows the short name and column *Full name* describes the model.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Eq.</th>
<th>Full name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td></td>
<td>Ordinary least square model</td>
</tr>
<tr>
<td>GLM</td>
<td></td>
<td>Generalized linear model</td>
</tr>
<tr>
<td>XGB&lt;sub&gt;def&lt;/sub&gt;</td>
<td></td>
<td>Gradient boosting model with default parameters</td>
</tr>
<tr>
<td>XGB&lt;sub&gt;tun&lt;/sub&gt;</td>
<td></td>
<td>Gradient boosting model with tuned parameters</td>
</tr>
<tr>
<td>IOQ&lt;sub&gt;par&lt;/sub&gt;</td>
<td>(4.3)</td>
<td>D-vine model using integration over quantiles with parametric pair-copulas</td>
</tr>
<tr>
<td>IOQ&lt;sub&gt;np&lt;/sub&gt;</td>
<td>(4.3)</td>
<td>D-vine model using integration over quantiles with nonparametric pair-copulas</td>
</tr>
<tr>
<td>CEE&lt;sub&gt;par&lt;/sub&gt;</td>
<td>(4.4)</td>
<td>D-vine model using calculation of estimating equation with parametric pair-copulas</td>
</tr>
<tr>
<td>CEE&lt;sub&gt;np&lt;/sub&gt;</td>
<td>(4.4)</td>
<td>D-vine model using calculation of estimating equation with nonparametric pair-copulas</td>
</tr>
<tr>
<td>MCS&lt;sub&gt;par&lt;/sub&gt;</td>
<td>(4.7)</td>
<td>D-vine model using Monte Carlo simulation with parametric pair-copulas</td>
</tr>
<tr>
<td>MCS&lt;sub&gt;np&lt;/sub&gt;</td>
<td>(4.7)</td>
<td>D-vine model using Monte Carlo simulation with nonparametric pair-copulas</td>
</tr>
<tr>
<td>ICD&lt;sub&gt;par&lt;/sub&gt;</td>
<td>(4.9)</td>
<td>D-vine model using integration over conditional copula density with parametric pair-copulas</td>
</tr>
<tr>
<td>ICD&lt;sub&gt;np&lt;/sub&gt;</td>
<td>(4.9)</td>
<td>D-vine model using integration over conditional copula density with nonparametric pair-copulas</td>
</tr>
<tr>
<td>SQI&lt;sub&gt;par&lt;/sub&gt;</td>
<td>(4.10)</td>
<td>D-vine model using simplified quantile integration with parametric pair-copulas</td>
</tr>
<tr>
<td>SQI&lt;sub&gt;np&lt;/sub&gt;</td>
<td>(4.10)</td>
<td>D-vine model using simplified quantile integration with nonparametric pair-copulas</td>
</tr>
</tbody>
</table>
6.4 Explanation of scenarios

As the goal of the simulation studies is to assess the finite sample performance of the different methods of D-vine mean regression, we will compare them with respect to each other and with respect to an OLS/GLM and a default and tuned XGB model. Therefore, we use many different scenarios starting by finding the optimal size of the Monte Carlo sample $m$ in Method MCS (cf. Chapter 7) and the optimal number of quantiles $n_{\text{quant}}$ in Method SQI (cf. Chapter 8). Afterwards, we analyze the effect of the size $n$ of the data set on the different methods (cf. Chapter 9). Then, using the results of these scenarios, we evaluate the effect of different data structures on the performance of the different methods in Chapter 10 for three covariates and in Chapter 11 for ten covariates. An overview of the scenarios, containing the used parameters and data structures, can be found in Table 6.4.
Table 6.4: Overview of all scenarios with used data structures and parameters.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Goal</th>
<th>Data</th>
<th>Data Structure</th>
<th>m (MCS)</th>
<th>$n_{quant}$ (SQI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS.1</td>
<td>Find optimal $m$</td>
<td>SL3 4</td>
<td>1500</td>
<td>100, 500, 1000, 5000, 10000</td>
<td>-</td>
</tr>
<tr>
<td>MCS.2</td>
<td>Find optimal $m$</td>
<td>QL3 4</td>
<td>1500</td>
<td>100, 500, 1000, 5000, 10000</td>
<td>-</td>
</tr>
<tr>
<td>MCS.3</td>
<td>Find optimal $m$</td>
<td>GR3 4</td>
<td>1500</td>
<td>100, 500, 1000, 5000, 10000</td>
<td>-</td>
</tr>
<tr>
<td>MCS.4</td>
<td>Find optimal $m$</td>
<td>DF4 4</td>
<td>1500</td>
<td>100, 500, 1000, 5000, 10000</td>
<td>-</td>
</tr>
<tr>
<td>SQL.1</td>
<td>Find optimal $n_{quant}$</td>
<td>SL3 4</td>
<td>1500</td>
<td>-</td>
<td>10, 25, 50, 75, 100</td>
</tr>
<tr>
<td>SQL.2</td>
<td>Find optimal $n_{quant}$</td>
<td>QL3 4</td>
<td>1500</td>
<td>-</td>
<td>10, 25, 50, 75, 100</td>
</tr>
<tr>
<td>SQL.3</td>
<td>Find optimal $n_{quant}$</td>
<td>GR3 4</td>
<td>1500</td>
<td>-</td>
<td>10, 25, 50, 75, 100</td>
</tr>
<tr>
<td>SQL.4</td>
<td>Find optimal $n_{quant}$</td>
<td>DF4 4</td>
<td>1500</td>
<td>-</td>
<td>10, 25, 50, 75, 100</td>
</tr>
<tr>
<td>DS.1</td>
<td>Analyze effect of $n$</td>
<td>SL3 4</td>
<td>100, 500, 1000, 1500, 2000</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>DS.2</td>
<td>Analyze effect of $n$</td>
<td>QL3 4</td>
<td>100, 500, 1000, 1500, 2000</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>DS.3</td>
<td>Analyze effect of $n$</td>
<td>GR3 4</td>
<td>100, 500, 1000, 1500, 2000</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>DS.4</td>
<td>Analyze effect of $n$</td>
<td>DF4 4</td>
<td>100, 500, 1000, 1500, 2000</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>SL.3</td>
<td>Compare fit</td>
<td>SL3 4</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>QL.3</td>
<td>Compare fit</td>
<td>QL3 4</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>GR.3</td>
<td>Compare fit</td>
<td>GR3 4</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>DF.4</td>
<td>Compare fit</td>
<td>DF4 4</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>DR.4</td>
<td>Compare fit</td>
<td>DR4 4</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>RR.4</td>
<td>Compare fit</td>
<td>RR4 4</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>SL.10</td>
<td>Compare fit</td>
<td>SL10 11</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>QL.10</td>
<td>Compare fit</td>
<td>QL10 11</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>GR.10</td>
<td>Compare fit</td>
<td>GR10 11</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
<tr>
<td>DF.11</td>
<td>Compare fit</td>
<td>DF11 11</td>
<td>1500</td>
<td>5000</td>
<td>10</td>
</tr>
</tbody>
</table>
Chapter 7

Selection of Monte Carlo sample size in Method MCS

We start our simulation studies by analyzing the effect of the Monte Carlo (MC) sample size in the MC simulation on the performance of Method MCS. Therefore, we let the sample size take the values \( m \in \{100, 500, 1000, 5000, 10000\} \) and consider the data structures SL3, QL3, GR3 and DF4, yielding Scenarios MCS.1, MCS.2, MCS.3 and MCS.4.

7.1 Scenario MCS.1: Monte Carlo sample size in simple linear regression

Figure 7.1 shows a boxplot of the RMSE for the training and test data with respect to the different sample sizes in the Monte Carlo simulation. The first thing that we observe is that the OLS yields the best results and the performance of Method MCS improves as \( m \) increases. Moreover, we can see many outliers for the nonparametric version which has also quite big upper quartiles in the test data. The reason for high values of the RMSE in the nonparametric version are a few bad predictions which yield large residuals. As the RMSE contains the squared residuals a single bad prediction can result in a huge RMSE. Further investigations have shown that bad predictions always occur when very unlikely values of \( Y \) are drawn during the Monte Carlo sampling process. At these values the nonparametric pair-copulas are not estimated well and therefore yield bad predictions of the conditional mean. The enormous effect of these outliers can also be observed when looking at the mean RMSE which can be found in the Appendix in Table B.1. Nevertheless, for all versions of Method MCS in the median we get quite close to the RMSE of the OLS which can be seen as a benchmark in this situation as our scenario meets all the assumptions from ordinary least squares estimation. However, we can see that the improvement from \( m = 5000 \) to \( m = 10000 \) is only very small.
Figure 7.1: Boxplot of RMSE of training and test data set in Scenario \textit{MCS.1} according to different sample sizes $m$ of the Monte Carlo simulation.
We get very similar results for the oRMSE and the rRMSE. From these results we could suggest to set $m = 5000$ as a default value for the size of the MC sample in Method MCS. In order to verify this, we now consider a scenario where a quadratic term is added in the data setup.

### 7.2 Scenario MCS.2: Monte Carlo sample size in quadratic linear regression

In this scenario we see a similar behaviour as in the previous one, this means for increasing $m$ the RMSE decreases. In Figure 7.2 we can moreover see that Method MCS outperforms the OLS for both versions in the median if the sample size in the Monte Carlo simulation is at least $m = 500$. The reason for this outperformance is that we did not include the quadratic term in the OLS model, so that it is not a benchmark in this scenario anymore. Again, the decrease of the RMSE is very small from $m = 5000$ to $m = 10000$ and the nonparametric version has many outliers and a big upper quartile in the test data. Once more, the outliers highly affect the mean RMSE which can be seen in Table B.2. The results for the oRMSE and the rRMSE have been very similar.

So far, our response was always given as a weighted sum of our covariates or squared versions of them. In the next scenario we will move from additive dependencies to multiplicative dependencies on the covariates and a non-normal response in order to see whether these have an effect on the needed Monte Carlo sample size.

### 7.3 Scenario MCS.3: Monte Carlo sample size in gamma regression

First of all, we note that in this scenario the OLS estimate is again not the benchmark for an optimal model. Instead, one could use a GLM which performs a gamma regression using the log-link (cf. 2.5.2) as a benchmark for this setup. But, we are still only interested in finding the required Monte Carlo sample size and not in comparing the fit of the D-vine mean regression with an optimal fit. This comparison will be done later in Sections 10.3 and 11.3.

Keeping this in mind we can now analyze the results of Scenario MCS.3 which are shown in Figure 7.3. We observe that in this scenario the RMSE already stabilizes for a Monte Carlo sample size of $m = 500$ and even for $m = 100$ the results are only slightly worse as Table B.3 shows. However, again in the context of the mean the nonparametric version is influenced by outliers, especially for the test data.
Figure 7.2: Boxplot of RMSE of training and test data set in Scenario MCS.2 according to different sample sizes of the Monte Carlo simulation.
Figure 7.3: Boxplot of RMSE of training and test data set in Scenario \textit{MCS.3} according to different sample sizes of the Monte Carlo simulation.
7.4 Scenario \textit{MCS.4}: Monte Carlo sample size in copula based model

After having analyzed three regression based scenarios, we now also want to investigate whether using a copula based data structure has a different effect on the required MC sample size. Figure 7.5 shows the results for the case where the data follows a D-vine copula with Gumbel copulas for all pair-copulas in the PCC. Again, we see that for increasing sample size $m$ the RMSE decreases. However, especially for the parametric version the improvement of the median is only very small from $m = 500$ to $m = 10000$. In contrast, the upper quartile of the nonparametric version is quite big for large $m$ which is again a result of single bad predictions within some of the replications. This behaviour can be explained with the fact that for larger MC samples sizes it is more likely that one of the sampled values comes from a low density region and therefore at this value the nonparametric copula is misspecified resulting in bad predictions. These do also affect the mean RMSE as Table B.4 in the Appendix shows.

We summarize the results from this and the previous three scenarios by deciding to set the Monte Carlo sample size to $m = 5000$ for all further simulation studies as the results only slightly improve for bigger values of $m$ and the computational time increases proportional to the size of $m$ as can be seen in Figure 7.4.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{computation_time}
\caption{Computation time for OLS and Method MCS for different Monte Carlo sample sizes $m$.}
\end{figure}
7.4. SCENARIO MCS.4

**Boxplot of RMSE for training data**

- **RMSE**
  - 0.5
  - 1.0
  - 1.5
- **m**
  - 100
  - 500
  - 1000
  - 5000
  - 10000

**Boxplot of RMSE for test data**

- **RMSE**
  - 0.5
  - 1.0
  - 1.5
- **m**
  - 100
  - 500
  - 1000
  - 5000
  - 10000

**Method**
- OLS
- MCS_par
- MCS_np

**Figure 7.5:** Boxplot of RMSE of training and test data set in Scenario MCS.4 according to different sample sizes of the Monte Carlo simulation.
Chapter 8

Selection of number of quantiles in Method SQI

After having analyzed the effect of the size of the Monte Carlo sample in Method MCS, we will now investigate whether increasing the number of calculated quantiles in Method SQI will significantly improve this method. Therefore, we again consider the four different data structures SL3, QL3, GR3 and DF4 and let the number of calculated quantiles be \( n_{quant} \in \{10, 25, 50, 75, 100\} \), resulting in Scenarios SQI.1, SQI.2, SQI.3 and SQI.4.

Figure 8.1 shows the results for Scenario SQI.1. We can see that the boxplots of the RMSE are almost identical for all choices of \( n_{quant} \). Moreover, in Table 8.1 we can see that the mean RMSE is basically the same for all values of \( n_{quant} \) within each method. The results for Scenarios SQI.2, SQI.3 and SQI.4 are very similar (cf. Appendix B.1.2) so we can conclude that setting the default value to \( n_{quant} = 10 \) is sufficiently large. This will also reduce computation time as it grows proportionally with the number of calculated quantiles.

Table 8.1: Mean RMSE for training and test data in Scenario SQI.1 for different number of quantiles in Method SQI.

<table>
<thead>
<tr>
<th>( n_{quant} )</th>
<th>OLS</th>
<th>SQI_par</th>
<th>SQI_up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>10</td>
<td>0.99691</td>
<td>1.00112</td>
<td>1.00729</td>
</tr>
<tr>
<td>25</td>
<td>0.99691</td>
<td>1.00112</td>
<td>1.00727</td>
</tr>
<tr>
<td>50</td>
<td>0.99691</td>
<td>1.00112</td>
<td>1.00724</td>
</tr>
<tr>
<td>75</td>
<td>0.99691</td>
<td>1.00112</td>
<td>1.00724</td>
</tr>
<tr>
<td>100</td>
<td>0.99691</td>
<td>1.00112</td>
<td>1.00724</td>
</tr>
</tbody>
</table>
Figure 8.1: Boxplot of RMSE of training and test data set in Scenario SQL.1 for different number of quantiles in Method SQI.
Chapter 9

Analyzing the effect of data set size

Now that we have found good choices for the Monte Carlo sample size $m$ in Method \textsc{MCS} and the number of quantiles $n_{\text{quant}}$ in Method \textsc{SQI} we will investigate which minimal size of the data set is needed in order to get reasonable results. Therefore, we analyze the prediction performance of all methods for the data structures \textsc{SL3}, \textsc{QL3}, \textsc{GR3} and \textsc{DF4} and let the overall data set size be $n \in \{100, 500, 1000, 1500, 2000\}$. This gives us Scenarios $\mathcal{DS}.1$, $\mathcal{DS}.2$, $\mathcal{DS}.3$ and $\mathcal{DS}.4$. The results for Scenario $\mathcal{DS}.1$ are shown in Figure 9.1. We decided to only plot the median, as we have already seen before that for Method \textsc{MCS} the mean is highly influenced by outliers. For each of the methods we can see two lines corresponding to the parametric (solid line) and nonparametric (dotted line) version. The plots indicate that in general already small sizes of the data set yield quite good results for the training data. However, for the test data the RMSE only stabilizes and gets close to the OLS benchmark if $n \geq 1500$. Moreover, in Table 9.1 we can additionally find the mean RMSE for all methods in Scenario $\mathcal{DS}.1$, revealing that also the nonparametric version of Method \textsc{ICD} is influenced by outliers. The behavior in Scenarios $\mathcal{DS}.2$, $\mathcal{DS}.3$ and $\mathcal{DS}.4$ is very similar and can therefore be found in Appendix B.1.3. Hence, we decided to set the size of the data set to $n = 1500$ for all further simulations, yielding sizes for the training and the test data set of $n_{\text{train}} = 1000$ and $n_{\text{test}} = 500$, respectively.
Figure 9.1: Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario DS.1 according to different sizes of the data set.
<table>
<thead>
<tr>
<th>Method</th>
<th>100 mean</th>
<th>100 median</th>
<th>500 mean</th>
<th>500 median</th>
<th>1000 mean</th>
<th>1000 median</th>
<th>1500 mean</th>
<th>1500 median</th>
<th>2000 mean</th>
<th>2000 median</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.969</td>
<td>0.970</td>
<td>0.986</td>
<td>0.986</td>
<td>0.993</td>
<td>0.989</td>
<td>0.997</td>
<td>0.999</td>
<td>0.995</td>
<td>0.997</td>
</tr>
<tr>
<td>test</td>
<td>1.047</td>
<td>1.043</td>
<td>1.002</td>
<td>1.005</td>
<td>1.004</td>
<td>1.007</td>
<td>1.001</td>
<td>0.998</td>
<td>1.001</td>
<td>1.001</td>
</tr>
<tr>
<td>IOQ_par</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>1.011</td>
<td>1.016</td>
<td>1.014</td>
<td>1.010</td>
<td>1.004</td>
<td>1.001</td>
<td>1.007</td>
<td>1.007</td>
<td>1.002</td>
<td>1.002</td>
</tr>
<tr>
<td>test</td>
<td>1.363</td>
<td>1.307</td>
<td>1.135</td>
<td>1.108</td>
<td>1.076</td>
<td>1.075</td>
<td>1.049</td>
<td>1.036</td>
<td>1.034</td>
<td>1.032</td>
</tr>
<tr>
<td>IOQ_np</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.911</td>
<td>0.911</td>
<td>0.975</td>
<td>0.961</td>
<td>0.983</td>
<td>0.981</td>
<td>0.993</td>
<td>0.996</td>
<td>0.991</td>
<td>0.991</td>
</tr>
<tr>
<td>test</td>
<td>1.357</td>
<td>1.265</td>
<td>1.107</td>
<td>1.090</td>
<td>1.050</td>
<td>1.052</td>
<td>1.037</td>
<td>1.031</td>
<td>1.027</td>
<td>1.027</td>
</tr>
<tr>
<td>CEE_par</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>1.058</td>
<td>1.026</td>
<td>1.007</td>
<td>1.009</td>
<td>1.002</td>
<td>1.000</td>
<td>1.004</td>
<td>1.007</td>
<td>1.001</td>
<td>1.001</td>
</tr>
<tr>
<td>test</td>
<td>1.336</td>
<td>1.284</td>
<td>1.121</td>
<td>1.102</td>
<td>1.072</td>
<td>1.074</td>
<td>1.047</td>
<td>1.038</td>
<td>1.035</td>
<td>1.033</td>
</tr>
<tr>
<td>CEE_np</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>1.007</td>
<td>0.967</td>
<td>0.974</td>
<td>0.969</td>
<td>0.985</td>
<td>0.981</td>
<td>0.990</td>
<td>0.994</td>
<td>0.990</td>
<td>0.991</td>
</tr>
<tr>
<td>test</td>
<td>1.306</td>
<td>1.262</td>
<td>1.095</td>
<td>1.087</td>
<td>1.050</td>
<td>1.050</td>
<td>1.034</td>
<td>1.028</td>
<td>1.026</td>
<td>1.027</td>
</tr>
<tr>
<td>MCS_par</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>1.202</td>
<td>1.080</td>
<td>1.044</td>
<td>1.025</td>
<td>1.046</td>
<td>1.024</td>
<td>1.029</td>
<td>1.024</td>
<td>1.033</td>
<td>1.024</td>
</tr>
<tr>
<td>test</td>
<td>1.499</td>
<td>1.349</td>
<td>1.351</td>
<td>1.193</td>
<td>1.166</td>
<td>1.120</td>
<td>1.143</td>
<td>1.087</td>
<td>1.144</td>
<td>1.081</td>
</tr>
<tr>
<td>MCS_np</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>4.851</td>
<td>1.012</td>
<td>2.040</td>
<td>1.028</td>
<td>3.096</td>
<td>1.035</td>
<td>1.414</td>
<td>1.033</td>
<td>1.328</td>
<td>1.044</td>
</tr>
<tr>
<td>test</td>
<td>56.848</td>
<td>1.838</td>
<td>5.677</td>
<td>1.194</td>
<td>7.877</td>
<td>1.128</td>
<td>11.319</td>
<td>1.135</td>
<td>5.366</td>
<td>1.119</td>
</tr>
<tr>
<td>ICD_par</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>1.102</td>
<td>1.064</td>
<td>1.018</td>
<td>1.010</td>
<td>1.004</td>
<td>1.000</td>
<td>1.008</td>
<td>1.007</td>
<td>1.001</td>
<td>1.002</td>
</tr>
<tr>
<td>test</td>
<td>1.281</td>
<td>1.247</td>
<td>1.108</td>
<td>1.086</td>
<td>1.052</td>
<td>1.052</td>
<td>1.035</td>
<td>1.024</td>
<td>1.024</td>
<td>1.023</td>
</tr>
<tr>
<td>ICD_np</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>2.043</td>
<td>1.037</td>
<td>1.019</td>
<td>0.992</td>
<td>1.004</td>
<td>0.996</td>
<td>1.014</td>
<td>1.001</td>
<td>1.001</td>
<td>0.998</td>
</tr>
<tr>
<td>test</td>
<td>24.785</td>
<td>1.926</td>
<td>2.859</td>
<td>1.149</td>
<td>1.134</td>
<td>1.069</td>
<td>1.231</td>
<td>1.062</td>
<td>1.507</td>
<td>1.041</td>
</tr>
<tr>
<td>SQL_par</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>1.011</td>
<td>1.018</td>
<td>1.014</td>
<td>1.010</td>
<td>1.004</td>
<td>1.001</td>
<td>1.007</td>
<td>1.007</td>
<td>1.002</td>
<td>1.002</td>
</tr>
<tr>
<td>test</td>
<td>1.368</td>
<td>1.318</td>
<td>1.137</td>
<td>1.110</td>
<td>1.077</td>
<td>1.076</td>
<td>1.049</td>
<td>1.038</td>
<td>1.035</td>
<td>1.034</td>
</tr>
<tr>
<td>SQL_np</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>train</td>
<td>0.907</td>
<td>0.907</td>
<td>0.974</td>
<td>0.960</td>
<td>0.983</td>
<td>0.980</td>
<td>0.993</td>
<td>0.995</td>
<td>0.990</td>
<td>0.990</td>
</tr>
<tr>
<td>test</td>
<td>1.361</td>
<td>1.270</td>
<td>1.111</td>
<td>1.095</td>
<td>1.051</td>
<td>1.053</td>
<td>1.038</td>
<td>1.032</td>
<td>1.028</td>
<td>1.028</td>
</tr>
</tbody>
</table>
Chapter 10

Comparison of estimation methods for three covariates

After having analyzed the parameter selection in Methods MCS and SQI and the effect of the size of the data set, we now focus on the comparison of our methods for D-vine mean regression from Section 4.2 with an OLS or GLM and estimates coming from a tuned gradient boosting model and a gradient boosting model with default settings. Therefore, we will compare the computational speed of the different methods, their prediction performance measured by the RMSE and their numerical stability. The data sets that are used throughout this chapter always contain three covariates and a response which depends on all the covariates. Therefore, we used the conditional log-likelihood as defined in (3.11) as our selection criterion for the construction of the D-vine structure.

10.1 Results of Scenario $SL.3$

We start by comparing the methods with respect to the run time for predictions. All our simulation studies ran on a Linux cluster with an Intel Xeon E5-2697 v3 ("Haswell") CPU. For each replication of our simulations we used a separate core which implies that within one replication the predictions ran in serial. The average prediction times for each of the methods can be found in Table 10.1. First we observe that the OLS and the XGB predictions are a lot faster than the ones of D-vine mean regression. The reason for this is that their implementations are highly sophisticated and optimized on the machine level. In contrast, the methods for D-vine mean regression have so far only been implemented straightforward in R. Therefore, we only focus on the comparison of the five methods for D-vine mean regression. Here, we can also see big differences and for all methods the parametric version is a lot faster than the nonparametric version. This can be explained by the fact that when the pair-copulas have been estimated parametrically we get a closed form solution which can be calculated very fast. Another observation is that Methods IOQ and ICD are very slow, which is a result of the numerical integration that is involved. Method SQI is by far the fastest for both parametric and nonparametric estimates as it does
not include numerical integration and can benefit from the fast implementation of D-vine quantile regression in the \texttt{vinereg}-package (Nagler, 2018b). As the relative difference in computation times does not change for other scenarios, we will not investigate the run times for the other scenarios.

\textbf{Table 10.1:} Average computation times in seconds over 100 repetitions for 1500 predictions in Scenario $SL_3$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Par./Tuned</th>
<th>Nonpar./Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>XGB</td>
<td>309.95</td>
<td>4224.80</td>
</tr>
<tr>
<td>IOQ</td>
<td>24.64</td>
<td>125.97</td>
</tr>
<tr>
<td>CEE</td>
<td>284.44</td>
<td>1278.79</td>
</tr>
<tr>
<td>MCS</td>
<td>191.82</td>
<td>1690.79</td>
</tr>
<tr>
<td>ICD</td>
<td>1.38</td>
<td>43.25</td>
</tr>
<tr>
<td>SQI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We continue by analyzing the prediction performance of the different methods in Scenario $SL_3$. Figure 10.1 shows a boxplot of the training and test RMSE of the different methods. First of all, we note that in this scenario the OLS serves as a benchmark as all assumptions from ordinary least square estimation are met. We can see that for the training data both XGB models yield clearly the smallest RMSE, however, for the test data their performance is a lot worse. This indicates that even the tuned model is still overfitting (cf. Section 2.8) the training data. In contrast, the median performance of all methods for D-vine mean regression is very similar for the training and the test data and is very close to the benchmark of OLS. But, especially for Method MCS we can again observe that the upper quartile of the nonparametric version is a lot bigger than for the parametric version and in Methods MCS and ICD the mean RMSE of the nonparametric version is influenced by big outliers in the predictions for the test data (cf. Table B.11).

Another important result is that Methods IOQ and ICD are numerically not stable. This means for some data $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ the numerical integration that is involved produces an error resulting in an NA for the prediction. Although Table 10.2 indicates that only for a very little amount of data the prediction produces NAs, this is a big drawback of Methods IOQ and ICD. Therefore, so far Methods CEE and SQI are our methods of choice, as they are fastest, numerically stable and yield a small RMSE. Moreover, we note that for each of the two methods the nonparametric version is slightly better in terms of prediction performance and the parametric version is better in terms of speed. In order to verify that these methods are the best we continue with our next scenario, which now also includes a quadratic effect of the first covariate.

\textbf{Table 10.2:} Average percentage of produced NAs for the different methods in Scenario $SL_3$.

<table>
<thead>
<tr>
<th>Method</th>
<th>IOQ</th>
<th>ICD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
</tr>
<tr>
<td>parametric</td>
<td>0.16%</td>
<td>0.16%</td>
</tr>
<tr>
<td>nonparametric</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
10.1. RESULTS OF SCENARIO SL.3

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td></td>
<td>tuned</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parametric</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nonparametric</td>
</tr>
</tbody>
</table>

**Boxplot of RMSE for training data**

**Boxplot of RMSE for test data**

**Figure 10.1**: Boxplot of RMSE of training and test data set in Scenario SL.3.
10.2 Results of Scenario $QL.3$

First of all, recall that in this scenario the OLS model is not a benchmark anymore, as we did not include the quadratic term in the model. As a result, the OLS estimates are now outperformed by all other methods as we can see in Figure 10.2. Again, the default XGB model seems to overfit the training data as the RMSE for the training data is a lot smaller than for the test data. However, the tuned model can now reach quite similar results for both training and test data. When looking at the D-vine mean regression methods one can clearly see that the nonparametric versions perform better than the parametric ones. The reason for this is that parametric pair-copulas can only model monotonic dependencies but in this scenario we have a quadratic dependency. This becomes especially clear when looking at the residual plots in Figure 10.3. For the OLS model we can clearly see that the quadratic influence of covariate $X_1$ can not be modeled. Also the parametric copula regression only tracks this dependency partially. In contrast, the nonparametric version can handle this a lot better and the residuals are nicely distributed around 0. For the default XGB model we can now also see the overfitting, as there are many residuals very close to zero which belong to the training data and others that are scattered around zero with a much larger variance, belonging to the test data.

As we could already observe in Scenario $SL.3$, Methods $CEE$ and $SQI$ yield very good results in terms of the RMSE. Moreover, Methods $MCS$ and $ICD$ show a big variation of the RMSE in the nonparametric version. However, if one only compares the parametric versions Method $ICD$ is best. But, one should keep in mind that it is as well as Method $IOQ$ not numerically stable as we can see in Table 10.3.

<table>
<thead>
<tr>
<th></th>
<th>IOQ</th>
<th>ICD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
</tr>
<tr>
<td>parametric</td>
<td>0.09%</td>
<td>0.08%</td>
</tr>
<tr>
<td>nonparametric</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
Figure 10.2: Boxplot of RMSE of training and test data set in Scenario QL.3.
Figure 10.3: Plot of covariate $X_1$, which has quadratic influence, versus the residuals of the different models for training data (grey) and test data (orange). The dotted lines indicate the 95%-confidence interval.
10.3 Results of Scenario $GR.3$

In this scenario the used data follows the assumptions of gamma regression as explained in 2.5.2. Therefore, we will now use a GLM performing a gamma regression instead of an OLS as a benchmark model. In Figure 10.4 we can see that the tuned XGB model as well as all methods of D-vine mean regression achieve very similar results as the benchmark GLM. The nonparametric version of Method ICD can even outperform the GLM. However, we should keep in mind that this Method is numerically not stable and produces NAs for some of the predictions. In general the parametric version performs a little worse than the nonparametric version for all of the methods except for Method MCS.

10.4 Results of Scenario $DF.4$

So far, we always considered our data to follow a regression structure and we have seen, that D-vine mean regression can fit all these scenarios very well. Now, we use a data set that indeed follows a D-vine copula and assess the prediction performance of all methods in this setup. We only allowed for bivariate Gumbel copulas in the pair-copula construction of the D-vine in order to get a data set with tail-dependence and a fully parametric representation of the D-vine. As one would expect, the parametric versions of D-vine mean regression can now outperform all other models on the test data as we can see in Figure 10.5. Especially Methods IOQ and SQI yield the lowest median RMSE and also show only little variation among all 100 repetitions. They also reach the lowest mean RMSE although the other methods mostly reach very similar results (cf. Table B.14).

As we have already seen in many scenarios before, the nonparametric version of Method MCS has a big upper quartile, especially for the training data. Again, this is a result of a bad estimation of a pair-copula in one of the tails of the distribution of the response $Y$. 
Figure 10.4: Boxplot of RMSE of training and test data set in Scenario $G\mathcal{R}.3$. 

**Boxplot of RMSE for training data**

**Boxplot of RMSE for test data**
10.4. RESULTS OF SCENARIO $D_F.4$

Figure 10.5: Boxplot of RMSE of training and test data set in Scenario $D_F.4$. 


10.5 Results of Scenario $\mathcal{D}R.4$

After having used only Gumbel copulas in the PCC of the D-vine in Scenario $\mathcal{D}F.4$, we now choose the family of the pair-copulas randomly from one of the families given in Table 6.1. Again, we can observe that the parametric versions of D-vine mean regression yield the best results, however, Method CEE performs a lot worse than all other methods. The reason for that is not clear yet. The best results on the test data are achieved by the parametric version of Method ICD. Nevertheless, when only considering the numerical stable methods the median RMSE is lowest for Method SQI which is also only slightly worse than the one of Method ICD. The exact values of the median and mean RMSE for all the methods can be found in Table B.14 in the appendix.

10.6 Results of Scenario $\mathcal{R}R.4$

In the last scenario with three covariates we switch from a D-vine to a R-vine as underlying copula, which allows for even more complex dependence structures. However, as we are only in four-dimensions the copulas are still special cases of R-vines, namely C-vines or D-vines. The findings for the resulting Scenario $\mathcal{R}R.4$ are shown in Figure 10.7. The most obvious observation is as in the previous scenario that Method CEE performs a lot worse than all other methods. The best results in terms of smallest mean and median RMSE and numerical stability are again reached by Method SQI (cf. Table B.16).

Until now, we have seen that D-vine mean regression works very well for many different setups and that in scenarios where we have an optimal benchmark it reaches similar results. Moreover, gradient boosting with regression trees cannot outperform the best method for D-vine mean regression which is – summarized over all scenarios so far – Method SQI. In the next chapter we will investigate whether D-vine mean regression still works well when the number of covariates increases and whether the algorithm detects unimportant covariates that are independent of the response. However, before that we shortly analyze which copula families have been selected in the pair-copula constructions of the scenarios, that have been investigated so far.
10.6. RESULTS OF SCENARIO RR.4

Figure 10.6: Boxplot of RMSE of training and test data set in Scenario RR.4.
Figure 10.7: Boxplot of RMSE of training and test data set in Scenario 𝑅𝑅.4.
10.7 Analyzing estimated pair-copulas

We conclude this chapter by having a short look at the pair-copulas that have been selected in the parametric vine copula models. In Table 10.4 we can see that for Scenario $\mathcal{S}L.3$ the copula family that was chosen most often is the Gaussian copula with 35.5% of all the pair-copulas. Keeping in mind the results of Section 6.2.3 this is what we would expect. As the t and the BB8 copula can attain very similar shapes as the Gaussian copula, it is not surprising that they have also been fitted many times. For the BB8 copula we have already seen in Section 3.3 that its contour plots are very similar to a Gaussian copula and that it allows for no tail dependence.

The results for Scenario $\mathcal{G}R.3$ are very similar which can be explained with the fact that the underlying copula of the joint distribution of $(Y, X)$ is again the Gaussian copula, as we have shown in Section 6.2.3.

Moreover, in Scenario $\mathcal{D}F.4$ we only allowed for Gumbel copulas during the generation of our data set, hence, mainly fitting BB1 and BB6 copulas is a reasonable result, as they have the Gumbel copula as a limiting/special case (cf. Section 3.3).

In all other scenarios we do not know the exact underlying copula structure and therefore we cannot give a statement which pair-copulas we would expect to be fitted by the algorithm. Furthermore, it is not surprising that the Joe, Frank, and Gumbel families are chosen only very rarely, as for each of them there exists at least one limiting case in the BB1, BB6, BB7 or BB8 copula families (cf. Section 3.3).

<table>
<thead>
<tr>
<th>Scenario</th>
<th>BB1</th>
<th>BB6</th>
<th>BB7</th>
<th>BB8</th>
<th>Clayton</th>
<th>Frank</th>
<th>Gaussian</th>
<th>Gumbel</th>
<th>Joe</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{S}L.3$</td>
<td>1.3%</td>
<td>0.2%</td>
<td>5.5%</td>
<td>27.0%</td>
<td>5.5%</td>
<td>0.8%</td>
<td>35.5%</td>
<td>0.0%</td>
<td>1.3%</td>
<td>22.8%</td>
</tr>
<tr>
<td>$\mathcal{Q}L.3$</td>
<td>2.0%</td>
<td>0.2%</td>
<td>9.3%</td>
<td>40.5%</td>
<td>5.7%</td>
<td>1.2%</td>
<td>8.8%</td>
<td>0.0%</td>
<td>1.5%</td>
<td>30.8%</td>
</tr>
<tr>
<td>$\mathcal{G}R.3$</td>
<td>9.5%</td>
<td>0.2%</td>
<td>6.0%</td>
<td>33.7%</td>
<td>5.5%</td>
<td>1.0%</td>
<td>26.3%</td>
<td>0.2%</td>
<td>1.3%</td>
<td>16.3%</td>
</tr>
<tr>
<td>$\mathcal{D}F.4$</td>
<td>37.5%</td>
<td>34.0%</td>
<td>9.4%</td>
<td>7.4%</td>
<td>0.7%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>8.1%</td>
<td>0.3%</td>
<td>2.5%</td>
</tr>
<tr>
<td>$\mathcal{D}R.4$</td>
<td>29.0%</td>
<td>9.2%</td>
<td>5.0%</td>
<td>10.5%</td>
<td>1.7%</td>
<td>7.8%</td>
<td>8.2%</td>
<td>4.0%</td>
<td>1.3%</td>
<td>23.3%</td>
</tr>
<tr>
<td>$\mathcal{R}R.4$</td>
<td>32.5%</td>
<td>8.3%</td>
<td>4.5%</td>
<td>13.8%</td>
<td>1.7%</td>
<td>6.0%</td>
<td>7.2%</td>
<td>1.5%</td>
<td>0.7%</td>
<td>23.8%</td>
</tr>
</tbody>
</table>
Chapter 11

Comparison of estimation methods for ten covariates

By now, all our simulation studies were based on rather simple data sets containing only three covariates. Now, we increase the number of covariates to ten, but let two of the covariates be independent of the response $Y$. Besides analyzing the prediction performance we will also investigate whether the independent covariates are recognized by D-vine mean regression and therefore not included in the vine structure. Hence, we now choose the AIC-corrected conditional log-likelihood as defined in (3.12) to be our selection criterion during the construction of the D-vine structure.

11.1 Results of Scenario $SL_{10}$

The first scenario that we analyze uses a data set that fulfills the assumptions of linear regression. So the OLS estimate can again be seen as the benchmark model. In Figure 11.1 we can see that for all methods the parametric version yields significantly better results than the nonparametric counterpart. This observation can be explained when looking at the vine structures that are fit in the models. In Table 11.1 we can see that the parametric D-vine recognizes the independent covariates $X_9$ and $X_{10}$ in more than 90% of the repetitions. In contrast, the nonparametric D-vine includes the independent covariates in about 60% of the repetitions yielding a misspecified model and therefore also worse predictions and hence a higher RMSE. Moreover, Methods IOQ and SQI achieve the lowest RMSE which gets quite close to the OLS benchmark and is in the median and mean even slightly less than the one of the tuned XGB model (cf. Table B.17).

Table 11.1: Appearance of covariates $X_1, \ldots, X_{10}$ in the vine structure within the 100 repetitions of Scenario $SL_{10}$.

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$X_5$</th>
<th>$X_6$</th>
<th>$X_7$</th>
<th>$X_8$</th>
<th>$X_9$</th>
<th>$X_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>parametric</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>7</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>nonparametric</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>59</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>
Figure 11.1: Boxplot of RMSE of training and test data set in Scenario $S\mathcal{L}.10$. 
11.2 Results of Scenario QL.10

Now, we consider a scenario where we have a very complex dependence structure between the response $Y$ and the covariates $X = (X_1, \ldots, X_{10})$ containing quadratic effects of the covariates and also interactions between two covariates. First of all, in Figure 11.2 we can see that both gradient boosting models highly overfit the training data, as the RMSE of the test data is a lot bigger than for the training data. Moreover, the RMSE of the OLS is very big, which is obvious since we did not include any quadratic terms or interactions in the model. The models from D-vine mean regression do also not perform well in this setup. One can explain this result by looking at the conditional pair-copulas that are fit in the PCC. The main assumption that was used in all our theory is the simplifying assumption, stating that the conditional copula densities in the PCC do not depend on the actual value of the conditioning variable, this means we assumed that $c_{i,j:D}(\cdot, \cdot; u_D) \equiv c_{i,j:D}(\cdot, \cdot)$. However, for this scenario we could show that the simplifying assumption might be violated and thus it is not surprising that D-vine mean regression does not perform well. Developing a method that allows to model D-vine copulas where the simplifying assumption is not fulfilled is beyond the scope of this thesis and hence a possible topic for further research.

11.3 Results of Scenario GR.10

In this scenario the underlying data structure follows again the assumptions of gamma regression (cf. 2.5.2). Hence, we will use a GLM instead of the OLS as a benchmark model. Again, D-vine mean regression achieves very similar results as the benchmark model and the tuned XGB model. Moreover, we can see that on the test data the parametric versions yield a slightly better performance than the nonparametric ones.

If we look at the vine order we can see that for the parametric model the influential variables are considered most of the time. Only covariates with small values of $\beta$ are sometimes disregarded. Moreover, the non-influential variables $X_9$ and $X_{10}$ are not included in the model most of the time. In contrast, the nonparametric model seems to have problems distinguishing between influential and non-influential variables (cf. Table 11.2).

| Table 11.2: Appearance of covariates $X_1, \ldots, X_{10}$ in the vine structure within the 100 repetitions of Scenario GR.10. |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| $X_1$       | $X_2$       | $X_3$       | $X_4$       | $X_5$       | $X_6$       | $X_7$       | $X_8$       | $X_9$       | $X_{10}$    |
| $\beta$     | 0.200       | 0.175       | 0.150       | 0.125       | 0.100       | 0.075       | 0.050       | 0.025       | -           |
| parametric   | 100         | 100         | 100         | 100         | 96          | 70          | 28          | 9           | 20          |
| nonparametric| 100         | 98          | 95          | 84          | 73          | 66          | 64          | 61          | 56          | 59          |
Chapter 11. Comparison for Ten Covariates

Figure 11.2: Boxplot of RMSE of training and test data set in Scenario QC.10.
11.3. RESULTS OF SCENARIO GR.10

Figure 11.3: Boxplot of RMSE of training and test data set in Scenario GR.10.
11.4 Results of Scenario $\mathcal{DF}_{11}$

The last scenario that we investigated is again based on data sets that are sampled from a D-vine copula where we only allowed for Gumbel copulas in the PCC. As we have already seen in the four-dimensional D-vine scenarios, the parametric D-vine methods achieve the highest prediction performance and in the median the RMSE is quite similar for all the methods. Moreover, D-vine mean regression can slightly outperform the tuned gradient boosting model. The median and mean RMSE of all the models can be found in Table B.20.

11.5 Conclusion of simulation studies

We conclude this chapter with a summary over the results that have been found throughout the simulation studies. First of all, we have seen that Method MCS does only work well in the parametric version, the nonparametric version has produced very high prediction errors in many different scenarios and should therefore not be used. Moreover, we have seen that Methods IOQ and ICD are most of the time very good in terms of having a low RMSE. Nevertheless, using these methods can result in NAs as the numerical integration is not stable and can thus yield errors. Hence, we remain with methods CEE and SQI which mostly achieve very similar results and are both numerically stable. However, in the scenarios where we had a vine copula as underlying data structure, Method CEE deteriorated a lot. Therefore, the overall best method is simplified quantile integration which has already been implemented in the R-package vinereg. In order to decide whether to use the parametric or the nonparametric version one has to analyze whether there are only monotonic dependencies or also more complex dependencies involved. If there are only monotonic dependencies the parametric version will yield the best results, otherwise one should choose the nonparametric version.
Figure 11.4: Boxplot of RMSE of training and test data set in Scenario $\mathcal D F.11$. 
Chapter 12

Application to abalone data set

After having verified that D-vine mean regression, as introduced in Chapter 4, yields very good results in many different setups, we now want to apply it to a real-world data set. Again, we will compare our results with a gradient boosting model. For our application we will use the abalone data set which is included in the R-package PivotalR (Pivotal Inc., 2017).

12.1 Introduction to data set

The data set contains measurements from a population of abalones. The goal is to predict the age of the abalone from these physical measurements. In general, the age can be determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope. As this is a very time-consuming task one tries to use other measurements to predict the age as these are a lot easier to obtain. The data set contains the following variables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Measurement Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex</td>
<td>nominal</td>
<td>–</td>
<td>M, F, and I (infant)</td>
</tr>
<tr>
<td>Length</td>
<td>continuous</td>
<td>mm</td>
<td>Longest shell measurement</td>
</tr>
<tr>
<td>Diameter</td>
<td>continuous</td>
<td>mm</td>
<td>perpendicular to length</td>
</tr>
<tr>
<td>Height</td>
<td>continuous</td>
<td>mm</td>
<td>with meat in shell</td>
</tr>
<tr>
<td>Whole weight</td>
<td>continuous</td>
<td>grams</td>
<td>whole abalone</td>
</tr>
<tr>
<td>Shucked weight</td>
<td>continuous</td>
<td>grams</td>
<td>weight of meat</td>
</tr>
<tr>
<td>Viscera weight</td>
<td>continuous</td>
<td>grams</td>
<td>gut weight (after bleeding)</td>
</tr>
<tr>
<td>Shell weight</td>
<td>continuous</td>
<td>grams</td>
<td>after being dried</td>
</tr>
<tr>
<td>Rings</td>
<td>integer</td>
<td>–</td>
<td>+1.5 gives the age in years</td>
</tr>
</tbody>
</table>

We will split the data set by variable ’Sex’ into three disjoint sets – one for each of the levels M (male), F (female) and I (infant) – and will do our analysis only for the subset containing all female abalones. We end up with a data set of $n = 1306$
observations. As the goal is to predict the age of an abalone given physical measurements, we will treat the variable ‘Rings’ as our response and all other variables will be the covariates.

12.2 Exploratory data analysis

First of all, we do some exploratory data analysis in order to see whether there are some obvious patterns that indicate dependence of the response ‘Rings’ on some of the other variables. Figure 12.1 shows a pairs plot of the observed data. We can clearly see that all covariates seem to have influence on the response.

![Figure 12.1: Pairs plot of abalone data on original scale.](image-url)
Further analyses have shown that 'Rings' and 'Height' only take a small number of integer values. Therefore, when applying D-vine mean regression we will also consider models where we use jittering (cf. 3.7) for these two variables. After applying the PIT to the marginal distributions, we can also get a pairs plot for the data on the copula scale. In Figure 12.2 we can see that the PIT for variables 'Rings' and 'Height' does not yield a perfectly uniform distribution which is a result of the discreteness of these two variables. Moreover, we can see that there is some dependence between the response and all the covariates and that the covariates itself highly depend on each other.

**Figure 12.2:** Pairs plot of abalone data on copula scale.
CHAPTER 12. APPLICATION TO ABALONE DATA SET

12.3 Model selection and comparison

Now, we want to fit different models to the data and compare their prediction performance. We used 5-fold cross validation (cf. Section 2.8) in order to fit the following models to the data:

Table 12.2: Models that are fitted to the abalone data set.

<table>
<thead>
<tr>
<th>Name</th>
<th>Method</th>
<th>Selection criterion</th>
<th>Jittering</th>
</tr>
</thead>
<tbody>
<tr>
<td>par.cll</td>
<td>Parametric D-vine mean regression</td>
<td>cll</td>
<td>No</td>
</tr>
<tr>
<td>par.AIC</td>
<td>Parametric D-vine mean regression</td>
<td>cll(^{\text{AIC}})</td>
<td>No</td>
</tr>
<tr>
<td>np.cll</td>
<td>Nonparametric D-vine mean regression</td>
<td>cll</td>
<td>No</td>
</tr>
<tr>
<td>np.AIC</td>
<td>Nonparametric D-vine mean regression</td>
<td>cll(^{\text{AIC}})</td>
<td>No</td>
</tr>
<tr>
<td>jit.cll</td>
<td>Nonparametric D-vine mean regression</td>
<td>cll</td>
<td>Yes</td>
</tr>
<tr>
<td>jit.AIC</td>
<td>Nonparametric D-vine mean regression</td>
<td>cll(^{\text{AIC}})</td>
<td>Yes</td>
</tr>
<tr>
<td>xgb.def</td>
<td>Boosting with default parameters</td>
<td>–</td>
<td>No</td>
</tr>
<tr>
<td>xgb.tun</td>
<td>Boosting with tuned parameters</td>
<td>–</td>
<td>No</td>
</tr>
</tbody>
</table>

For all models of D-vine mean regression we used Method SQI as the simulation studies have shown that it yields the best results. In order to fit the gradient boosting models we used again the R-packages \texttt{xgboost} (Chen et al., 2018) and \texttt{caret} (Kuhn et al., 2018). The column ‘Jittering’ of Table 12.2 indicates whether jittering was used for the discrete variables ‘Rings’ and ‘Height’. If jittering is used, we automatically have to use nonparametric D-vine mean regression as the parametric version does not work in this case (Nagler, 2018a).

When analyzing the vine-order of the D-vine mean regression models we find that all models having the cll as selection criterion include all variables in the model. In contrast, the models using the AIC-corrected cll only include two or three of the covariates. The two variables that are included in all of the models are ‘Shell weight’ which has the highest dependence with the response ‘Rings’ and ‘Shucked weight’ which has the least dependence with ‘Shell weight’ (cf. Figure 12.2). Table 12.3 summarizes the results, including also the RMSE for the training and test data. We can see that model \texttt{jit.aic} using the AIC-corrected cll and jittering for the discrete variables ‘Rings’ and ‘Height’ achieves the smallest RMSE on the test data. However, it is only slightly better than the nonparametric D-vine model \texttt{np.aic} that uses the AIC-corrected cll but regards ‘Rings’ and ‘Height’ as continuous. Moreover, the tuned gradient boosting model \texttt{xgb.tun} also shows a very similar performance. We highlighted the RMSE of the test data for these three models in gray. Figure 12.3 shows the contour plots of the pair-copulas that have been fit in model \texttt{jit.aic}. 
Table 12.3: Results of different models for the abalone data. The best models for D-vine mean regression with and without jittering and the tuned gradient boosting model are highlighted in grey.

<table>
<thead>
<tr>
<th>Model</th>
<th>RMSE train</th>
<th>RMSE test</th>
<th>Vine-order (vine models) / Feature importance (xgb models)</th>
</tr>
</thead>
<tbody>
<tr>
<td>par.cll</td>
<td>2.50</td>
<td>2.40</td>
<td>shell, shucked, whole, viscera, height, length, diameter</td>
</tr>
<tr>
<td>par.aic</td>
<td>2.53</td>
<td>2.34</td>
<td>shell, shucked, whole</td>
</tr>
<tr>
<td>np.cll</td>
<td>2.46</td>
<td>2.38</td>
<td>shell, shucked, length, viscera, height, diameter, whole</td>
</tr>
<tr>
<td>np.aic</td>
<td>2.52</td>
<td>2.31</td>
<td>shell, shucked</td>
</tr>
<tr>
<td>jit.cll</td>
<td>2.54</td>
<td>2.40</td>
<td>shell, shucked, length, whole, viscera, diameter, h.ord</td>
</tr>
<tr>
<td>jit.aic</td>
<td>2.61</td>
<td>2.28</td>
<td>shell, shucked</td>
</tr>
<tr>
<td>xgb.def</td>
<td>0.72</td>
<td>2.54</td>
<td>shell, shucked, whole, viscera, height, length, diameter</td>
</tr>
<tr>
<td>xgb.tun</td>
<td>2.23</td>
<td>2.36</td>
<td>shell, shucked, whole, viscera, length, height, diameter</td>
</tr>
</tbody>
</table>

Figure 12.3: Contour plots of pair-copulas in model jit.aic.
Chapter 13

Conclusion and outlook

In this thesis we investigated five different methods for conditional mean estimation, exploiting the approach of Kraus and Czado (2017) for D-vine quantile regression. Moreover, we gave a detailed explanation for gradient boosting with regression trees and continued by an extensive simulation study comparing D-vine mean regression and gradient boosting. We found that the five methods for D-vine mean regression do not all perform equally well. Overall, Method SQI performed best in terms of prediction performance, numerical stability and computational speed. When comparing D-vine mean regression to gradient boosting or other well established methods like ordinary least square estimation or generalized linear models, we could show that the models for D-vine mean regression yield comparably good results. It can handle classical regression problems including quadratic effects of covariates as well as more complex models with vine copulas as underlying dependence structures. In some of the scenarios D-vine mean regression could even outperform all other methods in terms of having the smallest RMSE.

In order to verify the results from the simulation studies, we also applied our methods to a real-world data set. Again D-vine mean regression showed a very good prediction performance and could outperform the gradient boosting model.

Nevertheless, there is still a lot of research that can be done in order to further improve D-vine mean regression. One possible improvement is to proceed from D-vine mean regression to R-vine mean regression. This means by fitting an R-vine copula to the data instead of a D-vine, we allow for a more general structure of the pair-copula construction which enables to model more general and complex dependence structures. Another interesting field of research is to investigate what happens if the simplifying assumption is no longer fulfilled, as this is an important assumption of the current methodology. If the presented approach can be extended to non-simplified vine copulas this would allow for the modelling of even more complex regression problems.
List of Figures

2.1 Example for a highly overfitted polynomial model (red) and good fitting least square model (green). The polynomial model fits the observed data perfectly, however, for future observation it will very likely yield no good predictions. .................................................. 11

3.1 Bivariate copula densities of Gaussian (left), Gumbel (middle) and Clayton (right) copula each which Kendall’s τ of 0.6. ......................... 17
3.2 Normalized contour plots of Gaussian (left), Gumbel (middle) and Clayton (right) copula each which Kendall’s τ of 0.6. ......................... 17
3.3 Normalized contour plots of BB8 copula with Kendall’s τ of 0 (left), 0.22 (middle) and 0.68 (right). ............................ 18
3.4 Graph theoretic representation of a 5-dimensional D-vine copula with order $X_1 - X_2 - X_3 - X_4 - X_5$. The pair-copulas of the PCC are written above each edge. .................................................. 21
3.5 Exemplary R-vine tree sequence on five elements. .......................... 23

5.1 Left: Partitioning of two-dimensional covariate space $\mathcal{X} = [0, 1]^2$ into five regions $R_1, \ldots, R_5$ via binary splits. Middle: Regression tree corresponding to the partitioning on the left. Right: Regression surface according to partitioning and regression tree on the left based on some exemplary data. (Hastie et al., 2001) ............................... 38
5.2 Illustration of algorithm for gradient boosting with regression trees under quadratic loss. .................................................. 46

7.1 Boxplot of RMSE of training and test data set in Scenario $\text{MCS.1}$ according to different sample sizes $m$ of the Monte Carlo simulation. 64
7.2 Boxplot of RMSE of training and test data set in Scenario $\text{MCS.2}$ according to different sample sizes of the Monte Carlo simulation. 66
7.3 Boxplot of RMSE of training and test data set in Scenario $\text{MCS.3}$ according to different sample sizes of the Monte Carlo simulation. 67
7.4 Computation time for OLS and Method MCS for different Monte Carlo sample sizes $m$. .................................................. 68
7.5 Boxplot of RMSE of training and test data set in Scenario $\text{MCS.4}$ according to different sample sizes of the Monte Carlo simulation. 69
LIST OF FIGURES

8.1 Boxplot of RMSE of training and test data set in Scenario $S_{Q1.1}$ for different number of quantiles in Method SQI. ........................................... 72

9.1 Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario $DS.1$ according to different sizes of the data set. 74

10.1 Boxplot of RMSE of training and test data set in Scenario $SL.3$. .................. 79
10.2 Boxplot of RMSE of training and test data set in Scenario $QC.3$. ............. 81
10.3 Plot of covariate $X_1$, which has quadratic influence, versus the residuals of the different models for training data (grey) and test data (orange). The dotted lines indicate the 95%-confidence interval. ............. 82
10.4 Boxplot of RMSE of training and test data set in Scenario $GR.3$. ............ 84
10.5 Boxplot of RMSE of training and test data set in Scenario $DF.4$. ............ 85
10.6 Boxplot of RMSE of training and test data set in Scenario $DR.4$. ............ 87
10.7 Boxplot of RMSE of training and test data set in Scenario $RR.4$. ............ 88

11.1 Boxplot of RMSE of training and test data set in Scenario $SL.10$. ............ 92
11.2 Boxplot of RMSE of training and test data set in Scenario $QC.10$. .......... 94
11.3 Boxplot of RMSE of training and test data set in Scenario $GR.10$. .......... 95
11.4 Boxplot of RMSE of training and test data set in Scenario $DF.11$. .......... 97

12.1 Pairs plot of abalone data on original scale. ........................................... 100
12.2 Pairs plot of abalone data on copula scale. ............................................ 101
12.3 Contour plots of pair-copulas in model $jit.aic$. ..................................... 103

B.1 Boxplot of RMSE of training and test data set in Scenario $S_{Q1.2}$ for different number of quantiles in Method SQI. ........................................... 130
B.2 Boxplot of RMSE of training and test data set in Scenario $S_{Q1.3}$ for different number of quantiles in Method SQI. ........................................... 131
B.3 Boxplot of RMSE of training and test data set in Scenario $S_{Q1.4}$ for different number of quantiles in Method SQI. ........................................... 132
B.4 Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario $DS.2$ according to different sizes of the data set. 134
B.5 Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario $DS.3$ according to different sizes of the data set. 135
B.6 Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario $DS.4$ according to different sizes of the data set. 135
## List of Tables

2.1 Exemplary data containing continuous variables *weight* and *age* and categorical variables *gender* and *country*. ............................................. 9

2.2 Exemplary data containing continuous variables *weight* and *age* and dummy variables for the former categorical variables *gender* and *country* after application of dummy variable encoding. ................. 10

5.1 Exemplary data set for Example 5.2 on gradient boosting with regression trees. ................................................................. 39

5.2 Initialization of boosting algorithm with residual sum of squares $SSR = 2642$. ................................................................. 40

5.3 Step one of boosting algorithm, decreasing residual sum of squares to $SSR = 2058.8$. ................................................................. 41

5.4 Step two of boosting algorithm, yielding residual sum of squares $SSR = 1395.25$. ................................................................. 41

5.5 Overview of all tuning parameters in gradient boosting with regression trees. ................................................................. 48

6.1 Allowed copula families and values of Kendall's $\tau$ for pair-copulas in tree $j$ in data setup (DR4) and (RR4). ............................................. 55

6.2 Overview of data structures which are used throughout the simulation studies. ........................................................................... 55

6.3 Overview over all the different methods/models that are used throughout the simulation studies. Column *Abbreviation* shows the short name and column *Full name* describes the model. ............................................. 59

6.4 Overview of all scenarios with used data structures and parameters. ................................................................. 61

8.1 Mean RMSE for training and test data in Scenario $SQI.1$ for different number of quantiles in Method SQI. ............................................. 71

9.1 Mean and median RMSE for training and test data in Scenario $DS.1$ according to different data set sizes. ............................................. 75

10.1 Average computation times in seconds over 100 repetitions for 1500 predictions in Scenario $SL.3$. ............................................. 78

10.2 Average percentage of produced NAs for the different methods in Scenario $SL.3$. ............................................. 78
10.3 Average percentage of produced NAs for the different methods in Scenario $QL.3$. ......................................................... 80
10.4 Frequency of appearances of different pair-copula families in the pair-copula constructions for different scenarios. ......................... 89

11.1 Appearance of covariates $X_1, \ldots, X_{10}$ in the vine structure within the 100 repetitions of Scenario $SL.10$. ........................................... 91
11.2 Appearance of covariates $X_1, \ldots, X_{10}$ in the vine structure within the 100 repetitions of Scenario $GR.10$. ................................. 93

12.1 Variables of abalone data set. .......................................................... 99
12.2 Models that are fitted to the abalone data set. ................................. 102
12.3 Results of different models for the abalone data. The best models for D-vine mean regression with and without jittering and the tuned gradient boosting model are highlighted in grey. ......................... 103

B.1 Mean and median RMSE for training and test data set in Scenario $MCS.1$ according to different sample sizes $m$ of the Monte Carlo simulation. .......................................................... 127
B.2 Mean and median RMSE for training and test data set in Scenario $MCS.2$ according to different sample sizes of the Monte Carlo simulation. .......................................................... 128
B.3 Mean and median RMSE for training and test data set in Scenario $MCS.3$ according to different sample sizes of the Monte Carlo simulation. .......................................................... 128
B.4 Mean and median RMSE for training and test data set in Scenario $MCS.4$ according to different sample sizes of the Monte Carlo simulation. .......................................................... 129
B.5 Mean RMSE for training and test data in Scenario $SQI.2$ for different number of quantiles in Method SQI. ........................................ 133
B.6 Mean RMSE for training and test data in Scenario $SQI.3$ for different number of quantiles in Method SQI. ........................................ 133
B.7 Mean RMSE for training and test data in Scenario $SQI.4$ for different number of quantiles in Method SQI. ........................................ 133
B.8 Mean and median RMSE for training and test data in Scenario $DS.2$ according to different data set sizes. ...................................... 136
B.9 Mean and median RMSE for training and test data in Scenario $DS.3$ according to different data set sizes. ...................................... 137
B.10 Mean and median RMSE for training and test data in Scenario $DS.4$ according to different data set sizes. ...................................... 138
B.11 Median and mean RMSE for training and test data in Scenario $SL.3$. 139
B.12 Median and mean RMSE for training and test data in Scenario $QL.3$. 139
B.13 Median and mean RMSE for training and test data in Scenario $GR.3$. 140
B.14 Median and mean RMSE for training and test data in Scenario $DF.4$. 140
B.15 Median and mean RMSE for training and test data in Scenario $DR.4$. 141
B.16 Median and mean RMSE for training and test data in Scenario $RR.4$. 141
B.17 Median and mean RMSE for training and test data in Scenario SL 10.142
B.18 Median and mean RMSE for training and test data in Scenario QC 10.142
B.19 Median and mean RMSE for training and test data in Scenario GR 10.143
B.20 Median and mean RMSE for training and test data in Scenario DF 11.143
Bibliography


Appendix A

Implementation of methods for D-vine mean regression in R

A.1 Integration over quantiles

```r
# Function to estimate cond. EV using integration over quantiles
predict.mean.IOQ <- function(object, newdata, uscale = FALSE)
{
    # object: an object coming from the function vinereg()
    # newdata: a matrix containing the values of the covariates.
    # Has to contain all the covariates which are in the
    # D-vine of object.
    # uscale: indicator whether newdata is already on u-scale

    # check if margins were estimated
    if (is.null(object$margins[[1]]) & (!uscale))
    {
        warning("no margins have been estimated,
            setting uscale = TRUE")
        uscale <- TRUE
    }

    # check if all variables in the model are in newdata
    missing_vars <- setdiff(colnames(object$model_frame)[-1],
                              colnames(newdata))
    if (length(missing_vars) > 0)
        stop("'newdata' is missing variables ",
             paste(missing_vars, sep = "", "", "", "")

    # remove unused variables
    selected_vars <- match(object$order, colnames(newdata))
    newdata <- newdata[, selected_vars, drop = FALSE]
}
```
n <- nrow(newdata)
y_hat <- numeric(n)

# function to calculate a single quantile for a single set of covariates
pred_quant <- function(x,i)
{
  quant <- as.numeric(predict(object, newdata[i,], alpha=x))
}

# estimate cond. EV for each set of covariates (for each row of newdata)
for(i in 1:n)
{
  # calculate the integral. In case of error NAs are produced
  y_hat[i] <- try(integrate(function(x)
    pred_quant(x,i),0,1)$value, TRUE)
}

return(as.numeric(y_hat))

A.2 Calculating the estimating equation

# Function to estimate conditional EV using estimating equation
predict.mean.CEE <- function(object, newdata, uscale = FALSE)
{
  # object: an object coming from the function vinereg()
  # newdata: a matrix containing the values of the covariates.
  # Has to contain all the covariates which are in the D-vine of object.
  # uscale: indicator whether newdata is already on u-scale

  # check if margins were estimated
  if (is.null(object$margins[[1]]) & (!uscale))
  {
    warning("no margins have been estimated, setting uscale = TRUE")
    uscale <- TRUE
  }
  if (!(is.null(object$margins[[1]])) & (uscale))
  {
A.2. Calculating the estimating equation

```r
warning("margins have been estimated, 
     setting uscale = FALSE")
uscale <- FALSE

# check if all variables in the model are in newdata
missing_vars <- setdiff(colnames(object$model.frame)[-1], 
colnames(newdata))
if (length(missing_vars) > 0)
  stop("'newdata' is missing variables ", 
paste(missing_vars, sep = "'", ","), ",")

# expand factors and make ordered variables numeric
x <- cctools::expand_as_numeric(newdata[, colnames(object$model.frame)[-1], 
drop = FALSE])

# remove unused variables
selected_vars <- match(object$order, colnames(x))
u <- x[, selected_vars, drop = FALSE]
d <- ncol(u)

# transform to uniform scale
if (!uscale)
  for (j in 1:d)
    { u[, j] <- pkde1d(x[, j], 
                      object$Margins[[selected_vars[j] + 1]])
    }  

# we need the y-values of the training data set and transform 
# it to u-scale
y <- object$model.frame$y
v <- pkde1d(y, object$Margins[[1]])

# write function that estimates the cond. EV for one set of 
# covariates u
pred_mean <- function(object, y, v, u)
{
  
```

APPENDIX A. IMPLEMENTATION OF D-VINE MEAN REGRESSION

vu <- cbind(v, matrix(u, nrow=length(v), ncol=d, byrow = TRUE))
cop_dens <- dvinecop(vu, object$vine)
y_hat <- sum(y*cop_dens)/sum(cop_dens)
return(y_hat)

# afterwards loop through all the rows of newdata applying
# this function
n <- nrow(newdata)
y_hat <- numeric(n)
for(i in 1:n)
{
y_hat[i] <- pred_mean(object, y, v, u[i,])
}
return(y_hat)

A.3 Monte Carlo simulation

# Function to estimate cond. mean using Monte Carlo Simulation
predict.mean.MCS <- function(object, newdata, m=5000, uscale = FALSE)
{
  # object: an object coming from the function vinereg()
  # newdata: a matrix containing the values of the covariates.
  # Has to contain all the covariates which are in the
  # D-vine of object.
  # uscale: indicator whether newdata is already on u-scale
  # m: sample size for Monte Carlo simulation

  # check if margins were estimated
  if (is.null(object$margins[[1]]) & (!uscale))
  {
    warning("no margins have been estimated,
             setting uscale = TRUE")
    uscale <- TRUE
  }
  if (!(is.null(object$margins[[1]])) & (uscale))
  {
    warning("margins have been estimated,
             setting uscale = FALSE")
  }

  # afterwards loop through all the rows of newdata applying
  # this function
  n <- nrow(newdata)
y_hat <- numeric(n)
  for(i in 1:n)
  {
y_hat[i] <- pred_mean(object, y, v, u[i,])
  }
  return(y_hat)
}
A.4. Integration over (conditional) copula density

# Function to estimate cond. mean using integration over conditional # copula density

predict.mean.ICD <- function(object, newdata, uscale = FALSE) {
  uscale <- FALSE

  # check if all variables in the model are in newdata
  missing_vars <- setdiff(colnames(object$model_frame)[-1],
                          colnames(newdata))
  if (length(missing_vars) > 0)
    stop("'newdata' is missing variables '",
         paste(missing_vars, sep = ", ", ", ")")

  # expand factors and make ordered variables numeric
  x <- cctools::expand_as_numeric(
    newdata[, colnames(object$model_frame)[-1],
            drop = FALSE])

  # remove unused variables
  selected_vars <- match(object$order, colnames(x))
  x <- x[, selected_vars, drop = FALSE]

  # sample from distribution of Y
  y <- rkde1d(m, object$margins[[1]])

  n <- nrow(newdata)
  y_hat <- numeric(n)

  for(i in 1:n) {
    y_hat[i] <- mean(y*cond_vine_dens(object, y,
                                   newdata[i,,drop = FALSE],
                                   uscale = uscale),)
  }

  return(y_hat)
}

# Function to estimate cond. mean using integration over conditional # copula density
APPENDIX A. IMPLEMENTATION OF D-VINE MEAN REGRESSION

# object: an object coming from the function vinereg()
# newdata: a matrix containing the values of the covariates.
# Has to contain all the covariates which are in the
# D-vine of object.
# uscale: indicator whether newdata is already on u-scale

# check if margins were estimated
if (is.null(object$margins[[1]]) & (!uscale))
{
  warning("no margins have been estimated,
  setting uscale = TRUE")
  uscale <- TRUE
}
if (!(is.null(object$margins[[1]])) & (uscale))
{
  warning("margins have been estimated,
  setting uscale = FALSE")
  uscale <- FALSE
}

# check if all variables in the model are in newdata
missing_vars <- setdiff(colnames(object$model_frame)[-1],
  colnames(newdata))
if (length(missing_vars) > 0)
  stop("'newdata' is missing variables '",
  paste(missing_vars, sep = "'", "'", "")")

# expand factors and make ordered variables numeric
x <- cctools::expand_as_numeric(
  newdata[, colnames(object$model_frame)[-1],
    drop = FALSE])

# remove unused variables
selected_vars <- match(object$order, colnames(x))
x <- x[, selected_vars, drop = FALSE]

# function to calculate cond. vine density for given y and x
C_y_x <- function(y, x) cond_vine_dens(object, y, x,
  uscale = uscale)

# function which needs to be integrated in order to get EV
integ <- function(y, x)
{
  y*C_y_x(y, x)*dkde1d(y,object$margins[[1]])}
A.5. SIMPLIFIED QUANTILE INTEGRATION

```r
n <- nrow(newdata)
y_hat <- numeric(n)
for(i in 1:n)
{
  # calculate the integral. In case of error NAs are produced
  y_hat[i] <- try(integrate(function(y)
    integ(y, newdata[i,,drop =FALSE]),
    -Inf , Inf)$value, TRUE)
}
return(as.numeric(y_hat))
```

A.5 Simplified quantile integration

predict.mean.SQI <- function(object, newdata, uscale = FALSE,
   n.quant = 10)
{
  # object: an object coming from the function vinereg()
  # newdata: a matrix containing the values of the covariates.
  # Has to contain all the covariates which are in the
  # D-vine of object.
  # uscale: indicator whether newdata is already on u-scale
  # n.quant: number of quantiles at which the quantile
  # regression should be performed

  return(rowMeans(predict(object, newdata,
    alpha = (1: n.quant) / (n.quant +1),
    uscale)))
}

A.6 Calculate conditional vine-copula density

This function is needed in the implementations of Methods MCS and ICD.

```r
# function to extract c_{Y|X} from vinereg output for given y and x
# in a general regression setup x should be a vector containing
# only the variables used in the vine copula model. Moreover, they
```
cond_vine_dens <- function(object, y, x, uscale = FALSE) {
  # object: an object coming from the vinereg() function
  # y: value of y where \( c_{Y|X} \) should be evaluated
  # (can be a vector)
  # x: a vector containing the values of the covariates X
  # where \( c_{Y|X} \) should be evaluated. It should only
  # contain the covariates which are used in the D-vine

  # check if margins were estimated
  if (is.null(object$margins[[1]]) & (!uscale))
    {
      warning("no margins have been estimated,
               setting uscale = TRUE")
      uscale <- TRUE
    }

  # check if all variables in the model are in newdata
  missing_vars <- setdiff(colnames(object$model_frame)[-1],
                           colnames(x))
  if (length(missing_vars) > 0)
    stop("'x' is missing variables ",
         paste(missing_vars, sep = "", ",") )

  # expand factors and make ordered x numeric
  x <- cctools::expand_as_numeric(
    x[, colnames(object$model_frame)[-1],
      drop = FALSE])

  # remove unused variables
  selected_vars <- match(object$order, colnames(x))
  x <- x[, selected_vars, drop = FALSE]

  vine <- object$vine
  d <- ncol(vine$matrix)
  m <- length(y)
  if (ncol(x) != d - 1)
    stop("Dimensions of x and vine are not compatible")

  # perform PIT for y-values
  v <- pkde1d(y, object$margins[[1]])
  u <- numeric(d-1)
A.6. CALCULATE CONDITIONAL VINE-COPULA DENSITY

# perform PIT for x-values
for(i in 1:(d-1))
{
  u[i] <- pkde1d(x[i], object$margins[[i+1]])
}

# putting together each value of y with all the values of x
vu <- cbind(v, matrix(u, nrow=m, ncol = (d-1), byrow = TRUE))

# calculating the h-functions which are needed to calculate
# the conditional copula densities of the pair-copulas of
# the D-vine
V <- array(NA, dim = c(d, d, m))
V[d,,] <- t(vu)
V2 <- V
for (j in (d - 1):1)
{
  for (k in (d - 1):j)
  {
    tmp <- cbind(V2[k + 1, j,], V[k + 1, j + 1,])
    V[k, j,] <- hbicop(tmp, 1,
                       vine$pair_copulas[[d - k]][[j]])
    V2[k, j,] <- hbicop(tmp, 2,
                        vine$pair_copulas[[d - k]][[j]])
  }
}

# extracting only part of the h-functions from above, as not
# all the values are needed to calculate the conditional
# pair-copula densities
tmp <- array(NA, dim = c(d, 2, m))
for(i in 1:m)
{
  tmp[,,i] <- cbind(V2[,1,i], V[,2,i])
}

# calculating the conditional copula density \( c_{Y|X} \) by
# multiplying the densities of the pair-copulas
c_y_x <- matrix(0, m, (d-1))
for(i in 1:(d-1))
{
  c_y_x[,i] <- predict(vine$pair_copulas[[i]][[1]],
                        t(tmp[d-i+1,,]), what = "pdf")
}
APPENDIX A. IMPLEMENTATION OF D-VINE MEAN REGRESSION

```r
# calculate copula density
c_y_x <- apply(c_y_x, 1, prod)

return(c_y_x)
```

Appendix B

Further results of simulation studies

B.1 Selection of parameters

B.1.1 Selection of MC sample size

Additional results for Scenario $MCS.1$

Table B.1: Mean and median RMSE for training and test data set in Scenario $MCS.1$ according to different sample sizes $m$ of the Monte Carlo simulation.

<table>
<thead>
<tr>
<th>m</th>
<th>data</th>
<th>OLS mean</th>
<th>OLS median</th>
<th>MCS_par mean</th>
<th>MCS_par median</th>
<th>MCS_np mean</th>
<th>MCS_np median</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>train</td>
<td>0.997</td>
<td>0.999</td>
<td>1.506</td>
<td>1.421</td>
<td>1.627</td>
<td>1.469</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.001</td>
<td>0.998</td>
<td>1.644</td>
<td>1.500</td>
<td>82.701</td>
<td>1.528</td>
</tr>
<tr>
<td>500</td>
<td>train</td>
<td>0.997</td>
<td>0.999</td>
<td>1.154</td>
<td>1.123</td>
<td>1.892</td>
<td>1.166</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.001</td>
<td>0.998</td>
<td>1.458</td>
<td>1.193</td>
<td>2.678</td>
<td>1.272</td>
</tr>
<tr>
<td>1000</td>
<td>train</td>
<td>0.997</td>
<td>0.999</td>
<td>1.091</td>
<td>1.077</td>
<td>1.534</td>
<td>1.098</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.001</td>
<td>0.998</td>
<td>1.277</td>
<td>1.162</td>
<td>9.169</td>
<td>1.222</td>
</tr>
<tr>
<td>5000</td>
<td>train</td>
<td>0.997</td>
<td>0.999</td>
<td>1.029</td>
<td>1.024</td>
<td>1.414</td>
<td>1.033</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.001</td>
<td>0.998</td>
<td>1.143</td>
<td>1.087</td>
<td>11.319</td>
<td>1.135</td>
</tr>
<tr>
<td>10000</td>
<td>train</td>
<td>0.997</td>
<td>0.999</td>
<td>1.017</td>
<td>1.018</td>
<td>1.186</td>
<td>1.026</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.001</td>
<td>0.998</td>
<td>1.135</td>
<td>1.074</td>
<td>13.283</td>
<td>1.120</td>
</tr>
</tbody>
</table>
### Additional results for Scenario $MCS.2$

**Table B.2:** Mean and median RMSE for training and test data set in Scenario $MCS.2$ according to different sample sizes of the Monte Carlo simulation.

<table>
<thead>
<tr>
<th>m</th>
<th>data</th>
<th>OLS mean</th>
<th>median</th>
<th>MCS_par mean</th>
<th>median</th>
<th>MCS_np mean</th>
<th>median</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>train</td>
<td>1.732</td>
<td>1.727</td>
<td>2.389</td>
<td>2.173</td>
<td>2.722</td>
<td>2.184</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.739</td>
<td>1.734</td>
<td>3.275</td>
<td>2.198</td>
<td>5.431</td>
<td>2.330</td>
</tr>
<tr>
<td>500</td>
<td>train</td>
<td>1.732</td>
<td>1.727</td>
<td>1.688</td>
<td>1.516</td>
<td>6.764</td>
<td>1.523</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.739</td>
<td>1.734</td>
<td>1.883</td>
<td>1.544</td>
<td>8.382</td>
<td>1.695</td>
</tr>
<tr>
<td>1000</td>
<td>train</td>
<td>1.732</td>
<td>1.727</td>
<td>1.480</td>
<td>1.404</td>
<td>4.056</td>
<td>1.329</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.739</td>
<td>1.734</td>
<td>1.821</td>
<td>1.442</td>
<td>10.984</td>
<td>1.556</td>
</tr>
<tr>
<td>5000</td>
<td>train</td>
<td>1.732</td>
<td>1.727</td>
<td>1.256</td>
<td>1.211</td>
<td>3.501</td>
<td>1.114</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.739</td>
<td>1.734</td>
<td>1.589</td>
<td>1.296</td>
<td>7.853</td>
<td>1.293</td>
</tr>
<tr>
<td>10000</td>
<td>train</td>
<td>1.732</td>
<td>1.727</td>
<td>1.222</td>
<td>1.181</td>
<td>2.035</td>
<td>1.056</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.739</td>
<td>1.734</td>
<td>1.466</td>
<td>1.241</td>
<td>5.451</td>
<td>1.197</td>
</tr>
</tbody>
</table>

### Additional results for Scenario $MCS.3$

**Table B.3:** Mean and median RMSE for training and test data set in Scenario $MCS.3$ according to different sample sizes of the Monte Carlo simulation.

<table>
<thead>
<tr>
<th>m</th>
<th>data</th>
<th>OLS mean</th>
<th>median</th>
<th>MCS_par mean</th>
<th>median</th>
<th>MCS_np mean</th>
<th>median</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>train</td>
<td>1.046</td>
<td>1.045</td>
<td>1.115</td>
<td>1.081</td>
<td>1.149</td>
<td>1.097</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.052</td>
<td>1.043</td>
<td>1.130</td>
<td>1.066</td>
<td>1.330</td>
<td>1.112</td>
</tr>
<tr>
<td>500</td>
<td>train</td>
<td>1.046</td>
<td>1.045</td>
<td>1.059</td>
<td>1.045</td>
<td>1.082</td>
<td>1.030</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.052</td>
<td>1.043</td>
<td>1.075</td>
<td>1.053</td>
<td>1.118</td>
<td>1.059</td>
</tr>
<tr>
<td>1000</td>
<td>train</td>
<td>1.046</td>
<td>1.045</td>
<td>1.046</td>
<td>1.037</td>
<td>1.045</td>
<td>1.029</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.052</td>
<td>1.043</td>
<td>1.060</td>
<td>1.049</td>
<td>1.092</td>
<td>1.054</td>
</tr>
<tr>
<td>5000</td>
<td>train</td>
<td>1.046</td>
<td>1.045</td>
<td>1.039</td>
<td>1.032</td>
<td>1.044</td>
<td>1.008</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.052</td>
<td>1.043</td>
<td>1.054</td>
<td>1.046</td>
<td>1.234</td>
<td>1.042</td>
</tr>
<tr>
<td>10000</td>
<td>train</td>
<td>1.046</td>
<td>1.045</td>
<td>1.037</td>
<td>1.032</td>
<td>1.027</td>
<td>1.011</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.052</td>
<td>1.043</td>
<td>1.055</td>
<td>1.046</td>
<td>1.236</td>
<td>1.047</td>
</tr>
</tbody>
</table>
### Additional results for Scenario \textit{MCS}.4

Table B.4: Mean and median RMSE for training and test data set in Scenario \textit{MCS}.4 according to different sample sizes of the Monte Carlo simulation.

<table>
<thead>
<tr>
<th>( m )</th>
<th>data</th>
<th>( \text{OLS} )</th>
<th>( \text{MCS}_\text{par} )</th>
<th>( \text{MCS}_\text{np} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mean median</td>
<td>mean median</td>
<td>mean median</td>
</tr>
<tr>
<td>100</td>
<td>train</td>
<td>0.476 0.475</td>
<td>0.722 0.585</td>
<td>106534.141 0.585</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.481 0.480</td>
<td>0.781 0.583</td>
<td>0.938 0.608</td>
</tr>
<tr>
<td>500</td>
<td>train</td>
<td>0.476 0.475</td>
<td>0.595 0.516</td>
<td>3052.833 0.522</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.481 0.480</td>
<td>0.603 0.521</td>
<td>1.383 0.549</td>
</tr>
<tr>
<td>1000</td>
<td>train</td>
<td>0.476 0.475</td>
<td>0.654 0.502</td>
<td>1190.937 0.504</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.481 0.480</td>
<td>0.587 0.517</td>
<td>4438.057 0.525</td>
</tr>
<tr>
<td>5000</td>
<td>train</td>
<td>0.476 0.475</td>
<td>0.511 0.486</td>
<td>8716.653 0.497</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.481 0.480</td>
<td>0.508 0.497</td>
<td>1600.505 0.516</td>
</tr>
<tr>
<td>10000</td>
<td>train</td>
<td>0.476 0.475</td>
<td>0.497 0.483</td>
<td>6012.433 0.490</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.481 0.480</td>
<td>0.497 0.490</td>
<td>9047.193 0.515</td>
</tr>
</tbody>
</table>
B.1.2 Number of quantiles in Method SQI

**Figure B.1:** Boxplot of RMSE of training and test data set in Scenario *SQI.2* for different number of quantiles in Method SQI.
Figure B.2: Boxplot of RMSE of training and test data set in Scenario $SQI.3$ for different number of quantiles in Method $SQI$. 

*Boxplot of RMSE for training data* 

*Boxplot of RMSE for test data*
APPENDIX B. FURTHER RESULTS OF SIMULATION STUDIES

Figure B.3: Boxplot of RMSE of training and test data set in Scenario $\mathcal{SQI}.4$ for different number of quantiles in Method SQI.
### B.1. SELECTION OF PARAMETERS

#### Table B.5: Mean RMSE for training and test data in Scenario $\text{SQI.2}$ for different number of quantiles in Method SQI.

<table>
<thead>
<tr>
<th></th>
<th>OLS</th>
<th>SQI_par</th>
<th>SQI_np</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>10</td>
<td>1.73207</td>
<td>1.73935</td>
<td>1.17534</td>
</tr>
<tr>
<td>25</td>
<td>1.73207</td>
<td>1.73935</td>
<td>1.17517</td>
</tr>
<tr>
<td>50</td>
<td>1.73207</td>
<td>1.73935</td>
<td>1.17505</td>
</tr>
<tr>
<td>75</td>
<td>1.73207</td>
<td>1.73935</td>
<td>1.17501</td>
</tr>
<tr>
<td>100</td>
<td>1.73207</td>
<td>1.73935</td>
<td>1.17499</td>
</tr>
</tbody>
</table>

#### Table B.6: Mean RMSE for training and test data in Scenario $\text{SQI.3}$ for different number of quantiles in Method SQI.

<table>
<thead>
<tr>
<th></th>
<th>OLS</th>
<th>SQI_par</th>
<th>SQI_np</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>10</td>
<td>1.04617</td>
<td>1.05108</td>
<td>1.03569</td>
</tr>
<tr>
<td>25</td>
<td>1.04617</td>
<td>1.05108</td>
<td>1.03427</td>
</tr>
<tr>
<td>50</td>
<td>1.04617</td>
<td>1.05108</td>
<td>1.03404</td>
</tr>
<tr>
<td>75</td>
<td>1.04617</td>
<td>1.05108</td>
<td>1.03406</td>
</tr>
<tr>
<td>100</td>
<td>1.04617</td>
<td>1.05108</td>
<td>1.03408</td>
</tr>
</tbody>
</table>

#### Table B.7: Mean RMSE for training and test data in Scenario $\text{SQI.4}$ for different number of quantiles in Method SQI.

<table>
<thead>
<tr>
<th></th>
<th>OLS</th>
<th>SQI_par</th>
<th>SQI_np</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>10</td>
<td>0.4764</td>
<td>0.4811</td>
<td>0.46828</td>
</tr>
<tr>
<td>25</td>
<td>0.4764</td>
<td>0.4811</td>
<td>0.46808</td>
</tr>
<tr>
<td>50</td>
<td>0.4764</td>
<td>0.4811</td>
<td>0.46805</td>
</tr>
<tr>
<td>75</td>
<td>0.4764</td>
<td>0.4811</td>
<td>0.46804</td>
</tr>
<tr>
<td>100</td>
<td>0.4764</td>
<td>0.4811</td>
<td>0.46804</td>
</tr>
</tbody>
</table>
B.1.3 Analyzing the effect of data set size

Figure B.4: Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario DS.2 according to different sizes of the data set.
B.1. SELECTION OF PARAMETERS

Figure B.5: Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario DS.3 according to different sizes of the data set.

Figure B.6: Median RMSE of training and test data set for parametric (solid line) and nonparametric (dotted line) versions of D-vine mean regression methods in Scenario DS.4 according to different sizes of the data set.
Table B.8: Mean and median RMSE for training and test data in Scenario DS.2 according to different data set sizes.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>1500</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>median</td>
<td>mean</td>
<td>median</td>
<td>mean</td>
<td>median</td>
</tr>
<tr>
<td>OLS</td>
<td>train</td>
<td>1.609</td>
<td>1.57</td>
<td>1.683</td>
<td>1.667</td>
<td>1.719</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.732</td>
<td>1.673</td>
<td>1.757</td>
<td>1.766</td>
<td>1.752</td>
</tr>
<tr>
<td>IOQ_par</td>
<td>train</td>
<td>1.196</td>
<td>1.156</td>
<td>1.169</td>
<td>1.152</td>
<td>1.193</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.495</td>
<td>1.389</td>
<td>1.314</td>
<td>1.272</td>
<td>1.294</td>
</tr>
<tr>
<td>IOQ_np</td>
<td>train</td>
<td>1.015</td>
<td>0.956</td>
<td>1.007</td>
<td>0.999</td>
<td>1.048</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.636</td>
<td>1.543</td>
<td>1.178</td>
<td>1.133</td>
<td>1.127</td>
</tr>
<tr>
<td>CEE_par</td>
<td>train</td>
<td>1.4</td>
<td>1.224</td>
<td>1.172</td>
<td>1.15</td>
<td>1.159</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.643</td>
<td>1.511</td>
<td>1.323</td>
<td>1.271</td>
<td>1.273</td>
</tr>
<tr>
<td>CEE_np</td>
<td>train</td>
<td>1.255</td>
<td>1.019</td>
<td>1.004</td>
<td>0.992</td>
<td>1.016</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.618</td>
<td>1.507</td>
<td>1.162</td>
<td>1.134</td>
<td>1.094</td>
</tr>
<tr>
<td>MCS_par</td>
<td>train</td>
<td>1.384</td>
<td>1.254</td>
<td>1.229</td>
<td>1.198</td>
<td>1.253</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.617</td>
<td>1.457</td>
<td>1.457</td>
<td>1.31</td>
<td>1.831</td>
</tr>
<tr>
<td>MCS_np</td>
<td>train</td>
<td>10.396</td>
<td>1.088</td>
<td>1.226</td>
<td>1.084</td>
<td>1.762</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>73.667</td>
<td>1.976</td>
<td>4.488</td>
<td>1.286</td>
<td>23.608</td>
</tr>
<tr>
<td>ICD_par</td>
<td>train</td>
<td>1.279</td>
<td>1.216</td>
<td>1.177</td>
<td>1.157</td>
<td>1.204</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.507</td>
<td>1.372</td>
<td>1.241</td>
<td>1.197</td>
<td>1.245</td>
</tr>
<tr>
<td>ICD_np</td>
<td>train</td>
<td>2.796</td>
<td>1.756</td>
<td>1.423</td>
<td>1.391</td>
<td>1.349</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>15.83</td>
<td>2.357</td>
<td>3.738</td>
<td>1.719</td>
<td>1.759</td>
</tr>
<tr>
<td>SQL_par</td>
<td>train</td>
<td>1.195</td>
<td>1.153</td>
<td>1.169</td>
<td>1.153</td>
<td>1.194</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.5</td>
<td>1.391</td>
<td>1.318</td>
<td>1.272</td>
<td>1.297</td>
</tr>
<tr>
<td>SQL_np</td>
<td>train</td>
<td>1.011</td>
<td>0.955</td>
<td>1.007</td>
<td>0.989</td>
<td>1.048</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.642</td>
<td>1.55</td>
<td>1.181</td>
<td>1.136</td>
<td>1.129</td>
</tr>
</tbody>
</table>
Table B.9: Mean and median RMSE for training and test data in Scenario DS.3 according to different data set sizes.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>500</td>
<td>1000</td>
<td>1500</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>mean</td>
<td>median</td>
<td>mean</td>
<td>median</td>
<td>mean</td>
<td>median</td>
<td>mean</td>
<td>median</td>
<td>mean</td>
<td>median</td>
<td>mean</td>
</tr>
<tr>
<td>OLS</td>
<td>train</td>
<td>1.018</td>
<td>0.987</td>
<td>1.044</td>
<td>1.036</td>
<td>1.043</td>
<td>1.04</td>
<td>1.046</td>
<td>1.045</td>
<td>1.041</td>
<td>1.038</td>
<td>1.05</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.074</td>
<td>1.002</td>
<td>1.063</td>
<td>1.026</td>
<td>1.05</td>
<td>1.034</td>
<td>1.052</td>
<td>1.043</td>
<td>1.05</td>
<td>1.046</td>
<td>1.05</td>
</tr>
<tr>
<td>IOQ_par</td>
<td>train</td>
<td>0.997</td>
<td>0.98</td>
<td>1.048</td>
<td>1.025</td>
<td>1.051</td>
<td>1.03</td>
<td>1.035</td>
<td>1.031</td>
<td>1.033</td>
<td>1.026</td>
<td>1.06</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.128</td>
<td>1.029</td>
<td>1.089</td>
<td>1.058</td>
<td>1.074</td>
<td>1.029</td>
<td>1.052</td>
<td>1.045</td>
<td>1.06</td>
<td>1.055</td>
<td>1.06</td>
</tr>
<tr>
<td>IOQ_np</td>
<td>train</td>
<td>0.968</td>
<td>0.91</td>
<td>1.055</td>
<td>1.008</td>
<td>1.05</td>
<td>1.004</td>
<td>1.014</td>
<td>1.003</td>
<td>1.012</td>
<td>1.007</td>
<td>1.048</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.193</td>
<td>1.13</td>
<td>1.126</td>
<td>1.06</td>
<td>1.088</td>
<td>1.062</td>
<td>1.044</td>
<td>1.036</td>
<td>1.048</td>
<td>1.036</td>
<td>1.048</td>
</tr>
<tr>
<td>CEE_par</td>
<td>train</td>
<td>0.976</td>
<td>0.966</td>
<td>1.015</td>
<td>1.011</td>
<td>1.027</td>
<td>1.019</td>
<td>1.03</td>
<td>1.027</td>
<td>1.035</td>
<td>1.024</td>
<td>1.057</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.106</td>
<td>1.028</td>
<td>1.067</td>
<td>1.037</td>
<td>1.052</td>
<td>1.02</td>
<td>1.05</td>
<td>1.041</td>
<td>1.057</td>
<td>1.053</td>
<td>1.057</td>
</tr>
<tr>
<td>CEE_np</td>
<td>train</td>
<td>1.006</td>
<td>0.968</td>
<td>1.027</td>
<td>1.002</td>
<td>1.025</td>
<td>1.003</td>
<td>1.015</td>
<td>1.009</td>
<td>1.009</td>
<td>1.004</td>
<td>1.043</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.171</td>
<td>1.103</td>
<td>1.089</td>
<td>1.07</td>
<td>1.058</td>
<td>1.047</td>
<td>1.044</td>
<td>1.034</td>
<td>1.043</td>
<td>1.036</td>
<td>1.061</td>
</tr>
<tr>
<td>MCS_par</td>
<td>train</td>
<td>1.006</td>
<td>0.987</td>
<td>1.05</td>
<td>1.023</td>
<td>1.053</td>
<td>1.034</td>
<td>1.039</td>
<td>1.032</td>
<td>1.036</td>
<td>1.031</td>
<td>1.061</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.126</td>
<td>1.032</td>
<td>1.104</td>
<td>1.059</td>
<td>1.082</td>
<td>1.034</td>
<td>1.054</td>
<td>1.046</td>
<td>1.061</td>
<td>1.055</td>
<td>1.061</td>
</tr>
<tr>
<td>MCS_np</td>
<td>train</td>
<td>1.016</td>
<td>0.948</td>
<td>1.091</td>
<td>1.016</td>
<td>1.058</td>
<td>1.011</td>
<td>1.044</td>
<td>1.008</td>
<td>1.021</td>
<td>1.01</td>
<td>1.073</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.641</td>
<td>1.176</td>
<td>1.185</td>
<td>1.066</td>
<td>1.095</td>
<td>1.082</td>
<td>1.234</td>
<td>1.042</td>
<td>1.073</td>
<td>1.05</td>
<td>1.073</td>
</tr>
<tr>
<td>ICD_par</td>
<td>train</td>
<td>0.983</td>
<td>0.941</td>
<td>0.984</td>
<td>0.962</td>
<td>1.046</td>
<td>0.995</td>
<td>1.016</td>
<td>0.973</td>
<td>1.049</td>
<td>1.005</td>
<td>1.082</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.078</td>
<td>1.012</td>
<td>1.029</td>
<td>0.995</td>
<td>1.066</td>
<td>1.004</td>
<td>1.019</td>
<td>0.984</td>
<td>1.082</td>
<td>1.074</td>
<td>1.082</td>
</tr>
<tr>
<td>ICD_np</td>
<td>train</td>
<td>0.989</td>
<td>0.912</td>
<td>0.939</td>
<td>0.935</td>
<td>0.973</td>
<td>0.918</td>
<td>0.936</td>
<td>0.944</td>
<td>0.931</td>
<td>0.922</td>
<td>0.986</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.659</td>
<td>1.108</td>
<td>1.057</td>
<td>1.006</td>
<td>1.03</td>
<td>0.995</td>
<td>0.98</td>
<td>0.97</td>
<td>0.986</td>
<td>0.972</td>
<td>0.986</td>
</tr>
<tr>
<td>SQI_par</td>
<td>train</td>
<td>0.993</td>
<td>0.979</td>
<td>1.043</td>
<td>1.022</td>
<td>1.049</td>
<td>1.03</td>
<td>1.036</td>
<td>1.033</td>
<td>1.034</td>
<td>1.027</td>
<td>1.061</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.125</td>
<td>1.026</td>
<td>1.086</td>
<td>1.059</td>
<td>1.074</td>
<td>1.028</td>
<td>1.053</td>
<td>1.044</td>
<td>1.061</td>
<td>1.058</td>
<td>1.061</td>
</tr>
<tr>
<td>SQI_np</td>
<td>train</td>
<td>0.966</td>
<td>0.905</td>
<td>1.054</td>
<td>1.007</td>
<td>1.049</td>
<td>1.006</td>
<td>1.015</td>
<td>1.003</td>
<td>1.012</td>
<td>1.007</td>
<td>1.048</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.194</td>
<td>1.124</td>
<td>1.126</td>
<td>1.059</td>
<td>1.089</td>
<td>1.071</td>
<td>1.044</td>
<td>1.03</td>
<td>1.048</td>
<td>1.038</td>
<td>1.048</td>
</tr>
</tbody>
</table>
Table B.10: Mean and median RMSE for training and test data in Scenario $\mathcal{DS}.4$ according to different data set sizes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>1500</th>
<th>2000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>mean</td>
<td>median</td>
<td>mean</td>
<td>median</td>
<td>mean</td>
</tr>
<tr>
<td>OLS</td>
<td>train</td>
<td>0.458</td>
<td>0.448</td>
<td>0.473</td>
<td>0.472</td>
<td>0.477</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.485</td>
<td>0.478</td>
<td>0.482</td>
<td>0.482</td>
<td>0.477</td>
</tr>
<tr>
<td>IOQ_par</td>
<td>train</td>
<td>0.443</td>
<td>0.431</td>
<td>0.463</td>
<td>0.460</td>
<td>0.467</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.515</td>
<td>0.510</td>
<td>0.479</td>
<td>0.477</td>
<td>0.473</td>
</tr>
<tr>
<td>IOQ_np</td>
<td>train</td>
<td>0.409</td>
<td>0.398</td>
<td>0.453</td>
<td>0.452</td>
<td>0.464</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.586</td>
<td>0.543</td>
<td>0.501</td>
<td>0.494</td>
<td>0.489</td>
</tr>
<tr>
<td>CEE_par</td>
<td>train</td>
<td>0.491</td>
<td>0.496</td>
<td>0.486</td>
<td>0.484</td>
<td>0.487</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.540</td>
<td>0.527</td>
<td>0.496</td>
<td>0.493</td>
<td>0.489</td>
</tr>
<tr>
<td>CEE_np</td>
<td>train</td>
<td>0.500</td>
<td>0.488</td>
<td>0.496</td>
<td>0.496</td>
<td>0.497</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.574</td>
<td>0.565</td>
<td>0.503</td>
<td>0.505</td>
<td>0.499</td>
</tr>
<tr>
<td>MCS_par</td>
<td>train</td>
<td>0.496</td>
<td>0.479</td>
<td>0.555</td>
<td>0.491</td>
<td>0.490</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.569</td>
<td>0.546</td>
<td>0.546</td>
<td>0.500</td>
<td>0.503</td>
</tr>
<tr>
<td>MCS_np</td>
<td>train</td>
<td>0.752</td>
<td>0.478</td>
<td>36.740</td>
<td>0.486</td>
<td>4183.469</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>2287.574</td>
<td>0.613</td>
<td>2228.278</td>
<td>0.510</td>
<td>111.863</td>
</tr>
<tr>
<td>ICD_par</td>
<td>train</td>
<td>0.450</td>
<td>0.433</td>
<td>0.464</td>
<td>0.461</td>
<td>0.467</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.501</td>
<td>0.483</td>
<td>0.478</td>
<td>0.476</td>
<td>0.471</td>
</tr>
<tr>
<td>ICD_np</td>
<td>train</td>
<td>0.514</td>
<td>0.495</td>
<td>0.485</td>
<td>0.480</td>
<td>0.550</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>1.039</td>
<td>0.622</td>
<td>0.529</td>
<td>0.526</td>
<td>0.547</td>
</tr>
<tr>
<td>SQL_par</td>
<td>train</td>
<td>0.442</td>
<td>0.431</td>
<td>0.464</td>
<td>0.460</td>
<td>0.468</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.516</td>
<td>0.509</td>
<td>0.479</td>
<td>0.478</td>
<td>0.474</td>
</tr>
<tr>
<td>SQL_np</td>
<td>train</td>
<td>0.409</td>
<td>0.398</td>
<td>0.453</td>
<td>0.452</td>
<td>0.464</td>
</tr>
<tr>
<td></td>
<td>test</td>
<td>0.588</td>
<td>0.539</td>
<td>0.502</td>
<td>0.496</td>
<td>0.489</td>
</tr>
</tbody>
</table>
B.2 Comparison for three covariates

B.2.1 Scenario $SL$.3

Table B.11: Median and mean RMSE for training and test data in Scenario $SL$.3.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>mean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>test</td>
<td></td>
</tr>
<tr>
<td>OLS</td>
<td>0.999</td>
<td>0.998</td>
<td>0.997</td>
<td>1.001</td>
<td></td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.801</td>
<td>1.068</td>
<td>0.801</td>
<td>1.068</td>
<td></td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.207</td>
<td>1.205</td>
<td>0.208</td>
<td>1.206</td>
<td></td>
</tr>
<tr>
<td>IOQ_par</td>
<td>1.007</td>
<td>1.036</td>
<td>1.007</td>
<td>1.049</td>
<td></td>
</tr>
<tr>
<td>IOQ_np</td>
<td>0.996</td>
<td>1.031</td>
<td>0.993</td>
<td>1.037</td>
<td></td>
</tr>
<tr>
<td>CEE_par</td>
<td>1.007</td>
<td>1.038</td>
<td>1.004</td>
<td>1.047</td>
<td></td>
</tr>
<tr>
<td>CEE_np</td>
<td>0.994</td>
<td>1.028</td>
<td>0.990</td>
<td>1.034</td>
<td></td>
</tr>
<tr>
<td>MCS_par</td>
<td>1.024</td>
<td>1.087</td>
<td>1.029</td>
<td>1.143</td>
<td></td>
</tr>
<tr>
<td>MCS_np</td>
<td>1.033</td>
<td>1.135</td>
<td>1.414</td>
<td>11.319</td>
<td></td>
</tr>
<tr>
<td>ICD_par</td>
<td>1.148</td>
<td>1.202</td>
<td>1.175</td>
<td>1.237</td>
<td></td>
</tr>
<tr>
<td>ICD_np</td>
<td>1.001</td>
<td>1.062</td>
<td>1.014</td>
<td>1.231</td>
<td></td>
</tr>
<tr>
<td>SQL_par</td>
<td>1.007</td>
<td>1.038</td>
<td>1.007</td>
<td>1.049</td>
<td></td>
</tr>
<tr>
<td>SQL_np</td>
<td>0.995</td>
<td>1.032</td>
<td>0.993</td>
<td>1.038</td>
<td></td>
</tr>
</tbody>
</table>

B.2.2 Scenario $QL$.3

Table B.12: Median and mean RMSE for training and test data in Scenario $QL$.3.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>mean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>test</td>
<td></td>
</tr>
<tr>
<td>OLS</td>
<td>1.727</td>
<td>1.734</td>
<td>1.732</td>
<td>1.739</td>
<td></td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.811</td>
<td>1.077</td>
<td>0.810</td>
<td>1.083</td>
<td></td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.222</td>
<td>1.226</td>
<td>0.222</td>
<td>1.234</td>
<td></td>
</tr>
<tr>
<td>IOQ_par</td>
<td>1.148</td>
<td>1.202</td>
<td>1.175</td>
<td>1.237</td>
<td></td>
</tr>
<tr>
<td>IOQ_np</td>
<td>1.005</td>
<td>1.061</td>
<td>1.030</td>
<td>1.093</td>
<td></td>
</tr>
<tr>
<td>CEE_par</td>
<td>1.144</td>
<td>1.200</td>
<td>1.156</td>
<td>1.212</td>
<td></td>
</tr>
<tr>
<td>CEE_np</td>
<td>1.005</td>
<td>1.051</td>
<td>1.011</td>
<td>1.070</td>
<td></td>
</tr>
<tr>
<td>MCS_par</td>
<td>1.211</td>
<td>1.296</td>
<td>1.256</td>
<td>1.589</td>
<td></td>
</tr>
<tr>
<td>MCS_np</td>
<td>1.114</td>
<td>1.293</td>
<td>3.501</td>
<td>7.853</td>
<td></td>
</tr>
<tr>
<td>ICD_par</td>
<td>1.153</td>
<td>1.157</td>
<td>1.187</td>
<td>1.203</td>
<td></td>
</tr>
<tr>
<td>ICD_np</td>
<td>1.235</td>
<td>1.368</td>
<td>1.243</td>
<td>1.451</td>
<td></td>
</tr>
<tr>
<td>SQL_par</td>
<td>1.149</td>
<td>1.204</td>
<td>1.175</td>
<td>1.240</td>
<td></td>
</tr>
<tr>
<td>SQL_np</td>
<td>1.005</td>
<td>1.060</td>
<td>1.030</td>
<td>1.094</td>
<td></td>
</tr>
</tbody>
</table>
### B.2.3 Scenario $\mathcal{GR}.3$

Table B.13: Median and mean RMSE for training and test data in Scenario $\mathcal{GR}.3$.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>mean</td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>GLM</td>
<td>1.011</td>
<td>1.005</td>
<td>1.012</td>
<td>1.015</td>
<td></td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.949</td>
<td>1.043</td>
<td>0.946</td>
<td>1.048</td>
<td></td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.185</td>
<td>1.168</td>
<td>0.183</td>
<td>1.182</td>
<td></td>
</tr>
<tr>
<td>IOQ_par</td>
<td>1.013</td>
<td>1.041</td>
<td>1.014</td>
<td>1.054</td>
<td></td>
</tr>
<tr>
<td>IOQ_np</td>
<td>0.999</td>
<td>1.021</td>
<td>0.998</td>
<td>1.031</td>
<td></td>
</tr>
<tr>
<td>CEE_par</td>
<td>1.013</td>
<td>1.035</td>
<td>1.013</td>
<td>1.041</td>
<td></td>
</tr>
<tr>
<td>CEE_np</td>
<td>0.997</td>
<td>1.028</td>
<td>0.996</td>
<td>1.033</td>
<td></td>
</tr>
<tr>
<td>MCS_par</td>
<td>1.017</td>
<td>1.031</td>
<td>1.017</td>
<td>1.038</td>
<td></td>
</tr>
<tr>
<td>MCS_np</td>
<td>1.002</td>
<td>1.041</td>
<td>1.475</td>
<td>2.563</td>
<td></td>
</tr>
<tr>
<td>ICD_par</td>
<td>1.001</td>
<td>0.992</td>
<td>0.999</td>
<td>1.006</td>
<td></td>
</tr>
<tr>
<td>ICD_np</td>
<td>0.910</td>
<td>0.950</td>
<td>0.915</td>
<td>0.960</td>
<td></td>
</tr>
<tr>
<td>SQI_par</td>
<td>1.019</td>
<td>1.044</td>
<td>1.019</td>
<td>1.059</td>
<td></td>
</tr>
<tr>
<td>SQI_np</td>
<td>1.003</td>
<td>1.025</td>
<td>1.001</td>
<td>1.034</td>
<td></td>
</tr>
</tbody>
</table>

### B.2.4 Scenario $\mathcal{DF}.4$

Table B.14: Median and mean RMSE for training and test data in Scenario $\mathcal{DF}.4$.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>mean</td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>OLS</td>
<td>0.365</td>
<td>0.366</td>
<td>0.449</td>
<td>0.453</td>
<td></td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.358</td>
<td>0.432</td>
<td>0.432</td>
<td>0.488</td>
<td></td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.069</td>
<td>0.393</td>
<td>0.092</td>
<td>0.501</td>
<td></td>
</tr>
<tr>
<td>IOQ_par</td>
<td>0.337</td>
<td>0.382</td>
<td>0.432</td>
<td>0.449</td>
<td></td>
</tr>
<tr>
<td>IOQ_np</td>
<td>0.347</td>
<td>0.345</td>
<td>0.426</td>
<td>0.447</td>
<td></td>
</tr>
<tr>
<td>CEE_par</td>
<td>0.573</td>
<td>0.577</td>
<td>0.860</td>
<td>0.871</td>
<td></td>
</tr>
<tr>
<td>CEE_np</td>
<td>0.546</td>
<td>0.555</td>
<td>0.799</td>
<td>0.812</td>
<td></td>
</tr>
<tr>
<td>MCS_par</td>
<td>0.374</td>
<td>0.439</td>
<td>0.463</td>
<td>0.503</td>
<td></td>
</tr>
<tr>
<td>MCS_np</td>
<td>0.508</td>
<td>0.500</td>
<td>2.826</td>
<td>8.666</td>
<td></td>
</tr>
<tr>
<td>ICD_par</td>
<td>0.347</td>
<td>0.330</td>
<td>0.440</td>
<td>0.444</td>
<td></td>
</tr>
<tr>
<td>ICD_np</td>
<td>0.389</td>
<td>0.424</td>
<td>0.473</td>
<td>0.961</td>
<td></td>
</tr>
<tr>
<td>SQI_par</td>
<td>0.337</td>
<td>0.397</td>
<td>0.432</td>
<td>0.451</td>
<td></td>
</tr>
<tr>
<td>SQI_np</td>
<td>0.348</td>
<td>0.345</td>
<td>0.426</td>
<td>0.448</td>
<td></td>
</tr>
</tbody>
</table>
### B.2.5 Scenario DR.4

**Table B.15:** Median and mean RMSE for training and test data in Scenario DR.4.

<table>
<thead>
<tr>
<th>Method</th>
<th>median train</th>
<th>median test</th>
<th>mean train</th>
<th>mean test</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>0.417</td>
<td>0.434</td>
<td>0.490</td>
<td>0.495</td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.358</td>
<td>0.432</td>
<td>0.432</td>
<td>0.488</td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.069</td>
<td>0.393</td>
<td>0.092</td>
<td>0.501</td>
</tr>
<tr>
<td>IOQ_par</td>
<td>0.385</td>
<td>0.412</td>
<td>0.470</td>
<td>0.486</td>
</tr>
<tr>
<td>IOQ_np</td>
<td>0.400</td>
<td>0.435</td>
<td>0.466</td>
<td>0.489</td>
</tr>
<tr>
<td>CEE_par</td>
<td>0.524</td>
<td>0.520</td>
<td>0.722</td>
<td>0.730</td>
</tr>
<tr>
<td>CEE_np</td>
<td>0.541</td>
<td>0.558</td>
<td>0.696</td>
<td>0.703</td>
</tr>
<tr>
<td>MCS_par</td>
<td>0.396</td>
<td>0.507</td>
<td>0.495</td>
<td>0.522</td>
</tr>
<tr>
<td>MCS_np</td>
<td>0.509</td>
<td>0.516</td>
<td>1.634</td>
<td>3.383</td>
</tr>
<tr>
<td>ICD_par</td>
<td>0.388</td>
<td>0.404</td>
<td>0.472</td>
<td>0.478</td>
</tr>
<tr>
<td>ICD_np</td>
<td>0.389</td>
<td>0.413</td>
<td>0.476</td>
<td>0.528</td>
</tr>
<tr>
<td>SQI_par</td>
<td>0.384</td>
<td>0.413</td>
<td>0.470</td>
<td>0.488</td>
</tr>
<tr>
<td>SQI_np</td>
<td>0.400</td>
<td>0.436</td>
<td>0.466</td>
<td>0.489</td>
</tr>
</tbody>
</table>

### B.2.6 Scenario RR.4

**Table B.16:** Median and mean RMSE for training and test data in Scenario RR.4.

<table>
<thead>
<tr>
<th>Method</th>
<th>median train</th>
<th>median test</th>
<th>mean train</th>
<th>mean test</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>0.258</td>
<td>0.262</td>
<td>0.306</td>
<td>0.308</td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.236</td>
<td>0.287</td>
<td>0.248</td>
<td>0.330</td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.048</td>
<td>0.290</td>
<td>0.058</td>
<td>0.338</td>
</tr>
<tr>
<td>IOQ_par</td>
<td>0.251</td>
<td>0.264</td>
<td>0.298</td>
<td>0.312</td>
</tr>
<tr>
<td>IOQ_np</td>
<td>0.251</td>
<td>0.273</td>
<td>0.293</td>
<td>0.319</td>
</tr>
<tr>
<td>CEE_par</td>
<td>1.005</td>
<td>1.023</td>
<td>1.035</td>
<td>1.043</td>
</tr>
<tr>
<td>CEE_np</td>
<td>0.994</td>
<td>1.007</td>
<td>1.023</td>
<td>1.029</td>
</tr>
<tr>
<td>MCS_par</td>
<td>0.309</td>
<td>0.322</td>
<td>1.143</td>
<td>0.479</td>
</tr>
<tr>
<td>MCS_np</td>
<td>0.489</td>
<td>0.461</td>
<td>203.151</td>
<td>33.253</td>
</tr>
<tr>
<td>ICD_par</td>
<td>0.263</td>
<td>0.268</td>
<td>1.049</td>
<td>0.447</td>
</tr>
<tr>
<td>ICD_np</td>
<td>0.305</td>
<td>0.338</td>
<td>0.462</td>
<td>0.865</td>
</tr>
<tr>
<td>SQI_par</td>
<td>0.251</td>
<td>0.264</td>
<td>0.298</td>
<td>0.313</td>
</tr>
<tr>
<td>SQI_np</td>
<td>0.251</td>
<td>0.273</td>
<td>0.293</td>
<td>0.320</td>
</tr>
</tbody>
</table>
B.3 Comparison for ten covariates

B.3.1 Scenario $\mathcal{SL}.10$

Table B.17: Median and mean RMSE for training and test data in Scenario $\mathcal{SL}.10$.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>median</td>
<td>mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>OLS</td>
<td>0.998</td>
<td>0.999</td>
<td>0.997</td>
<td>1.005</td>
</tr>
<tr>
<td>XGB_{tun}</td>
<td>0.786</td>
<td>1.209</td>
<td>0.787</td>
<td>1.210</td>
</tr>
<tr>
<td>XGB_{def}</td>
<td>0.066</td>
<td>1.890</td>
<td>0.067</td>
<td>1.897</td>
</tr>
<tr>
<td>IOQ_{par}</td>
<td>1.037</td>
<td>1.198</td>
<td>1.042</td>
<td>1.207</td>
</tr>
<tr>
<td>IOQ_{np}</td>
<td>1.107</td>
<td>1.580</td>
<td>1.110</td>
<td>1.631</td>
</tr>
<tr>
<td>CEE_{par}</td>
<td>1.043</td>
<td>1.212</td>
<td>1.047</td>
<td>1.214</td>
</tr>
<tr>
<td>CEE_{np}</td>
<td>1.242</td>
<td>1.352</td>
<td>1.244</td>
<td>1.368</td>
</tr>
<tr>
<td>MCS_{par}</td>
<td>1.151</td>
<td>1.323</td>
<td>1.174</td>
<td>1.642</td>
</tr>
<tr>
<td>MCS_{np}</td>
<td>1.499</td>
<td>1.745</td>
<td>2.395</td>
<td>324.258</td>
</tr>
<tr>
<td>ICD_{par}</td>
<td>1.087</td>
<td>1.217</td>
<td>1.101</td>
<td>1.247</td>
</tr>
<tr>
<td>ICD_{np}</td>
<td>1.376</td>
<td>1.922</td>
<td>2.794</td>
<td>267.621</td>
</tr>
<tr>
<td>SQI_{par}</td>
<td>1.037</td>
<td>1.200</td>
<td>1.042</td>
<td>1.209</td>
</tr>
<tr>
<td>SQI_{np}</td>
<td>1.106</td>
<td>1.585</td>
<td>1.109</td>
<td>1.635</td>
</tr>
</tbody>
</table>

B.3.2 Scenario $\mathcal{QL}.10$

Table B.18: Median and mean RMSE for training and test data in Scenario $\mathcal{QL}.10$.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>median</td>
<td>mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
</tr>
<tr>
<td>OLS</td>
<td>24.112</td>
<td>24.637</td>
<td>24.139</td>
<td>24.631</td>
</tr>
<tr>
<td>XGB_{tun}</td>
<td>0.737</td>
<td>13.556</td>
<td>0.757</td>
<td>14.055</td>
</tr>
<tr>
<td>XGB_{def}</td>
<td>0.437</td>
<td>18.172</td>
<td>0.447</td>
<td>18.325</td>
</tr>
<tr>
<td>IOQ_{par}</td>
<td>20.394</td>
<td>21.226</td>
<td>20.583</td>
<td>21.278</td>
</tr>
<tr>
<td>IOQ_{np}</td>
<td>27.820</td>
<td>28.353</td>
<td>27.513</td>
<td>28.524</td>
</tr>
<tr>
<td>CEE_{par}</td>
<td>19.880</td>
<td>20.037</td>
<td>19.953</td>
<td>20.478</td>
</tr>
<tr>
<td>CEE_{np}</td>
<td>27.653</td>
<td>27.686</td>
<td>27.513</td>
<td>27.706</td>
</tr>
<tr>
<td>MCS_{par}</td>
<td>20.600</td>
<td>20.901</td>
<td>20.645</td>
<td>21.253</td>
</tr>
<tr>
<td>MCS_{np}</td>
<td>27.851</td>
<td>28.334</td>
<td>27.653</td>
<td>28.360</td>
</tr>
<tr>
<td>ICD_{par}</td>
<td>15.806</td>
<td>16.016</td>
<td>424.280</td>
<td>19.137</td>
</tr>
<tr>
<td>ICD_{np}</td>
<td>27.926</td>
<td>31.433</td>
<td>29.242</td>
<td>33.071</td>
</tr>
<tr>
<td>SQI_{par}</td>
<td>19.873</td>
<td>20.342</td>
<td>19.847</td>
<td>20.551</td>
</tr>
<tr>
<td>SQI_{np}</td>
<td>27.487</td>
<td>27.907</td>
<td>27.126</td>
<td>28.109</td>
</tr>
</tbody>
</table>
### B.3.3 Scenario $\mathcal{GR}.10$

Table B.19: Median and mean RMSE for training and test data in Scenario $\mathcal{GR}.10$.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>test</td>
</tr>
<tr>
<td>GLM</td>
<td>0.759</td>
<td>0.772</td>
<td>0.761</td>
<td>0.772</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.682</td>
<td>0.796</td>
<td>0.685</td>
<td>0.802</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.040</td>
<td>0.875</td>
<td>0.041</td>
<td>0.879</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOQ_par</td>
<td>0.761</td>
<td>0.811</td>
<td>0.763</td>
<td>0.816</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOQ_np</td>
<td>0.747</td>
<td>0.826</td>
<td>0.747</td>
<td>0.829</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CEE_par</td>
<td>0.763</td>
<td>0.791</td>
<td>0.765</td>
<td>0.792</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CEE_np</td>
<td>0.792</td>
<td>0.830</td>
<td>0.795</td>
<td>0.833</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS_par</td>
<td>0.768</td>
<td>0.787</td>
<td>0.772</td>
<td>1.075</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS_np</td>
<td>0.771</td>
<td>0.847</td>
<td>1.054</td>
<td>0.894</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICD_par</td>
<td>0.753</td>
<td>0.770</td>
<td>0.754</td>
<td>0.777</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICD_np</td>
<td>0.713</td>
<td>0.813</td>
<td>0.717</td>
<td>0.831</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQI_par</td>
<td>0.765</td>
<td>0.815</td>
<td>0.767</td>
<td>0.821</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQI_np</td>
<td>0.749</td>
<td>0.831</td>
<td>0.750</td>
<td>0.834</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B.3.4 Scenario $\mathcal{DF}.11$

Table B.20: Median and mean RMSE for training and test data in Scenario $\mathcal{DF}.11$.

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>median</td>
<td>mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>test</td>
<td>train</td>
<td>test</td>
</tr>
<tr>
<td>OLS</td>
<td>0.466</td>
<td>0.467</td>
<td>0.467</td>
<td>0.467</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XGB_tun</td>
<td>0.430</td>
<td>0.475</td>
<td>0.433</td>
<td>0.473</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XGB_def</td>
<td>0.025</td>
<td>0.519</td>
<td>0.025</td>
<td>0.520</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOQ_par</td>
<td>0.457</td>
<td>0.469</td>
<td>0.459</td>
<td>0.465</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOQ_np</td>
<td>0.484</td>
<td>0.623</td>
<td>0.492</td>
<td>0.625</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CEE_par</td>
<td>0.475</td>
<td>0.476</td>
<td>0.477</td>
<td>0.475</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CEE_np</td>
<td>0.498</td>
<td>0.495</td>
<td>0.501</td>
<td>0.499</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS_par</td>
<td>0.475</td>
<td>0.477</td>
<td>4.768</td>
<td>0.483</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS_np</td>
<td>0.501</td>
<td>0.506</td>
<td>0.853</td>
<td>607074.381</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICD_par</td>
<td>0.456</td>
<td>0.464</td>
<td>4.648</td>
<td>0.461</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICD_np</td>
<td>0.471</td>
<td>0.497</td>
<td>0.512</td>
<td>0.615</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQI_par</td>
<td>0.457</td>
<td>0.470</td>
<td>0.460</td>
<td>0.465</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQI_np</td>
<td>0.485</td>
<td>0.623</td>
<td>0.492</td>
<td>0.625</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>