AN APPLICATION OF CREDIT SCORING BASED ON LOGISTIC REGRESSION

by

Berker Moralar

B.S., Telecommunications Engineering, Istanbul Technical University, 2010

B.S., Control Engineering, Istanbul Technical University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2019

ACKNOWLEDGEMENTS

When I changed my thesis topic to carry out studies in my new job field, credit scoring, as per the suggestion of my advisor, I was not fully aware of the challenges in front of me. It took so much time to define the scope of the thesis while training myself in the field. I sincerely appreciate understanding, guidance and patience of Assoc. Prof. Wolfgang Hörmann throughout the whole period.

I am grateful to my Mom, Dad and sister for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without you.

I also want to thank my friends, who have been exposed to me talking about my never-ending thesis for a long time, for supporting and motivating me.

Sometimes, I was lost in details to make this work better; however, at the end, I believe I finished it in a way I am proud of. This process turned me into an expert in the field for which I am passionate. This work is just a step of my future works in this dynamic and challenging field. Thank you.

ABSTRACT

AN APPLICATION OF CREDIT SCORING BASED ON LOGISTIC REGRESSION

As the importance of processing data to extract added value continuously grows, financial institutions, which are able to collect customer data from a variety of sources, invest more and more resources to move towards a data-based decision-making approach. One of the areas where data have been used extensively in recent years and tremendous benefits are provided is credit scoring models for lending institutions, remarkably. Thus, they not only automate but also manage application evaluation processes in a way that is more objective, reliable and in accordance with the regulations. In this thesis, a study was conducted using logistic regression which is one of the most frequently used prediction techniques in credit scoring systems. The study mainly consists of three pillars. Firstly, a data set of consumer loan records was prepared for the specified model development period using the credit bureau data where member financial institutions share credit histories and payment behaviors of their customers. Secondly, after determining that the prepared data set is capable of representing the whole sector, a random sample of records was selected, and it has been shown that such a study can be carried out with open source programs like R, without using popular data analytics and statistical programs, where large budgets are allocated for their licenses, installations and maintenance. In addition to visual and numerical analyses, algorithms and functions needed in the model development process were examined extensively. Thirdly, automatic calculation of interaction terms using Weight of Evidence (WOE) method is discussed, and the so-called Neighbours' approach is proposed to calculate WOE values of covariate patterns that are not observed in the development data set. In short, we elaborated in this study on how to develop an end-to-end credit scoring model using data which can portray the whole credit sector in a given period.

ÖZET

LOJİSTİK REGRESYON KULLANILARAK BİR KREDİ SKORLAMA UYGULAMASI

Verileri işleyerek katma değer üretmenin önemi her geçen gün artarken, müşterileri hakkında çok farklı kaynaklardan verilere ulaşabilen finansal kurumlar da veriye dayalı karar alma kültürünü oluşturubilmek ve geliştirebilmek için büyük kaynak ayırıyorlar. Son yıllarda verilerin yoğun olarak kullanıldığı ve önemli fayda sağlanan alanlardan biri, kredi veren kurumlar için kredi skorlama modelleri olarak dikkat çekiyor. Böylelikle süreci otomatize etmenin yanında, daha objektif, güvenilir ve özellikle de regülasyonlara uygun şekilde başvuru değerlendirme süreçlerini yönetiyorlar. Bu tezde de kredi skorlamada en sık kullanılan öngörü tekniklerinden lojistik regresyon kullanılarak bir çalışma yürütüldü. Çalışma temel olarak üç ana yapısal bloktan oluşuyor. İlk olarak, üye olan finansal kurumların müşterilerinin kredi geçmişleri ve ödeme davranışlarıyla ilgili verileri paylaştıkları kredi kayıt bürosu kayıtlarından, model geliştirme periyodunda kullandırılmış olan tüketici kredisi kayıtlarıyla bir veri seti hazırlandı. İkinci olarak, hazırlanan veri setinin, tüm sektörü temsil edebilecek şekilde olduğunun tespiti sonrası alınan örneklemden, lisansları, kurulumları ve yönetilmeleri için büyük bütçeler ayrılan popüler veri analitiği ve istatistiksel programları kullanılmadan, bu tür bir çalışmanın, R kullanılarak, açık kaynak kodlu programlarla da yürütülebileceği gösterildi. Görsel ve sayısal analizlerle birlikte model geliştirme sürecinde ihtiyaç duyulabilecek algoritmalar ve fonksiyonlar kapsamlı biçimde ele alındı. Üçüncü olarak, kanıt ağırlığı yönteminin etkileşim terimleri için nasıl otomatik olarak hesaplanabileceği gösterildi ve geliştirme setinde görülmeyen kategorik değişken konfigürasyonları için de kanıt ağırlığı hesaplanabilmesi için komşular yaklaşımı olarak isimlendirdiğimiz yeni bir yöntem önerildi. Kısacası, kurum bağımsız veriler kullanılarak uçtan uca bir kredi skorlama modelinin nasıl geliştirileceği ayrıntılarıyla ele alındı.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

AUROC	Area under the ROC curve
BRSA	Banking Regulation and Supervision Agency
EDW	Enterprise Data Warehouse
GRP variable	Grouped variable
IQR	Interquartile Range
NA	Not Available
ROC curve	Receiver Operating Characteristic curve
TBB	Türkiye Bankalar Birliği (Union of Banks in Turkey)
TRY	Turkish Citizen ID Number
TCKN	Turkish Lira
WOE variable	Weight of Evidence variable

1. INTRODUCTION

In the digital age, also called the information age, we live in, data are one of the most valuable resources, that mankind strives to understand and process, to acquire knowledge to be used for every aspect of life. As more people and devices get connected to each other, the size of global data sphere is growing at an exponential rate, doubling every two years. According to IDC, ten times more data than the size of data generated in 2016 is expected to be created and reproduced in 2025 [1].

As the volume of data generated grows fast, handling data in order to extract knowledge has become critical and major area of interest in mathematics, statistics and computer science. As a result, data science has emerged as an interdisciplinary field. Thanks to advancement in computational power, institutions and organizations from various industries have been putting an effort to exploit the data they generate in order to increase efficiency in their processes, profits, and cut costs and minimize their loss in their business activities.

Financial sector is one of the major sectors that is able to collect huge amounts of data, store and treat them as a business asset. Not only banks but also other financial institutions put great importance on data and build business units and train them to adapt to the contemporary data-driven age. Marketing analytics departments make use of data for campaign management, customer churn models, etc. Inspection units develop fraud detection models, credit monitoring and collection units create systems for early detection of future delinquent customers, call center agents are allocated accordingly to achieve maximum business benefit. More examples from the field can be given, whereas usage of data analytics as a credit appraisal tool, the so-called credit scoring models, should be emphasized as the subject of this study.

One of the main activities of commercial banks as intermediary financial institutions is raising financial resources and lending them to third parties, individuals or corporations of different sizes. Apart from fees and commissions for their various

services, big share of banks' income come from lending credits.

Bank credits can be classified as consumer loans, loans granted to micro, small and medium-sized enterprises (SMEs), and commercial and corporate loans. Consumer loans include a vast range of different loans and appeal to many areas that individuals may need from cash loans to mortgages. Commercial and corporate are loans for larger companies than SMEs and project finance can be given as an example.

Banking Regulation and Supervision Agency (BRSA) of Turkey quarterly publishes main indicators of Turkish banking sector. As of June 2018, realizations of loan types are broken down as in Figure 1.1. The yearly growth rate of each type of credits is clearly distinguishable. Consumer loans and credit cards amount to almost 22% of total loans lent by banks in the Turkish Banking Sector.

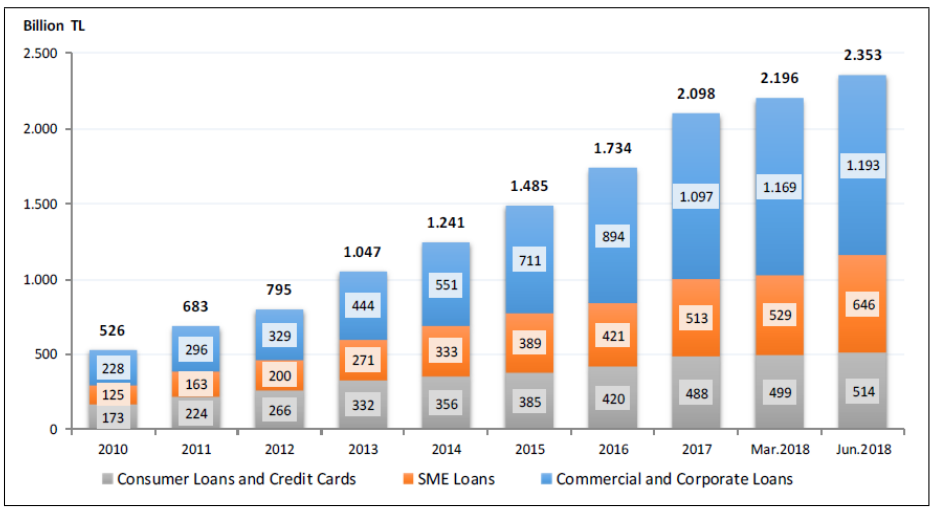


Figure 1.1. Development of loans by types.

As of June 2018, it can be seen that the increase of consumer loans and credit cards continues. Breakdown of these loans is given in Figure 1.2. The asset being purchased, i.e. the house or car, is used as collateral for these housing loans and auto loans. However, personal finance credits are usually unsecured credits, that is they are granted without collateral. Thus, we put our attention on them.

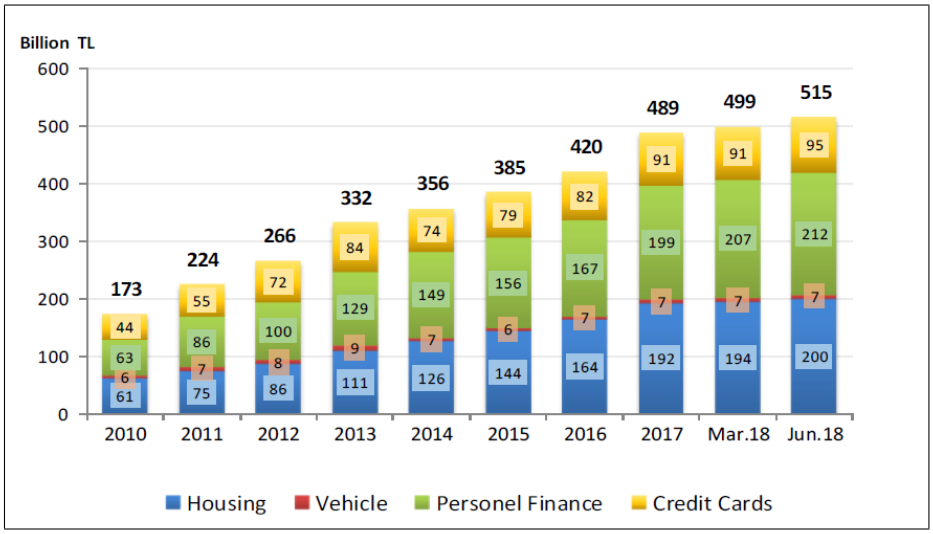


Figure 1.2. Breakdown of consumer loans and credit cards.

Figure 1.3 shows the ratio of non-performing loans to total loans trend for consumer loans in Turkish banking sector.

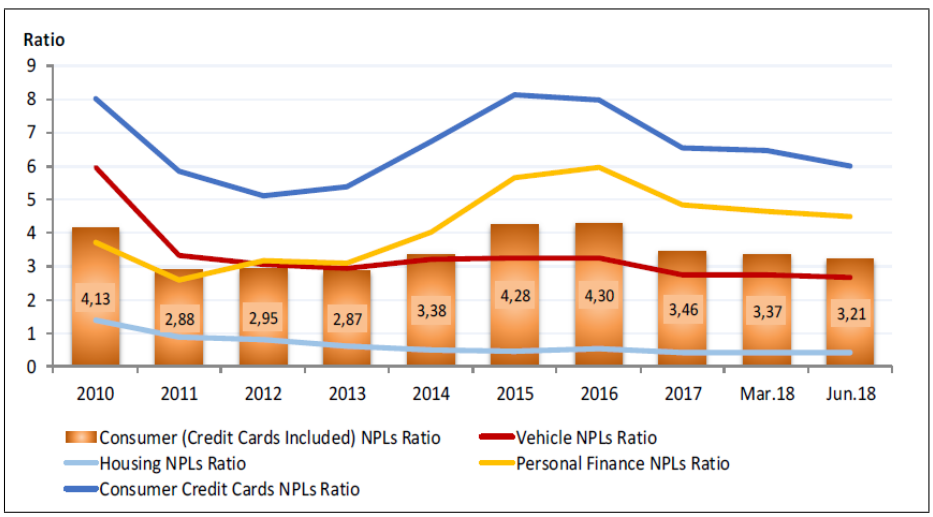


Figure 1.3. Consumer NPL ratios.

Each borrower has different characteristics in the context of their repayment abilities. Hence banks undertake the risk related to the borrowers not fulfilling their respective financial obligations and issue an appropriate rate of interest. For riskier customers, the bank may choose not to give credit or grant it with a higher interest rate. Thus, the spread, which is the difference between the interest rate of the loan

and the cost of money lent to the bank, gets bigger.

Credit scoring models are used to determine the applicants or customers who will probably fail to pay their loan back. Approval of a credit application does not solely depend on the output of the scoring model; expert's view, business rules and policies are other decisive factors in assessment of a loan.

Credit scoring assumes that the historical data can be used to predict the future behavior. Thus, future creditworthy applicants will tend to resemble past creditworthy applicants. A credit scoring model attains a probability of default to each applicant or each existing customer who satisfies particular conditions. This enables banks to make more objective, consistent, quick and efficient lending decisions and manage loans with less administrative work. Besides, it is also a requirement of governing institutions of banking sector.

The scorecard learns the relationship between the explanatory variables and the probability of default of an applicant. After implementation, the trained model assigns a score, which is obtained by using the probability of default, to a new application or each applicant or each existing customer who satisfies particular conditions.

Various state-of-the-art machine learning and sophisticated statistical methods have been proposed for credit scoring [2]. Logistic regression has been the most widely used method of estimation in the industry as it provides sufficient balance of accuracy, efficiency and interpretability of the outputs [3].

The thesis is organized as follows: in Chapter 2 a brief information on the history of credit scoring, widely-used methods and approaches are presented. In Chapter 3, an extensive summary of logistic regression's mathematical background and interpretation of its outputs are shared. Feature extraction and elimination techniques used in this study are introduced in Chapter 4. In Chapter 5, it is explained how the model fit is assessed and predictive power is calculated. Preparation of the development dataset using Credit Bureau data is shown step by step in Chapter 6. In Chapter 7, the focus

is on the raw, unbinned variables. Outlier detection and elimination techniques and various univariate analysis techniques are shared, in addition to quantifying the loss of information through categorizing. We examine the contribution of interaction terms in the context of Gini indices in Chapter 8, in case categorized variables are used. In addition to the standard approach to include interaction terms of GRP variables in a regression model, calculation of their WOE versions and neigbours' approach are described. In Chapter 9, we sum up the experiments that we carried out in earlier chapters, and interpret the performances of final models. Chapter 10 contains the conclusion.

2. CREDIT SCORING MODEL

The risk associated with a prospective borrower is measured using a credit scorecard by analyzing his data to determine the likelihood that he will default on a loan [2].

Using process automation by means of credit scoring models reduces bank's personnel costs, simplifies and accelerates the credit appraisal processes and provides better control over the approval decision-making processes [4].

Credit scorecards are developed for not only retail customers but also corporates and institutions. Two main types can be given as:

- Application scorecards are used for approval / rejection decision of the customer's credit application. If the application is approved regarding the score, specifications of the credit such as collateral, amount of credit, interest rate, and duration of credit are determined.
- Behavioural scorecards are used for limit increase, cross-sell, up-sell, limit blocking / monitoring / collection operations for existing customers, covering retail (including credit cards) and corporate loans portfolios.

Various state-of-the-art machine learning and sophisticated statistical methods can be employed to develop credit scoring models. Some of them can be grouped as below [5]:

- Statistical models such as linear probability models and discriminant analysis.
- Machine learning models such as neural networks, decision trees, support vector machines.
- Methods derived from financial theory such as option pricing models.
- Combination of various individual classification models, namely ensemble models.

Primary interest of this study is not evaluation of different modelling techniques; thus details of the methods except for the logistic regression will not be shared. In [6] and [5], reviews and comparisons of a variety of classification algorithms for credit scoring are inclusively shared. In [5] it is shown that logistic regression generally performs well and is even better than the most advanced machine learning algorithms. Today, logistic regression is still widely used, because reality checks are relatively easy. For example, only the sign of a variable coefficient can be considered to see if the results are intuitively meaningful.

[7] and [8] are reference books that guide practitioners in the credit scoring field successfully. In this thesis, the approach is also influenced greatly by these references.

2.1. Data Sources

For each applicant, credit history of the applicant is inquired from the Credit Bureau. The applicant may be a customer of the bank, in this case the bank has some useful information about the customer's credit history in the bank, such as his pay-back behaviour, spending habits, trends in limit usage etc. However, it is also important to get historical data of this customer at other financial institutions to determine whether a loan should be granted to this specific customer, if positive, the amount, number of installments and even interest rate are specified. If the applicant is not a customer of the bank, historical data in the financial industry is more crucial. That's why, each financial institution shares its data with this institution, namely Credit Bureau, and Credit Bureau supplies the accumulating data for each customer, individual or corporate. Thus, information on types of credit used, current level of indebtedness, length of credit history, payment history and new credit accounts for a borrower can be obtained from the Credit Bureau.

Demographic information such as age, gender, marital status, education level, employment type, profession, residential status, city/district information are also added to the dataset.

Except for the more conventional data sources stated above, contemporary data sources are being studied by the institutions. Customers' social media accounts are seen an important data source to determine their socioeconomic status, interests and activities which would be related to their pay-back behavior. Location based features generated from data which includes where the client lives, works and goes in his social life may also significant. Another data source can be other individuals or corporations in the close circle of the client, since related people like parents, siblings and couples may behave similarly. If corporations that the borrower company makes business with go financially troubled, then the borrower will be affected with no surprise. However, use of this type of data are not always possible for financial institutions because the authorization of regulatory institutions is necessary to use them in the scoring models. As the informational privacy is a constitutional right, borrower's content is the uttermost important.

2.2. Target

A model is trained using a dataset consisting of records whose performance can be observed. That is, a defined amount of time, i.e. "performance window", should pass after a loan was lent in order to take that loan record into the development dataset. Their status after the performance window is defined as their default status.

According to the Basel accords, default is defined as being classified as a non-performing loan (NPL) or missing three payments in a row which is typically 90 days of delinquency on the loan.

2.3. Splitting the Data into Training, Validation and Test Samples

The development dataset is usually split into three sets: a training set, a validation set and a test set. The training set is used to estimate the models. In order to ensure that the trained models are generalizable, it is important to test the models with data different from those used for training to prevent an underestimation of the

true error, usually in the following two stages [9]:

(i) Validation is used for model selection by comparing the performance of different models to find the best one.

(ii) Testing is employed for model assessment to estimate the trained model's generalization error on new data, given that the best fitting model has been found.

3. LOGISTIC REGRESSION

Simple logistic regression is used to describe the relationship between a binary (dichotomous) dependent variable and an explanatory independent variable. A univariate logistic regression can have a nominal output variable with more than two categories, i.e. multinomial logistic regression, whereas in this study the output variable is binary depicting default or non-default. Logistic regression with more than one independent variable is called as multiple logistic regression. There is a widespread confusion that the terms multiple and multivariate are used interchangeably in the literature, however multivariate logistic regression has more than one nominal output variable and it is not a subject of this thesis.

In this chapter of the thesis, we will focus on binary multiple logistic regression. Structure and content of this chapter is extensively based on [10]. Assume the binomial response variable can take either 0 or 1, then it follows a Bernoulli distribution. The mean of the Bernoulli distribution is the proportion of 1's in the distribution which is equivalently the probability of having response 1, denoted as p that is restricted to the interval (0, 1). Thus, the proportion of 0's is the probability of having outcome 0 which is equal to $(1 - p)$. The variance of the Bernoulli distribution is $p\,(1 - p)$.

3.1. Why Linear Regression Is Not Applicable Directly in Case of a Binary Response Variable

A linear regression is not applicable in case of a nominal dependent variable, because several of the model assumptions would be violated.

There is not a linear relationship between the independent variable(s) and the dependent variable; thus, the fundamental assumption of linear regression is already violated.

Since the predicted values are probabilities, they have to be between 0 and 1. However, predicted values fall outside of the restricted interval when using linear regression.

Constant variance of the dependent variable across the independent variables is not constant for logistic regression. As stated earlier, the variance of Bernoulli distribution is $p(1-p)$. Assume 50% of the observed sample has 1 as dependent variable, the variance is at its maximum of $0.5 * 0.5 = 0.25$. As one moves to the extremes, the variance decreases. If $p = 0.1$, the variance is $0.1 * 0.9 = 0.09$ so as p approaches 0 or 1, the variance approaches zero. Thus, homoscedasticity assumption of linear regression is also violated.

In a linear regression model, it is assumed that the dependent variable can be expressed as $y = \beta_0 + \beta_1 x + \epsilon$. The assumption in linear regression is that this ϵ is distributed with a mean of zero and a constant variance. With a binary variable, the dependent variable given x is now expressed as $y = \pi(x) + \epsilon$. The error ϵ can now assume only one of two possible values. If $y = 1$, then $\epsilon = 1 - \pi(x)$ with probability $\pi(x)$, and if $y = 0$ then $\epsilon = -\pi(x)$ with probability $1 - \pi(x)$. Thus, ϵ is distributed with a mean of zero and a variance of $\pi(x)[1 - \pi(x)]$. Since the error term has this distribution, it is now deduced that the conditional distribution of the dependent variable follows a binomial distribution with a probability given by the conditional mean $\pi(x)$.

3.2. Definition of Logit Transformation and Regression

This section is taken from [10] and slightly modified. In linear regression, it is assumed that the conditional mean of the dependent variable y, given the independent variable x can be expressed as an Equation 3.1 linear in x.

$$E(y|x) = \beta_0 + \beta_1 x \tag{3.1}$$

This expression implies that it is possible for $E(y|x)$ to take any value as x ranges between $-\infty$ and ∞. With a binary dependent variable, this conditional mean must be greater than 0 and less than 1.

$$0 < E(y|x) < 1 \tag{3.2}$$

Consider a vector $\mathbf{x} = [1\ x_1\ x_2\ \ldots\ x_k]^T$ which denotes a collection of k independent variables and 1 that stands for the intercept.

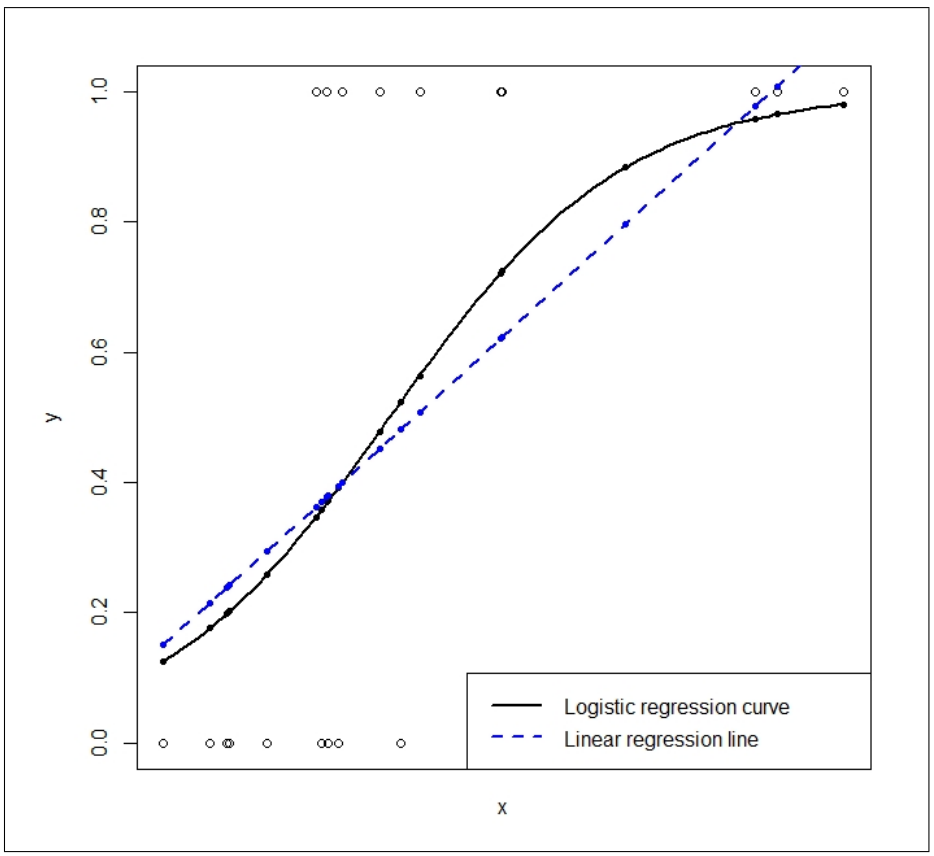


Figure 3.1. Logistic regression curve vs. linear regression line.

The S-shaped curve of logistic regression given in Figure 3.1 is defined by

$$P(y = 1|\mathbf{x}) = \pi(\mathbf{x}) = \frac{e^{\beta^{\mathbf{T}}\mathbf{x}}}{1 + e^{\beta^{\mathbf{T}}\mathbf{x}}} \tag{3.3}$$

Here, $\beta = [\beta_0\ \beta_1\ \beta_2\ \ldots\ \beta_k]^T$ is a vector of unknown parameters to be estimated, and $\pi(\mathbf{x})\, represents E(y|\mathbf{x})$, the proportion of 1's or the probability of 1. This equation is rearranged as

$$\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})} = e^{\beta^{\mathbf{T}}\mathbf{x}} \tag{3.4}$$

Accordingly, the following equation is obtained where $g(\mathbf{x})$ is called the logit transformation.

$$g(\mathbf{x}) = log\left(\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right) = \beta^{\mathbf{T}}\mathbf{x} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k \tag{3.5}$$

$g(\mathbf{x})$ has many of the desirable properties of a linear regression model. It is linear in its parameters, may be continuous and may range from $-\infty and \infty$, depending on the range of $\mathbf{x}$. This means the logistic regression is given by

$$\pi(\mathbf{x}) = \frac{e^{g(\mathbf{x})}}{1+e^{g(\mathbf{x})}} \tag{3.6}$$

To fit a logistic regression model $\pi(\mathbf{x})$ to a set of data requires that the values for the unknown parameters of vector β be estimated. There is no mathematical solution that produces explicit expressions for least squares estimates of the parameters which minimize the sum of squared errors (SSE). Hence, maximum likelihood approach is employed to obtain the parameters. This approach returns values for the unknown parameters that maximize the probability of acquiring the observed data set. To apply this approach, a likelihood function must be constructed. This function expressed the probability of the observed data as a function of the unknown parameters. The maximum likelihood estimators of these parameters are chosen that this function is

maximized, hence the resulting estimators will agree most closely with the observed data.

If y is coded as 0 and 1, the expression for $\pi(\mathbf{x}) = \frac{e^{g(\mathbf{x})}}{1+e^{g(\mathbf{x})}}$ provides the conditional probability that $y = 1\ given\ \mathbf{x}, P(y = 1|\mathbf{x})$. It follows that $1-\pi(\mathbf{x})$ gives the conditional probability that $y = 0\ given\ \mathbf{x}, P(y = 0|\mathbf{x})$. For an observation $(\mathbf{x}_i,\ y_i)$ where $y_i = 1$, the contribution to the likelihood function is $\pi(\mathbf{x}_i)$ and where $y_i = 0$, the contribution to the likelihood function is $1 - \pi(\mathbf{x}_i)$. Now this can be expressed for the observation $(\mathbf{x}_i,\ y_i)$ as:

$$\pi(\mathbf{x}_i)^{y_i} [1 - \pi(\mathbf{x}_i)]^{1-y_i} \tag{3.7}$$

Assume that a sample of n independent observations $(\mathbf{x}_i,\ y_i),\ i = 1, 2, \ldots, n$. The likelihood function is obtained as a product of the terms given by the above expression

$$L(\beta) = \prod_{i=1}^{n} \pi(\mathbf{x}_i)^{y_i} [1 - \pi(\mathbf{x}_i)]^{1-y_i} \tag{3.8}$$

Now β has to be estimated so that $L(\beta)$ is maximized. It is mathematically easier to work with the natural logarithm of this equation. The log likelihood function is defined as:

$$l(\beta) = log(L(\beta)) = \sum_{i=1}^{n} \{y_i\ log(\pi(\mathbf{x}_i)) + (1 - y_i)\ \ log(1 - \pi(\mathbf{x}_i))\} \tag{3.9}$$

The $k+1$ likelihood equations will be obtained by differentiating the log likelihood function with respect to the $k+1$ coefficients. In linear regression, the normal equations

obtained by minimizing the SSE, was linear in the unknown parameters that are easily solved. In logistic regression, minimizing the log likelihood yields equations that are nonlinear in the unknowns, so numerical methods are used to obtain their solutions.

Let $\hat{\beta}$ denote the solution to these equations. The method of estimating the variances and covariances of the estimated coefficients follows from the theory that estimators are obtained from the matrix of second partial derivatives of the log likelihood function.

Let the $(k+1)\,(k+1)$ matrix containing the negative of these partial derivatives be denoted by $\mathbf{I}(\beta)$. This matrix is called the observed information matrix. The variances and covariances are obtained from the inverse of the matrix, which is denoted by:

$$\mathbf{var}(\hat{\beta}) = \mathbf{I}^{-1}(\beta) \tag{3.10}$$

In most cases, if not always, it is not possible to write explicit expressions for the elements in this matrix. $var(\hat{\beta}_j)$ will be used to denote the jth diagonal element of the matrix, which is the variance of $\hat{\beta}_j$ and $Cov(\hat{\beta}_j,\ \hat{\beta}_i)$ to denote the covariance of $\hat{\beta}_j$ and $\hat{\beta}_i$. The estimators of the variances and covariances are obtained by evaluating $\mathbf{var}(\beta)$ at $\hat{\beta}$.

The estimated standard errors of the estimated coefficients will mostly be used, which are:

$$SE\ \left(\hat{\beta}_j\right) = \sqrt{Var\left(\hat{\beta}_j\right)},\ for\ j = 1,\ 2,\ \ldots, k \tag{3.11}$$

A useful formulation of the information matrix is:

$$\hat{\mathbf{I}}\left(\hat{\beta}\right) = \mathbf{X}^T\mathbf{V}\mathbf{X} \tag{3.12}$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1k} \\ 1 & x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix} \tag{3.13}$$

$$\mathbf{V} = \begin{bmatrix} \hat{\pi}_1\left(1-\hat{\pi}_1\right) & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \hat{\pi}_2\left(1-\hat{\pi}_2\right) & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \hat{\pi}_n\left(1-\hat{\pi}_n\right) \end{bmatrix} \tag{3.14}$$

3.3. Testing Significance of the Model

This section with its subsections is taken from [10] and slightly modified. After estimating the coefficients for the model, significance of the variables should be assessed. This involves the formulation and testing of statistical hypotheses to determine whether the independent variables in the model are significantly related to the dependent variable.

3.3.1. Deviance

The significance of the slope coefficient is evaluated by comparing the observed values of the response variable to predicted values obtained from models with and without the predictors. In logistic regression, comparison of observed to predicted values is based on the log likelihood function.

To better understand this comparison, it is helpful conceptually to think of an observed value of the response variable as also being a predicted value resulting from a saturated model. A saturated model is one that contains as many parameters as there are predictors.

The comparison of the observed to predicted values using the likelihood function is based on the following expression:

$$D = -2\ log\left[\frac{likelihood\,(fitted)}{likelihood\,(saturated)}\right] \tag{3.15}$$

The quantity inside the large brackets in the above expression is called the likelihood ratio. Using minus 2 times the log is necessary to obtain a quantity whose distribution is known and can therefore be used for hypothesis testing. Such a test is called the likelihood ratio test and is described fully below.

Substituting the likelihood function gives us the deviance statistic:

$$D = -2\ \sum_{i=1}^{n}\left\{y_i\ log\left(\frac{\hat{\pi}\,(\mathbf{x}_i)}{y_i}\right) + (1-y_i)\ \ log\left(\frac{1-\hat{\pi}\,(\mathbf{x}_i)}{1-y_i}\right)\right\} \tag{3.16}$$

Where the values of the outcome variable are either 0 or 1, the likelihood of the saturated model is 1. Specifically, it follows from the definition of a saturated model $\hat{\pi}_i = y_i$ and the likelihood is:

$$likelihood\,(saturated) = \prod_{i=1}^{n} {y_i}^{y_i} \;\; [1 - y_i]^{1-y_i} = 1 \tag{3.17}$$

Thus, it follows that the deviance is:

$$D = -2 \; log\,[likelihood\,(saturated)] \tag{3.18}$$

3.3.2. Likelihood Ratio Test

Likelihood ratio test is used to assess the overall significance of the k independent variables in the model by comparing the likelihood of the fitted model (L1) to the likelihood of a constant only model (L0).

The likelihood-ratio test uses the ratio of the maximized value of the likelihood function for the full model (L1) over the maximized value of the likelihood function for the simpler model (L0). The full model has all the parameters of interest in it. The simpler model is said to be a nested, reduced model, where all independent variables are dropped from the overall model. The likelihood-ratio test tests if the logistic regression coefficients for the dropped variables can be treated as zero, thereby justifying the dropping of the variables from the model. A non-significant likelihood-ratio test indicates no difference between the full model and the reduced model, hence justifying dropping the given variables so as to have a more parsimonious model that works just

as well. The likelihood-ratio test statistic equals:

$$G = -2log\left(\frac{L_0}{L_1}\right) = -2\left[log\left(L_0\right) - log\left(L_1\right)\right] = -2\left(l_0 - l_1\right) \tag{3.19}$$

This log transformation of the likelihood function yields an approximate chi-square statistic.

The hypothesis test:

$$H_0 : \beta_i = 0,\ i = 1,\ 2,\ \ldots,\ k \tag{3.20}$$

$H_0 : There\ is\ no\ difference\ between\ the\ fitted\ and\ full\ (/intercept\ only)\ model$

Under H_0, the test statistic G will have a chi-square distribution with k degrees of freedom.

Note that if H_0 is rejected, the conclusion is that at least one or perhaps all k coefficients are significantly different from zero.

3.3.3. Wald Test

Likelihood ratio test indicates that any one of the variables or all of the variables are significant but does not give an indication of their individual significance. Before concluding that any or all of the coefficients are nonzero, one needs to look at the univariate Wald test statistics.

The Wald test is used to test the statistical significance of each coefficient (β_j) in the model. A Wald test calculates a Z statistic which is:

$$W_j = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \tag{3.21}$$

This value is squared which yields a chi-square distribution and is used as the Wald test statistic. (Alternatively, the value can be directly compared to a normal distribution.)

Under the null hypothesis that the coefficient is equal to zero, this statistic follows a standard normal distribution. Alternatively, this statistic can be squared and will then follow a chi-square distribution with 1 degree of freedom. Either way, it gives equivalent results.

Several statisticians have identified problems with this statistic. For large logit coefficients, the standard error is inflated, lowering the Wald statistic and leading to Type II errors (false negatives: thinking the effect is not significant, when it is).

If a variable is found to be not significant, a reduced model can be fitted and compared to the full model. The aim is to get the best fitting model, while keeping the number of variables to a minimum. The likelihood ratio test can again be used. If the null hypothesis is not rejected, the conclusion is that the reduced model is just as good as the full model. Note that statistical significance isn't the only consideration when variables are included or excluded in the model in practice. Sometimes business might require a variable to be included or legally a variable must be excluded. These considerations might prove to be more important than statistical significance.

The Wald test is obtained from the following matrix calculation:

$$W = \hat{\beta}^T \left[var(\hat{\beta}) \right]^{-1} \hat{\beta} = \hat{\beta}^T \left[\mathbf{X}^T \mathbf{V} \mathbf{X} \right]^{-1} \hat{\beta} \tag{3.22}$$

Which will be distributed as chi-square with $k+1$ degrees-of-freedom under the hypothesis that each of the $k+1$ coefficients are equal to zero. Tests for just the k slope coefficients are obtained by eliminating $\hat{\beta}_0$ from $\hat{\beta}$ and the relevant row and column from $\mathbf{X}^T\mathbf{V}\mathbf{X}$. Since evaluation of this test requires the capability to perform vector-matrix operations and to obtain $\hat{\beta}$, there is no gain over the likelihood ratio test of the significance of the model.

3.3.4. Score Test

The Score test for the significance of the model is based on the distribution of the k derivatives of $L(\beta)$ with respect to β. The Score test does not require the computation of the maximum likelihood estimates for the coefficients.

Suppose that $\hat{\beta}$ is the maximum likelihood estimate of β under the null hypothesis $H_0 : \beta = \hat{\beta}$. The chi-square score statistic for testing H_0 is defined by,

$$U^T \left(\hat{\beta} \right) \; I^{-1} \left(\hat{\beta} \right) \; U \left(\hat{\beta} \right) \; \sim \chi^2 (k) \tag{3.23}$$

asymptotically under H_0, where k is the number of constraints imposed by the null hypothesis and the score is

$$U \left(\hat{\beta} \right) = \frac{\partial log L \left(\hat{\beta} | \mathbf{x} \right)}{\partial \beta} \tag{3.24}$$

The observed Fisher information is:

$$U^T\left(\hat{\beta}\right)\ I^{-1}\left(\hat{\beta}\right)\ U\left(\hat{\beta}\right)\ \sim \chi^2\left(k\right) \tag{3.25}$$

$$I\left(\hat{\beta}\right) = -E\left[\frac{\partial^2 logL\left(\hat{\beta}|\mathbf{x}\right)}{\partial\beta\partial\beta^T}\right] \tag{3.26}$$

The computation of this test has the same level of complexity as the calculation of the Wald test.

3.4. Selection Methods

In order to select the best subset of predictors for the fitted model, one of the following methods can be applied:

- Forward selection method starts with a null model that does not contain any predictors, but the intercept. It adds variables to the model one at a time. At each step, the most significant variable according to likelihood ratio test is selected to enter the model. If none of the remaining variables have a p-value less than the selected threshold, it stops. A variable cannot be deleted after it has been included in the model.
- In contrast to forward selection, backward selection begins with all the candidate variables in the model and then iteratively removes the least significant predictor. This process continues until there is no more significant variable according to significance level specified by the user.
- Stepwise regression is a combination of forward and backward selection methods. After each step where a variable is added, it is checked to see if the significance of all candidate variables in the model has fallen below the specified threshold. If

a nonsignificant variable is found, it is removed from the model. This addresses the situation in which the variables were added or removed in the early stages of the process, and we would like to consider it later. At each stage, a variable can be added or removed. Stepwise regression requires two levels of significance. One for adding variables, one for removing variables. The cut off probability for adding variables should be less than the cut off probability for removing variables to prevent the process from entering an infinite loop.

AIC (Akaike's Information Criterion) and SBC (Schwarz's Bayesian Criterion, or Bayesian Information Criterion, abbreviated as BIC) provide an indication of model quality relative to other models. They are used for model selection, and penalize for adding parameters to the model. Smaller values of these criteria are preferred [7].

3.5. Interpretation of the Fitted Model

This section is taken from [10] and slightly modified. After fitting a model, the emphasis shifts from the computation and assessment of significance of the estimated coefficients to the interpretation of their values. It will now be assumed that the logistic regression model has been fit, that the variables in the model are significant in either a statistical or business sense, and that the model fits according to some statistical measure of fit.

The answer to the following question is now needed: "What do the estimated coefficients in the model tell one about the question that motivated the model?" Interpretation involves two issues: determining the functional relationship between the dependent variable and the independent variables, and appropriately defining the units of change for the independent variables.

The first step is to determine the link function, i.e. what function of the dependent variable yields a linear function of the independent variables. In linear regression, it is the identity function, as the dependent variable is by definition linear in its parameters.

In logistic regression, it is the logit transformation:

$$g(\mathbf{x}) = log\left(\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right) = \beta^{\mathbf{T}}\mathbf{x} \tag{3.27}$$

For simplicity, suppose a simple logistic regression, i.e. a binary logistic regression with a single independent variable is of concern.

In linear regression, the slope coefficient β_1 for a single independent variable x_1 is equal to the difference between the value of the dependent variable at $x_1 + 1$ and the value of the dependent variable at x_1, for any value of x_1. Thus, the interpretation of the coefficient is relatively straightforward as it expresses the resulting change in the measurement scale of the dependent variable for a unit change in the independent variable.

In the logistic regression model, the slope coefficient represents the change in the logit corresponding to a unit change in the independent variable, $\beta_1 = g(x_1 + 1) - g(x_1)$. A meaning needs to be placed on the difference between the two logits, to interpret the coefficient.

A basic example will first be used, as this will provide the conceptual foundation for all other situations. Assume that the independent variable x_1, is coded as either 1 or 0. The difference in the logit for a subject for $x_1 = 1$ and $x_1 = 0$ is:

$$\beta_1 = g(1) - g(0) = (\beta_1 + \beta_0) - \beta_0 = \beta_1 \tag{3.28}$$

In this case the logit difference is β_1. In order to interpret this, a measure of association, called the odds ratio needs to be introduced.

The possible values of the logistic regression are displayed in Table 3.1.

Table 3.1. Probabilities of the different possible outcomes in logistic regression.

	Independent variable (x_1)	
Outcome variable (**y**)	$x_1 = 1$	$x_1 = 0$
$y = 1$	$\pi(1) = \frac{e^{\beta_0+\beta_1}}{1+e^{\beta_0+\beta_1}}$	$\pi(0) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$
$y = 0$	$1-\pi(1) = \frac{1}{1+e^{\beta_0+\beta_1}}$	$1-\pi(0) = \frac{1}{1+e^{\beta_0}}$
Total	1	1

The odds of the outcome being present among individuals with $x_1 = 1$ is defined as:

$$\frac{\pi(1)}{1-\pi(1)} \tag{3.29}$$

Similarly, the odds of the outcome being present among individuals with $x_1 = 0$ is defined as:

$$\frac{\pi(0)}{1-\pi(0)} \tag{3.30}$$

The odds ratio, OR, is defined as the ratio of the odds for $x_1 = 1$ and the odds for $x_1 = 0$ and is given by the equation:

$$OR = \frac{\frac{\pi(1)}{1-\pi(1)}}{\frac{\pi(0)}{1-\pi(0)}} \tag{3.31}$$

Substituting the expressions in the above table gives:

$$OR = \frac{\frac{\frac{e^{\beta_0+\beta_1}}{1+e^{\beta_0+\beta_1}}}{\frac{1}{1+e^{\beta_0+\beta_1}}}}{\frac{\frac{e^{\beta_0}}{1+e^{\beta_0}}}{\frac{1}{1+e^{\beta_0}}}} = \frac{e^{\beta_0+\beta_1}}{e^{\beta_0}} = e^{\beta_1} \tag{3.32}$$

Hence, for a logistic regression with a dichotomous independent variable, coded zero and one, the relationship between the odds ratio and the regression coefficient is:

$$OR = e^{\beta_1} \tag{3.33}$$

The odds ratio is a measure of association which approximates how much more likely it is for the outcome to be present among those with $x_1 = 1$ than those with $x_1 = 0$.

The odds ratio (OR) is usually the parameter of interest in a logistic regression due to its ease of interpretation. Its estimate $\widehat{OR}$ tends to have a distribution that is skewed, due to the fact that the possible values range between 0 and ∞. Inferences are usually based on the sampling distribution of $\ln\left(\widehat{OR}\right) = \hat{\beta}_1$. A 100 $(1-\alpha)$ % confidence interval estimate for the odds ratio is obtained by first calculating the interval for the coefficient, β_1, then taking the exponent of these values. In general, the endpoints are given by:

$$\exp[\hat{\beta}_1 \pm z_{1-\frac{\alpha}{2}} SE\left(\hat{\beta}_1\right)] \tag{3.34}$$

Because of the importance of the odds ratio as a measure of association, software packages, like SAS, automatically provide point and confidence interval estimates based on the exponent of each coefficient in a fitted logistic regression model.

It is also important to consider the effect that coding of the variable has on the computation of the odds ratio. It was noted that the estimate of the odds ratio was $\widehat{OR} = e^{\hat{\beta}_1}$. This is correct when the independent variable is coded as 0 or 1. Other coding requires that the value of the logit difference for the specific coding used is calculated and then this difference exponent is taken to estimate the odds ratio.

The estimate of the log of the odds ratio for any independent variable at two levels, say $x_1 = a$ versus $x_1 = b$ is the difference between the estimated logits computed at these two values:

$$\begin{aligned} \ln\left[\widehat{OR}\,(a,b)\right] &= \hat{g}\,(a) - \hat{g}\,(b) \\ &= \left(\hat{\beta}_0 + \hat{\beta}_1\ a\right) - \left(\hat{\beta}_0 + \hat{\beta}_1\ b\right) \\ &= \hat{\beta}_1\ \ (a-b) \end{aligned} \tag{3.35}$$

The estimate for the odds ratio is obtained by taking the exponent of the logit difference:

$$\widehat{OR}\,(a,b) = e^{\hat{\beta}_1\ (a-b)} \tag{3.36}$$

If only the exponent of the coefficient from the computer output is taken, one would have obtained the wrong estimate of the odds ratio. The method of coding also influences the calculation of the endpoints of the confidence interval. In general, it is given by:

$$\exp[\hat{\beta}_1\,(a-b) \pm z_{1-\frac{\alpha}{2}}\ \ |a-b|\ \ SE\left(\hat{\beta}_1\right)] \tag{3.37}$$

This relationship between the logistic regression coefficient and the odds ratio provides the foundation of the interpretation of all logistic regression results.

4. FEATURE EXTRACTION AND ELIMINATION

After the data of interest from various sources are gathered, the data are converted into a meaningful form to extract features to be used during the model development phase. Techniques such as grouping, creating flag-type indicator variables can be employed to obtain a development dataset.

After analyzing the good-bad ratio, looking at the significance of the variables and their predictive power some of the weak variables are first eliminated. One needs to pay attention that the correlation between the independent variables should be low, whereas their correlation with the dependent variable is high. In particular, the correlation is high for similarly calculated variables such as financial ratios. Hence these should be carefully examined and the ones with less explanatory power should be excluded.

As summarized in [11] characteristics that will be used in the model development should be:

(i) Logical: Predictors in the fitted model should make logical sense, so that they work not only when implemented, but also for a significant time period thereafter, and the resulting model can be conveniently explained to the business units.

(ii) Predictive: Covariates with a significant degree of predictive power should be included into the model. Gini coefficient and Information Value are measures of predictive power of variables which are explained in this chapter.

(iii) Available and stable: Variables should only be used if they will be available in future, have been stable since the sample was taken and are expected to be stable in the future. They should be excluded if they will not be captured in the future, are new and poorly populated, unstable due to infrastructure changes or problems, sensitive to inflation like income, or if they are easily manipulated by either the clients or staff.

(iv) Compliant: Banks are strictly regulated institutions. If the models are to support decision-making, the developer must ensure that the variables used are compliant with any legal, policy or ethical restrictions on their use.

(v) Customer related: When rating clients' risk, the variables should relate to them and not the lender's strategy. Lenders' interest is in customer risk, independent of the decision made, so that the decision can be made. For example, in application scoring, the customer demographics, indebtedness, loan purpose and payment behaviour are acceptable variables, but the product offered, loan term and the credit limit granted are not.

(vi) If excluded, result in acceptable levels of information loss. When reducing the number of variables, it should be done with the minimum information loss. There may be variables that are seemingly weak and highly correlated with other variables, whose exclusion reduces the final model's power.

4.1. Multicollinearity

In many instances, a lot of variables will be highly correlated with each other, especially those calculated using the same, or similar, base inputs. This gives rise to potential multicollinearity, which can lead to poor out-of-sample performance as the model is over-fitted to the data. Over-fitting is typically characterized by extremely large coefficients, high standard errors, or changes in coefficient for some variables that do not make logic sense. In instances where coefficients in the model have the "wrong" sign, a careful investigation of the relationships between the variables are necessary.

4.2. Weight of Evidence (WOE)

Each decision made is based on the probability of some event occurring. One assesses the circumstances and determines a weight of evidence. The weight of evidence converts the risk associated with a particular choice into a linear scale that is easier

for the human mind to assess. WOE of an attribute (category) of a variable is:

$$WOE_i = \ln\left(\frac{DistributionGood}{DistributionBad}\right) = \ln\left(\frac{\left(\frac{num\_good_i}{total\_num\_good}\right)}{\left(\frac{num\_bad_i}{total\_num\_bad}\right)}\right) \tag{4.1}$$

where i is the index of the attribute being evaluated. The precondition is non-zero values for all num_good_i and num_bad_i.

The WOE is used to assess the relative risk of different attributes for a characteristic and as a means to transform characteristics into variables. It is also a very useful tool for binning.

The WOE formula discussed above is the one most often used. It can be restated as:

$$WOE_i = \ln\left(\frac{num\_good_i}{num\_bad_i}\right) - \ln\left(\frac{total\_num\_good}{total\_num\_bad}\right) \tag{4.2}$$

which illustrates two components: a variable portion for the odds of that group and a constant portion for the sample or population odds. The WOE for any group with average odds is zero. A negative WOE indicates that the proportion of defaults is higher for that attribute than the overall proportion and indicates higher risk.

For a characteristic transformation, the WOE variable has a linear relationship with the logistic function, making it well suited for representing the characteristic when using logistic regression.

The WOE does not consider the proportion of observations with that attribute, only the relative risk. Other tools are used to determine the relative contribution of each attribute and the total information value.

Here, it is also necessary to mention the relationship between odds and WOE. As we fit the model to predict the default (1), i.e. bad loans, response variable of the logit regression is $\ln\left(\frac{num\_bad}{num\_good}\right)$. Thus, WOE can be defined as the logarithm of the ratio of the odds of Good-to-Bad in the attribute level to the odds of Good-to-Bad in the entire sample:

$$WOE_i = \frac{\ln\left(\frac{num\_good_i}{num\_bad_i}\right)}{\ln\left(\frac{total\_num\_good}{total\_num\_bad}\right)} \tag{4.3}$$

4.3. Information Value

The information value (IV) is used to rank order variables in terms of their predictive power. It is extended in the credit scoring context by looking at the non-occurrence (N) as non-defaults (goods) and the occurrence (P) as defaults (bads). A high information value indicates a high ability to discriminate. Values for the information value will always be positive and may be above 3 when assessing highly predictive characteristics and is typically seen in behavioural scorecards.

As shared in [12], if the IV statistic is

- Less than 0.01 then the predictor is not useful for modeling.
- 0.01–0.1 then the predictor has only a weak relationship with the response variable.
- 0.1–0.2, then the predictor has a medium strength relationship with the response variable
- 0.2 or higher, then the predictor has a strong relationship with the response variable.

It should be noted that weak characteristics may provide value in combination with others or have individual attributes that could provide value as dummy variables.

They should thus not be discarded indiscriminately. It can be difficult to interpret the information value because there are no associated statistical tests. As a general rule, it is best to use the information value in combination with other measures for final selection.

The information value, as it is known in credit scoring, is technically referred to as the Kullback divergence measure. It measures the difference between two distributions. It is expressed as:

$$IV = \sum_{i=1}^{n} \left[\left(\frac{num\_good_i}{\sum_{i=1}^{n} num\_good_i} - \frac{num\_bad_i}{\sum_{i=1}^{n} num\_bad_i} \right) WOE_i \right] \tag{4.4}$$

where i is the index of the attribute being evaluated and n is the total number of attributes.

Information value is sensitive to how the variable is binned and the number of bins. Different binning algorithms and/or different bin sizes result in different information value.

4.4. Gini Coefficient

The Gini coefficient is often referred to as an accuracy ratio or power ratio in credit scoring. It is used as a measure of how well a scorecard or variable is able to distinguish between goods and bads.

The Gini Index is a Gini coefficient expressed as a percentage. A Gini coefficient ranges from 0 to 1, so the Gini Index ranges from 0 to 100. A Gini Index is a measure of the uniformity or non-uniformity of a distribution.

Its calculation is given as:

- Categories of a variable are sorted in descending order of their event rates (here, bad rates)
- For each of these sorted categories, number of bads and goods are counted

$$Gini\_index = \left(1 - \frac{2*\sum_{i=2}^{m}\left(num\_bad_i*\sum_{j=1}^{i-1} num\_good_j\right)+\sum_{k=1}^{m}\left(num\_bad_k*num\_good_k\right)}{total\_num\_bad*total\_num\_good}\right) \quad (4.5)$$

There is not a "statistical" cut-off for the Gini index, but a Gini index of 10% or more, or for sparse data (low default rate), 5% or more can be used as cut-offs. In terms of the information value, characteristics with values of less than 0.1 are typically viewed as weak, while values over 0.3 are sought after.

5. ASSESSING THE FIT AND PREDICTIVE POWER OF THE MODEL

Predictive power of a model is its ability to generalize the rules it has learned from the training data set to an unseen data set. In this section, various techniques are introduced to assess the fit and measure the predictive power of the model developed.

5.1. Classification Table

In order to summarize the results of a fitted logistic regression model, a classification table can be utilized. This table is the result of cross-classifying the outcome variable, y, with a dichotomous variable whose values are derived from the estimated logistic probabilities. To obtain the derived dichotomous variable one must define a cut-off point, c, and compare each estimated probability to c. If the estimated probability exceeds c then let the derived variable be equal to 1; otherwise it is equal to zero. In Table 5.1, a classification table is depicted.

Table 5.1. Confusion matrix and common performance measurements.

		Value observed	
		1	0
Value Predicted	1	True Positive (TP)	False Positive (FP)
	0	False Negative (FN)	True Negative (TN)

Sensitivity $=\frac{TP}{TP+FN}$
Specificity $=\frac{TN}{TN+FP}$

- TP: True positive - Number of bad credits classified as bad
- TN: True negative - Number of good credits classified as good
- FP: False positive - Number of good credits classified as bad: Type I Error
- FN: False negative: Number of bad credits classified as good: Type II Error

Misclassification costs associated with Type II errors are usually much higher than those associated with Type I errors [13].

5.2. Receiver Operating Characteristic (ROC) Curve

In a Receiver Operating Characteristic (ROC) curve, the true positive rate (sensitivity) is plotted as a function of the false positive rate (1-specificity) for various cut-off points. Each point on the ROC curve represents a sensitivity / specificity pair corresponding to a particular decision threshold.

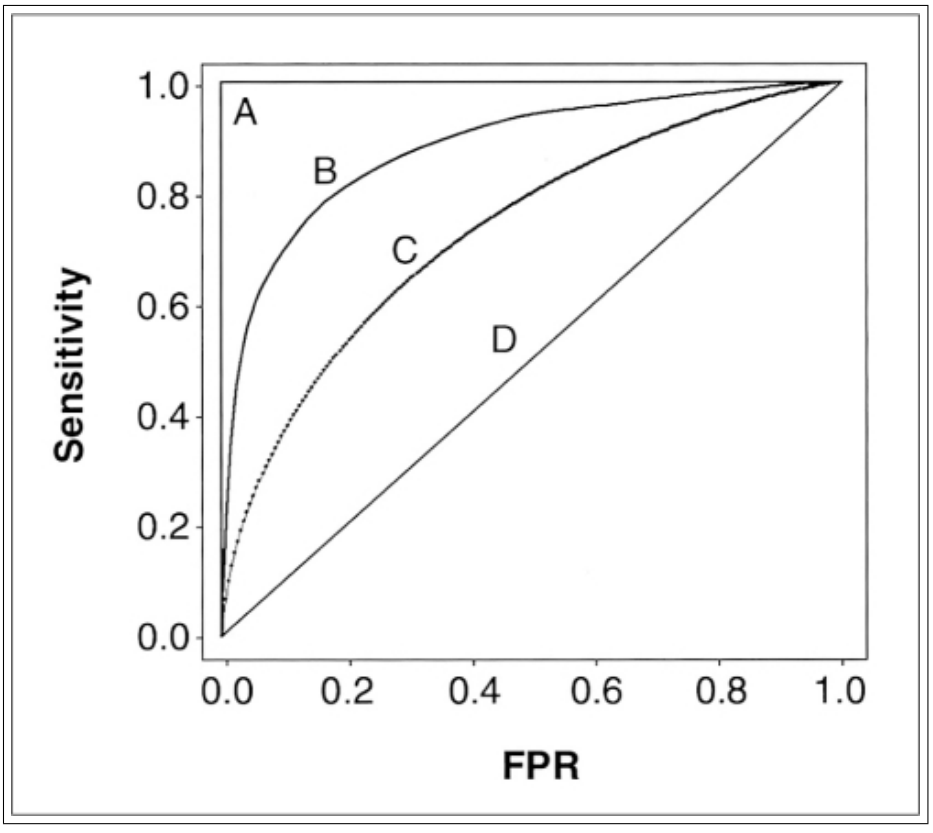


Figure 5.1. A Receiver Operating Characteristic (ROC) curve depicting four cases.

In Figure 5.1 [14], four ROC curves are presented with different areas under the ROC curve (AUROC). In a test with perfect discrimination (A), i.e. in case of two non-overlapping distributions, ROC curve passes through the upper left corner and has an AUROC that is equal to 1. The closer the ROC curve is to the upper left, the greater the area under the ROC curve and subsequently better the predictive ability of the model. If we were to rely on pure chance to distinguish default and nondefault observations, the resulting ROC curve would fall along the diagonal line (the line segment from (0, 0) to (1, 1)), which is referred to as the chance diagonal (D). The chance diagonal has an AUROC of 0.5. ROC curves of tests with some ability to distinguish between default and nondefault observations (B, C) lie between A and D. Test B with higher AUROC has a better discriminatory power than test C.

In order to compare alternative models in this study, Gini coefficient is used which is equivalent to 2 * AUROC – 1, i.e. twice the area between the curve and the chance diagonal [15].

5.3. The Kolmogorov-Smirnov (KS) Test Statistic

Let $P_g(s)\, and P_b(s)$ be the cumulative distribution functions of the scores for the good class and the bad class, respectively [15]. The Kolmogorov-Smirnov (KS) statistic is then defined as

$$KS = \max_s |P_g(s) - P_b(s)| \tag{5.1}$$

When calculating the KS-statistic cumulative distributions of goods and bads, $p_g(s)\, and p_b(s)$, can be plotted.

5.4. From Odds to Score Transformation

Scaling refers to the range and format of scores in a scorecard and the rate of change in odds for increases in score [7]. Scorecard scores can take several forms with

decimal or discrete number scores:

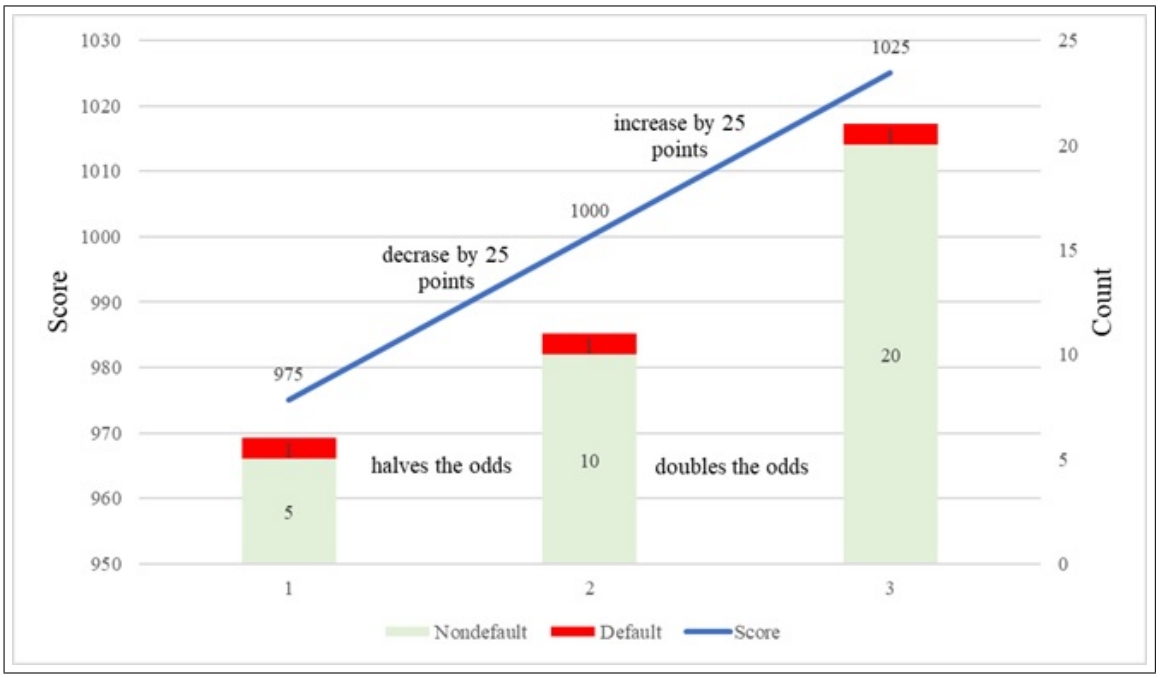


Figure 5.2. Relationship between odds ratio and the respective credit score.

- Where the score is the good/bad odds or probability of bad (e.g., score of 1000 means a 10:1 odds)
- With some defined numerical minimum/maximum scale with a specified odds ratio at a certain point (odds of 10:1) and specified rate of change of odds (double every 25 points)

The choice of scaling does not affect the predictive strength of the scorecard.

5.4.1. Score Calculation - Scaling

The relationship between log of *odds* and scores can be presented with the following linear transformation:

$$\text{Score} = \text{offset} + \text{factor} * \log(odds) \tag{5.2}$$

Where the scores are calculated using *odds* at a Score and specified "points to double the odds" (*pdo*), the factor and offset can easily be calculated by using equations (5.2) and (5.3) simultaneously.

$$\text{Score} + pdo = \text{offset} + \text{factor} * \log(2 * odds) \tag{5.3}$$

Solving the equations above for *pdo*, we get

$$pdo = \text{factor} * \log(2) \tag{5.4}$$

Therefore,

$$\text{factor} = \frac{pdo}{\log(2)} \tag{5.5}$$

$$\text{offset} = \text{Score} - \text{factor} * \log(odds) \tag{5.6}$$

Score is equivalently defined with the following equation.

$$\text{score} = pdo * (ln(odds) - ln(pdo)) \ / \ ln(2) + \text{Score} \tag{5.7}$$

5.5. Reject Inference

A credit scoring model is trained only using the originated loans, i.e. applications approved earlier. If a loan application is rejected, it is not possible to know the

future performance of the rejected applicant. This leads to one of the biggest problems regarding the evaluation of credit scoring models, namely a rejection bias. The easiest way to handle rejection bias is to approve all applications. However, this is virtually impossible because the risk associated with this process is very high. If a loan application of the rejected customers is approved by other banks, the payment history of the rejected customers can be collected from the credit bureaus. On the other hand, banks generally apply similar scoring techniques and it is unlikely that rejected applicants will be approved by other banks for other loan applications. While the credit history of rejected customers is hard to find by the credit bureaus, the members of the credit bureau also have to pay each customer's payment history. In order to cope with the negative effects of rejection, there are other techniques in the literature, such as reclassification, reweighting and extrapolation

In this study, we could not carry out reject inference, because it is simply impossible to specify the rejected customers for all the institutions. Each institution stores this type of information in its own database which we were not allowed to use in the scope of this study.

6. OBTAINING THE RESPONSE VARIABLE AND THE EXPLANATORY VARIABLES

Today, importance of data for institutions in every sector as well as for financial institutions is obvious. Financial institutions invest huge amounts of money to build and maintain data warehouses (enterprise data warehouse, abbreviated as EDW). Not only they collect and store data from their existing and prospective customers, but also have access to external sources of data. One of the most important external sources is the Credit Bureau, if it exists, in the country they operate.

In this thesis, our aim is developing a universal consumer loans application scorecard, i.e. a scorecard which is applicable in any financial institution that is eligible to obtain data from the Credit Bureau in Turkey. Consequently, we decided not to use confidential or in-house data which are collected during application or derived from consumer specific records in a particular bank's database. Thus, we carry out the studies without a complete data set which would also include customer demographics such as gender, age and marital status, deposits and wealth data, and additional data sources which get richer day by day such as GPS-based data, data based on social media and technology usage.

6.1. Preparation of Credit Bureau Data for the Scorecard Experiments

For each distinct application for a consumer loan, credit history of the applicant is inquired from the Credit Bureau. The applicant may be a customer of the bank, in this case the bank has some useful information about the customer's credit history in the bank, such as his pay-back behavior, spending habits, trends in limit usage. However, it is also important to get historical data of this customer at other financial institutions to determine whether a loan should be granted to this specific customer, if positive, the amount, number of installments and even interest rate are specified. If the applicant is not a customer of the bank, historical data in the financial industry are

more crucial. That's why, each member financial institution shares its data with this institution, namely Credit Bureau, and the Credit Bureau supplies the accumulating data for each customer, individual or corporate.

We decided to develop an application scorecard using the consumer loans granted between 01/07/2014 and 30/06/2015 with the following restrictions:

- Only consumer loans in Turkish Liras (TRY).
- TCKN (Turkish Citizen ID Number) should not be empty, i.e. only Turkish citizens.
- The Credit Bureau inquiry should not be older than 6 months (day difference between inquiry date and start date of consumer loan should be less than 181 days).
- The applicant should be the main applicant, i.e. he/she should be the person who is going to use the loan personally.
- Interest rate should be constant during the lifetime of the credit.

Throughout the study, we have used consumer loans and cash loans interchangeably. How we obtained the final development data set is explained step by step in this chapter.

6.1.1. Determination of Consumer Loans Between 01/07/2014 and 30/06/2015

A bank inquires a customer's Credit Bureau records when he/she applies for a product such as credit cards and any type of loans, or for marketing activities periodically. An inquiry returned includes his/her historic records in several categories. In this thesis, we make use of two categories to obtain the development data set:

- GGCP includes information about customer's applications to any bank in the system for particular products 6 months before the inquiry date.

- GGCI includes historic usage information such as initial amount, total outstanding balance and payment behavior from the beginning of customer's first financial product usage from any bank until the inquiry date.

Big banks store each inquiry and information about the related applications in their EDW without deleting the previous inquiries made for any customer. This is necessary to use the data later, because if one wants to evaluate the application at any time in the future, most up-to-date inquiry would probably not express the customer's characteristics at that time.

We utilized one of the Turkey's big banks Credit Bureau inquiries stored on its EDW. These data are its own property, and also reflect its customer base. We paid utmost attention not to violate the Bank's confidentiality. Thus, we cannot publish the share of loans lent by the Bank and any other bank specific data.

In an inquiry, all the loans granted to that person, identified with his/her TCKN, in his financial history are listed. So, we first determined the oldest inquiry, which has the smallest inquiry number, in which a specific REFERENCEKEY is seen for the first time. So, the distinctive ID for a consumer loan is "TCKN || REFERENCEKEY".

First, we specified the consumer loans lent by the Bank with respect to the restrictions stated earlier. There are some fields we can only observe for the Bank's credits in an inquiry: CUSTOMERACCOUNTNUMBER, USERCOMPANYBRANCH, APPLICATION-REFERENCENUMBER, etc. Using these fields, we compared the data obtained with the actual data in the Bank's own tables. We observed that the actual number of consumer loans and the obtained number are close to each other. The reason for difference is possibly the constraints listed above such as only TRY credits for Turkish citizens. When we compared the data using the APPLICATION-REFERENCENUMBER field, then almost all of them intersect. The difference is due to movement of credits from one branch to another (When a branch closes or unites with another one, then loans lent by that branch take on another CREDIT_REFERENCE.) In the intersection, STARTDATE of a consumer loan exactly the same for the

actual and obtained data. Since they are consistent, we made sure that the obtained data are correct and reliable.

Union of Banks in Turkey (Türkiye Bankalar Birliği, abbreviated as TBB) publishes reports about loans granted every quarter. For the development period we specified, the figures for the consumer loans are summarized in Table 6.1.

Table 6.1. TBB data on consumer loans granted between September 2014 and June 2015.

	Total amout granted, in million TRY		**Number of people**	
Term	**Consumer Loan**	**Other**	**Consumer Loan**	**Other**
2014 - 3 (September)	22,718	6,773	2,176,135	618,527
2014 - 4 (December)	23,197	6,927	2,038,246	602,179
2015 - 1 (March)	27,549	2,731	2,333,476	272,572
2015 - 2 (June)	26,509	3,128	2,287,027	308,788
Total	99,973	19,559	8,834,884	1,802,066
	119,532		10,636,950	

Other credits are probably also stated as consumer loans to the Credit Bureau, since there is not any other specific FINANCETYPE field for these credits. For consumer loans FINANCETYPE = '2' and for example for credit cards FINANCETYPE = '23'. However, there is not any specific FINANCETYPE field for "other loans".

Finally, we specified the loans granted by other banks, whose names are masked by letters. Considering the intersections between consumer loans and other loans, and between quarters, we figured out that we have enough data to represent the other banks' data successfully.

6.1.2. Determination of Credit Inquiries for Consumer Loans Granted to Be Used During Scoring Process

After we determined all the credits granted, for each consumer loan we kept the inquiry number, where this credit is seen for the first time. Subsequently, we searched for any previous credit bureau inquiry for that customer which we would use to create variables depicting the customer's characteristics right before he/she was granted for the loan.

First, the relevant credit bureau inquiry for a consumer loan granted by the Bank was found as the most recent credit bureau inquiry before the credit bureau inquiry where the loan was first observed. For most of consumer loans, a previous credit bureau record was found which was not older than 181 days (6 months). Then the inquiries found were compared with the actual data. For the intersection stated earlier, the inquiry number used is the same for most of the loans. The difference is caused by mostly newer inquiries than the actual inquiries used, so there is no drawback to use the ones obtained, on the contrary, it is correct to use them for the new scoring process.

After we made sure, that this method applies, we obtained the credit bureau inquiries for consumer loans granted by other banks. For consumer loans, a previous credit bureau record was found which was not older than 181 days (6 months) are taken into account for the next step.

6.1.3. Assigning the Target Variable

In order to assign a target variable for each loan we first specified the most recent Credit Bureau inquiry after the loan was granted. This inquiry should satisfy these conditions:

(i) If CLOSEDATE field in the most recent inquiry related to the consumer loan has a date value, then it is not important when the inquiry is done.

(ii) If CLOSEDATE field in the most recent inquiry related to the consumer loan is empty, then the MONTHOFLASTUPDATE field should satisfy one of the following conditions:

- If the MONTHOFLASTUPDATE has a date value (e.g. 20160502) then this field should be at least 12 months later after the STARTDATE of the credit.
- If the MONTHOFLASTUPDATE has not a date value (e.g. 20160500: depicting only the month of the last update related to this credit) then this field is transformed into date, as the first day of that month and this date should be at least 12 months later after the first day of the month following the STARTDATE. For example, if the STARTDATE is 17/03/2015, then the MONTHOFLASTUPDATE should be equal to or later than 01/04/2016.

If the inquiry satisfies the conditions above, then first the NPL date is calculated. If any of CLOSEDATE, LITIGATIONDATE, DEFAULTDATE, WRITEOFFRECOVEREDDATE in this inquiry has a date value and ACCOUNTSTATUS takes any of the values given in Table 6.2, then the minimum of CLOSEDATE, LITIGATIONDATE, DEFAULTDATE, WRITEOFFRECOVEREDDATE is the NPL date.

Table 6.2. Descriptions of values ACCOUNTSTATUS may take.

ACCOUNTSTATUS	**Description**
32	Closed for usage with risk
81	Closed, default or legal, still recovering
82	Closed, default or legal, recovered
83	Closed, default or legal, written off
84-88	Reserved for future default or legal statuses

If the NPL date between the STARTDATE and the date exactly 12 months after the STARTDATE, then the target variable is 1: NPL in 12 months, otherwise 0.

Previously, we shared the target definition according to the Basel II accords: being classified as NPL or being delinquent at least 90 days in 12 months after the

loan was granted. Days of delinquency information are stored in banks' own EDW in order to take early action when a borrower shows signals of being default. However, we could not specify exact days of delinquency from the Credit Bureau tables. The field ACCOUNTPAYMENTSTATUS shows a string of delinquent payments for at most 36 months. Suppose for a consumer loan without delinquency for the last 3 months, a payment is delinquent for the first time, the field takes a string of 1000. If the following payment is also delinquent, then the string becomes 21000. If the third payment is again delinquent, it takes 321000, however we cannot distinguish 63 days of delinquency from 78 days. Lots of consumer loans with close to 90 days of delinquency are managed by professionals and "saved" before they are classified as NPL. Briefly, we could not specify at least 90 days of delinquency from the tables and concluded to continue with a target definition of being classified as NPL.

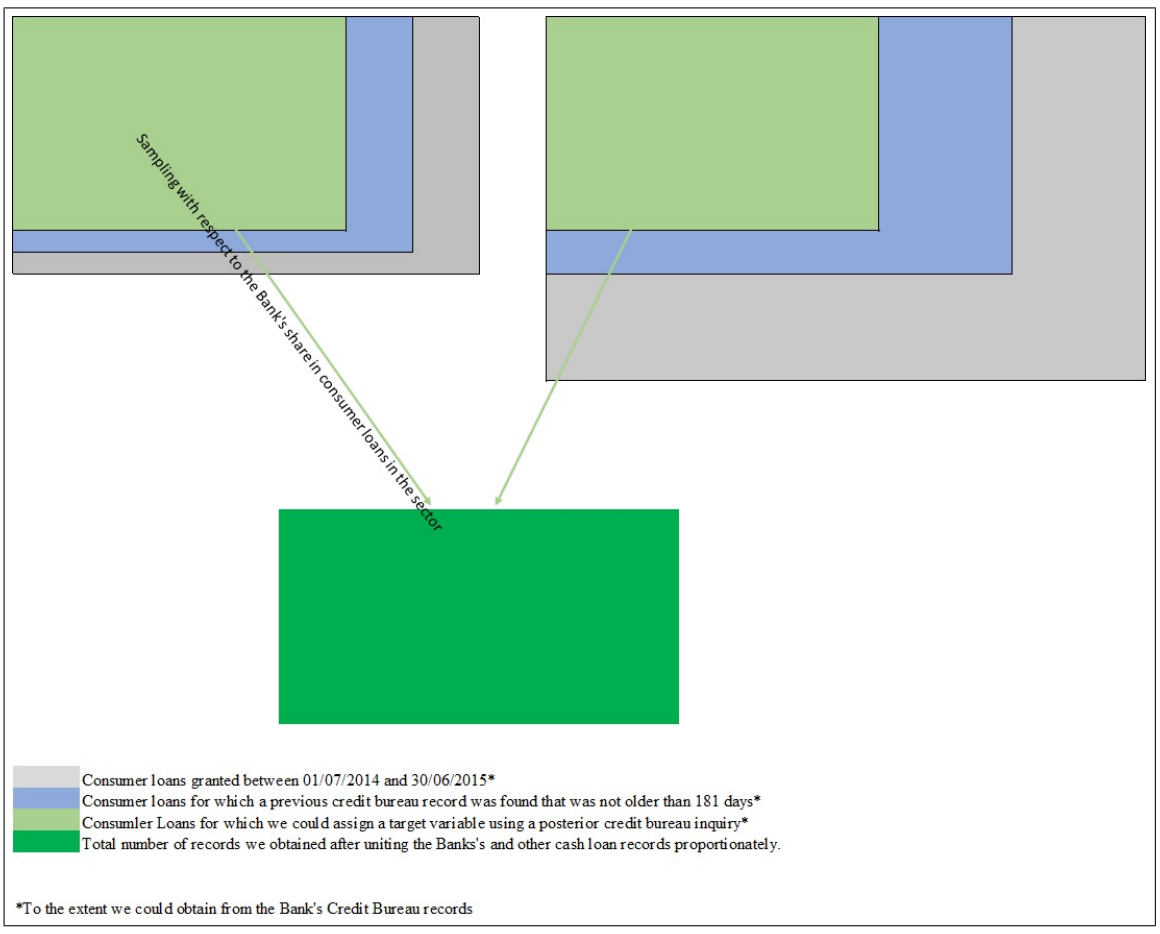


Figure 6.1. Diagram summarizing the creation of the data set.

After we compared the target obtained with the target information in the Bank's database, we made sure that they significantly intersect, so this approach can be used to determine the target variable. Finally, we obtained several million of consumer loan records granted between 01/07/2014 and 30/06/2015 that had a previous credit bureau record which was not older than 181 days (6 months) and we could assign a target variable using a posterior credit bureau inquiry that satisfy the conditions listed above. To summarize the whole process, the diagram in Figure 6.1 would be beneficial.

6.2. Calculation of Explanatory Variables

After the consumer loans granted and relevant credit bureau inquiries were determined and their target was calculated, we had several millions of records. Using the fields in GGCI and GGCP tables that we previously introduced, a large set of variables were calculated for each consumer loan. The initial credit bureau variables data set consists of several thousands of variables. These variables were created and derived from each other by the Bank's modelling team. Since it is the Bank's intellectual property, we would not share their calculation in this thesis. However, any professional who is familiar with the Credit Bureau tables, can create thousands of variables depending on his/her experience and creativity. It is not easy to work with that number of variables, because firstly a table in SQL cannot have more than 1000 columns, secondly running a logistic regression using any selection method (forward, backward, stepwise) with that number of variables is extremely time consuming. Therefore, we worked on the elimination of variables and reduced the number of variables.

Before we began to study further with R, we made use of SAS Enterprise Miner (SAS EM) in order to initially eliminate some of the variables and reduce the size of the data set.

In this section of the report, we will give brief information on the following SAS EM nodes and their outputs:

(i) Interactive grouping node from Credit Scoring add-on to create categorized variables and eliminate the variables which possess low discriminatory power.
(ii) Scorecard node from Credit Scoring add-on runs a logistic regression according to the user specified settings and then a scorecard is built.

There is also another node called Variable Clustering node which is widely used by the practitioners to construct clusters of similar variables in order to eliminate variables which are highly correlated. Thus, variable reduction is carried out and multicollinearity is prevented. We will not give detailed information on this node, because for this study, we did not utilize it.

6.2.1. Advantages of Using Categorized Variables

In practice, raw variables, variables that were used as is without binning, are usually not used directly as covariates in a credit scoring model. Instead, variables are categorized. The reasons can be summarized with the following points:

(i) Through categorizing, one can easily handle missing values by putting them together in a category.
(ii) A preprocessing step to eliminate the outliers is also not needed.
(iii) Categorized variables are easy to interpret, which is of great importance for the business units.
(iv) Some information is lost; however, a consistent behavior of the variable is provided with respect to the target. For example, for an interval variable, it is secured that the target rate in a category monotonically increases/decreases as the value of the variable increases/decreases.

Categorizing is executed by Interactive Grouping node of SAS Miner, that creates two types of variables for each variable: WOE (weight of evidence) variable and

Group variable. One can adjust the settings to minimize the information loss and thus meaningful categorized variables are created. Some of the settings are listed below:

(i) Binning method: bucket or quantile: First, a variable can be binned into buckets, i.e. it can be divided into equal intervals. For example, a variable that takes a value between 0 and 100 can be divided into 10 parts like 0-10, 11-20, etc. Alternatively, a variable can be cut into quantiles according to their values, so that first 10% of the records that take values between 0 and 7 form the first group, second 10% of the records that take values between 8 and 11 form the second group, etc. After this step, buckets or quantiles are combined with a decision tree algorithm running behind according to their target ratios.

(ii) Maximum number of groups of a variable can be specified, so that the buckets/quantiles are combined with respect to this setting.

(iii) How the missing values are treated can be specified:
- A separate branch can be formed if there is any missing value in the data.
- They can be used in the split search
- They can be assigned to the largest branch
- They can be assigned to the branch that minimizes SSE among observations with missing values.

(iv) Minimum group size can be specified as count or percentage.

The node is also interactive so that one can fine-tune the categories of a variable, merge or resplit, and change borders of the categories to make business sense (rounding amount variables etc.).

With the following example the process can easily be explained. One of the variables in our final data set is CCOL_VAR_3: Total limit utility of open credit cards and overdraft loans. Approximately 3% of the records take NULL values in SQL notation for this variable, i.e. they are blank. In R notation, these fields are represented with NA. These do not arise from any error in the database, yet they have a meaning that the customer has neither overdraft limit nor credit card limit. If one's either credit card or overdraft limit is nonzero, but his/her usage is 0, then

this variable takes a value of 0. That is, they have different meanings. In such a case, categorizing a variable is helpful, so that all the missing values form a characteristic in a categorical variable. According to our settings, this variable was first split into 50 quantiles, and similar quantiles are combined with respect to "their event rate" and finally 4 groups are constructed as given in Table 6.3. Each group takes an integer number in an increasing order of the original values as given in Table 6.4.

Table 6.3. CCOL_VAR_3 in raw, GRP and WOE versions with response variable, NPL_IN_12M, for 10 observations.

	NPL_IN_12M	CCOL_VAR_3	GRP_CCOL_VAR_3	WOE_CCOL_VAR_3
1	0	0.2939	1	0.67
2	0	NA	4	-0.84
3	0	0.4758	2	0.35
4	0	0.6834	3	0
5	0	0.1356	1	0.67
6	0	0.865	4	-0.84
7	1	0.1274	1	0.67
8	0	0.5311	2	0.35
9	0	NA	4	-0.84
10	0	0.0047	1	0.67

Table 6.4. Summary of binning CCOL_VAR_3.

		Group	Weight of Evidence	Event Rate NPL_IN_12M	Percentage of Population
CCOL_VAR_3	<0.38	1	0.67	1.36%	46.9%
	[0.38, 0.56)	2	0.35	1.86%	16.8%
	[0.56, 0.74)	3	0	2.62%	14.2%
	>= 0.74 or NA	4	-0.84	5.88%	22.1%

One can use either WOE variables which are similar to continuous variables, or group variables that are encoded with dummy variables (one-hot variables) in later steps as covariates.

Table 6.4 shows that if current limit utility in the last month is less than 0.38%, then group variable GRP_CCOL_VAR_3 takes 1, WOE_CCOL_VAR_3 takes 0.67. It is clear that the categories of the variable differentiate from each other with regard to event rate.

In [16], one can find detailed information about the Interactive Grouping node.

6.2.2. Reduction and Elimination of Initial Set of Variables

Before we obtained WOE versions of each variable, we eliminated the variables with emptiness of higher than 95%. Following that, we eliminated the variables with GINI value less than 0.05, because they show little discriminatory power. Furthermore, we excluded the variables that differentiate whether any product is granted by the Bank or other banks regarding our purpose of developing a scorecard without bank-specific information. Additionally, we eliminated variables created and calculated for only overdraft loans, i.e. data set includes variables for credit cards and overdraft loans together. Thereby, we obtained a data set with relatively strong variables, however we could only reduce the number of variables to several hundreds. Therefore, we utilized the Scorecard node of SAS Miner which runs a logistic regression algorithm in order to create a scoring model.

After the first run of logistic regression algorithm with stepwise selection method and Schwarz-Bayesian Criterion, we reduced the number of variables to several dozens. Then, we sorted the variables according to their GINI coefficients in a decreasing order and calculated the pairwise Pearson correlation between each of the WOE variables. Then we eliminated the correlated variables (with an absolute value of correlation higher than 0.7) so that the variable with the highest GINI coefficient stayed in the data set but the others correlated to this variable were taken out. This approach is similar to clustering the variables according to their correlation between each other, i.e. highly correlated variables build up a cluster. Afterwards, we reran the Scorecard node with the same settings and finally reduced the number of variables to 28, which is too much for a practical scorecard but it is useful for our research. We examined each variable and their groupings and all of them exhibit meaningful groupings.

In [17] one can find detailed information about the Scorecard node.

6.3. Introduction of the Development Data Set

In practice, all the "healthy" records are used for the development process. However, for our purpose of academic studies, we decided to deal with a sample of the records. Following a random sampling, we obtained a data set of 84221 records with 28 covariates and a response variable (NPL_IN_12M). 2239 of the records take NPL_IN_12M = 1, that is default rate in our data set is approximately 2.66%.

In the final data set, each covariate has 3 versions: raw version (without applying binning), GROUP version and WOE version. As an example, the raw variable is CCOL_VAR_3, whereas GROUP version of it is GRP_CCOL_VAR_3 and WOE of it is WOE_CCOL_VAR_3. GROUP versions of the raw variables, which will be referred to as GRP variables hereinafter, that are created by SAS Interactive Grouping node as stated earlier, have been used without any change throughout this thesis. Nevertheless, WOE versions of the variables have been recalculated, because the frequencies of each group of a variable and their respective event rate have slightly changed after sampling.

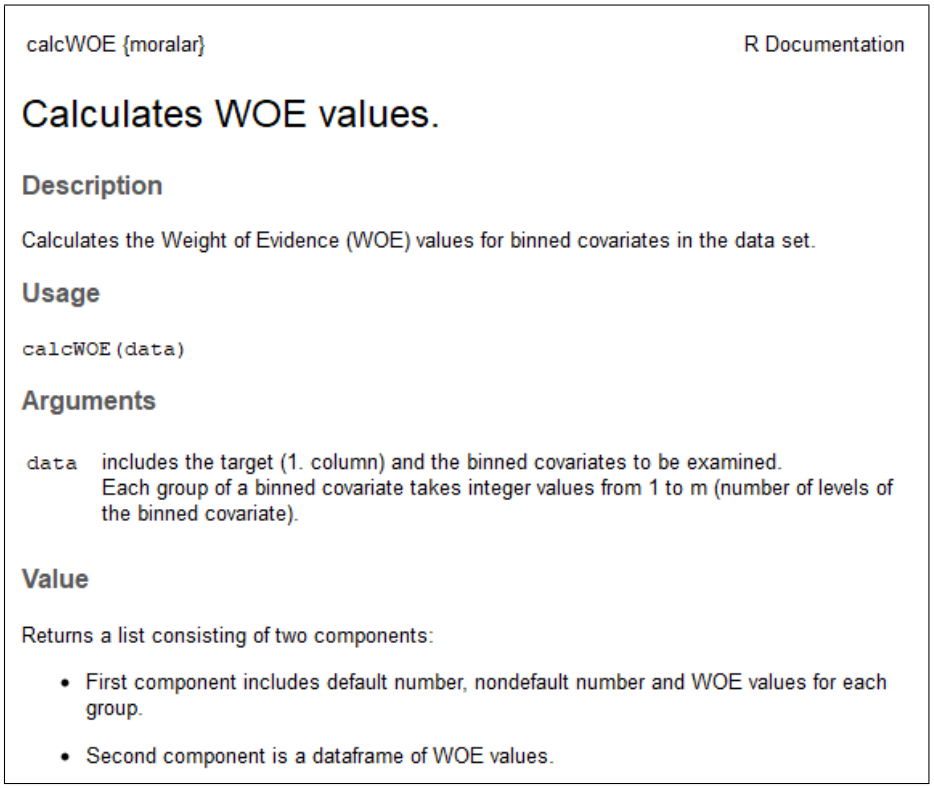
6.3.1. Recalculation of WOE Variables After Sampling

After the GRP variables are loaded, createDataPartition() function from caret package [18] is used to split the data set into training and test samples with percentages of 70% and 30%, respectively. A seed is also specified as 1234 in order to obtain the same splits each time afterwards. WOE values are calculated with calcWOE() function for the training data set, and then the calculated WOE values are also assigned with assignWOE() function to the test data set. Finally training and test data sets are bound by row, and reordered to obtain the data set of WOE variables ready to be studied further. R expressions for these operations are given in Figure 6.2.

Screenshots from the R documentations of calcWOE() and assignWOE() functions are given in Figure 6.3 and Figure 6.4, respectively.

```
library(moralar)
library(caret)

#Load masked GRP variables
load("Model_VAR_ALL_GRP_masked.RData")

data ← Var_all_GRP_masked[, -c(30)]

set.seed(1234)
pct ← 0.7
splitIndex ← createDataPartition(data$NPL_IN_12M, p = pct, list = FALSE,
times = 1)
trainSplit ← data[splitIndex, ]
testSplit ← data[-splitIndex, ]

trainSplit_WOE ← calcWOE(trainSplit)[[2]]
testSplit_WOE ← assignWOE(trainSplit, testSplit)[[2]]

rownames(trainSplit_WOE) ← rownames(trainSplit)
rownames(testSplit_WOE) ← rownames(testSplit)

Var_all_WOE_masked ← rbind(trainSplit_WOE, testSplit_WOE)

Var_all_WOE_masked ← Var_all_WOE_masked [order(as.numeric
(row.names(Var_all_WOE_masked))), ]

save(Var_all_WOE_masked, file = "Model_VAR_ALL_WOE_-
masked.RData")
```

Figure 6.2. Principal Component Analysis Algorithm.



calcWOE {moralar} R Documentation

Calculates WOE values.

Description

Calculates the Weight of Evidence (WOE) values for binned covariates in the data set.

Usage

```
calcWOE(data)
```

Arguments

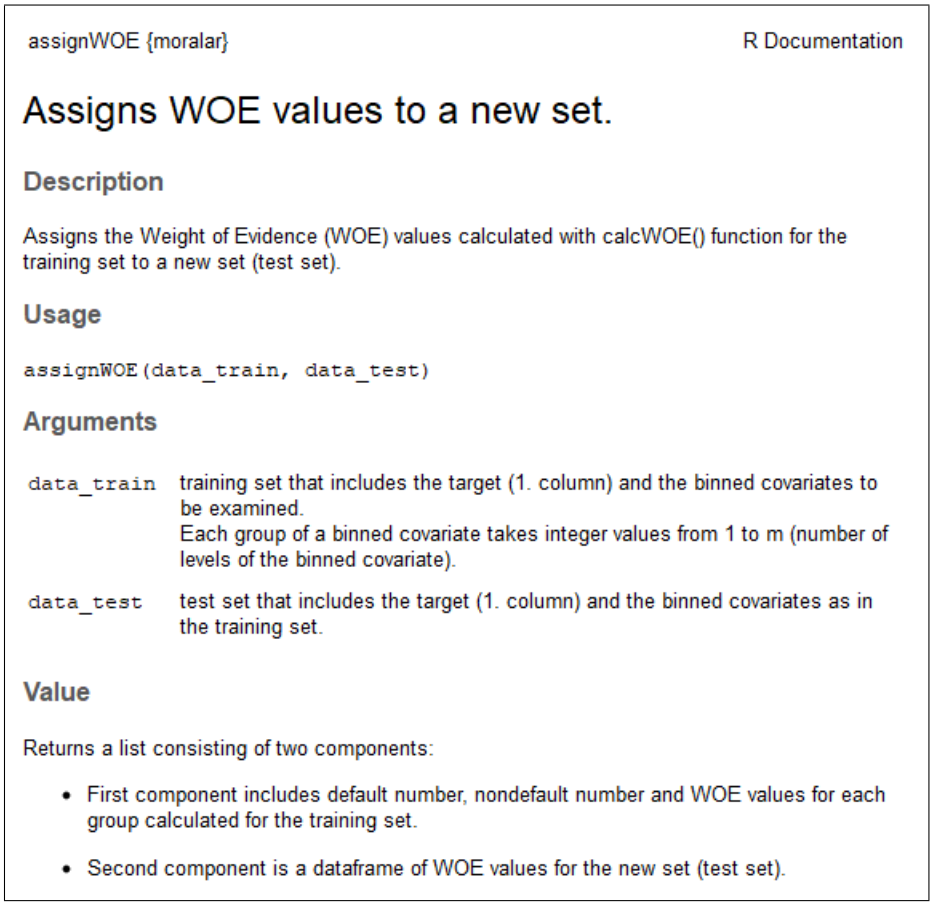
`data`	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).

Value

Returns a list consisting of two components:

- First component includes default number, nondefault number and WOE values for each group.
- Second component is a dataframe of WOE values.

Figure 6.3. Screenshot from the R documentation of calcWOE() function.



assignWOE {moralar} R Documentation

Assigns WOE values to a new set.

Description

Assigns the Weight of Evidence (WOE) values calculated with calcWOE() function for the training set to a new set (test set).

Usage

```
assignWOE(data_train, data_test)
```

Arguments

`data_train`	training set that includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
`data_test`	test set that includes the target (1. column) and the binned covariates as in the training set.

Value

Returns a list consisting of two components:

- First component includes default number, nondefault number and WOE values for each group calculated for the training set.
- Second component is a dataframe of WOE values for the new set (test set).

Figure 6.4. Screenshot from the R documentation of assignWOE() function.

In order to observe the slight change in the WOE values, with the R expressions in Figure 6.5, both components returned by the calcWOE() function are presented.

```
# First component: Summary table for groups of CCOL_VAR_3
calcWOE(trainSplit)[[1]]$GRP_CCOL_VAR_3

# Second component: WOE_CCOL_VAR_3
head(calcWOE(trainSplit)[[2]][,c("NPL_IN_12M","WOE_CCOL_VAR_3")],
10)
```

Figure 6.5. R expressions to calculate the WOE version of CCOL_VAR_3 for the training split and print response variable and WOE_CCOL_VAR_3 for the first ten observations.

In Table 6.5, the first component returned by the calcWOE() function displays nondefault number, default number and WOE values for each group of GRP_CCOL-_VAR_3. As one can see, after sampling and then recalculating WOE values, they changed to some extent in comparison to the values given in Table 6.4.

Table 6.5. First component returned by the calcWOE() function for GRP_CCOL_VAR_3.

	1	**2**	**3**	**4**
numNonDef	27241	9751	8035	12367
numDef	369	193	219	780
WOE	0.6971	0.3178	-0.0021	-0.8411

In Table 6.6, first ten observations of the second component returned by the calcWOE() function displays their WOE values.

Table 6.6. First ten observations of the second component returned by the calcWOE() function for GRP_CCOL_VAR_3 and respective NPL status

	NPL_IN_12M	WOE_CCOL_VAR_3
1	0	0.3178
2	0	-0.8411
3	0	-0.0021
4	0	-0.8411
5	0	0.6971
6	0	-0.0021
7	0	-0.0021
8	0	0.3178
9	0	0.6971
10	0	0.3178

6.3.2. Splitting the Data Sets of GRP and WOE Variables into Training and Test Data Sets

Training and test splits of data sets including GRP and WOE variables are obtained with the R expressions in Figure 6.6. These are used throughout the whole study.

```
# Load packages
library(caret)
library(moralar)

#Load masked GRP variables
load("Model_VAR_ALL_GRP_masked.RData")

data ← Var_all_GRP_masked[, -30]

# Data set is splitted into training and test sets.
set.seed(1234)
pct ← 0.7
splitIndex ← createDataPartition(data$NPL_IN_12M, p = pct, list = FALSE,
times = 1)
Var_all_GRP_train ← data[splitIndex, ]
Var_all_GRP_test ← data[-splitIndex, ]


#Load masked WOE variables
load("Model_VAR_ALL_WOE_masked.RData")

data ← Var_all_WOE_masked

set.seed(1234)
pct ← 0.7
splitIndex ← createDataPartition(data$NPL_IN_12M, p = pct, list = FALSE,
times = 1)
Var_all_WOE_train ← data[splitIndex, ]
Var_all_WOE_test ← data[-splitIndex, ]
```

Figure 6.6. Splitting the data sets of GRP and WOE variables into training and test sets.

For the same sample variable CCOL_VAR_3, it is beneficial to visualize GRP and WOE concepts for the training set. With the R expressions in Figure 6.7, we obtained Figure 6.8.

```
# Plot categories and WOE values
par(mfrow = c(1, 2))

plot(as.factor(Var_all_GRP_train$GRP_CCOL_VAR_3),
     as.factor(Var_all_GRP_train$NPL_IN_12M),
     ylab = "NPL_IN_12M", xlab = "Category",
     main = "GRP_CCOL_VAR_3 ~ NPL_IN_12M")

WOE_CCOL_VAR_3_summary <- calcWOE(Var_all_GRP_train)[[1]]
[["GRP_CCOL_VAR_3"]]

barplot_WOE_CCOL_VAR_3 <-
 barplot(WOE_CCOL_VAR_3_summary[3, ],
         col = "brown",
         main = "WOE_CCOL_VAR_3",
         xlab = "Category", ylab = "WOE")

text(x = barplot_WOE_CCOL_VAR_3, y = c(0.5, 0.2, 0,-0.6),
     label = WOE_CCOL_VAR_3_summary[3, ],
     adj = 0.5, cex = 1, col = "green")

dev.off()
```

Figure 6.7. Visualizing the GRP and WOE concepts for CCOL_VAR_3.

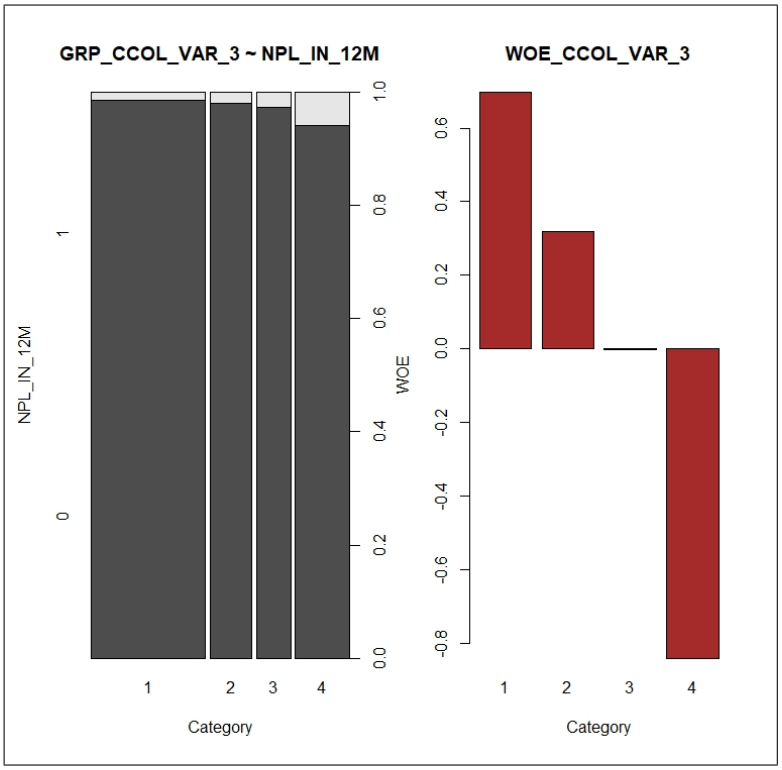


Figure 6.8. Categories of GRP_CCOL_VAR_3 with their respective NPL ratios and WOE values.

6.3.3. List of Covariates Used in the Scorecard Experiments

The covariates are listed in Table 6.7 with their descriptions. In the remaining chapters of the document, we will utilize the following variable names, and just for necessary cases, we will reshare their description.

Table 6.7. Covariates in the data set and their description.

Variable Name	**Description**
ALL_VAR_1	Number of distinct banks applied for any type of loan.
ALL_VAR_2	Number of applications for any type of loan in the last 365 days.
ALL_VAR_3	Number of good-closed loans of any type without a single delinquency in the last 90 days.
ALL_VAR_4	Ratio of the sum of the arrears balance to the sum of the total outstanding balance of open loans of any type.
ALL_VAR_5	Total arrears balance of open loans of any type.
ALL_WOHL_VAR_1	Number of applications for any type of loan excluding house loans that were not accepted or originated in the last 180 days.
CC_VAR_1	Number of credit card applications that were not accepted in the last 180 days.
CC_VAR_2	Worst payment status in credit cards.
CC_VAR_3	Total sum of limits of good-closed credit cards with at most one delinquent payment in a row.
CC_VAR_4	Age of the oldest credit card.
CC_VAR_5	Number of open credit cards whose limit usage is greater than 100%.
CC_VAR_6	Ratio of the payment in the last 3 months to its limit for open credit cards.

Table 6.7. Covariates in the data set and their description (cont.)

CCOL_VAR_1	Number of distinct banks that granted credit cards or overdraft loans whose limit usage is greater than 80%.
CCOL_VAR_2	Number of open, active credit cards and overdraft loans used before at least 182 days and having a worst payment status of 0 (i.e. without any delinquency) in the last 6 months.
CCOL_VAR_3	Total limit utility of open credit cards and overdraft loans.
CHL_VAR_1	Age of the oldest loan among cash loans and house loans granted.
CHL_VAR_2	Age of the oldest loan among open cash loans or house loans granted with at most one delinquent payment in a row.
CHL_VAR_3	Maximum number of months without delinquency in the last six months amongst open cash loans and house loans.
CHL_VAR_4	Number of open cash loans and house loans that are granted in the last 91 to 365 days and currently not delinquent.
CHL_VAR_5	Ratio of the sum of the arrears balance to the sum of the total outstanding balance of open cash loans and house loans.
CL_VAR_1	Number of distinct banks applied for cash loans.
CL_VAR_2	Worst payment status in cash loans.
CL_VAR_3	Total sum of usage amounts of cash loans that closed good in the last 365 days with at most one delinquent payment in a row.
CL_VAR_4	Total sum of usage amounts of open cash loans granted in the last 365 days.
CL_VAR_5	Ratio of total installment amount of open cash loans to their total outstanding balance.

Table 6.7. Covariates in the data set and their description (cont.)

OL_VAR_1	Worst payment status in overdraft loans.
OL_VAR_2	Age of the oldest overdraft loan excluding the bad-closed ones.
OL_VAR_3	Number of open overdraft loans whose limit usage is greater than 100%.

Limit usage term in the variable descriptions denotes the ratio of total outstanding balance to the limit. In Table 6.8, it is displayed how variables are binned and which values are included in each bin.

Table 6.8. Intervals of grouped variables and their respective group values.

Variable Name	**Value**	**Group**
ALL_VAR_1	<3	1
	[3 , 4)	2
	>= 4	3
ALL_VAR_2	<1	1
	[1 , 2)	2
	[2 , 4)	3
	>= 4	4
ALL_VAR_3	<1	1
	>= 1	2
ALL_VAR_4	<0.03	1
	>= 0.03 or NA	2
ALL_VAR_5	<611	1
	>= 611	2

Table 6.8. Intervals of grouped variables and their respective group values (cont.).

Variable Name	**Value**	**Group**
ALL_WOHL_VAR_1	<2	1
	[2 , 3)	2
	[3 , 5)	3
	>= 5	4
	NA	5
CC_VAR_1	<1	1
	[1 , 2)	2
	>= 2	3
CC_VAR_2	<1	1
	[1 , 2)	2
	[2 , 3)	3
	>= 3 or NA	4
CC_VAR_3	<1500	1
	[1500 , 4000)	2
	>= 4000	3
	NA	4
CC_VAR_4	<1596 or NA	1
	[1596 , 2783)	2
	[2783 , 3705)	3
	[3705 , 4976)	4
	>= 4976	5
CC_VAR_5	<1	1
	>= 1	2

Table 6.8. Intervals of grouped variables and their respective group values (cont.).

Variable Name	**Value**	**Group**
CC_VAR_6	<0.06	1
	[0.06 , 0.11)	2
	[0.11 , 0.17)	3
	[0.17 , 0.27)	4
	>= 0.27	5
	NA	6
CCOL_VAR_1	<1	1
	[1 , 3)	2
	>= 3	3
CCOL_VAR_2	<1	1
	[1 , 2)	2
	>= 2	3
CCOL_VAR_3	<0.38	1
	[0.38 , 0.56)	2
	[0.56 , 0.74)	3
	>= 0.74 or NA	4
CHL_VAR_1	<960	1
	[960 , 1455)	2
	[1455 , 1895)	3
	>= 1895	4
	NA	5
CHL_VAR_2	<316	1
	[316 , 514)	2
	[514 , 771)	3
	>= 771	4
	NA	5

Table 6.8. Intervals of grouped variables and their respective group values (cont.).

Variable Name	**Value**	**Group**
CHL_VAR_3	<6	1
	>= 6	2
	NA	3
CHL_VAR_4	<1	1
	>= 1	2
CHL_VAR_5	<0.01	1
	>= 0.01	2
	NA	3
CL_VAR_1	<2	1
	>= 2	2
CL_VAR_2	<1	1
	[1 , 2)	2
	>= 2	3
	NA	4
CL_VAR_3	<16411	1
	>= 16411	2
	NA	3
CL_VAR_4	<7183	1
	[7183 , 21500)	2
	>= 21500	3
	NA	4
CL_VAR_5	<0.04	1
	[0.04 , 0.06)	2
	>= 0.06	3
	NA	4
OL_VAR_1	<1	1
	[1 , 2)	2
	>= 2 or NA	3

Table 6.8. Intervals of grouped variables and their respective group values (cont.).

Variable Name	**Value**	**Group**
OL_VAR_2	<700 or NA	1
	[700 , 1347)	2
	[1347 , 1956)	3
	[1956 , 2675)	4
	>= 2675	5
OL_VAR_3	<1	1
	>= 1	2

7. STUDIES ON RAW (UNBINNED) VARIABLES

In this chapter, we will focus on raw, original, unbinned variables and first examine the extent of the information loss, when they are grouped into a few categories. Then, we will perform outlier elimination techniques widely used by practitioners. Finally, we will share some R codes and figures which would be beneficial, in case raw variables are used to fit the model.

In this chapter, as in the remaining chapters of the thesis, we will make use of the data sets obtained in Chapter 6, specifically in Subsection 6.3.2.

7.1. Information Loss in Case of Binning Variables

In order to assess the loss of information after binning the variables, we determined eleven raw variables in our data set that do not include NA values. These variables are determined with the R code given in Figure 7.1 and they are listed in Table 7.1.

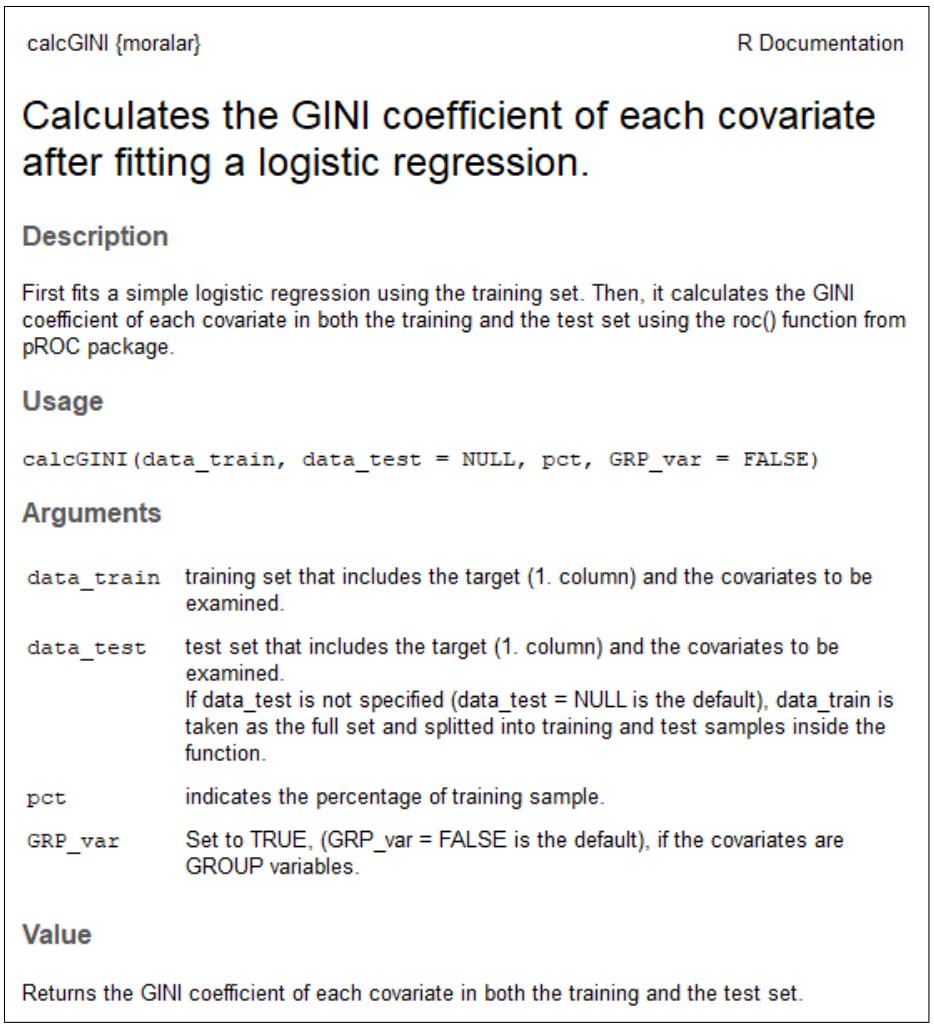
```
# Load all masked raw variables
load("Model_VAR_ALL_RAW_masked.RData")

# Determine the variables that include NA values in the data set
colnames(Var_all_raw_masked[sapply(Var_all_raw_masked,      function(x)
sum(is.na(x))) == 0])[-1]
```

Figure 7.1. R code to load the data set of raw variables and determine the variables which do not include NA values.

Table 7.1. The raw variables that do not include NA values.

Variable Name
ALL_VAR_1
ALL_VAR_2
ALL_VAR_3
ALL_VAR_5
CC_VAR_1
CC_VAR_5
CCOL_VAR_1
CCOL_VAR_2
CHL_VAR_4
CL_VAR_1
OL_VAR_3

Gini indices for these variables are calculated with calcGINI() function whose description and usage is given in Figure 7.2. This function first splits the data set into training and test sets according to the specified percentage with createDataPartition() function from caret package [18]. Then, a logistic regression is fitted using the training set. Using the fitted model, predictions for both training and test sets are determined and roc() function from pROC package [19] is employed to calculate the Gini indices.



calcGINI {moralar} R Documentation

Calculates the GINI coefficient of each covariate after fitting a logistic regression.

Description

First fits a simple logistic regression using the training set. Then, it calculates the GINI coefficient of each covariate in both the training and the test set using the roc() function from pROC package.

Usage

```
calcGINI(data_train, data_test = NULL, pct, GRP_var = FALSE)
```

Arguments

data_train	training set that includes the target (1. column) and the covariates to be examined.
data_test	test set that includes the target (1. column) and the covariates to be examined. If data_test is not specified (data_test = NULL is the default), data_train is taken as the full set and splitted into training and test samples inside the function.
pct	indicates the percentage of training sample.
GRP_var	Set to TRUE, (GRP_var = FALSE is the default), if the covariates are GROUP variables.

Value

Returns the GINI coefficient of each covariate in both the training and the test set.

Figure 7.2. Screenshot from the R documentation of calcGINI() function.

70% of the data set is selected for the training set, whereas the remaining 30% as the test set. Seed is specified as 1234, in order to obtain the same splits each time. In Figure 7.3, R code to calculate the Gini indices of the selected raw variables for training and test data sets are shared.

```
Var_all_wo_NA ← Var_all_raw_masked[ , colnames (Var_all_raw_-
masked[sapply(Var_all_raw_masked, function(x) sum(is.na(x))) == 0])]

# Calculate Gini indices for raw variables that do not include NA values
set.seed(1234)
pct ← 0.7
round(calcGINI(data_train = Var_all_wo_NA, data_test = NULL, pct,
GRP_var = FALSE) * 100, 2)
```

Figure 7.3. R code to calculate Gini indices of raw variables without NA values for training and test splits with calcGINI() function.

In order to examine whether the Gini indices change after the variables are standardized, expressions stated in Figure 7.4 are run.

```
# Calculate Gini indices for standardized raw variables that do not include NA
values
scaled.Var_all_wo_NA ← Var_all_wo_NA
scaled.Var_all_wo_NA[, -1] ← scale(Var_all_wo_NA[, -1]) #First column is
the target.

set.seed(1234)
pct ← 0.7
round(calcGINI(data_train = scaled.Var_all_wo_NA, data_test = NULL,
pct, GRP_var = FALSE) * 100, 2)
```

Figure 7.4. R code to calculate Gini indices of standardized raw variables without NA values for training and test splits with calcGINI() function.

In case of using the GRP variables, one has to set the parameter GRP_var of calcGINI() to TRUE. Otherwise, the logistic regression is fitted as if variables take integer values. Suppose, a variable was binned into three groups. Values in the first

bin take 1, values in the second bin take 2, and so on. If one runs a logistic regression before transforming the integer values to factors, this gives incorrect results. If GRP_var is set to TRUE, calcGINI() can correctly calculate the Gini indices for both training and test sets, using dummy variables. In Figure 7.5, R code to calculate the Gini indices of the selected variables in GRP versions are shown.

```
#Load masked GRP variables
load("Model_VAR_ALL_GRP_masked.RData")

Var_GRP_wo_NA ← Var_all_GRP_masked[, c("NPL_IN_12M",
paste("GRP_", colnames(Var_all_raw_masked [sapply(Var_all_raw_-
masked, function(x) sum(is.na(x))) == 0]), sep = "")[-1])]

set.seed(1234)
pct ← 0.7
round(calcGINI(data_train = Var_GRP_wo_NA, data_test = NULL, pct,
GRP_var = TRUE) * 100, 2)
```

Figure 7.5. R code to load the data set of GRP variables whose raw version do not include NA values and to calculate their Gini indices for training and test splits with calcGINI() function.

Finally, Gini indices of these variables in WOE versions are calculated with the R code in Figure 7.6.

```
#Load masked WOE variables
load("Model_VAR_ALL_WOE_masked.RData")

Var_WOE_wo_NA ← Var_all_WOE_masked[, c("NPL_IN_12M",
paste("WOE_", colnames(Var_all_raw_masked[sapply(Var_all_raw_-
masked, function(x) sum(is.na(x))) == 0]), sep = "")[-1])]

set.seed(1234)
pct ← 0.7
round(calcGINI(data_train = Var_WOE_wo_NA, data_test = NULL, pct,
GRP_var = FALSE) * 100, 2)
```

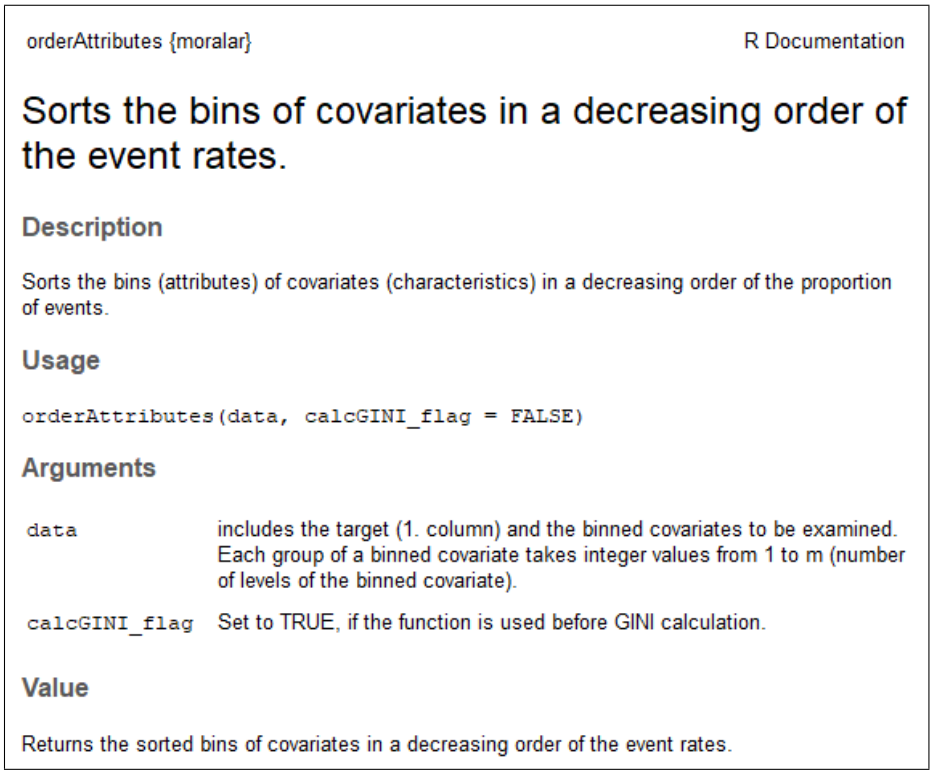
Figure 7.6. R code to load the data set of WOE variables whose raw version do not include NA values and to calculate their Gini indices for training and test splits with calcGINI() function.

Calculated Gini indices for different versions of the variables are tabulated in Figure 7.2. One can see, that just standardizing the variables does not change the explanatory power of the variables in terms of Gini. On the other hand, binning variables generally reduced the predictive power of the variable. Especially for ALL_VAR_5, a significant decrease in Gini indices is noted for both splits. However, in some cases binning variables improves the Gini index of the variable for both the training and test sets, e.g. ALL_VAR_1 and CHL_VAR_4. It can be concluded that using grouped variables instead of raw variables does not necessarily cause a loss of information. Besides, for the selected variables, Gini indices for GRP and WOE versions give similar results. Although, this is often the case, sometimes monotonicity is not preserved, in case of using GRP version, i.e. with increasing rate of defaults, predicted scores should decrease.

Table 7.2. Gini indices of variables in various versions whose raw version do not include NA values.

	Raw variable		**Standardized**		**GRP**		**WOE**	
	Training	**Test**	**Training**	**Test**	**Training**	**Test**	**Training**	**Test**
ALL_VAR_1	11.46	10.38	11.46	10.38	12.72	11.16	12.72	11.16
ALL_VAR_2	14.63	10.94	14.63	10.94	13.83	10.44	13.83	10.44
ALL_VAR_3	4.83	4.87	4.83	4.87	4.73	4.80	4.73	4.80
ALL_VAR_5	23.36	24.27	23.36	24.27	19.62	18.92	19.62	18.92
CC_VAR_1	22.44	17.93	22.44	17.93	22.01	17.58	22.01	17.58
CC_VAR_5	20.42	18.15	20.42	18.15	19.48	17.39	19.48	17.39
CCOL_VAR_1	29.53	25.59	29.53	25.59	27.84	24.51	27.84	24.51
CCOL_VAR_2	19.77	23.22	19.77	23.22	20.73	23.99	20.73	23.99
CHL_VAR_4	7.48	10.49	7.48	10.49	8.52	11.13	8.52	11.13
CL_VAR_1	8.51	6.51	8.51	6.51	7.61	6.80	7.61	6.80
OL_VAR_3	14.19	14.20	14.19	14.20	13.88	13.84	13.88	13.84

An alternative function to calculate Gini indices for GRP variables is calcGINI_var(). This calculates the Gini index without fitting a logistic regression, exclusively make use of default and nondefault numbers in a bin of a GRP variable. First, using the orderAttributes() function, the bins of a covariate are sorted in decreasing order of the proportion of events and bins are renamed accordingly, if calcGINI_flag is set to TRUE. That is, the bin with the highest default ratio takes the bin number 1, the bin with the lowest default ratio takes the bin number m, where the number of bins of the variable equals to m. A screenshot from the R documentation of orderAttributes() is presented in Figure 7.7, whereas Figure 7.8 depicts the usage of calcGINI_var() function.



orderAttributes {moralar} R Documentation

Sorts the bins of covariates in a decreasing order of the event rates.

Description

Sorts the bins (attributes) of covariates (characteristics) in a decreasing order of the proportion of events.

Usage

```
orderAttributes(data, calcGINI_flag = FALSE)
```

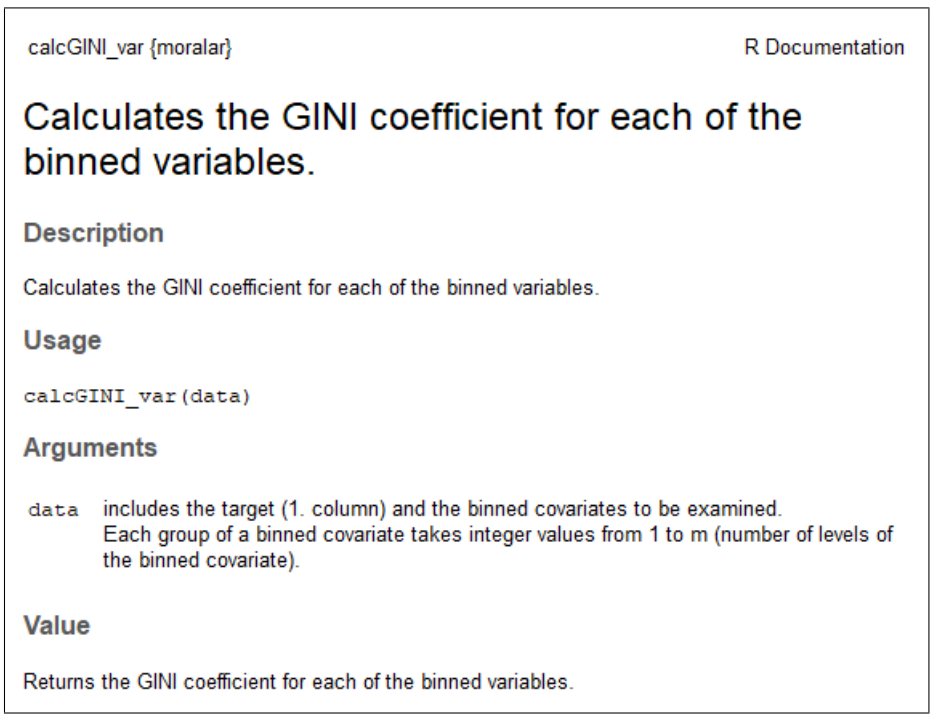
Arguments

`data`	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
`calcGINI_flag`	Set to TRUE, if the function is used before GINI calculation.

Value

Returns the sorted bins of covariates in a decreasing order of the event rates.

Figure 7.7. Screenshot from the R documentation of orderAttributes() function.



calcGINI_var {moralar} R Documentation

Calculates the GINI coefficient for each of the binned variables.

Description

Calculates the GINI coefficient for each of the binned variables.

Usage

```
calcGINI_var(data)
```

Arguments

`data`	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).

Value

Returns the GINI coefficient for each of the binned variables.

Figure 7.8. Screenshot from the R documentation of calcGINI_var() function.

In Figure 7.9, first the data set of GRP variables is splitted into training and test sets and calcGINI_var() is applied separately for training and test sets.

```
# caret package of R is loaded for createDataPartition() function.
library(caret)

set.seed(1234)
pct ← 0.7

splitIndex ← createDataPartition(Var_GRP_wo_NA[ , 1], p = pct, list =
FALSE, times = 1)
trainSplit GRP_wo_NA ← Var_GRP_wo_NA[splitIndex, ]
testSplit GRP_wo_NA ← Var_GRP_wo_NA[-splitIndex, ]

round(calcGINI_var(trainSplit GRP_wo_NA), 2)
round(calcGINI_var(testSplit GRP_wo_NA), 2)
```

Figure 7.9. R code to split the data set of GRP variables whose raw version do not include NA values into training and test sets and to calculate their Gini indices for each split with calcGINI_var() function.

When the Gini indices of the GRP variables are compared in Table 7.2 and Table 7.3, it can be seen that Gini indices for the training split are the same for each variable. On the other hand, for ALL_VAR_1 and ALL_VAR_2, there is a difference in Gini indices for the test split.

Table 7.3. Gini indices of GRP variables whose raw version do not include NA values, calculated with calcGINI_var() function.

	GRP	
	Training	**Test**
ALL_VAR_1	12.72	12.25
ALL_VAR_2	13.83	10.51
ALL_VAR_3	4.73	4.80
ALL_VAR_5	19.62	18.92
CC_VAR_1	22.01	17.58
CC_VAR_5	19.48	17.39
CCOL_VAR_1	27.84	24.51
CCOL_VAR_2	20.73	23.99
CHL_VAR_4	8.52	11.13
CL_VAR_1	7.61	6.80
OL_VAR_3	13.88	13.84

7.1.1. Summary of the Findings on Categorizing Variables

In this section, firstly, raw variables that do not include NA values in the development dataset were determined. Then, Gini indices of these raw variables were calculated with/out standardizing. We concluded that standardizing variables does not change the explanatory power of the variables in terms of Gini. Afterwards, we calculated the Gini indices for their GRP versions. We underlined that dummy/design variables should be utilized to calculate correct Gini indices, in case categorized variables take integer values such as 1, 2 depicting their categories. We showed that using GRP variables instead of raw variables does not always cause a loss of information, and the difference often is not substantial in the event of a decrease for the variables in our dataset. Finally, we calculated Gini indices for WOE versions of the categorized variables and deduced that Gini indices for GRP and WOE versions often give similar results.

7.2. Outlier Treatment Techniques for Raw Variables

In case of using raw variables, outlier treatment is of great importance, although in Section 7.1, we did not perform any outlier elimination technique. Suppose, there is a single covariate (x) and the response variable (y). In practice, unusual values of the covariate, i.e. very high or very low values, are eliminated or restricted with a maximum or minimum value it can take according to the expert's view. This process is called as outlier treatment by many practitioners. Although this is often helpful, it demonstrates a confusion of the following statistical terms [20]:

(i) Outlier: If an observation unconditionally takes an unusual value for either its covariate or response variable, it is called a univariate outlier [21]. On the other hand, a regression outlier is an observation that has an unusual value of the dependent variable, conditional on its value of the independent variable. A regression outlier will not necessarily affect the regression slope coefficient. The Bonferroni-adjusted outlier test in car package [22] in R tests the largest absolute studentized residual [23].

(ii) Leverage: An observation with an unusual covariate value, i.e. it is far from the mean of x, can potentially influence the regression line [21]. Thus, it has leverage on the regression line. The degree of the leverage depends on how far is its value away from the mean of x, either in a positive or negative direction. High leverage does not necessarily mean that it influences the regression coefficients. An observation may have a high leverage, but follow a straight in line with the pattern of the rest of the data. Most common measure of leverage is the hat-value, h_i. The hat value h_i measures the potential leverage of y_i on all the fitted values [23].

(iii) Influential observations: A regression outlier that has high leverage influences the regression line, that is both the intercept and the slope are affected [21].

Most practitioners directly eliminate outliers in order to get rid of possibly influential observations. This can also be a source of information loss, because some

observations with unexpected values are taken out from the development set, although they are not influential. Here, we attempted to practice three approaches in order to eliminate observations that take unusual values from the variables we examined in previous section before fitting logistic regression and calculating their discriminatory powers in terms of Gini index. These approaches are:

(i) First, we identify the first quartile (Q1) and third quartile (Q3) of the values of a variable and then calculate the interquartile range (IQR = Q3 – Q1). It is common to eliminate observations that are below Q1 - 1.5 * IQR and above Q3 + 1.5 * IQR.

(ii) We specified the values of variables at their 99th, 99.5th and 99.9th percentiles and later assigned the larger values to the specified values, so that they are capped at these maximum values determined.

(iii) Cook's distance is widely used to identify the influential outlier variables. There are different opinions regarding what cut-off values to use for spotting highly influential points. In practice, many statisticians use the rule of thumb that Cook's distances bigger than the 10th percentile of an F distribution with p and n-p degrees of freedom represent potential problems, where n is the number of observations, and p is the number of parameters estimated. Since this value is close to 1 for large n, a simple operational guideline of $D\_i > 1$ has been suggested, i.e. if the Cook's distance of ith observation, D_i, is greater than 1, it is considered as an influential observation [23].

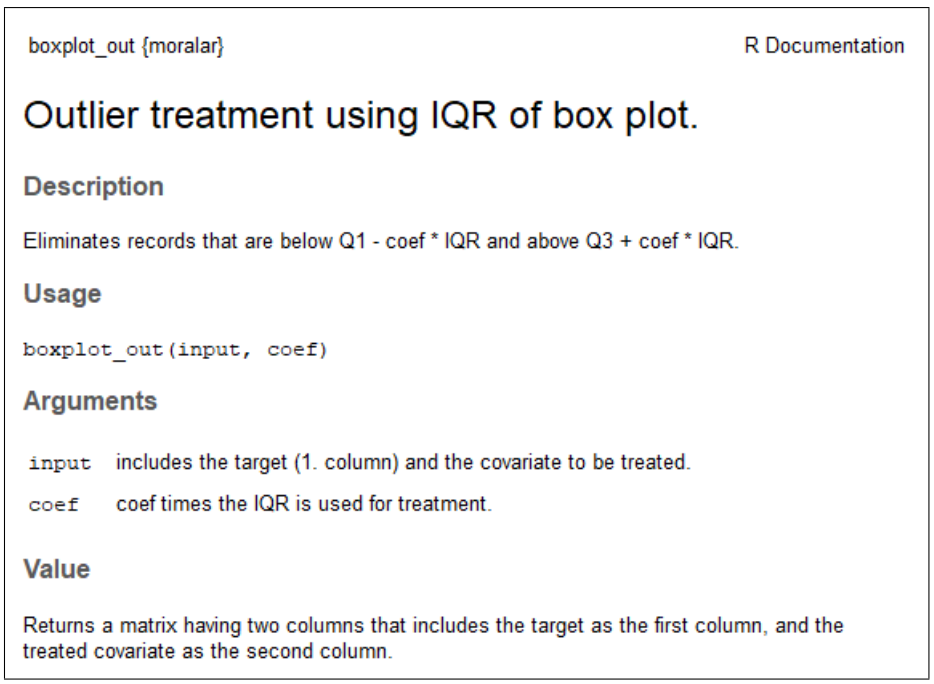
For further studies, we split the dataset of raw variables which do not include NA values into training and test sets as in Figure 7.10.

7.2.1. Outlier Detection and Elimination Using Interquartile Range

In order to eliminate observations that take values below and above a multiple of IQR, boxplot_out() function can be used whose screenshot from the R documentation is given in Figure 7.11.

```
set.seed(1234)
pct ← 0.7
splitIndex ← createDataPartition(Var_all_wo_NA$NPL_IN_12M, p = pct,
list = FALSE, times = 1)
trainSplit_wo_NA ← Var_all_wo_NA[splitIndex, ]
testSplit_wo_NA ← Var_all_wo_NA[-splitIndex, ]
```

Figure 7.10. R code to split the data set of raw (unbinned) variables which do not include NA values into training and test sets.



boxplot_out {moralar} R Documentation

Outlier treatment using IQR of box plot.

Description

Eliminates records that are below Q1 - coef * IQR and above Q3 + coef * IQR.

Usage

```
boxplot_out(input, coef)
```

Arguments

input	includes the target (1. column) and the covariate to be treated.
coef	coef times the IQR is used for treatment.

Value

Returns a matrix having two columns that includes the target as the first column, and the treated covariate as the second column.

Figure 7.11. Screenshot from the R documentation of boxplot_out() function.

We could apply this approach for only the variables whose IQR does not equal to zero. As an example, ALL_VAR_3, i.e. number of good-closed loans of any type without a single delinquency in the last 90 days, takes zero for most of the observations, so that both first and third quartiles are zero, and consequently IQR equals to zero. As given in Figure 7.12, boxplot_out() function returns NA which indicates this approach cannot be utilized for this variable.

```
> print(colnames(trainSplit)[4])
[1] "ALL_VAR_3"

> print(boxplot_out(trainSplit[, c(1, 4)], coef = 3))
[1] NA
```

Figure 7.12. boxplot_out() returns NA for ALL_VAR_3, because IQR equals to zero.

For the variables whose IQR is different than zero, boxplot_out() is applied and then Gini indices are calculated for both training and test splits with the expression in Figure 7.13.

```
for(i in c(2, 3, 6, 8, 9, 10, 11) {
  data ← boxplot_out(trainSplit[, c(1, i)], coef = 1.5)
  print(round(calcGINI(data_train = data, data_test = testSplit[, c(1, i)]) *
100, 2))
}
```

Figure 7.13. Usage of boxplot_out() function.

In Table 7.4, Gini indices of raw variables are tabulated for training and test splits, after observations that are below and above 1.5 times IQR are excluded from the data set with boxplot_out() function, i.e. coef is specified as 1.5. It can be seen that the Gini indices for the test split after applying boxplot_out() function are the same as the Gini indices without any outlier treatment in Table 7.2. On the other hand, Gini indices differ for the training set. For some variables like ALL_VAR_1, there is an increase and for some other variables like CCOL_VAR_1, there is a decrease. Especially for CCOL_VAR_1, the decrease from 29.53 to 20.17 is enormous.

Table 7.4. Gini indices of raw variables after observations that are below Q1 - 1.5 * IQR and above Q3 + 1.5 * IQR are eliminated with boxplot_out() function.

	>= Q1-1.5*IQR and <= Q3+1.5*IQR	
	Training	**Test**
ALL_VAR_1	13.85	10.38
ALL_VAR_2	12.70	10.94
ALL_VAR_3	Not applicable	Not applicable
ALL_VAR_5	Not applicable	Not applicable
CC_VAR_1	16.14	17.93
CC_VAR_5	Not applicable	Not applicable
CCOL_VAR_1	20.17	25.59
CCOL_VAR_2	22.55	23.22
CHL_VAR_4	9.61	10.49
CL_VAR_1	7.97	6.51
OL_VAR_3	Not applicable	Not applicable

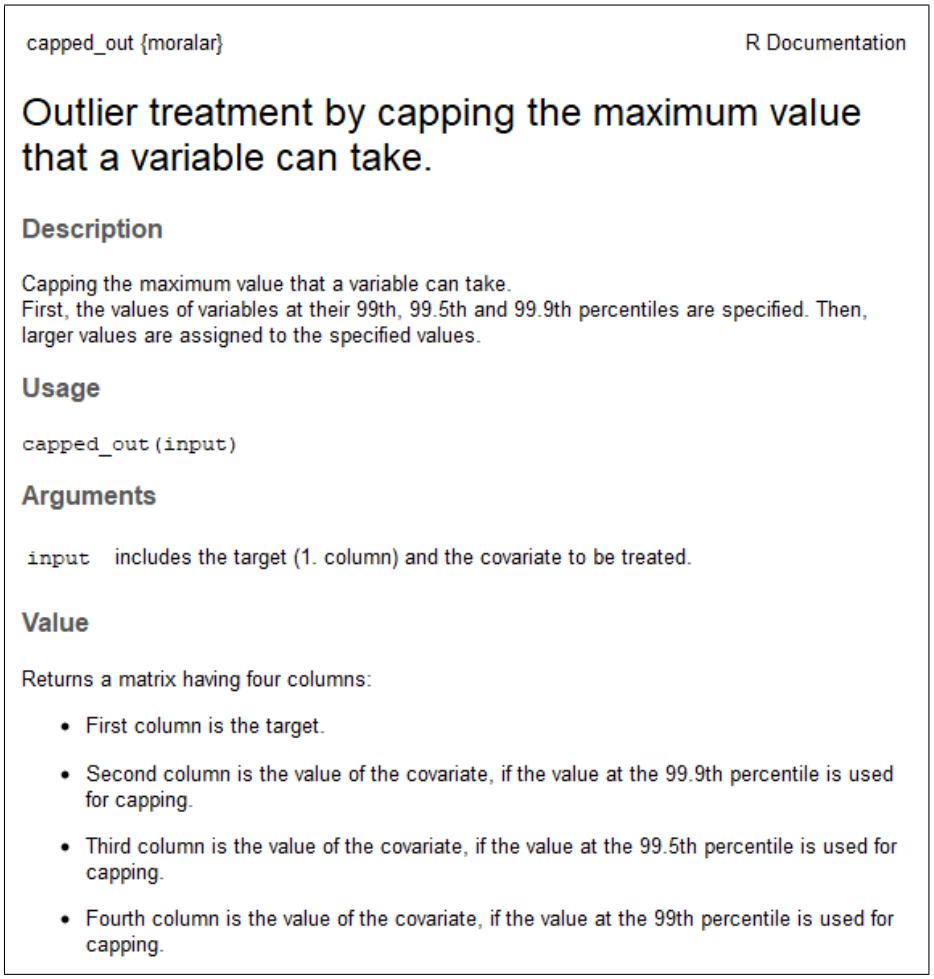
In Table 7.5, Gini indices of raw variables are tabulated for training and test splits, after observations that are below and above three times IQR are excluded from the data set with boxplot_out() function, i.e. coef is specified as 3. Again, the Gini indices for the test split do not change after applying boxplot_out(). On the other hand, Gini indices differ for the training set, but the extent of change is not as high as in the previous example, i.e. where coef is specified as 1.5. Hence, it can be concluded that applying this approach may cause significant information loss, especially when the upper and lower boundaries are closer to each other.

Table 7.5. Gini indices of raw variables after observations that are below Q1 - 3 * IQR and above Q3 + 3 * IQR are eliminated with boxplot_out() function

	>= Q1-3*IQR and <= Q3+3*IQR	
	Training	**Test**
ALL_VAR_1	11.50	10.38
ALL_VAR_2	14.58	10.94
ALL_VAR_3	Not applicable	Not applicable
ALL_VAR_5	Not applicable	Not applicable
CC_VAR_1	20.08	17.93
CC_VAR_5	Not applicable	Not applicable
CCOL_VAR_1	25.27	25.59
CCOL_VAR_2	20.13	23.22
CHL_VAR_4	8.20	10.49
CL_VAR_1	8.45	6.51
OL_VAR_3	Not applicable	Not applicable

7.2.2. Outlier Detection and Elimination Capping the Maximum Value

In order to cap the maximum value that a variable can take, capped_out() function can be used whose screenshot from the R documentation is given in Figure 7.14. With this function the values of variables at their 99th, 99.5th and 99.9th percentiles are specified and the larger values later assigned to the specified values.



capped_out {moralar} R Documentation

Outlier treatment by capping the maximum value that a variable can take.

Description

Capping the maximum value that a variable can take.
First, the values of variables at their 99th, 99.5th and 99.9th percentiles are specified. Then, larger values are assigned to the specified values.

Usage

```
capped_out(input)
```

Arguments

input	includes the target (1. column) and the covariate to be treated.

Value

Returns a matrix having four columns:

- First column is the target.
- Second column is the value of the covariate, if the value at the 99.9th percentile is used for capping.
- Third column is the value of the covariate, if the value at the 99.5th percentile is used for capping.
- Fourth column is the value of the covariate, if the value at the 99th percentile is used for capping.

Figure 7.14. Screenshot from the R documentation of capped_out() function.

The usage of capped_out() function is given in Figure 7.15. As stated in the documentation of the function, if a variable in the training data set is capped with its value at the 99.9th percentile, then the second column gives its treated version. Then, for each alternative percentile, Gini indices are calculated for both training and test splits which are displayed in Table 7.6. Gini indices for the test split does not change after applying capped_out(). On the other hand, Gini indices hardly differ for the training set. Especially for higher percentiles, change is very limited. Accordingly, it can be deduced that applying this approach does not cause great information loss, particularly for the higher percentile values.

```
for(i in 2:length(trainSplit) {
  set.seed(1234)
  pct ← 0.7
  #data ← capped_out(trainSplit[, c(1, i)])[, c(1, 2)] #capped_999
  #data ← capped_out(trainSplit[, c(1, i)])[, c(1, 3)] #capped_995
  data ← capped_out(trainSplit[, c(1, i)])[, c(1, 4)] #capped_990
  colnames(data) ← c(colnames(testSplit[, c(1, i)]))
  print(round(calcGINI(data_train = data, data_test = testSplit[, c(1, i)]) *
100, 2))
}
```

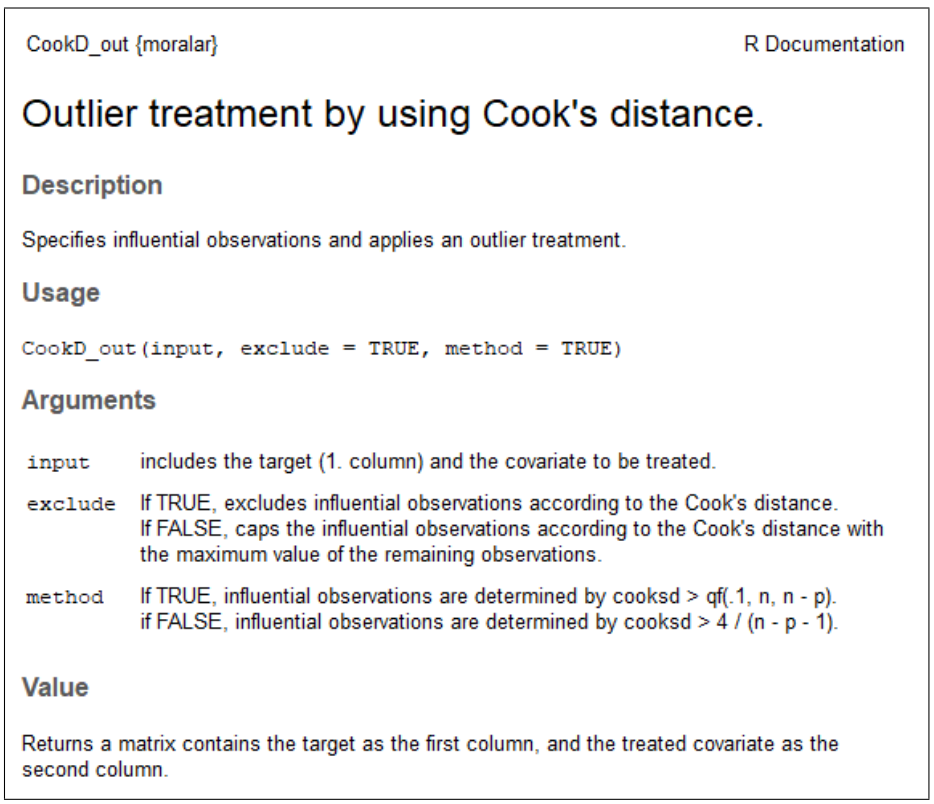
Figure 7.15. Usage of capped_out() function.

Table 7.6. Gini indices of raw variables after observations that are capped at different percentiles with capped_out() function.

	Cap at 99.9% Percentile		**Cap at 99.5% Percentile**		**Cap at 99% Percentile**	
	Training	**Test**	**Training**	**Test**	**Training**	**Test**
ALL_VAR_1	11.46	10.38	11.46	10.38	11.47	10.38
ALL_VAR_2	14.63	10.94	14.63	10.94	14.63	10.94
ALL_VAR_3	4.83	4.87	4.83	4.87	4.83	4.87
ALL_VAR_5	23.36	24.27	23.36	24.27	23.36	24.27
CC_VAR_1	22.44	17.93	22.44	17.93	22.43	17.93
CC_VAR_5	20.42	18.15	20.42	18.15	20.37	18.15
CCOL_VAR_1	29.53	25.59	29.52	25.59	29.50	25.59
CCOL_VAR_2	19.77	23.22	19.77	23.22	19.77	23.22
CHL_VAR_4	7.48	10.49	7.48	10.49	7.48	10.49
CL_VAR_1	8.51	6.51	8.51	6.51	8.48	6.51
OL_VAR_3	14.19	14.20	14.19	14.20	14.19	14.20

7.2.3. Outlier Detection and Elimination Using Cook's Distance

In order to eliminate influential observations, CookD_out() function can be used whose screenshot from the R documentation is given in Figure 7.16. One can pick the threshold to be used for the treatment with "method" parameter, whereas "exclude" parameter is used to decide whether detected influential observations with the selected method should be directly excluded from the data set, or capped with the remaining highest value.



CookD_out {moralar} R Documentation

Outlier treatment by using Cook's distance.

Description

Specifies influential observations and applies an outlier treatment.

Usage

```
CookD_out(input, exclude = TRUE, method = TRUE)
```

Arguments

input	includes the target (1. column) and the covariate to be treated.
exclude	If TRUE, excludes influential observations according to the Cook's distance. If FALSE, caps the influential observations according to the Cook's distance with the maximum value of the remaining observations.
method	If TRUE, influential observations are determined by cooksd > qf(.1, n, n - p). if FALSE, influential observations are determined by cooksd > 4 / (n - p - 1).

Value

Returns a matrix contains the target as the first column, and the treated covariate as the second column.

Figure 7.16. Screenshot from the R documentation of CookD_out() function.

As in Figure 7.17, one can specify the parameters and apply the CookD_out() function to a covariate and then Gini indices are calculated for both training and test splits. If the threshold is selected as 1 as usual, the Gini indices are calculated as in Table 7.7. Just for ALL_VAR_5, a single observation happens to be influential for this threshold, and whether excluding or capping this observation, does not change Gini indices for neither training nor test splits.

```
for(i in 2:length(trainSplit) {
  data ← CookD_out(trainSplit[, c(1, i)], exclude = FALSE, method = FALSE)
  print(round(calcGINI(data_train = data, data_test = testSplit[, c(1, i)]) *
100, 2))
}
```

Figure 7.17. Usage of CookD_out() function.

Table 7.7. Gini indices of raw variables after observations that are treated with CookD_out() function, where threshold is specified as 1.

	If Cook's distance >1			
	Influential records eliminated		**Influential records capped by the remaining maximum value**	
	Training	**Test**	**Training**	**Test**
ALL_VAR_1	11.46	10.38	11.46	10.38
ALL_VAR_2	14.63	10.94	14.63	10.94
ALL_VAR_3	4.83	4.87	4.83	4.87
ALL_VAR_5	23.36	24.27	23.36	24.27
CC_VAR_1	22.44	17.93	22.44	17.93
CC_VAR_5	20.42	18.15	20.42	18.15
CCOL_VAR_1	29.53	25.59	29.53	25.59
CCOL_VAR_2	19.77	23.22	19.77	23.22
CHL_VAR_4	7.48	10.49	7.48	10.49
CL_VAR_1	8.51	6.51	8.51	6.51
OL_VAR_3	14.19	14.20	14.19	14.20

If another threshold is selected, 4 / (n - p - 1) where n is the number of observations, and p is the number of parameters estimated, then the number of influential observations detected increases. Hence, Gini indices change slightly for the training split.

Table 7.8. Gini indices of raw variables after observations that are treated with CookD_out() function, where threshold is specified as 4 / (n - p - 1).

	Cook's distance >4 / (n - p - 1)			
	Influential records eliminated		**Influential records capped by the remaining maximum value**	
	Training	**Test**	**Training**	**Test**
ALL_VAR_1	11.45	10.38	11.46	10.38
ALL_VAR_2	15.02	10.94	14.63	10.94
ALL_VAR_3	4.69	4.87	4.83	4.87
ALL_VAR_5	23.04	24.27	23.36	24.27
CC_VAR_1	22.87	17.93	22.44	17.93
CC_VAR_5	20.31	18.15	20.42	18.15
CCOL_VAR_1	29.39	25.59	29.53	25.59
CCOL_VAR_2	19.75	23.22	19.77	23.22
CHL_VAR_4	7.25	10.49	7.48	10.49
CL_VAR_1	8.71	6.51	8.51	6.51
OL_VAR_3	14.19	14.20	14.19	14.20

7.2.4. Summary and Comparison of the Outlier Elimination Techniques

In this section, we mentioned that it is necessary to perform outlier detection and elimination techniques, in case raw, unbinned variables are used. We first defined related statistical terms: outlier, leverage and influential observations, and pointed out that an outlier is not necessarily an influential observation. Then, we introduced three outlier elimination techniques widely used by practitioners: using interquartile range, capping the extreme values and using Cook's distance. We concluded that using interquartile range to detect outliers may cause significant information loss, especially when the upper and lower boundaries are closer to each other. We deduced that capping the extreme values does not cause great information loss, particularly for the higher percentile values. Cook's distance is used directly to determine influential observations; but the threshold selection plays an important role. Through specifying a sensible threshold, this approach is superior to the previous two approaches which focused on eliminating outliers ignoring whether they are influential observations or not.

7.3. Univariate Analyses on Raw Variables and Investigating Nonlinearity

In Section 7.2, we examined some frequently used approaches used by the practitioners for outlier treatment. There exist various alternatives, and each alternative gives different results. One can choose one of them, if he/she knows the data well. On the other hand, there are some graphical tools which are beneficial, when working with raw variables.

In this section, two raw covariates out of the variables that do not include NA values will be examined thoroughly: ALL_VAR_5 as an example of continuous numerical variables and CCOL_VAR_1 as an example of discrete numerical variables. We chose to carry out further analysis on these variables, because they are more suitable representatives for our experiments according to the values they take. One can easily apply the following analyses in this section to any other raw variable and interpret the outputs accordingly.

7.3.1. An Example of Continuous Numerical Variable: ALL_VAR_5

ALL_VAR_5 is a continuous numerical variable that indicates total arrears balance of open loans of any type. ALL_VAR_5 takes values between 0 and 265121 in the training set. It is often recommended to investigate histogram and boxplot of a covariate to understand the distribution of the values it takes.

In Figure 7.18, R code to create histogram of ALL_VAR_5 for nondefault and default observations separately is shared using the ggplot2 [24] and plyr [25] packages in R.

```
library(ggplot2)
library(plyr)

classmeans ← ddply(trainSplit_wo_NA_wo_NA, "NPL_IN_12M", summarise, covariate.mean=mean(ALL_VAR_5))

ggplot(trainSplit_wo_NA_wo_NA, aes(x=ALL_VAR_5)) +
  geom_histogram(breaks=seq(0, 10000, by = 500), colour="black", fill="white") +
  facet_grid(NPL_IN_12M ~ ., scales="free") +
  geom_vline(data=classmeans, aes(xintercept=covariate.mean),
  linetype="dashed", size=1, colour="red")
```

Figure 7.18. R code to plot histogram of ALL_VAR_5 for nondefault and defaults observations separately.

In Figure 7.19, it can be seen that most of the observations pile up in the interval [0, 500). For nondefault observations, average amount is smaller than the average amount for default observations, as expected. Additionally, higher amounts of arrears balance are present for default observations, that is absolutely logical.

In Figure 7.20, R code to create boxplot of ALL_VAR_5 for nondefault and default observations separately is shared using the ggplot() function of ggplot2 package [24] in R.

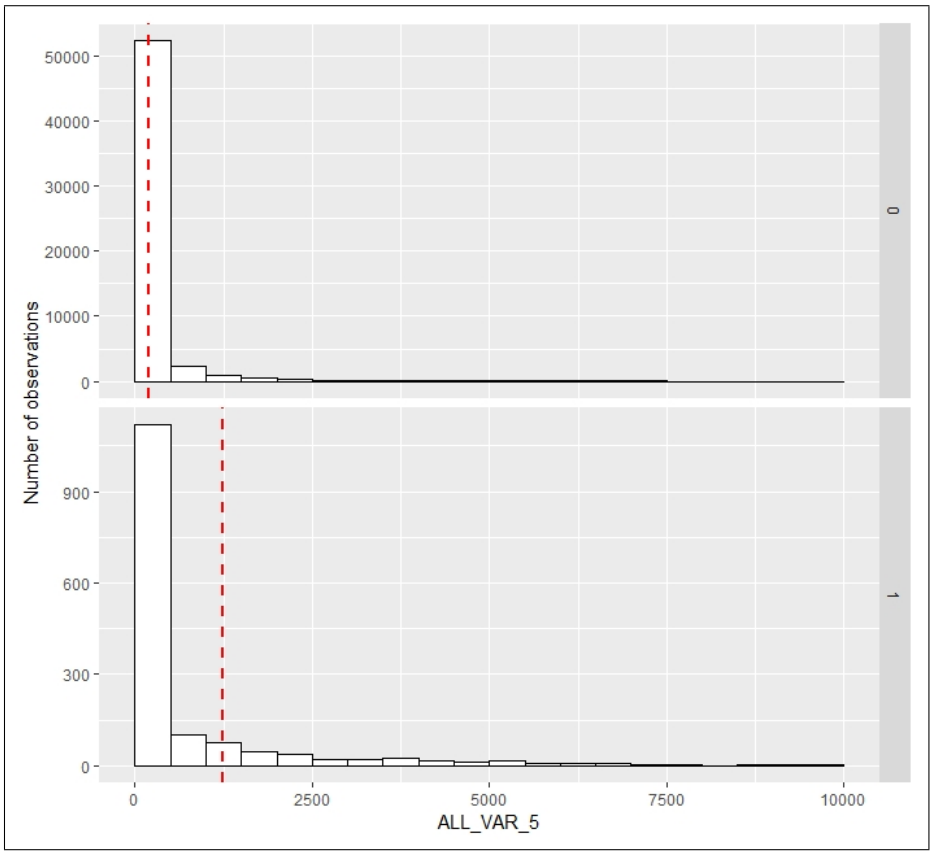


Figure 7.19. Histogram of ALL_VAR_5 for nondefault and default observations.

```
ggplot(trainSplit_wo_NA_wo_NA[trainSplit_wo_NA_wo_NA$ALL_-
VAR_5 <= 10000, ], aes(x = as.factor(NPL_IN_12M), y = ALL_VAR_5, fill
= as.factor(NPL_IN_12M))) +
  geom_boxplot(outlier.colour = "red", outlier.shape = 1) +
  stat_summary(fun.y = mean, geom = "point", shape = 5, size = 4) +
  guides(fill = FALSE) +
  ylab("ALL_VAR_5") +
  xlab("NPL_IN_12M")
```

Figure 7.20. R code to plot histogram of ALL_VAR_5 for nondefaults and defaults separately.

Boxplots of nondefault and default observations for ALL_VAR_5 are given in Figure 7.21. For nondefault observations, IQR is zero, as we determined also in the previous section. However, IQR is approximately 600 TL for default observations. Since IQR is zero for nondefault observations, it is not surprising that all values greater than zero are marked as outliers. Means are also shown with hollow diamonds.

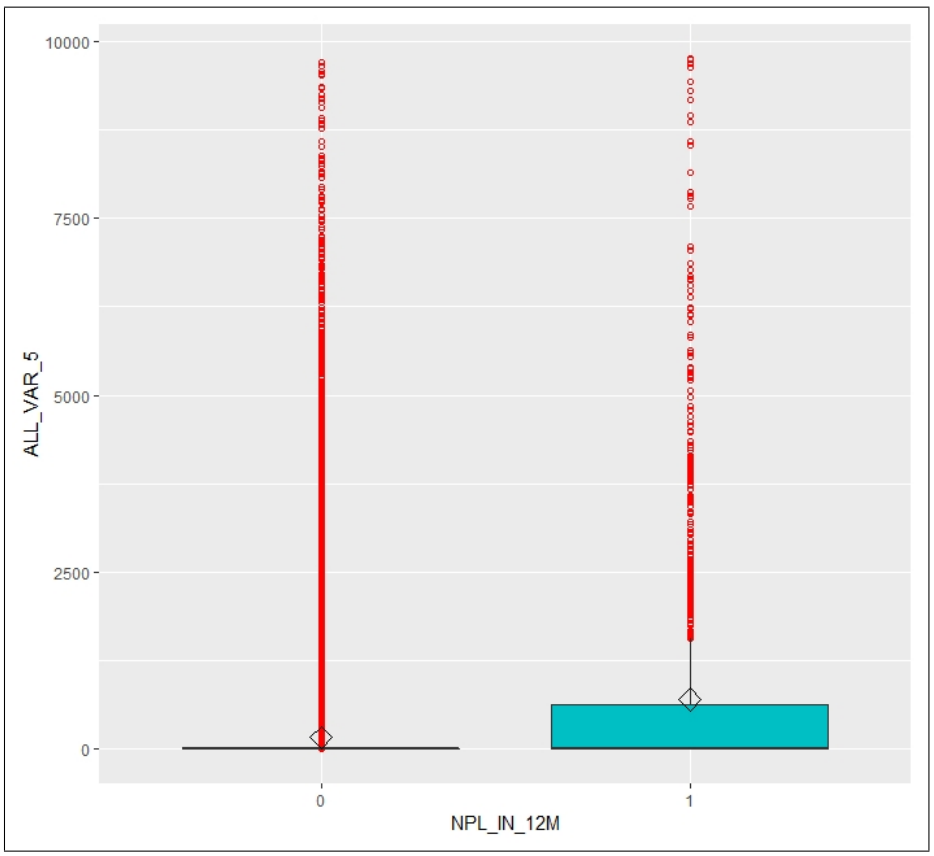


Figure 7.21. Boxplot of ALL_VAR_5 for nondefault and default observations.

After we fitted a logistic regression with the whole training data set, outlierTest() from car package [22] reveals that the observation that takes 265121 is an outlier according to the Bonferroni adjusted p-value. If we exclude this observation, slope of the covariate changes from 0.000166 to 0.000206. The magnitude of the slope is small, since we did not apply any transformation to the variable until now. Related R expressions and outputs are displayed in Figure 7.22.

```
> mod_fit_raw1 ← glm(NPL_IN_12M ~ ALL_VAR_5, data = trainSplit_wo_NA, family = binomial(link = "logit"))

> outlierTest(mod_fit_raw1)
```

	rstudent	unadjusted p-value	Bonferonni p
41592	-9.0703	1.19E-19	7.00E-15

```
> which(rownames(trainSplit_wo_NA) == "41592")
[1] 29149

> trainSplit_wo_NA[29149, "ALL_VAR_5"]
[1] 265121

> mod_fit_raw2 ← update(mod_fit_raw1, subset=-c(29149))

> outlierTest(mod_fit_raw2)
```

	rstudent	unadjusted p-value	Bonferonni p
32989	-4.5346	5.77E-06	0.3402

```
> compareCoefs(mod_fit_raw1, mod_fit_raw2)
Calls:
1: glm(formula = NPL_IN_12M ~ ALL_VAR_5, family = binomial(link = "logit"), data = trainSplit_wo_NA)
2: glm(formula = NPL_IN_12M ~ ALL_VAR_5, family = binomial(link = "logit"), data = trainSplit_wo_NA, subset = -c(29149))
```

	Model 1	Model 2
(Intercept)	-3.6674	-3.6864
SE	0.0265	0.0267
ALL_VAR_5	1.66E-04	2.06E-04
SE	1.21E-05	1.26E-05

Figure 7.22. Functions from car package in R are utilized for outlier detection, after fitting a logistic regression.

Since this variable represents monetary amount and can take very high values, it is meaningful to log transform it first for further analyses. However, many observations take value zero and log transform of zero is indefinite. Thus, we first add a very small number to each value, i.e. 0.1 which is a negligible amount of money and then the covariate is log tranformed. In order to detect nonlinearity, i.e. whether higher order terms of this variable are required, after the covariate is centered and scaled, a logistic regression is fitted with both first and second order terms. If we look at the summary of the fitted model, it suggests that the second order term should also be included into the model, because the p-value is very small and we reject the null hypothesis that the coefficient related to the second order term equals to zero. Is it really the case? We will investigate the distribution of values closely to draw a conclusion.

```
Call:
glm(formula = NPL_IN_12M ~ poly(center_scale(log(ALL_VAR_5 +
    0.1)), 2), family = binomial(link = "logit"), data = trainSplit_wo_NA[-c(29149),
    ])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4368  -0.2006  -0.2006  -0.2006   2.9965

Coefficients:
                                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                               -3.73624    0.02804 -133.25   <2e-16 ***
poly(center_scale(log(ALL_VAR_5 + 0.1)), 2)1 85.94037    5.40414   15.90   <2e-16 ***
poly(center_scale(log(ALL_VAR_5 + 0.1)), 2)2 59.14447    3.96728   14.91   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 14418  on 58953  degrees of freedom
Residual deviance: 13750  on 58951  degrees of freedom
AIC: 13756

Number of Fisher Scoring iterations: 6
```

Figure 7.23. A logistic regression model is fitted to detect whether a second order effect is significant.

The histogram in Figure 7.19, shows the distribution of nondefault and default observations for ALL_VAR_5. However, it is not adequate to detect the nonlinearity of the variable. For this purpose, one can make use of the R code in Figure 7.24, to split the values into bins, calculate event rate for each bin, and finally visualize the output. The following R code may be customized easily and very helpful to understand the distribution of defaults with respect to values of the covariate. One can also utilize

such a plot to split variables into bins and obtain GRP variables.

```
# Bins & percentage of NPL_IN_12M in each bin
x ← trainSplit_wo_NA_wo_NA$ALL_VAR_5 [trainSplit_wo_NA_wo_-
NA$ALL_VAR_5 > 0]

y ← table(cut(trainSplit_wo_NA_wo_NA$ALL_VAR_5, breaks = c(0, quan-
tile(x, probs = seq(0, 1, 0.1), na.rm = TRUE, names = TRUE)), include.lowest
= TRUE, right = FALSE, dig.lab = 10))

y0 ← table(cut(trainSplit_wo_NA_wo_NA$ALL_VAR_5 [trainSplit_wo_-
NA_wo_NA$NPL_IN_12M==0], breaks = c(0, quantile(x, probs = seq(0,
1, 0.1), na.rm = TRUE, names = TRUE)), include.lowest = TRUE, right =
FALSE, dig.lab = 10))

y1 ← table(cut(trainSplit_wo_NA_wo_NA$ALL_VAR_5[trainSplit_wo_-
NA_wo_NA$NPL_IN_12M==1], breaks =
c(0, quantile(x, probs = seq(0, 1, 0.1), na.rm = TRUE, names = TRUE)),
include.lowest = TRUE, right = FALSE, dig.lab = 10))

par(mar = c(5,5,2,5))
obj ← barplot(y, beside=TRUE, ann=FALSE)
par(new=TRUE)
plot(y1/y * 100,type="l",col="red",axes=FALSE,ann=FALSE)
axis(4,at=seq(0,20,1))
mtext(side = 1, line = 3, 'ALL_VAR_5 (binned)')
mtext(side = 2, line = 3, 'Number of records')
mtext(side = 3, line = 1, 'Bins of ALL_VAR_5 & percentage of NPL_IN_12M
in each bin')
mtext(side = 4, line = 3, 'Percent of NPL_IN_12M')
box()
```

Figure 7.24. A barplot is drawn to visually investigate whether a better binning would be possible for ALL_VAR_5.

R code in Figure 7.24 produce Figure 7.25. From this figure, we cannot conclude there is a nonlinearity for this variable. It is seen that for interval [417, 573), event rate is lower than for the neighboring bins. However, it does not have a special meaning for the data. Thus, the general assumption that the event rate increases with increasing amount of arrears balance still applies.

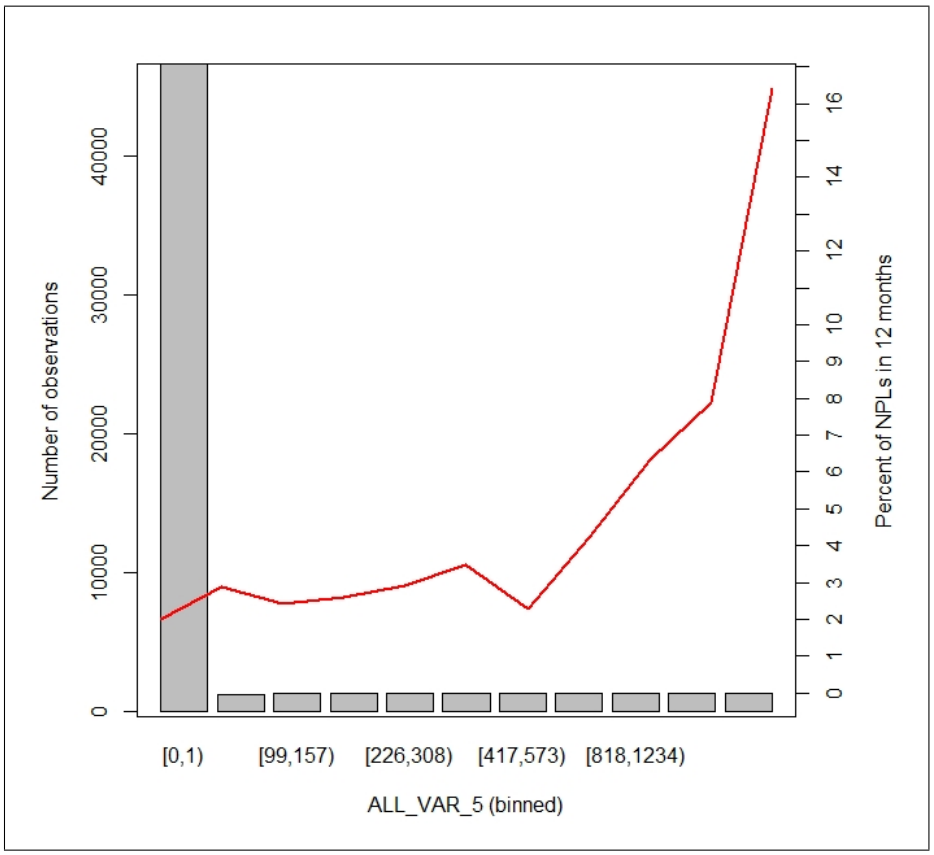


Figure 7.25. Bins of ALL_VAR_5 & percentage of NPL_IN_12M in each bin.

Finally diagnostic plots should be examined after fitting the model. Using the influenceIndexPlot() function of car package [22], one can obtain plots of Cook's distances, studentized residuals, Bonferroni adjusted p-values, and hat-values for each index [23]. Thus, one can graphically determine the possibly influential observations to exclude from the development data set. These plots are shared together in Figure 7.26. Figure 7.27 is an alternative to Figure 7.26. The size of the circle represents the Cook's distance. One can easily detect influential observations from the figure.

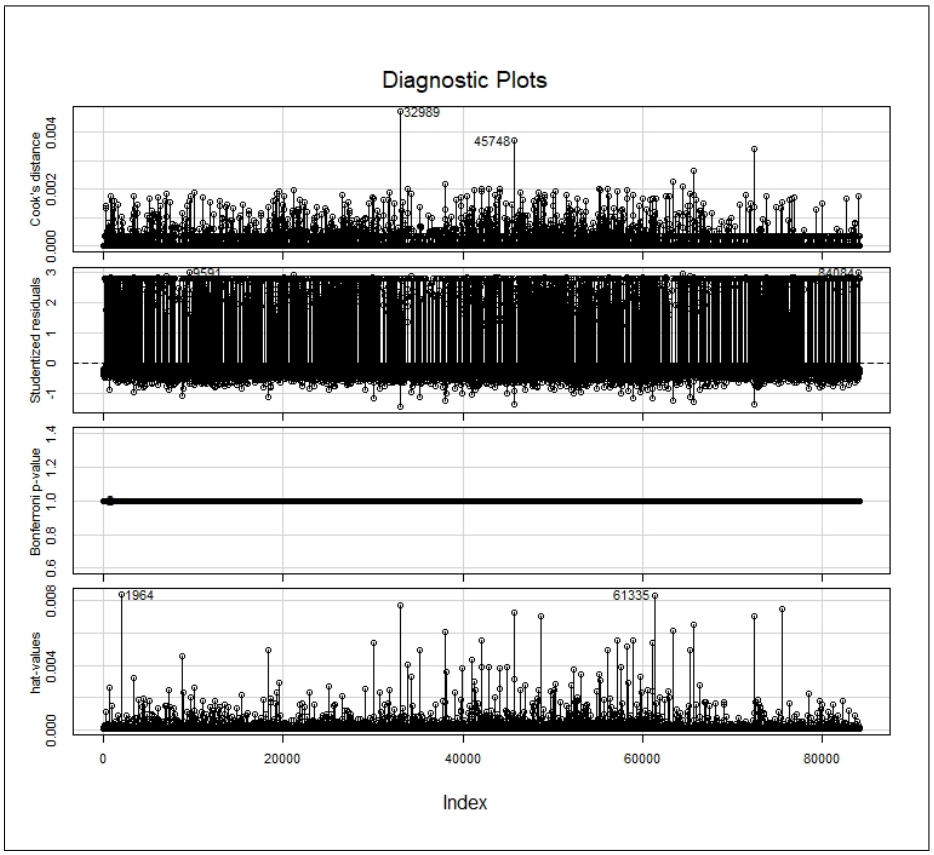


Figure 7.26. Output of influenceIndexPlot(mod_fit_raw3) to visually detect influential observations.

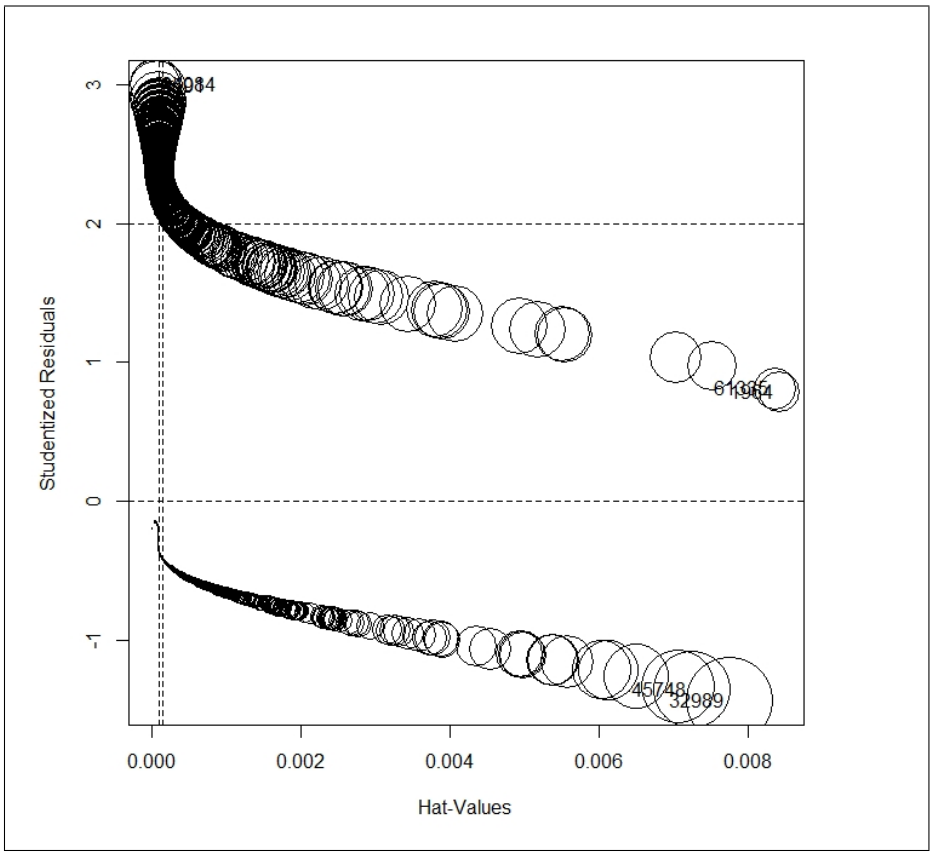


Figure 7.27. Output of influencePlot(mod_fit_raw3) to visually detect influential observations as an alternative to Figure 7.26.

7.3.2. An Example of Discrete Numerical Variable: CCOL_VAR_1

CCOL_VAR_1 is a discrete numerical variable which indicates the number of distinct banks that granted credit cards or overdraft loans whose limit usage is greater than 80%. CCOL_VAR_1 takes values between 0 and 13 in the training set.

Modifying the R code in Figure 7.18, histogram of CCOL_VAR_1 can be created for nondefault and default observations separately. From the figure, it can be seen that number of distinct banks that granted credit cards or overdraft loans whose limit usage is greater than 80% is more for defaulted customer, as expected. For nondefault observations, average number of banks is smaller than the average number of banks that that granted credit cards or overdraft loans whose limit usage is greater than 80% for default observation, as expected.

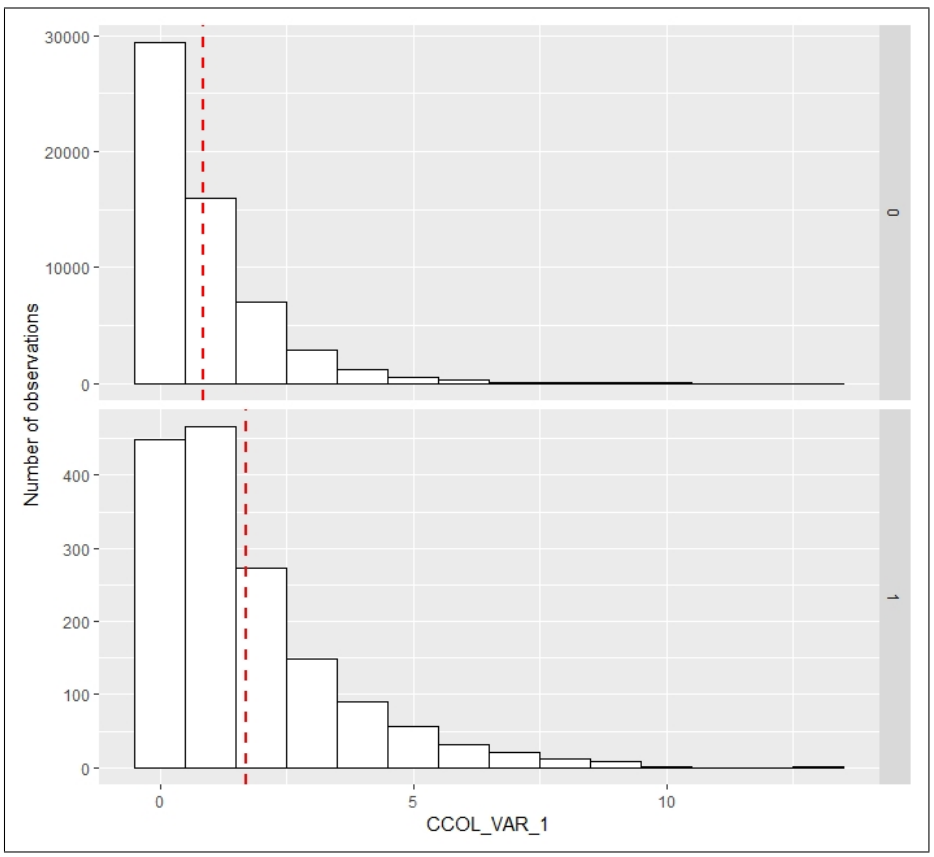


Figure 7.28. Histogram of CCOL_VAR_1 for nondefault and default observations.

Boxplots of nondefault and default observations for CCOL_VAR_1 are given in Figure 7.29. For nondefault observations, IQR is smaller, whereas it gets wider for default observations. Since IQR is small for both groups, it is not surprising that higher values are marked as outliers. Means are also shown with hollow diamonds.

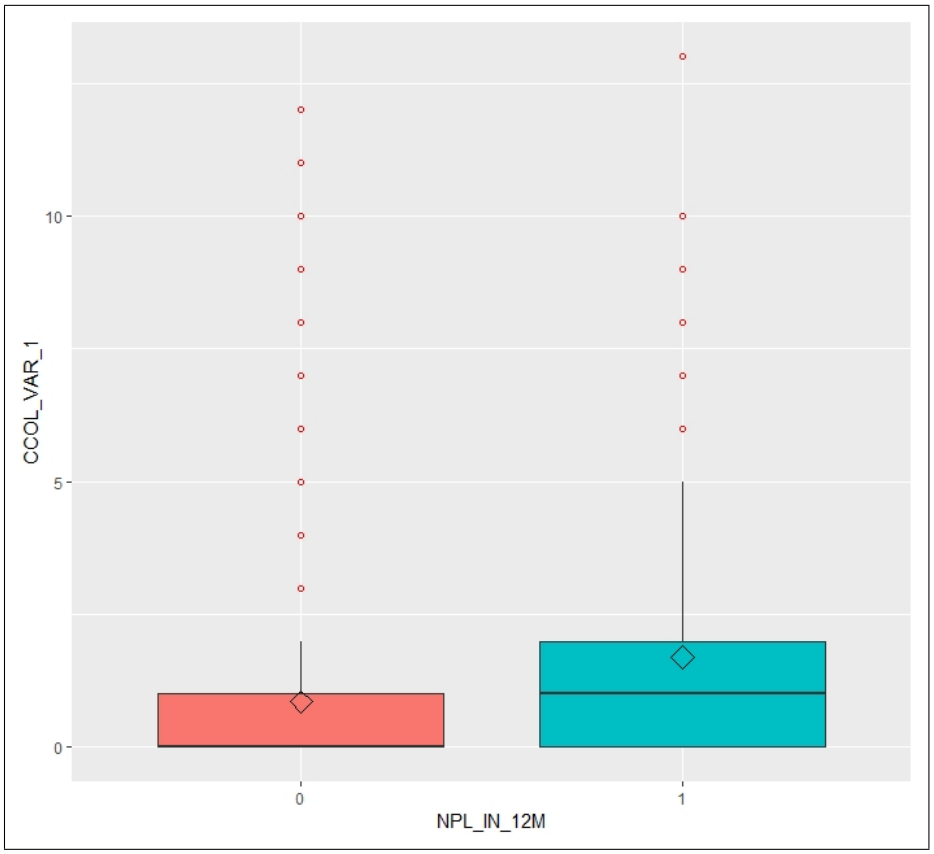


Figure 7.29. Boxplot of CCOL_VAR_1 for nondefault and default observations.

One can wonder why CCOL_VAR_1 that takes integer values is also categorized, although it does not take many different values. If Figure 7.30, which is obtained after modifying the R code in Figure 7.24, is examined, it can be seen that event rate makes zig zags in bigger values, i.e. after 7. However, it does not have a special meaning for the data. Thus, the general assumption that the event rate rises, if number of distinct banks that granted credit cards or overdraft loans whose limit usage is greater than 80% increases. However, one can also induce that values greater than 7 would also be grouped to secure the monotonicity.

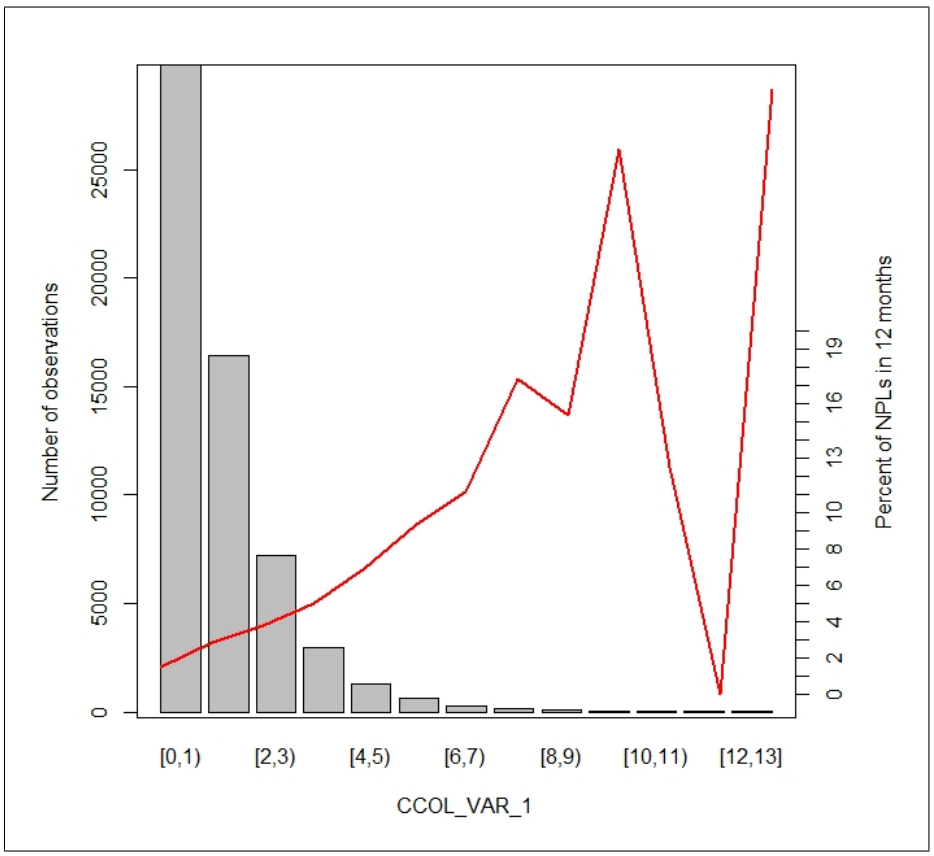


Figure 7.30. CCOL_VAR_1 & percentage of NPL_IN_12M at each value.

After a simple logistic regression is fitted with CCOL_VAR_1 as the predictor, outlierTest() function is utilized whether there are any outliers amongst the observations. Since the Bonferroni adjusted p-values for studentized residuals are greater than 0.05, we can conclude that all the observations can be included into the model, if CCOL_VAR_1 is used in its raw form.

It is beneficial to take a look at the plot of the probability of being classified as an NPL in 12 months versus CCOL_VAR_1 in its raw version in order to understand the influence of the covariate on the response variable. With the R code given in Figure 7.31, Figure 7.32 is obtained.

```
ggplot(trainSplit_wo_NA, aes(x = CCOL_VAR_1, y = NPL_IN_12M)) +
  geom_point() +
  stat_smooth(method = "glm", method.args = list(family = "binomial"), se
= TRUE)
```

Figure 7.31. R code to plot the probability of being classified as an NPL in 12 months versus CCOL_VAR_1 in raw version.

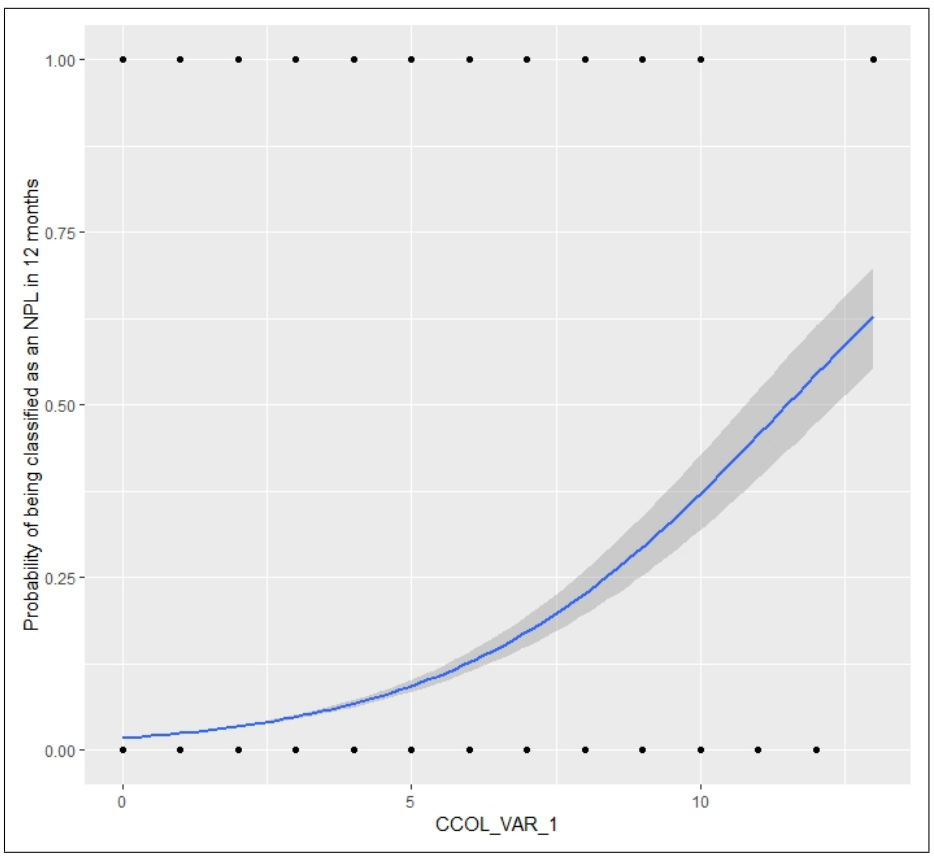


Figure 7.32. Probability of being classified as an NPL in 12 months versus CCOL_VAR_1.

If the variable is binned, the relationship between the covariate and the response variable is not as obvious as in Figure 7.32. Figure 7.32 is adjusted for categorized variables with the R code given in Figure 7.33 and subsequently Figure 7.34 is created.

```
# Plot of GRP variable
mod_fit_GRP ← glm(NPL_IN_12M ∼ GRP_CCOL_VAR_1, data = Var_-
all_GRP_train, family = binomial(link = "logit"))

xCCOL_VAR_1 ← as.factor(seq(1, 3, 1))
yCCOL_VAR_1 ← predict(mod_fit_GRP, list(GRP_CCOL_VAR_1 = xC-
COL_VAR_1), type="response")

plot(Var_all_GRP_train$GRP_CCOL_VAR_1,          Var_all_GRP_-
train$NPL_IN_12M, pch = 16, xlab = "GRP_CCOL_VAR_1", ylab =
"NPL_IN_12M"), lines(xCCOL_VAR_1, yCCOL_VAR_1, col = "red", lwd
= 2)
```

Figure 7.33. R code to plot the probability of being classified as an NPL in 12 months versus binned CCOL_VAR_1.

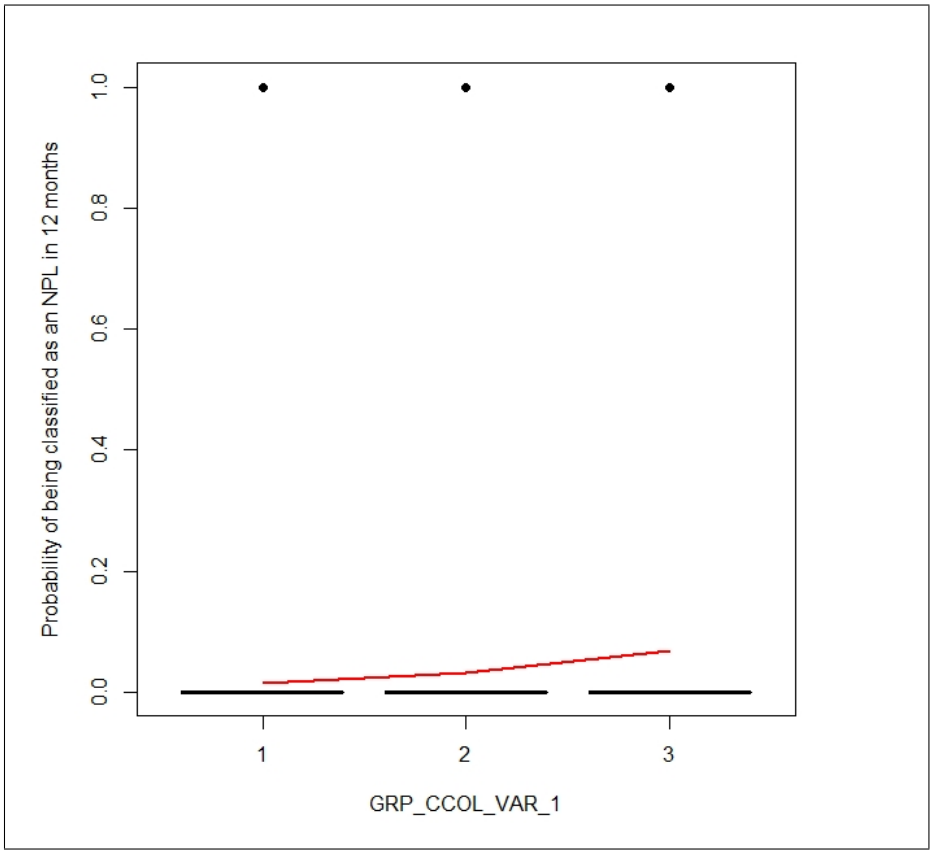


Figure 7.34. Probability of being classified as an NPL in 12 months versus CCOL_VAR_1.

7.3.3. Summary of the Studies on Raw Variables

This section is of great importance, if one decides to work with unbinned variables. As univariate analysis techniques, histogram, boxplot and bar plot of sample raw variables were shared. Especially, it was shown how the bar plot together with an event rate line can be used to detect the relationship between increasing values of a variable and the respective event rate. Although, higher order terms may be statistically significant looking at the logistic regression output in some cases, using bar plots, one can decide whether the nonlinearity observed is meaningful. This type of a plot can also be used to categorize variables without applying any intelligent method like decision trees. Besides, we obtained diagnostic plots to determine outliers and influential observations, which are really helpful graphical tools to examine covariates individually. Finally, we showed the probability of default with respect to values of

a covariate before and after categorizing. Such a plot also gives insights about the information loss as a consequence of categorizing.

8. EXAMINATION ON INTERACTION TERMS

GRP variables are binned variables. A continuous variable can be transformed into a GRP variable easily whether with expert opinion or using some smart algorithms like decision trees. Consider a raw variable is binned into 3 categories: 1 (low values), 2 (middle values), 3 (high values) are the values assigned to the GRP variable. The newly obtained variable is then transformed into a factor using as.factor() function of R, so that it has three categories.

WOE values of each category of a GRP variable are calculated using non-default ratio and default ratio in that category. WOE variables are interval variables and easy to interpret and they often do not differ much from GRP variables in terms of explanatory power, thus they do not incur great disadvantage to use them instead of GRP variables, in case a large number of observations to represent the population is available. Yet, monotonicity is ensured as stated earlier in Chapter 6, in case WOE variables are used as covariates.

Our assumption was that we lose information, when variables are binned. On the other hand, categorizing variables is beneficial for interpretability and the representation of NAs. It should be once again emphasized that NAs for credit bureau variables are not missing values and imputing these would also be a source of information loss. In order to compensate the loss, we deal with interaction terms in this chapter. Hence, we introduce new approaches and practical functions to automatically calculate WOE values for two-way and three-way interactions of categorized variables and eventually use the obtained interaction WOE variables as predictors to assess their contribution.

Throughout this chapter, we will use R notations also in headings. In R, when only main effects are to be included in the model fitted, a plus sign (+) between the covariate names is used. When in addition to main effects, the interaction term is also included, then asterisk (*) is used between the covariate names. If one is only interested in the interaction term of two main effects, then colon (:) between the covariate names

is placed. This notation will also be used as subsection headings for readability.

In this chapter, we examine the contribution of interaction terms in the context of Gini indices. We used the training and test data sets of GRP variables and WOE variables that were created in Chapter 6. In Section 8.1, we investigate two-way interaction terms for GRP variables, whereas in Section 8.2, we focus on obtaining WOE variables for interaction terms. In Section 8.3, we study three-way interaction terms for GRP variables, and finally in Section 8.4 we concentrate on calculation of WOE variables for three-way interaction terms.

8.1. Two-way Interactions Using GRP Variables

Two independent variables interact if the effect of one of the variables differs depending on the level of the other variable [26]. In this section, logistic regression outputs of two models, with only main variables and also with interaction term, respectively, are shared in the first step. In the tabulated summaries, we can see that the interaction terms are significant. Then, interactions are plotted for each case. Plots also support the significance of the interaction terms, as the moderator variable has an effect on the slope of the line of the other variable. The presence of a significant interaction indicates that the effect of one predictor variable on the response variable is different at different values of the other predictor.

In this section, we will make use the standard method used by practitioners to include the interaction term into a regression model using categorized variables. We will present three samples of interactions to explain our approach and show the usage of functions created. It is not recommended to study every combination of main effects to find interactions neglecting the expert's view, because sometimes data may reveal by chance as if there is an interaction, although in reality such an interaction is not logical and meaningful. Besides, it is also time consuming and complicated to examine each possible interaction, when there are hundreds of variables.

8.1.1. GRP_CC_VAR_6 * GRP_CCOL_VAR_1

In this subsection, we examine the presence of an interaction between two main effects, namely GRP_CC_VAR_6 and GRP_CCOL_VAR_1. These are the binned versions of the following variables:

(i) CC_VAR_6: Ratio of the payment in the last 3 months to its limit for open credit cards.
(ii) CCOL_VAR_1: Number of distinct banks that granted credit cards or overdraft loans whose limit usage is greater than 80%.

The conditional hypothesis is the relationship between the response variable, NPL_IN_12M, and GRP_CC_VAR_6 depends on the value of GRP_CCOL_VAR_-1.

In Figure 8.1, necessary R packages for the studies in this subsection are shared, namely stargazer [27], effects [28], ggplot2 [24] and pROC [19]. First the training and test data sets of GRP variables that were obtained in Chapter 6 are transformed into categorical variables with as.factor() function. Subsequently, two logistic regressions are fitted with/out interaction term. The summaries of the fitted models are tabulated using the stargazer() function of stargazer package [27] in a proper and formatted style.

The summaries of both fitted models with/out interaction term are presented in Figure 8.2. Both main effects are significant, when interaction term is not included into the model. Using design variables, in order to detect whether a categorized variable is significant, significance of a single category of it is sufficient. For example, GRP_CCOL_VAR_1 has three categories, thus two design variables, or dummy variables, are necessary. When the first category of this variable is the reference, then the second category is represented with GRP_CCOL_VAR_12, and the third with GRP_CCOL_VAR_13. If one of them is significant, then we can conclude that GRP_CCOL_VAR_1 is significant and has an influence on the response variable. GRP_CCOL_VAR_12 is significant, because the p-value related to its coefficient is

```
# Necessary packages are loaded.
library(stargazer)
library(effects)
library(ggplot2)
library(pROC)

# GRP variables are transformed into factors
# Training split
for(i in 2:length (Var_all_GRP_train){
  Var_all_GRP_train[, i] ← as.factor(Var_all_GRP_train[, i])
}

# Test split
for(i in 2:length(Var_all_GRP_test){
  Var_all_GRP_test[, i] ← as.factor(Var_all_GRP_test[, i])
}

# Two logistic regression models are fitted using the training split with/out
interaction term.
# Without interaction term
mod_fit_woI ← glm(NPL_IN_12M ∼ GRP_CC_VAR_6 + GRP_CCOL_-
VAR_1, data = Var_all_GRP_train, family = binomial(link = "logit"))

# With interaction term
mod_fit_wI ← glm(NPL_IN_12M ∼ GRP_CC_VAR_6 * GRP_CCOL_-
VAR_1, data = Var_all_GRP_train, family = binomial(link = "logit"))

# Summary tables for both fitted models
stargazer(mod_fit_woI, mod_fit_wI,
  type = "text",
  column.labels = c("only Main Effects", "with Interaction"),
  intercept.bottom = FALSE,
  single.row=TRUE,
  notes.append = FALSE,
  omit.stat=c("ser"),
  star.cutoffs = c(0.05, 0.01, 0.001),
  header=FALSE)
```

Figure 8.1. R code to fit logistic regression models with/out interaction term, and to print the summaries of fitted models.

smaller than 0.05, in a 95% confidence interval, and we reject the null hypothesis that regression coefficient related to it is zero. Consequently, GRP_CCOL_VAR_1 is also significant. This explanation applies also for the GRP_CC_VAR_6 and the interaction term.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-4.504*** (0.074)	-5.063*** (0.136)
GRP_CC_VAR_62	-0.166 (0.094)	0.337 (0.197)
GRP_CC_VAR_63	-0.199* (0.098)	0.450* (0.216)
GRP_CC_VAR_64	-0.115 (0.096)	0.846*** (0.207)
GRP_CC_VAR_65	0.326*** (0.086)	1.100*** (0.199)
GRP_CC_VAR_66	1.080*** (0.086)	1.723*** (0.153)
GRP_CCOL_VAR_12	0.943*** (0.064)	1.626*** (0.162)
GRP_CCOL_VAR_13	1.886*** (0.080)	2.980*** (0.190)
GRP_CC_VAR_62:GRP_CCOL_VAR_12		-0.753** (0.239)
GRP_CC_VAR_63:GRP_CCOL_VAR_12		-0.763** (0.255)
GRP_CC_VAR_64:GRP_CCOL_VAR_12		-1.117*** (0.246)
GRP_CC_VAR_65:GRP_CCOL_VAR_12		-0.821*** (0.229)
GRP_CC_VAR_66:GRP_CCOL_VAR_12		-0.891*** (0.200)
GRP_CC_VAR_62:GRP_CCOL_VAR_13		-0.777** (0.268)
GRP_CC_VAR_63:GRP_CCOL_VAR_13		-1.249*** (0.285)
GRP_CC_VAR_64:GRP_CCOL_VAR_13		-1.642*** (0.274)
GRP_CC_VAR_65:GRP_CCOL_VAR_13		-1.534*** (0.260)
GRP_CC_VAR_66:GRP_CCOL_VAR_13		-1.719** (0.568)
Observations	58,955	58,955
Log Likelihood	-6,841.117	-6,809.626
Akaike Inf. Crit.	13,698.230	13,655.250
Note:		*p<0.05; **p<0.01; ***p<0.001

Figure 8.2. Summaries of fitted logistic regression models with/out interaction term, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the main effects.

It is also important to emphasize that if the interaction term is significant, it is necessary to include the constructive main effects into the model, whether they are significant or not. Hence, one can calculate the predicted probability using 17 coefficients and an intercept for this case. This calculation is done with statistical packages, of course; however, for anybody who uses the output of the logistic regression but does not possess deep knowledge on regression analyses may experience great difficulty to

```
# The Interaction term
Inter.mod_fit_wI ← effect('GRP_CC_VAR_6 * GRP_CCOL_VAR_1',
mod_fit_wI, se = TRUE)
Inter.mod_fit_wI.DF ← as.data.frame(Inter.mod_fit_wI)

# Relable them to put them back in order
Inter.mod_fit_wI.DF$GRP_CC_VAR_6 ← factor(Inter.mod_fit_wI.DF-
$GRP_CC_VAR_6,
  level = c("1", "2", "3", "4", "5", "6"),
  labels = c("<0.06", "[0.06 , 0.11)", "[0.11 , 0.17)", "[0.17 , 0.27)", "0.27+",
"NA"))
Inter.mod_fit_wI.DF$GRP_CCOL_VAR_1 ← factor(Inter.mod_fit_-
wI.DF$GRP_CCOL_VAR_1,
  level = c("1", "2", "3"),
  labels = c("0 or NA", "1-2", "3+"))

# Create plot
ggplot(data = Inter.mod_fit_wI.DF, aes(x = GRP_CC_VAR_6, y = fit, color
= GRP_CCOL_VAR_1, group = GRP_CCOL_VAR_1)) +
  geom_point(size = 3.6) + geom_line(size = 1.2) +
  geom_ribbon(aes(ymin = lower, ymax = upper, fill = GRP_CCOL_VAR_-
1), alpha = 0.3) +
# labs(title = "GRP_CC_VAR_6 and GRP_CCOL_VAR_1 as NPL_IN_-
12M Predictors",
  labs(x = "GRP_CC_VAR_6",
  y = "Predicted probability of being classified as NPL in 12 months",
  color = "GRP_CCOL_VAR_1", fill = "GRP_CCOL_VAR_1") +
  theme_bw() + theme(text = element_text(size = 12),
  legend.text = element_text(size = 12),
  legend.direction = "horizontal",
  panel.grid.major = element_blank(),
  panel.grid.minor = element_blank(),
  legend.position = "top")
```

Figure 8.3. R code to print marginal effects, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the main effects.

draw conclusions. [29] is a very useful paper to understand interaction effects and how models can be fitted with interaction terms.

Looking at the log likelihood, we can also conclude that the model with interaction term is better, because we want to maximize the log likelihood, the higher value is better. Besides, we want to minimize the AIC and not surprisingly the model with interaction term has a smaller AIC value.

In order to visualize the interaction term, effect() function of effects package [28] is utilized. Then, the ggplot() function of ggplot2 package [24] is used for plotting Figure 8.4. effect() function calculates the predicted probability of being classified as an NPL in 12 months for each combination of categories of constructive main effects. Additionally, upper and lower boundaries for a 95% confidence interval are obtained with this function.

From Figure 8.4, one can see that the lines, whose colors depend on the level of the moderator variable, are not exactly parallel to each other. Namely, at different levels of GRP_CCOL_VAR_1, the effect of GRP_CC_VAR_6 differs. The figure explains a lot about the effects of covariates. When GRP_CC_VAR_6 takes on category 1, i.e. ratio of the payment in the last 3 months to its limit for open credit cards is less than 6%, the probability of default increases with the increasing number of distinct banks that granted credit cards or overdraft loans whose limit usage is greater than 80%. GRP_CCOL_VAR_1 is zero, that is the customer who has no credit cards or overdraft loans whose limit usage is greater than 80% is the least risky. With the increasing ratio of payment, probability of default decreases when the customer has at least one credit card or overdraft loan whose limit usage is greater than 80%. However, there is a slight increase in the probability of default in this case, if the customer has no credit cards or overdraft loans whose limit usage is greater than 80%. To sum up, looking at the lines of effects, one can draw various conclusions about the effects of the main effects and the interaction.

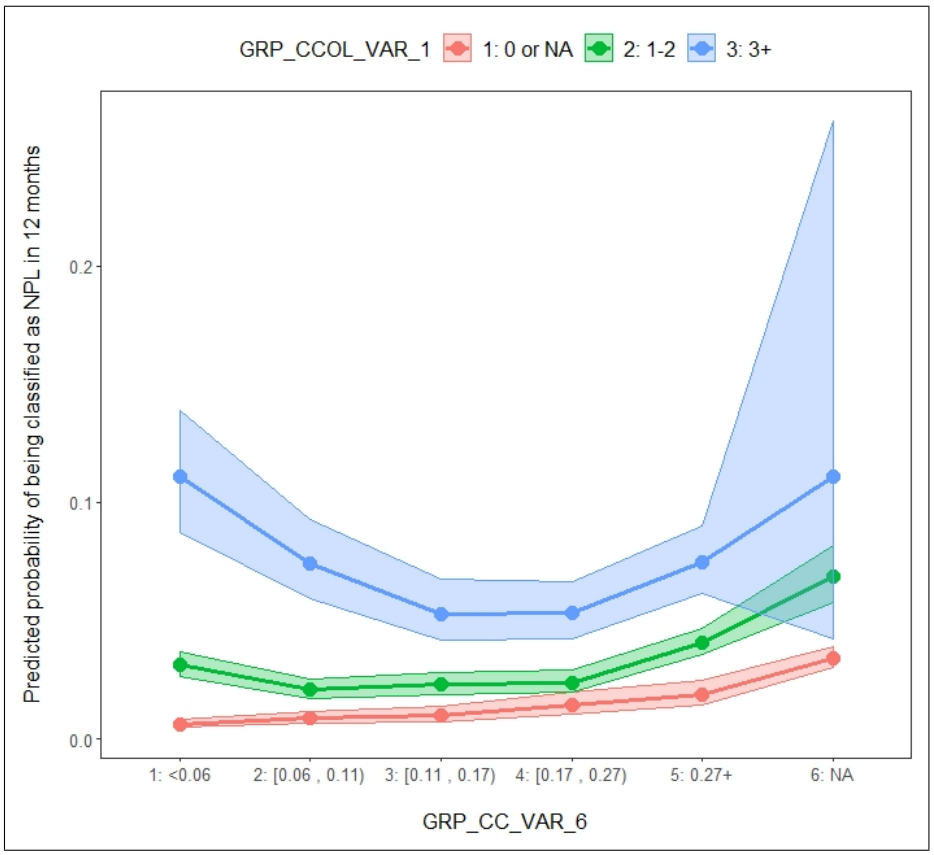


Figure 8.4. Plot of marginal effects, when GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are NPL_IN_12M predictors.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits. R code is given in Figure 8.5.

```
# Gini calculation for training and test splits using roc() from pROC package

# Without Interaction #
pred_train_woI ← predict(mod_fit_woI, type = "response")
round ((roc(Var_all_GRP_train$NPL_IN_12M, pred_train_woI)$auc * 2 -
1) * 100, 2)
# outputs the Gini index: 38.46

pred_test_woI ← predict(mod_fit_woI, Var_all_GRP_test, type = "re-
sponse")
round ((roc(Var_all_GRP_test$NPL_IN_12M, pred_test_woI)$auc * 2 - 1)
* 100, 2)
# outputs the Gini index: 36.3

# With Interaction #
pred_train_wI ← predict(mod_fit_wI, type = "response")
round ((roc(Var_all_GRP_train$NPL_IN_12M, pred_train_wI)$auc * 2 - 1)
* 100, 2)
# outputs the Gini index: 39.82

pred_test_wI ← predict(mod_fit_wI, Var_all_GRP_test, type = "response")
round ((roc(Var_all_GRP_test$NPL_IN_12M, pred_test_wI)$auc * 2 - 1) *
100, 2)
# outputs the Gini index: 37.18
```

Figure 8.5. R code to calculate Gini indices of fitted model alternatives for training and test splits, in case GRP variables are utilized.

Gini indices calculated in Figure 8.5 are given in Table 8.1. It can be seen that the Gini index of the model is increased by more than 1.3 when the interaction term is included, whereas the increase for the test split is approximately 0.9. The amount of increase is not great, however one point of increase in Gini index may have a significant impact on the profits of a bank.

Table 8.1. Gini indices of fitted models with/out interaction term, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the main effects.

	Without interaction term	**With interaction term**
Training	38.46	39.82
Test	36.3	37.18

8.1.2. GRP_ALL_VAR_5 * GRP_CC_VAR_6

In this subsection, we examine the presence of an interaction between two main effects, namely GRP_ALL_VAR_5 and GRP_CC_VAR_6. These are the binned versions of the following variables:

- ALL_VAR_5: Total arrears balance of open loans of any type.
- CC_VAR_6: Ratio of the payment in the last 3 months to its limit for open credit cards.

Two logistic regressions are fitted with/out interaction term modifying the R code in Figure 8.1, in case GRP_ALL_VAR_5 and GRP_CC_VAR_6 are constructive main effects. The summaries of the fitted models with/out interaction term are presented in Figure 8.6. Both main effects are significant, when interaction term is not included into the model, since significance of a single category of a binned variable is sufficient to acknowledge the significance of the variable, as stated in Subsection 8.1.1. The interaction term is significant, as well.

Looking at the log likelihood, we can also conclude that the model with interaction term is better, because we want to maximize the log likelihood, the higher value is better. Besides, we want to minimize the AIC and not surprisingly the model with interaction term has a smaller AIC value.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-4.209*** (0.066)	-4.420*** (0.082)
GRP_ALL_VAR_52	1.630*** (0.060)	2.276*** (0.132)
GRP_CC_VAR_62	-0.004 (0.094)	0.201 (0.117)
GRP_CC_VAR_63	0.065 (0.097)	0.348** (0.120)
GRP_CC_VAR_64	0.230* (0.095)	0.478*** (0.118)
GRP_CC_VAR_65	0.749*** (0.084)	1.044*** (0.103)
GRP_CC_VAR_66	0.970*** (0.085)	1.181*** (0.101)
GRP_ALL_VAR_52:GRP_CC_VAR_62		-0.629** (0.201)
GRP_ALL_VAR_52:GRP_CC_VAR_63		-0.879*** (0.209)
GRP_ALL_VAR_52:GRP_CC_VAR_64		-0.763*** (0.201)
GRP_ALL_VAR_52:GRP_CC_VAR_65		-0.993*** (0.188)
GRP_ALL_VAR_52:GRP_CC_VAR_66		-0.651** (0.220)
Observations	58,955	58,955
Log Likelihood	-6,830.540	-6,814.196
Akaike Inf. Crit.	13,675.080	13,652.390
Note:	*p<0.05; **p<0.01; ***p<0.001	

Figure 8.6. Summaries of fitted logistic regression models with/out interaction term, in case GRP_ALL_VAR_5 and GRP_CC_VAR_6 are the main effects.

From Figure 8.7, one can see that the lines, whose colors depend on the level of the moderator variable, are not exactly parallel to each other. Namely, at different levels of GRP_ALL_VAR_5, the effect of GRP_CC_VAR_6 differs. The figure explains a lot about the effects of covariates. First of all, if GRP_ALL_VAR_5 takes second category, i.e. total arrears balance of open loans of any type is greater than 611 TL, or there is not an open loan at the time of inquiry, probability of default is comparably greater. It is comprehensible that a customer paying on time is less risky. Secondly, with the increasing ratio of the payment in the last 3 months to its limit for open credit cards, probability of default tends to increase. This can be interpreted so that customers with high limit utilization are riskier. Another point attracts attention that riskiest customers are those who do not have an open credit card in the last three months, and if they have any other open loans, their total arrears balance is greater than 611 TL. To sum up, looking at the lines of effects, one can draw various conclusions about the effects of the main effects and the interaction and decide whether they are

sensible.

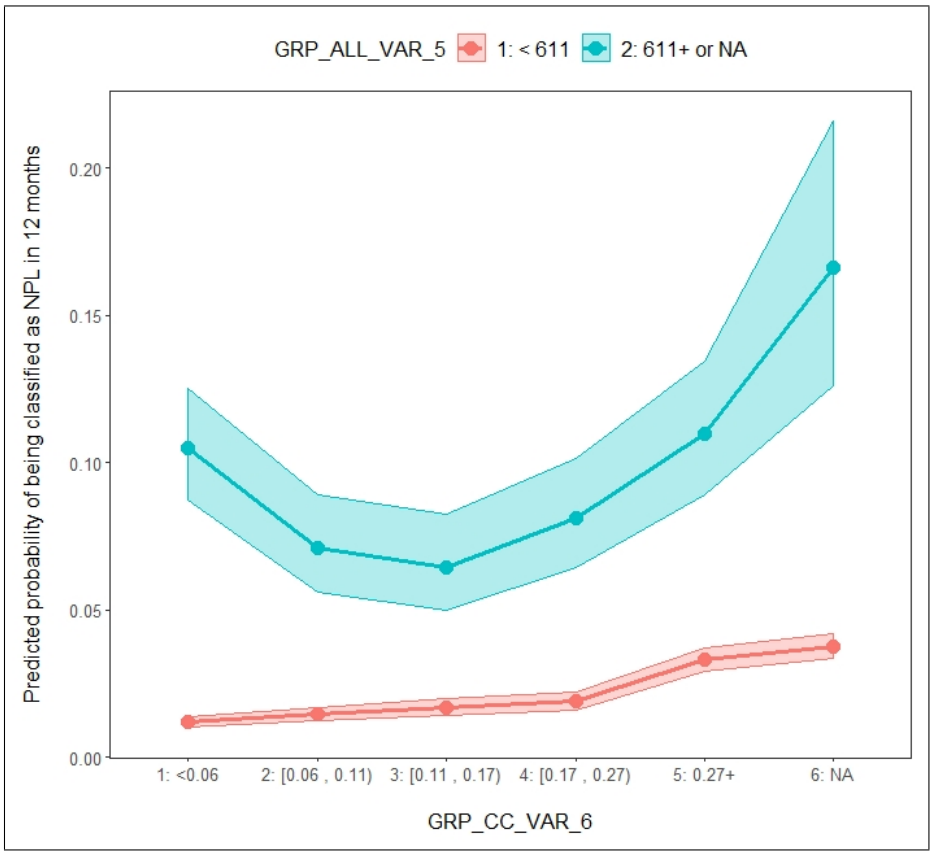


Figure 8.7. Plot of marginal effects, when GRP_ALL_VAR_5 and GRP_CC_VAR_6 are NPL_IN_12M predictors.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits, as usual. From Table 8.2, it can be seen that the Gini index of the model is increased by just under 0.9, when the interaction term is included, whereas the increase for the test split is even bigger, i.e. 1.3.

Table 8.2. Gini indices of fitted models with/out interaction term, in case GRP_ALL_VAR_5 and GRP_CC_VAR_6 are the main effects.

	Without interaction term	**With interaction term**
Training	35.69	36.58
Test	36.19	37.49

8.1.3. GRP_CC_VAR_2 * GRP_OL_VAR_2

In this subsection, we examine the presence of an interaction between two main effects, namely GRP_CC_VAR_2 and GRP_OL_VAR_2. These are the binned versions of the following variables:

- CC_VAR_2: Worst payment status in credit cards.
- OL_VAR_2: Age of the oldest overdraft loan excluding the bad-closed ones.

Two logistic regressions are fitted with/out interaction term modifying the R code in Figure 8.1, in case GRP_CC_VAR_2 and GRP_OL_VAR_2 are constructive main effects. The summaries of the fitted models with/out interaction term are presented in Figure 8.8. Both main effects are significant, when interaction term is not included into the model, since significance of a single category of a binned variable is sufficient to acknowledge the significance of the variable. The interaction term is significant, as well.

Looking at the log likelihood, we can also conclude that the model with interaction term is better, because we want to maximize the log likelihood, the higher value is better. Besides, we want to minimize the AIC and again the model with interaction term has a smaller AIC value.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.439*** (0.062)	-3.094*** (0.073)
GRP_CC_VAR_22	0.163* (0.073)	-0.094 (0.124)
GRP_CC_VAR_23	0.668*** (0.078)	0.272 (0.141)
GRP_CC_VAR_24	1.058*** (0.069)	0.344** (0.113)
GRP_OL_VAR_22	-0.371*** (0.076)	-1.036*** (0.159)
GRP_OL_VAR_23	-0.613*** (0.079)	-0.930*** (0.150)
GRP_OL_VAR_24	-0.761*** (0.080)	-1.506*** (0.179)
GRP_OL_VAR_25	-1.119*** (0.077)	-1.770*** (0.155)
GRP_CC_VAR_22:GRP_OL_VAR_22		0.571* (0.223)
GRP_CC_VAR_23:GRP_OL_VAR_22		0.832*** (0.235)
GRP_CC_VAR_24:GRP_OL_VAR_22		1.184*** (0.210)
GRP_CC_VAR_22:GRP_OL_VAR_23		0.175 (0.223)
GRP_CC_VAR_23:GRP_OL_VAR_23		0.032 (0.251)
GRP_CC_VAR_24:GRP_OL_VAR_23		0.910*** (0.204)
GRP_CC_VAR_22:GRP_OL_VAR_24		0.725** (0.239)
GRP_CC_VAR_23:GRP_OL_VAR_24		0.874*** (0.255)
GRP_CC_VAR_24:GRP_OL_VAR_24		1.261*** (0.230)
GRP_CC_VAR_22:GRP_OL_VAR_25		0.444* (0.220)
GRP_CC_VAR_23:GRP_OL_VAR_25		0.835*** (0.234)
GRP_CC_VAR_24:GRP_OL_VAR_25		1.295*** (0.208)
Observations	58,955	58,955
Log Likelihood	-6,925.283	-6,884.453
Akaike Inf. Crit.	13,866.570	13,808.910
Note:		*p<0.05; **p<0.01; ***p<0.001

Figure 8.8. Summaries of fitted logistic regression models with/out interaction term, in case GRP_CC_VAR_2 and GRP_OL_VAR_2 are the main effects.

From Figure 8.9, one can see that the lines, whose colors depend on the level of the moderator variable, are not exactly parallel to each other. Namely, at different levels of GRP_CC_VAR_2, the effect of GRP_OL_VAR_2 differs. The figure explains a lot about the effects of covariates. First of all, longer the time has passed after the customer opened an overdraft account, less risky he is according to the plot. It is sensible that this can be accepted as financial literacy and capability to use banking products. Secondly, probability of default obviously increases with the number of credit card payments past due. Especially, jagged line for the case with two payment past due, may be studied whether it makes a good business sense. As stressed earlier, one

can draw various conclusions about the effects of the main effects and the interaction looking at the interaction plot.

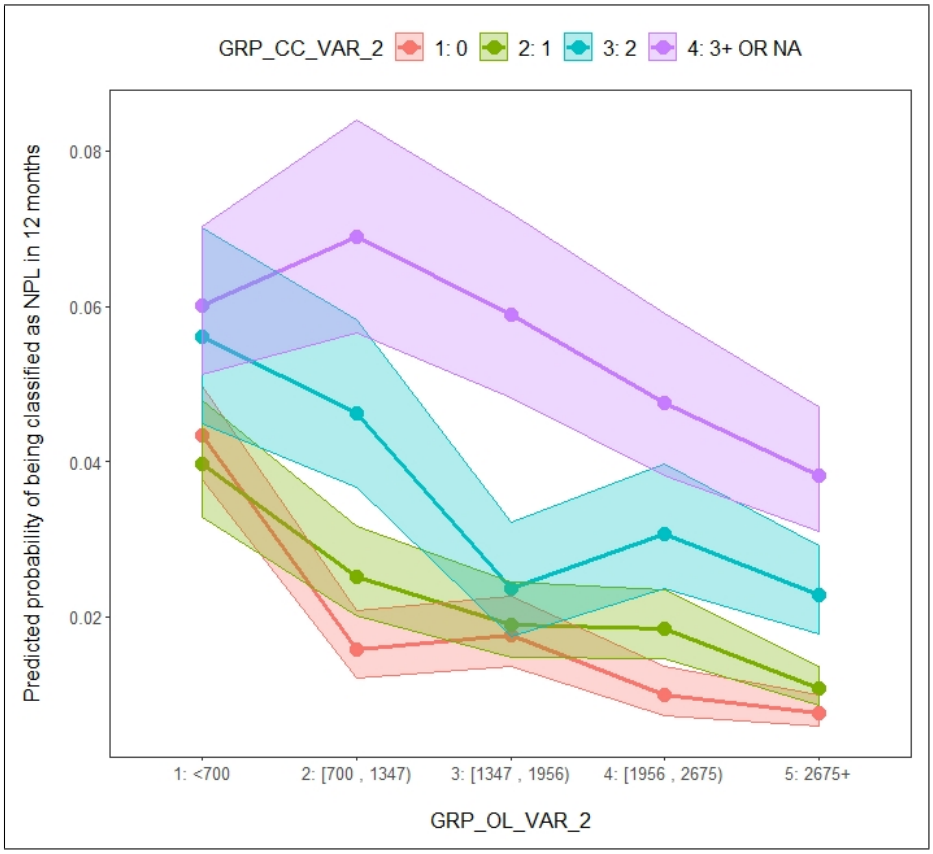


Figure 8.9. Plot of marginal effects, when GRP_CC_VAR_2 and GRP_OL_VAR_2 are NPL_IN_12M predictors.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits as usual. From Table 8.3, it can be seen that the Gini index of the model is increased by just above 1.1, when the interaction term is included, whereas the Gini index barely changes for the test split.

Table 8.3. Gini indices of fitted models with/out interaction term, in case GRP_OL_VAR_2 and GRP_CC_VAR_2 are the main effects.

	Without interaction term	**With interaction term**
Training	35.54	36.68
Test	38.31	38.36

8.2. Two-way Interactions Using WOE Variables

In Section 8.1, we showed how we can make use of interaction terms, in case the main effects are binned, GRP, variables. When we want to proceed with WOE variables, calculating the WOE values for interaction terms is an issue to be solved. In this section, we share how we calculate the WOE values for two-way interactions, overcome the problems we faced and also present their contribution to the models.

8.2.1. Description of the Approach to Calculate WOE Values for Interaction Terms

In this subsection, we will briefly describe our approach to calculate WOE values for interaction terms, in case one chooses to proceed with WOE variables for model development and automatically perform necessary prior calculations to obtain WOE versions of interaction terms.

In Section 8.1, we showed the standard method, if one aims to include the interaction of categorized variables into the model using categorized variables. This is straightforward and often applied by anyone who carry out regression analysis. On the other hand, concept of Weight of Evidence (WOE) is more popular amongst practitioners, when it comes to develop credit scoring models in financial institutions where a large number of observations are available. Although, GRP variables are better able to withstand future changes in the distribution, WOE variables do not cause a great disadvantage, considering the frequency of remodeling activities of institutions that especially operate in countries more exposed to economic and demographic changes.

Some practitioners state the reason of their choice of using directly GRP variables as the ease of introducing interactions effects into the model and discover nonlinear relationships, etc. At this very point, we introduce a new approach to automatically calculate WOE versions of interaction terms. In this subsection, we describe this approach for a two-way interaction; however, it can be easily extended for higher order interaction terms.

In order to calculate WOE values for each combination of categories of main effects, GRP variables are utilized. A GRP variable takes integer values depicting its bins. Consider a raw variable is divided into four bins with regards to event rate. Event rate in the first bin, i.e. GPR value 1, is not necessarily highest or lowest; besides, event rate may not monotonically decrease or increase with increasing bin values, as nonlinearity may be in question. Thus, to provide interpretable results, firstly, bins of each main effect are sorted in a decreasing order of event rates. Subsequently, the worst group of the first variable is broken up into bins regarding the sorted categories of second variable. If a higher order interaction term is to be obtained, the resulting first bin, which is formed by observations that take worst categories of both covariates, is broken up into bins regarding the sorted categories of third variable, and so on. Consequently, the very last bin includes the observations which take the best group values of each variable. Afterwards, WOE calculations can be carried out using the default and nondefault numbers of each category, as shown earlier.

Normally, when a raw variable is binned, each bin includes, generally, at least 5% of observations. However, a newly obtained category of an interaction term, may include less observations. In order to calculate a WOE value for a bin, we determined a rule that there should be at least 100 observations in that bin, and at least one sample from each label, i.e. nondefault and default in this case, should be represented.

Observations, for which WOE values could not be calculated, are excluded from the training data set. This is not a huge problem during model development, yet it may cause a financial loss in real life, because customers that demonstrate a pattern for which a WOE value could not be calculated, would directly be rejected. In order

to solve this issue, we propose an approach, the so-called Neighbours' approach. For a two-way interaction, a covariate pattern consists of category of the first variable (C1) and category of the second variable (C2) = (C1, C2). If this pattern does not include sufficient default and/or non-default records as stated earlier, then we search for its neighbours in two dimensions. Consider a three by three square, it has 9 cells. At the center is the pattern in question, thus (C1, C2) may have up to eight neighbours: (C1±1, C2±1) The logic depends on the decreasing/increasing order of default ratios of categories of a covariate. Thus, default ratio of a category is between its upper and lower categories. As the number of covariates is increased by one, neighbours' approach is shifted to third dimension, from a 3x3 square to a 3x3x3 cube.

After addressing the issue of uncalculated WOE values and introducing Neighbours' approach, we formulized our treatment policy of imputing uncalculated WOE values. If a bin does not include any defaults, a reasonable number, 100 as an expert's view, then its WOE value cannot be calculated directly given divisor is zero in the equation. However, there are sufficient number of observations in that bin to draw conclusions. We can estimate that the proportion of defaults is close to 0 but "0.7/number of observations" is also a sensible (and relatively high) estimate for the proportion of defaults. Therefore, it makes sense to calculate WOE for that cell using 0.7 instead of 0 defaults. Otherwise, after specifying the neighbours, if available, the weighted average of WOE values of its neighbours is calculated to impute the missing WOE value.

After obtaining the WOE values for each of the bins, WOE version of interaction term is obtained for each observation by assigning the calculated value to respective configuration of categories of constructive main effects.

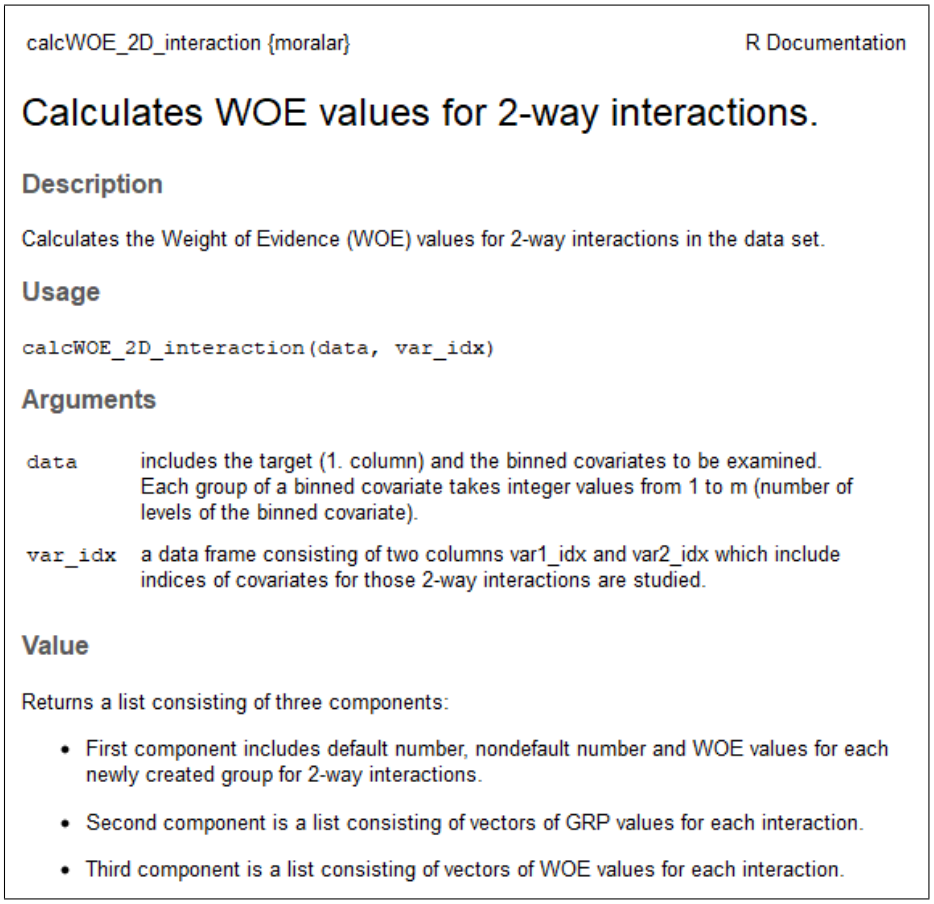
It should be stressed again, that WOE values are calculated using only the training data set. Thus, after completing the steps above, we implemented the necessary functions to assign obtained WOE values for interaction terms to other data sets for which predictions are needed.

An explicit implementation of our approach using R functions is shared in Subsection 8.2.2 in case of a two-way interaction.

8.2.2. WOE_CC_VAR_6 * WOE_CCOL_VAR_1

In Subsection 8.1.1, we showed how the interaction term can be included into a regression model, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the main effects. When we want to fit a regression with WOE variables, each combination of main effects should be taken into consideration. GRP_CC_VAR_6 is binned into six categories, whereas GRP_CCOL_VAR_1 is binned into three categories. That is, there are 18 new categories for which we need to calculate WOE values, when we want to utilize WOE variables. calcWOE_2D_interaction() function calculates the WOE values easily. This function produces three components that are used during the scorecard model development process. A screenshot of the R documentation for this function is given in Figure 8.10.

In Figure 8.11, one can see the usage of this function. First, the indices of the main effects are specified. Afterwards, these indices are stored in a data frame as a parameter for the function. Here, it should be stressed that the WOE values are calculated for the training split. The object generated by the function is a list that includes three components.



calcWOE_2D_interaction {moralar} R Documentation

Calculates WOE values for 2-way interactions.

Description

Calculates the Weight of Evidence (WOE) values for 2-way interactions in the data set.

Usage

```
calcWOE_2D_interaction(data, var_idx)
```

Arguments

data	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
var_idx	a data frame consisting of two columns var1_idx and var2_idx which include indices of covariates for those 2-way interactions are studied.

Value

Returns a list consisting of three components:

- First component includes default number, nondefault number and WOE values for each newly created group for 2-way interactions.
- Second component is a list consisting of vectors of GRP values for each interaction.
- Third component is a list consisting of vectors of WOE values for each interaction.

Figure 8.10. Screenshot from the R documentation of calcWOE_2D_interaction() function.

```
grep("GRP_CC_VAR_6", colnames(Var_all_GRP_train))
# outputs the index of GRP_CC_VAR_6 in the data set: 13

grep("GRP_CCOL_VAR_1", colnames(Var_all_GRP_train))
# outputs the index of GRP_CCOL_VAR_1 in the data set: 14

# WOE values of the two-way interaction term is calculated.
data_WOE_2D_interaction ← calcWOE_2D_interaction(Var_all_GRP_-
train, var_idx = data.frame(var1_idx = c(13), var2_idx = c(14)))

# The first component of the object is printed, that is, nondefault number,
default number and WOE value of each bin of the interaction term is tabulated.
data_WOE_2D_interaction[[1]]$‘GRP_CC_VAR_6 X GRP_CCOL_-
VAR_1‘
```

Figure 8.11. R code to calculate WOE version of the two-way interaction term, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the constructive main effects.

The first component is given in Table 8.4, that includes nondefault number, default number and WOE values for each combination. calcWOE_2D_interaction() function utilizes the orderAttributes() function introduced in Chapter 7, so that the bins of each main effect are sorted in a decreasing order of default rates. Subsequently, the worst group of the first variable is broken up into bins regarding the value the second variable takes. Hence, the resulting first bin includes the observations which take on the worst group values of both variables. Consequently, the very last bin includes the observations which take on the best group values of both variables. In order to calculate a WOE value for a bin, we determined a rule that there should be at least 100 observations in that bin, and there must be default and nondefault samples, at least one for each.

Table 8.4. First component of the object produced by calcWOE_2D_interaction() function, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the constructive main effects.

	6 x 3	6 x 2	6 x 1	5 x 3	5 x 2	5 x 1
numNonDef	32	1611	6155	1252	4634	2579
numDef	4	119	218	101	197	49
WOE	NA	-0.9991	-0.2641	-1.0872	-0.4466	0.3587

	4 x 3	4 x 2	4 x 1	3 x 3	3 x 2	3 x 1
numNonDef	1264	4038	2851	1107	3955	3629
numDef	71	99	42	62	93	36
WOE	-0.7253	0.1038	0.6131	-0.7223	0.1455	1.0086

	2 x 3	2 x 2	2 x 1	1 x 3	1 x 2	1 x 1
numNonDef	885	4478	5642	514	4230	8538
numDef	71	95	50	64	136	54
WOE	-1.0817	0.2484	1.1214	-1.5213	-0.1673	1.4587

The second component of the object includes GRP values of the newly created interaction term. In Table 8.4, it can be seen that the first component is a matrix that has 18 (6 x 3) columns. Each observation takes a proper GRP value according to its column index. In Figure 8.12, R code to print the GRP values of the interaction term for the first ten observations is shared. The output of this expression is given in Table 8.5.

```
# The second component, that includes GRP values of the interaction term, of
the object is printed for the first ten observations.
head(data_WOE_2D_interaction[[2]]$‘GRP_CC_VAR_6 X GRP_CCOL_-
VAR_1‘, 10)
```

Figure 8.12. R expression to print the second component of the returned object by calcWOE_2D_interaction() function, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the constructive main effects.

Table 8.5. GRP values of the interaction term for the first 10 observations, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the constructive main effects.

	GRP
1	12
2	14
3	4
4	5
5	3
6	14
7	9
8	15
9	3
10	15

The third component of the object includes the WOE values of the newly created interaction term. For each GRP value, i.e. column index of the matrix in the first component, the value in the third row, namely WOE value, is assigned. In Figure 8.13, R code to print the WOE values of the interaction term for the first ten observations is shared. The output of this expression is given in Table 8.6.

```
# The third component, that includes WOE values of the interaction term, of the object is printed for the first ten observations.
head(data_WOE_2D_interaction[[3]]$`GRP_CC_VAR_6 X GRP_CCOL_VAR_1`, 10)
```

Figure 8.13. R expression to print the third component of the returned object by calcWOE_2D_interaction() function, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the constructive main effects.

Table 8.6. WOE values of the interaction term for the first 10 observations, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the constructive main effects.

	WOE
1	1.0086
2	0.2484
3	-1.0872
4	-0.4466
5	-0.2641
6	0.2484
7	0.6131
8	1.1214
9	-0.2641
10	1.1214

After obtaining the WOE values by means of calcWOE_2D_interaction() function easily, the third component of the returned object is first renamed in a syntactically valid way and then combined with the main effects and the response variable. In Figure 8.14, one can see R expressions to fit logistic regression models with/out interaction term and print the summaries in an elegant way.

```
# Response variable, main effects and interaction term are joined together.
data_train ← cbind(Var_all_WOE_train[,c(1, 13, 14)], data_WOE_2D_in-
teraction[[3]][1])

length(data_train[is.na(data_train[, 4]) == 1, 4])
# There 36 records taking NA

# Interaction term is renamed.
colnames(data_train)[4]←"Interaction_var_2D"

# Logistic regression models are fitted using the training split for two cases:
# Without interaction term
mod_fit_2D_woI ← glm(NPL_IN_12M ∼ ., data = data_train[, -c(4)], family
= binomial(link = "logit"))

# With interaction term
mod_fit_2D_wI ← glm(NPL_IN_12M ∼ ., data = data_train, family = bi-
nomial(link = "logit"))

# Summary tables for both fitted models
stargazer(mod_fit_2D_woI, mod_fit_2D_wI,
  type = "text",
  column.labels = c("only Main Effects", "with Interaction"),
  intercept.bottom = FALSE,
  single.row=TRUE,
  notes.append = FALSE,
  omit.stat=c("ser"),
  star.cutoffs = c(0.05, 0.01, 0.001),
  header=FALSE)
```

Figure 8.14. R code to fit logistic regression models with/out interaction term using the WOE variables, and to print the summaries of fitted models, in case WOE_CC_VAR_6 and WOE_CCOL_VAR_1 are main effects.

The resulting summaries in Figure 8.15 show that the interaction term is significant. Its coefficient is 1 as expected and the main effects can be ignored, because their coefficients are very small. Here, we need to draw attention again to the coefficients of the predictors. In case GRP variables are used, a coefficient for each of the design variables is calculated, and because some categories are more influential, magnitudes of their coefficient become larger. On the other hand, in case WOE variables are used, a single coefficient for each predictor is calculated. Hence, in some cases, loss is incurred in terms of discriminatory power, although using WOE variables also has various advantages such as ease of interpretability, preserved monotonicity.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.607*** (0.027)	-3.605*** (0.028)
WOE_CC_VAR_6	-1.061*** (0.076)	-0.00005 (0.113)
WOE_CCOL_VAR_1	-1.011*** (0.046)	-0.00003 (0.095)
Interaction_var_2D		-1.000*** (0.081)
Observations	58,955	58,919
Log Likelihood	-6,882.580	-6,797.068
Akaike Inf. Crit.	13,771.160	13,602.140
Note:	*p<0.05; **p<0.01; ***p<0.001	

Figure 8.15. Summaries of fitted logistic regression models with/out interaction term, in case WOE_CC_VAR_6 and WOE_CCOL_VAR_1 are the main effects.

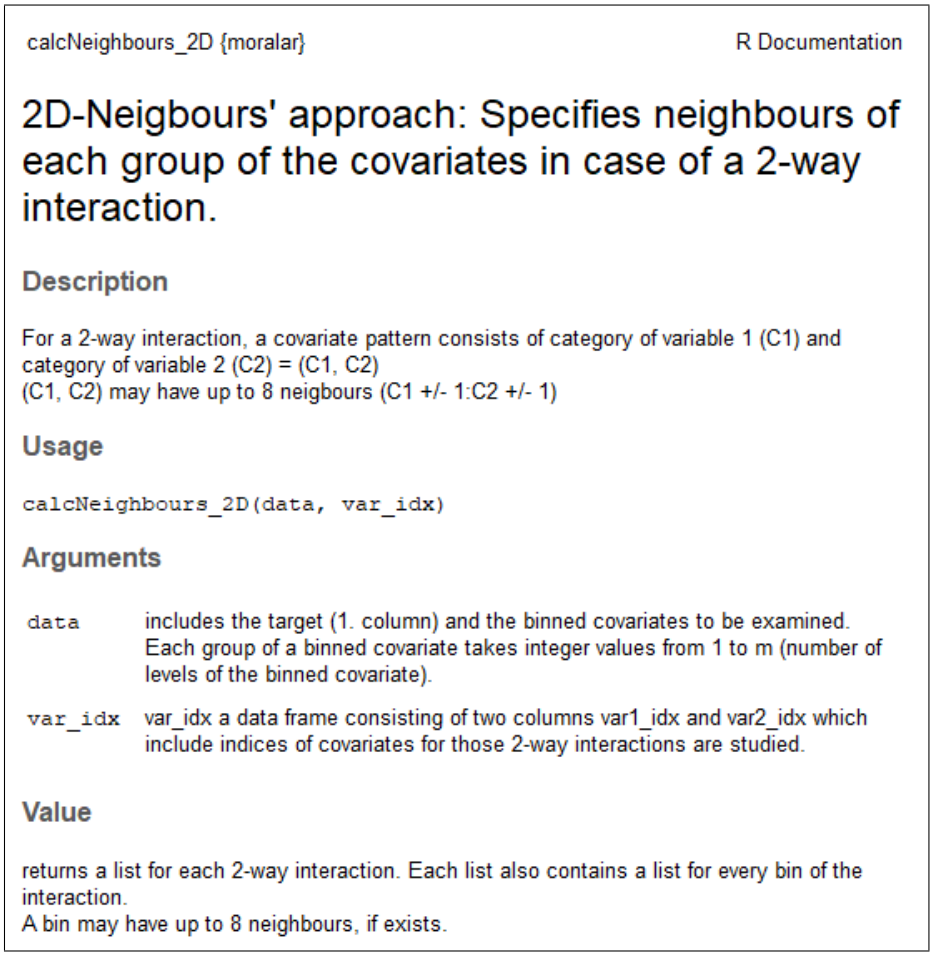
In Figure 8.15, it can be seen that we encounter an issue, that 32 observations, for which WOE values could not be calculated, were excluded from the training data set. This is not a huge problem during model development, yet it may cause a financial loss in real life, because customers that demonstrate a pattern for which a WOE value could not be calculated, would directly be rejected. This issue of uncalculated WOE brings us to an approach we called 2D-Neighbours' approach. For a two-way interaction, a covariate pattern consists of category of variable 1 (C1) and category of variable 2

(C2) = (C1, C2). If this pattern does not include sufficient default and/or non-default records as stated earlier, then we search for its neighbours in two dimensions. Consider a three by three square, it has 9 cells. At the center is the pattern in question, thus it has eight neighbours as given in Table 8.7. The logic depends on the decreasing/increasing order of default ratios of categories of a covariate. Thus, default ratio of a category is between its upper and lower categories.

Table 8.7. Neighbours of a bin with respect to 2D-Neighbours' approach in terms of the category indices of the constructive main effects.

	Variable 1 Category	**Variable 2 Category**
ITSELF	-	-
Neighbour - 1	-	+1
Neighbour - 2	-	-1
Neighbour - 3	+1	+1
Neighbour - 4	+1	-
Neighbour - 5	+1	-1
Neighbour - 6	-1	+1
Neighbour - 7	-1	-
Neighbour - 8	-1	-1

According to Table 8.7, calcNeighbours_2D() function specifies the neighbours of a covariate pattern. A screenshot from its R documentation summarizes the function's usage in Figure 8.16.



calcNeighbours_2D {moralar} R Documentation

2D-Neigbours' approach: Specifies neighbours of each group of the covariates in case of a 2-way interaction.

Description

For a 2-way interaction, a covariate pattern consists of category of variable 1 (C1) and category of variable 2 (C2) = (C1, C2)
(C1, C2) may have up to 8 neigbours (C1 +/- 1:C2 +/- 1)

Usage

```
calcNeighbours_2D(data, var_idx)
```

Arguments

data	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
var_idx	var_idx a data frame consisting of two columns var1_idx and var2_idx which include indices of covariates for those 2-way interactions are studied.

Value

returns a list for each 2-way interaction. Each list also contains a list for every bin of the interaction.
A bin may have up to 8 neighbours, if exists.

Figure 8.16. Screenshot from the R documentation of calcNeighbours_2D() function.

In Figure 8.17, R expression the calculate the neighbours of each covariate pattern for GRP_CC_VAR_6 and GRP_CCOL_VAR_1 is shared. Since the WOE value of the first bin of the interaction term, i.e. observations that take 6 as GRP_CC_VAR_6 and 3 as GRP_CCOL_VAR_1, could not be calculated, its neighbours are displayed in Table 8.8.

```
# Neighbours of each bin of the interaction term are specified.
Neighbours_2D_all_bins ← calcNeighbours_2D(Var_all_GRP_train, var_-
idx = data.frame(var1_idx = c(13), var2_idx = c(14)))

# Neighbours of the first bin are tabulated.
Neighbours_2D_all_bins$‘GRP_CC_VAR_6 X GRP_CCOL_VAR_1‘$‘6 x
3‘
# or
Neighbours_2D_all_bins$‘GRP_CC_VAR_6 X GRP_CCOL_VAR_1‘[[1]]
```

Figure 8.17. R code to specify neighbours of each bin of the interaction term with regards to 2D-Neighbours' approach.

From Table 8.8, it can be seen that not all of the possible neighbours may exist for a covariate pattern. Since the categories of both main effects are the worst categories in terms of default ratio in this bin, neighbours associated with even worse categories are not present for this bin.

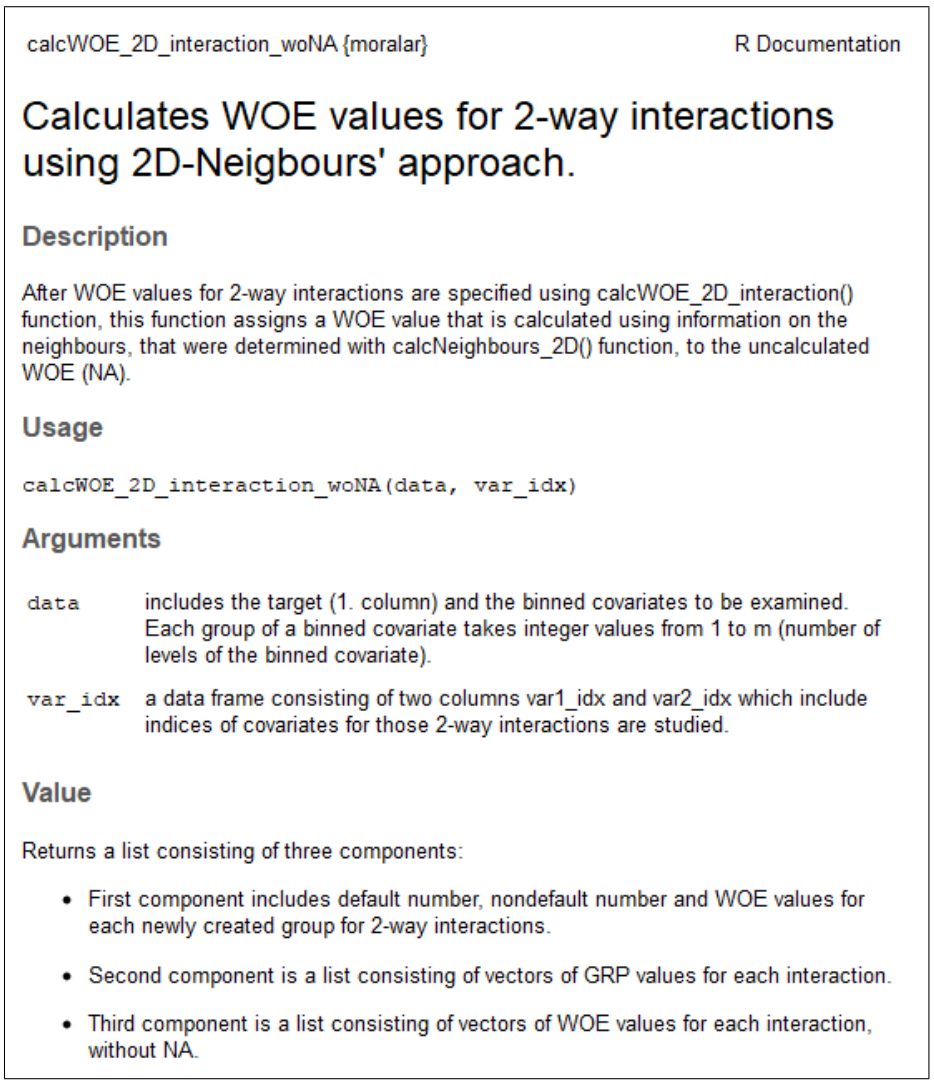
Table 8.8. Neighbours of the first bin of the interaction term, i.e. for the bin where GRP_CC_VAR_6 = 6 and GRP_CCOL_VAR_1 = 3.

	Neighbour - 1	**Neighbour - 2**	**Neighbour - 3**	**Neighbour - 4**
numNonDef	1252	NA	4634	1611
numDef	101	NA	197	119
WOE	-1.0872	NA	-0.4466	-0.9991

	Neighbour - 5	**Neighbour - 6**	**Neighbour - 7**	**Neighbour - 8**
numNonDef	NA	NA	NA	NA
numDef	NA	NA	NA	NA
WOE	NA	NA	NA	NA

Using the 2D-neighbours' approach, calcWOE_2D_interaction() is modified and the resulting calcWOE_2D_interaction_woNA() function addresses the issue of un-calculated WOE value. If a bin does not include any defaults but a reasonable number, 100 as an expert's view, of non-defaults, then we concluded that the number of defaults can be presumed to be 0.7. Otherwise, after specifying the neighbours, if available, the weighted average of WOE values of its neighbours are calculated to impute the missing

WOE value. Screenshot from the R documentation of calcWOE_2D_interaction_-woNA() function is given in Figure 8.18.



calcWOE_2D_interaction_woNA {moralar} R Documentation

Calculates WOE values for 2-way interactions using 2D-Neigbours' approach.

Description

After WOE values for 2-way interactions are specified using calcWOE_2D_interaction() function, this function assigns a WOE value that is calculated using information on the neighbours, that were determined with calcNeighbours_2D() function, to the uncalculated WOE (NA).

Usage

```
calcWOE_2D_interaction_woNA(data, var_idx)
```

Arguments

data	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
var_idx	a data frame consisting of two columns var1_idx and var2_idx which include indices of covariates for those 2-way interactions are studied.

Value

Returns a list consisting of three components:

- First component includes default number, nondefault number and WOE values for each newly created group for 2-way interactions.
- Second component is a list consisting of vectors of GRP values for each interaction.
- Third component is a list consisting of vectors of WOE values for each interaction, without NA.

Figure 8.18. Screenshot from the R documentation of calcWOE_2D_interaction_woNa() function.

In Figure 8.19, we shared the R expression of this function's usage. In Table 8.9, the comparison for the bin in question is tabulated before and after applying the 2D-Neighbours' approach.

```
# Interaction term is obtained using the 2D-Neighbours' approach.
data_WOE_2D_interaction_woNA ← calcWOE_2D_interaction_woNA
(Var_all_GRP_train, var_idx = data.frame(var1_idx = c(13), var2_idx =
c(14)))
# Comparison before and after applying the 2D-Neighbours' approach.
cbind (data_WOE_2D_interaction[[1]]$`GRP_CC_VAR_6 X GRP_-
CCOL_VAR_1` [, 1],
  data_WOE_2D_interaction_woNA[[1]]$`GRP_CC_VAR_6 X GRP_-
CCOL_VAR_1`[, 1])
```

Figure 8.19. R code to calculate WOE version of the two-way interaction term taking account of the 2D-Neighbours' approach, in case GRP_CC_VAR_6 and GRP_CCOL_VAR_1 are the constructive main effects.

Table 8.9. Comparison before and after applying the 2D-Neighbours' approach for the first bin of the interaction term, i.e. for the bin where GRP_CC_VAR_6 = 6 and GRP_CCOL_VAR_1 = 3.

	Actual	**After using 2D-Neighbours' approach**
	6 x 3	6 x 3
numNonDef	32	32
numDef	4	4
WOE	NA	-0.7154411

After obtaining the WOE values by means of calcWOE_2D_interaction_woNA() function easily, the third component of the produced object is renamed in a syntactically valid way and combined with the main effects and the response variable as in Figure 8.20. Subsequently, R expressions in Figure 8.14 are revisited to fit logistic regression models with/out interaction term and print the summaries in an elegant way.

The summaries of fitted logistic regression models with/out interaction term are shared in Figure 8.21 taking account of the 2D-Neighbours' approach, in case WOE_CC_VAR_6 and WOE_CCOL_VAR_1 are the main effects. It can be seen that the coefficient of the interaction variable is no longer one, but very close to one. Coefficients

```
# Response variable, main effects and interaction term are joined together.
data_train_woNA ← cbind(Var_all_WOE_train[, c(1, 13, 14)], data_-
WOE_2D_interaction_woNA[[3]][1])

# Interaction term is renamed.
colnames(data_train_woNA)[4]←"Interaction_var_2D"
```

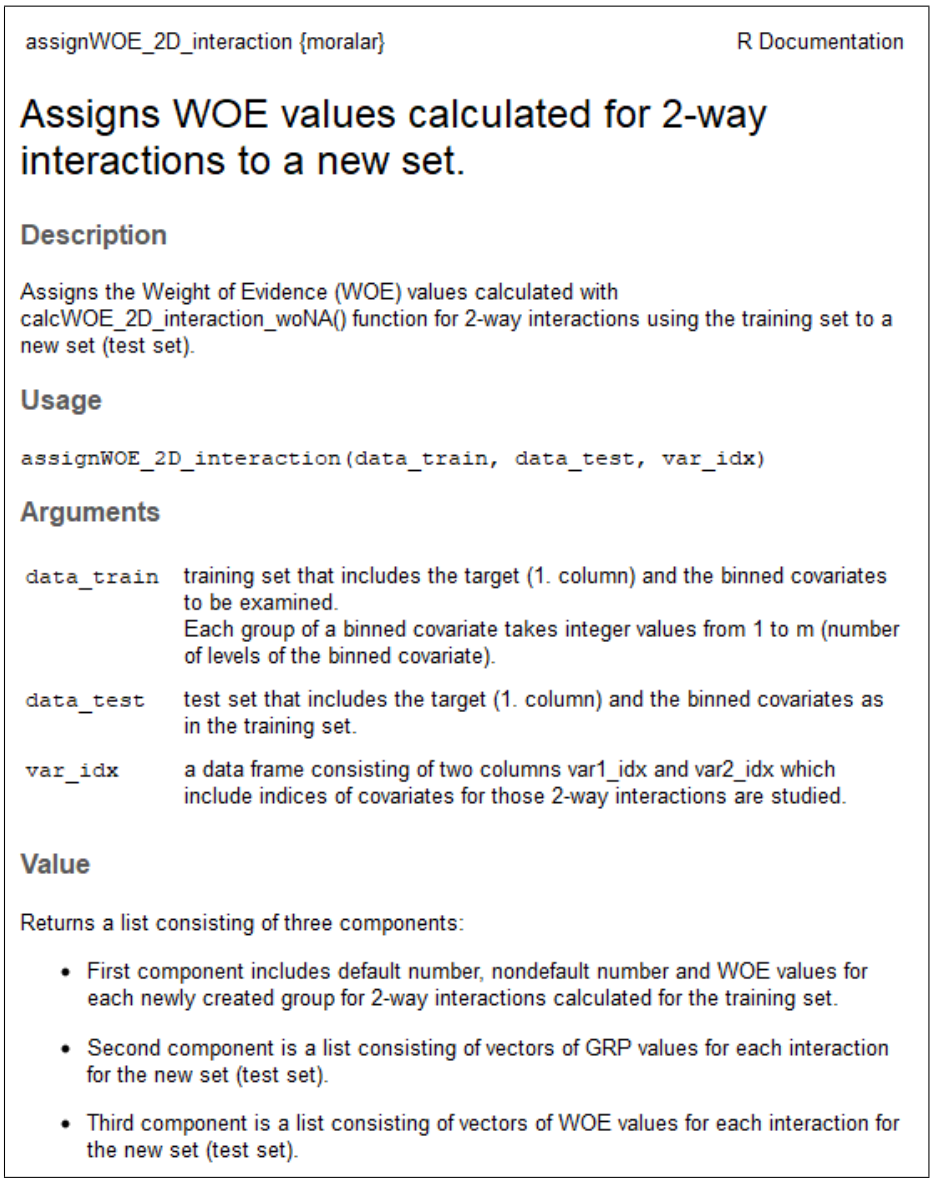
Figure 8.20. R code to prepare the data set with WOE version of the interaction term calculated with calcWOE_2D_interaction_woNA() function besides of constructive main effects and the response variable.

of the main effects are again very small but bigger than the coefficients in Figure 8.15, in case the interaction term is taken into the model. The reason lies in the manipulation with neigbouring approach.

```
=================================================
                        Dependent variable:
                    ---------------------------
                              NPL_IN_12M
                    only Main Effects with Interaction
                          (1)             (2)
-------------------------------------------------
Constant            -3.607*** (0.027) -3.604*** (0.027)
WOE_CC_VAR_6        -1.061*** (0.076)  -0.018 (0.112)
WOE_CCOL_VAR_1      -1.011*** (0.046)  -0.016 (0.094)
Interaction_var_2D                    -0.989*** (0.080)
-------------------------------------------------
Observations            58,955            58,955
Log Likelihood        -6,882.580        -6,810.555
Akaike Inf. Crit.     13,771.160        13,629.110
=================================================
Note:                      *p<0.05; **p<0.01; ***p<0.001
```

Figure 8.21. Summaries of fitted logistic regression models with/out interaction term taking account of the 2D-Neighbours' approach, in case WOE_CC_VAR_6 and WOE_CCOL_VAR_1 are the main effects.

It should be stressed again, that WOE values are calculated from the training data set, and test data set is not used anywhere else except assessing the discriminatory power of the fitted model. Thus, another function is necessary to assign obtained WOE values for two-way interaction to the test set, or any other set for which predictions are needed. We created a function named assignWOE_2D_interaction() for this purpose. Screenshot from the R documentation of this function is presented in Figure 8.22.



assignWOE_2D_interaction {moralar} R Documentation

Assigns WOE values calculated for 2-way interactions to a new set.

Description

Assigns the Weight of Evidence (WOE) values calculated with calcWOE_2D_interaction_woNA() function for 2-way interactions using the training set to a new set (test set).

Usage

```
assignWOE_2D_interaction(data_train, data_test, var_idx)
```

Arguments

data_train	training set that includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
data_test	test set that includes the target (1. column) and the binned covariates as in the training set.
var_idx	a data frame consisting of two columns var1_idx and var2_idx which include indices of covariates for those 2-way interactions are studied.

Value

Returns a list consisting of three components:

- First component includes default number, nondefault number and WOE values for each newly created group for 2-way interactions calculated for the training set.
- Second component is a list consisting of vectors of GRP values for each interaction for the new set (test set).
- Third component is a list consisting of vectors of WOE values for each interaction for the new set (test set).

Figure 8.22. Screenshot from the R documentation of assignWOE_2D_interaction() function.

After assigning the WOE values by means of assignWOE_2D_interaction() function to the test set, the third component of the produced object is renamed in a syntactically valid way and combined with the main effects and the response variable as in Figure 8.23.

```
# Interaction term is obtained for the test split using the 2D-Neighbours' approach.
data_WOE_2D_interaction_woNA_test ← assignWOE_2D_interaction(Var_all_GRP_train, Var_all_GRP_test, var_idx = data.frame(var1_idx = c(13), var2_idx = c(14)))

# Response variable, main effects and interaction term are joined together.
data_test_woNA ← cbind(Var_all_WOE_test[, c(1, 13, 14)], data_WOE_2D_interaction_woNA_test[[3]][1])

# Interaction term is renamed.
colnames(data_test_woNA)[4] ← "Interaction_var_2D"
```

Figure 8.23. Assigning WOE values for the interaction term calculated with calcWOE_2D_interaction_woNA() function to the test split.

In order to assess the contribution of the interaction term in its WOE version with regards to Gini index, we calculated the Gini indices of the fitted model alternatives for both training and test splits. These alternatives are a model with only main effects, a model with main effects and the two-way interaction term besides a model with only the two-way interaction term. Related R code is given in Figure 8.24.

Gini indices calculated in Figure 8.24 are tabulated in Table 8.10 under WOE columns, along with Gini indices obtained using GRP variables, as shared in Table 8.1, under GRP columns. It can be seen that the Gini index of the model is increased by approximately 3.4 when the interaction term is included, whereas the increase for the test split is approximately 3. The amount of increase is evident and also including the interaction term compensates the loss of discriminatory power resulted from using WOE variables instead of GRP variables. Additionally, we can also infer that using only interaction term in WOE version is sufficient, when working with WOE variables.

```
# Gini calculation for training and test splits using roc() from pROC package

# Without Interaction #
pred_train_woI ← predict(mod_fit_2D_woI, type = "response")
round ((roc(data_train_woNA$NPL_IN_12M, pred_train_woI)$auc * 2 - 1)
* 100, 2)
# outputs the Gini index: 36.44

pred_test_woI ← predict(mod_fit_2D_woI, data_test_woNA, type = "re-
sponse")
round ((roc(data_test_woNA$NPL_IN_12M, pred_test_woI)$auc * 2 - 1) *
100, 2)
# outputs the Gini index: 34.01

# With Interaction #
pred_train_wI ← predict(mod_fit_2D_wI, type = "response")
round ((roc(data_train_woNA$NPL_IN_12M, pred_train_wI)$auc * 2 - 1) *
100, 2)
# outputs the Gini index: 39.8

pred_test_wI ← predict(mod_fit_2D_wI, data_test_woNA, type = "re-
sponse")
round ((roc(data_test_woNA$NPL_IN_12M, pred_test_wI)$auc * 2 - 1) *
100, 2)
# outputs the Gini index: 37.07

# Only Interaction #
pred_train_onlyI ← predict(mod_fit_2D_onlyI, type = "response")
round ((roc(data_train_woNA$NPL_IN_12M, pred_train_onlyI)$auc * 2 -
1) * 100, 2)
# outputs the Gini index: 39.79

pred_test_onlyI ← predict(mod_fit_2D_onlyI, data_test_woNA, type = "re-
sponse")
round ((roc(data_test_woNA$NPL_IN_12M, pred_test_onlyI)$auc * 2 - 1)
* 100, 2)
# outputs the Gini index: 37.04
```

Figure 8.24. R code to calculate Gini indices of fitted model alternatives for training and test splits, in case WOE variables are utilized.

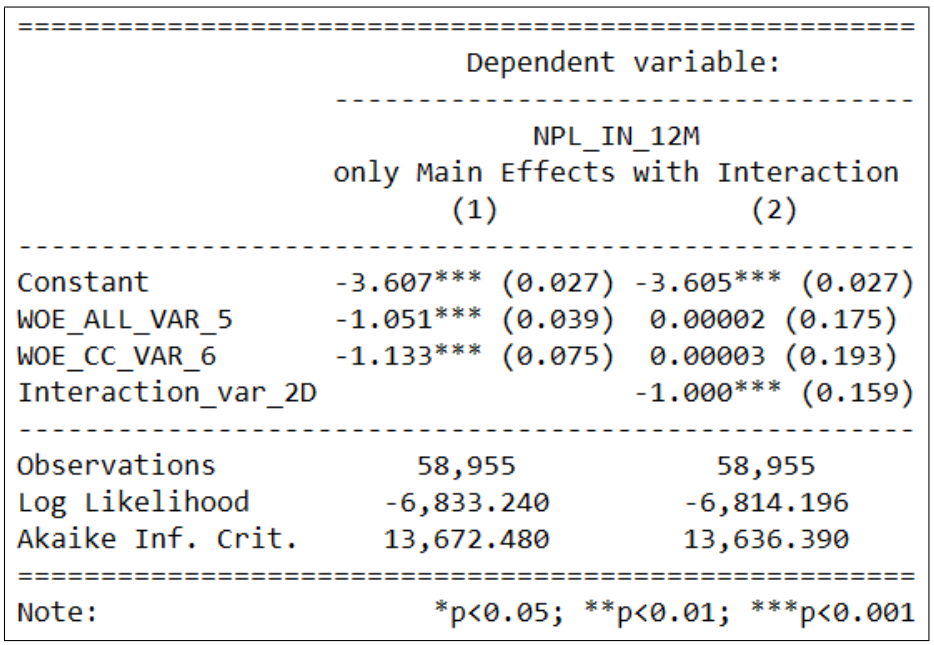
Table 8.10. Gini indices of fitted models with/out interaction term, in case WOE_CC_VAR_6 and WOE_CCOL_VAR_1 are the main effects.

	Without interaction term		**With interaction term**		**Only interaction term**	
	GRP	**WOE**	**GRP**	**WOE**	**GRP**	**WOE**
Training	38.46	36.44	39.82	39.8	-	39.79
Test	36.3	34.01	37.18	37.07	-	37.04

8.2.3. WOE_ALL_VAR_5 * WOE_CC_VAR_6

In Subsection 8.1.2, we showed the impact of the interaction term when included into a regression model, in case GRP_ALL_VAR_5 and GRP_CC_VAR_6 are the main effects. When we want to fit a regression with WOE variables, calcWOE_2D_-interaction_woNA() function is used to obtain WOE version of the two-way interaction term for the training split. Then, R expressions in Figure 8.14 are revisited to fit logistic regression models with/out interaction term and print the summaries in an elegant way. The resulting summaries in Figure 8.25 show that the interaction term is significant. Its coefficient is -1, which is also a sign that WOE values could be calculated for each covariate pattern without utilizing the 2D-Neighbours' approach. Besides, main effects can be ignored, because their coefficients are very small.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits as usual. In Table 8.11, these are presented under WOE columns, along with Gini indices obtained using GRP variables, as shared in Table 8.2, under GRP columns. It can be seen that the Gini indices barely change for the training and test splits, when the interaction terms are included. Additionally, we can, as well, infer that using only interaction term in WOE version is sufficient, when working with WOE variables.



	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.607*** (0.027)	-3.605*** (0.027)
WOE_ALL_VAR_5	-1.051*** (0.039)	0.00002 (0.175)
WOE_CC_VAR_6	-1.133*** (0.075)	0.00003 (0.193)
Interaction_var_2D		-1.000*** (0.159)
Observations	58,955	58,955
Log Likelihood	-6,833.240	-6,814.196
Akaike Inf. Crit.	13,672.480	13,636.390
Note:		*p<0.05; **p<0.01; ***p<0.001

Figure 8.25. Summaries of fitted logistic regression models with/out interaction term, in case WOE_ALL_VAR_5 and WOE_CC_VAR_6 are the main effects.

Table 8.11. Gini indices of fitted models with/out interaction term, in case WOE_ALL_VAR_5 and WOE_CC_VAR_6 are the main effects.

	Without interaction term		**With interaction term**		**Only interaction term**	
	GRP	**WOE**	**GRP**	**WOE**	**GRP**	**WOE**
Training	35.69	36.37	36.58	36.58	-	36.58
Test	36.19	37.28	37.49	37.49	-	37.49

8.2.4. WOE_CC_VAR_2 * WOE_OL_VAR_2

In Subsection 8.1.3, we showed the impact of the interaction term when included into a regression model, in case GRP_CC_VAR_2 and GRP_OL_VAR_2 are the main effects. When we want to fit a regression with WOE variables, calcWOE_2D_interaction_woNA() function is used to obtain WOE version of the two-way interaction term. Then, R expressions in Figure 8.14 are revisited to fit logistic regression models with/out interaction term and print the summaries in an elegant way. The resulting summaries in Figure 8.26 show that the interaction term is significant. Its coefficient is -1, which is also a sign WOE values could be calculated for each covariate pattern

without utilizing the 2D-Neighbours' approach. Besides, main effects can be ignored, because their coefficients are very small.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.603*** (0.027)	-3.605*** (0.027)
WOE_CC_VAR_2	-0.919*** (0.054)	-0.00005 (0.117)
WOE_OL_VAR_2	-0.923*** (0.059)	-0.0001 (0.120)
Interaction_var_2D		-1.000*** (0.112)
Observations	58,955	58,955
Log Likelihood	-6,926.373	-6,884.453
Akaike Inf. Crit.	13,858.750	13,776.910
Note:	*p<0.05; **p<0.01; ***p<0.001	

Figure 8.26. Summaries of fitted logistic regression models with/out interaction term, in case WOE_CC_VAR_2 and WOE_OL_VAR_2 are the main effects.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits as usual. In Table 8.12, these are presented under WOE columns, along with Gini indices obtained using GRP variables, as shared in Table 8.3, under GRP columns. It can be seen that the Gini index of the model is increased by more than 0.8, when the interaction term is included, whereas the Gini index barely changes for the test split. Additionally, we can also infer that using only interaction term in WOE version is sufficient, when working with WOE variables.

Table 8.12. Gini indices of fitted models with/out interaction term, in case WOE_CC_VAR_2 and WOE_OL_VAR_2 are the main effects.

	Without interaction term		**With interaction term**		**Only interaction term**	
	GRP	**WOE**	**GRP**	**WOE**	**GRP**	**WOE**
Training	35.54	35.84	36.68	36.68	-	36.68
Test	38.31	38.31	38.36	38.36	-	38.36

8.3. Three-way Interactions Using GRP Variables

A three-way interaction means that there is a two-way interaction that varies across levels of a third variable. Similar to two-way interactions, but a third variable is also in question. First, logistic regression outputs of two models, with only main variables and also with interaction terms, respectively, are shared. In the tabulated summaries, we can see that the interaction terms are significant. Then, three-way interactions are plotted for each case. A three-way interaction plot depicts an upgraded version of a two-way interaction plot, considering two-way interaction plots are faceted by a third variable. Difference in the slopes (trend of lines) represents the existence of interaction.

Normally, it is hard to find variables whose three-way interaction is significant. Even for 10 variables there are c(10,3) = 240 alternatives. As stated earlier in Section 8.2, it is also not a statistically right approach to examine every possible combination of three elements and find supposed significant three-way interactions. It may cause overfitting the training data set. Thus, selection of prospective interaction effects is mainly based on expert's view. Sample three-way interactions presented in this section may not be as meaningful as how an expert expects or hopes to have, however they are suitable candidates to demonstrate our approach to the problem.

8.3.1. GRP_ALL_VAR_1 * GRP_CHL_VAR_5 * GRP_OL_VAR_1

In this subsection, we examine the presence of a three-way interaction between three main effects, namely GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_-OL_VAR_1. These are the binned versions of the following variables:

(i) ALL_VAR_1: Number of distinct banks applied for any type of loan.
(ii) CHL_VAR_5: Ratio of the sum of the arrears balance to the sum of the total outstanding balance of open cash loans and house loans.
(iii) OL_VAR_1: Worst payment status in overdraft loans.

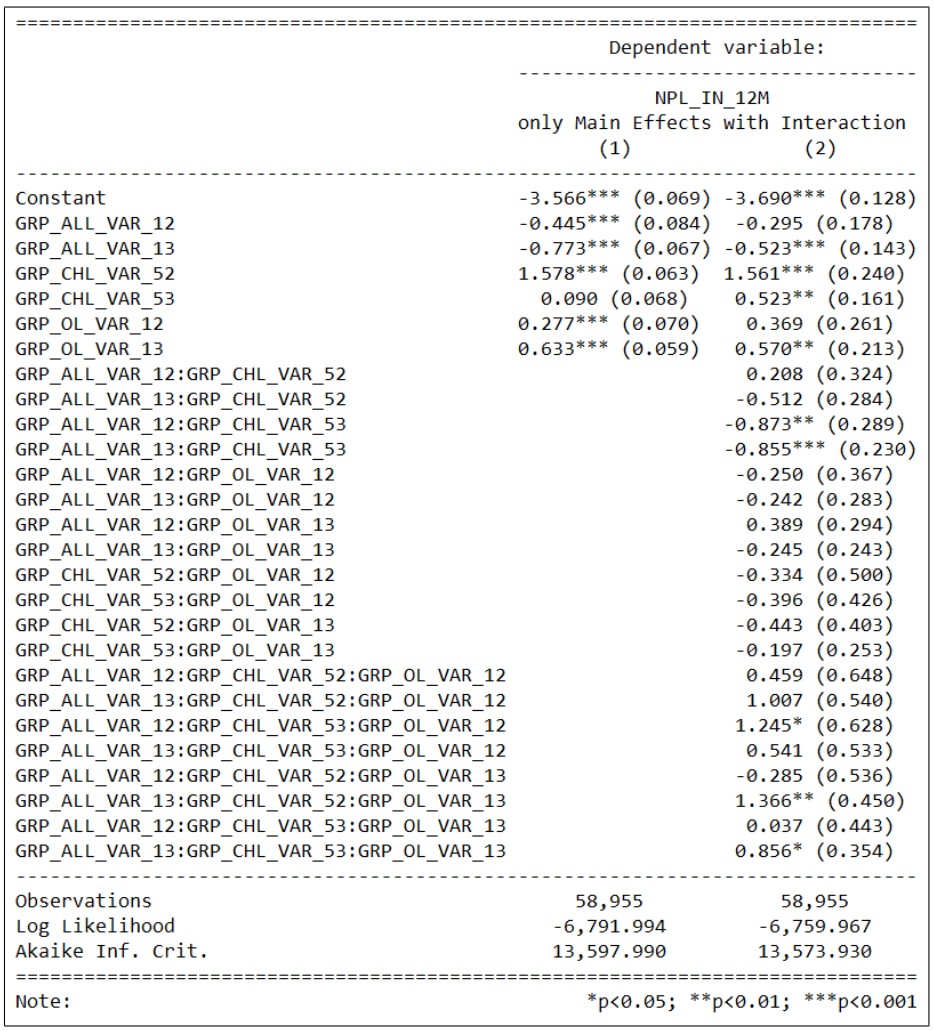
In Figure 8.27, R code to fit two logistic regressions with/out interaction terms is presented. In this section, we use plural, i.e. interaction terms, because there are three two-way interactions and one three-way interaction, in case there are three constructive main effects. As mentioned earlier, in order to include a three-way interaction term to the model, all lower order terms, i.e. main effects and two-way interactions in this case, should also be included into the model. The summaries of the fitted models are tabulated using the stargazer() function of stargazer package [27] in a proper and formatted style.

The summaries of both fitted models with/out interaction terms are presented in Figure 8.28. All main effects are significant, when interaction terms are not included into the model, since at least one of their categories is significant. Three-way interaction is also significant when included into the model. Although two-way interactions "GRP_ALL_VAR_1 : GRP_OL_VAR_1" and "GRP_CHL_VAR_5 : GRP_OL_VAR_1" are not significant, they should be included into the model as lower order terms of three-way interaction in question. Consequently, 26 coefficients and an intercept are used to calculate the predicted probability of default, when categorized variables are used. Of course, this type of calculations is done automatically by computers; however, explaining such an output gets more difficult with increasing number of predictors. That's why WOE variables are preferred in the industry.

```
# Two logistic regression models are fitted using the training split with/out
interaction term.
# Without interaction term
mod_fit_woI ← glm(NPL_IN_12M ∼ GRP_ALL_VAR_1 + GRP_CHL_-
VAR_5 + GRP_OL_VAR_1, data = Var_all_GRP_train, family = bino-
mial(link = "logit"))

# With interaction term
mod_fit_wI ← glm(NPL_IN_12M ∼ GRP_ALL_VAR_1 * GRP_CHL_-
VAR_5 * GRP_OL_VAR_1, data = Var_all_GRP_train, family = bino-
mial(link = "logit"))

# Summary tables for both fitted models
stargazer(mod_fit_woI, mod_fit_wI,
type = "text",
column.labels = c("only Main Effects", "with Interaction"),
intercept.bottom = FALSE,
single.row=TRUE,
notes.append = FALSE,
omit.stat=c("ser"),
star.cutoffs = c(0.05, 0.01, 0.001),
header=FALSE)
```

Figure 8.27. R code to print marginal effects, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the main effects.

Looking at the log likelihood, we can also conclude that the model with interaction terms is better, because we want to maximize the log likelihood, the higher value is better. Besides, we want to minimize the AIC and not surprisingly the model with interaction terms has a smaller AIC value.



	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.566*** (0.069)	-3.690*** (0.128)
GRP_ALL_VAR_12	-0.445*** (0.084)	-0.295 (0.178)
GRP_ALL_VAR_13	-0.773*** (0.067)	-0.523*** (0.143)
GRP_CHL_VAR_52	1.578*** (0.063)	1.561*** (0.240)
GRP_CHL_VAR_53	0.090 (0.068)	0.523** (0.161)
GRP_OL_VAR_12	0.277*** (0.070)	0.369 (0.261)
GRP_OL_VAR_13	0.633*** (0.059)	0.570** (0.213)
GRP_ALL_VAR_12:GRP_CHL_VAR_52		0.208 (0.324)
GRP_ALL_VAR_13:GRP_CHL_VAR_52		-0.512 (0.284)
GRP_ALL_VAR_12:GRP_CHL_VAR_53		-0.873** (0.289)
GRP_ALL_VAR_13:GRP_CHL_VAR_53		-0.855*** (0.230)
GRP_ALL_VAR_12:GRP_OL_VAR_12		-0.250 (0.367)
GRP_ALL_VAR_13:GRP_OL_VAR_12		-0.242 (0.283)
GRP_ALL_VAR_12:GRP_OL_VAR_13		0.389 (0.294)
GRP_ALL_VAR_13:GRP_OL_VAR_13		-0.245 (0.243)
GRP_CHL_VAR_52:GRP_OL_VAR_12		-0.334 (0.500)
GRP_CHL_VAR_53:GRP_OL_VAR_12		-0.396 (0.426)
GRP_CHL_VAR_52:GRP_OL_VAR_13		-0.443 (0.403)
GRP_CHL_VAR_53:GRP_OL_VAR_13		-0.197 (0.253)
GRP_ALL_VAR_12:GRP_CHL_VAR_52:GRP_OL_VAR_12		0.459 (0.648)
GRP_ALL_VAR_13:GRP_CHL_VAR_52:GRP_OL_VAR_12		1.007 (0.540)
GRP_ALL_VAR_12:GRP_CHL_VAR_53:GRP_OL_VAR_12		1.245* (0.628)
GRP_ALL_VAR_13:GRP_CHL_VAR_53:GRP_OL_VAR_12		0.541 (0.533)
GRP_ALL_VAR_12:GRP_CHL_VAR_52:GRP_OL_VAR_13		-0.285 (0.536)
GRP_ALL_VAR_13:GRP_CHL_VAR_52:GRP_OL_VAR_13		1.366** (0.450)
GRP_ALL_VAR_12:GRP_CHL_VAR_53:GRP_OL_VAR_13		0.037 (0.443)
GRP_ALL_VAR_13:GRP_CHL_VAR_53:GRP_OL_VAR_13		0.856* (0.354)
Observations	58,955	58,955
Log Likelihood	-6,791.994	-6,759.967
Akaike Inf. Crit.	13,597.990	13,573.930
Note:		*p<0.05; **p<0.01; ***p<0.001

Figure 8.28. Summaries of fitted logistic regression models with/out interaction terms, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the main effects.

In order to visualize the three-way interaction term with the R code given in Figure 8.29, first, effect() function of effects package [28] is utilized. Then the ggplot() function of ggplot2 package [24] is used for plotting. effect() function calculates the predicted probability of being classified as an NPL in 12 months for each combination of categories of constructive main effects. Additionally, upper and lower boundaries for a 95% confidence interval are obtained with this function.

```
# The Interaction term
Inter.mod_fit_wI ← effect('GRP_ALL_VAR_1 * GRP_CHL_VAR_5 *
GRP_OL_VAR_1', mod_fit_wI, se = TRUE)
Inter.mod_fit_wI.DF ← as.data.frame(Inter.mod_fit_wI)
# Relable them to put them back in order
Inter.mod_fit_wI.DF$GRP_ALL_VAR_1 ← factor(Inter.mod_fit_wI.DF$
GRP_ALL_VAR_1,
level = c("1", "2", "3"),
labels = c("1-2", "3", "4+"))
Inter.mod_fit_wI.DF$GRP_CHL_VAR_5 ← factor(Inter.mod_fit_wI.DF$
GRP_CHL_VAR_5,
level = c("1", "2", "3"),
labels = c("<0.01", "+0.01", "NA"))
Inter.mod_fit_wI.DF$GRP_OL_VAR_1 ← factor(Inter.mod_fit_wI.DF$
GRP_OL_VAR_1,
level = c("1", "2", "3"),
labels = c("0", "1", "2+ or NA"))
# Create plot
ggplot(data = Inter.mod_fit_wI.DF, aes(x = GRP_CHL_VAR_5, y = fit,
color = GRP_OL_VAR_1, group = GRP_OL_VAR_1)) +
geom_point(size = 3.6) + geom_line(size = 1.2) +
geom_ribbon(aes(ymin = lower, ymax = upper, fill = GRP_OL_VAR_1),
alpha = 0.3) +
facet_wrap( ~ GRP_ALL_VAR_1) +
facet_grid(. ~ GRP_ALL_VAR_1, labeller = label_both) +
labs(x = "GRP_CHL_VAR_5",
y = "NPL_IN_12M",
color = "GRP_OL_VAR_1",
fill = "GRP_OL_VAR_1") +
theme_bw() + theme(text = element_text(size = 12),
# plot.title = element_text(hjust = 0.5),
legend.text = element_text(size = 12),
legend.direction = "horizontal",
panel.grid.major = element_blank(),
panel.grid.minor = element_blank(),
legend.position = "top")
```

Figure 8.29. R code to print marginal effects, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the main effects.

In Figure 8.30, one can see plots of two-way interactions between GRP_CHL_VAR_5 and GRP_OL_VAR_1 with regards to levels of GRP_ALL_VAR_1. This can be stated as a plot of two-way interactions facetted by a third variable. Since the lines are not shifted versions of each other for each value of GRP_ALL_VAR_1, we can firstly infer that two-way interaction between GRP_CHL_VAR_5 and GRP_OL_VAR_1 is significant, which we also conclude from the summary of the fitted model.

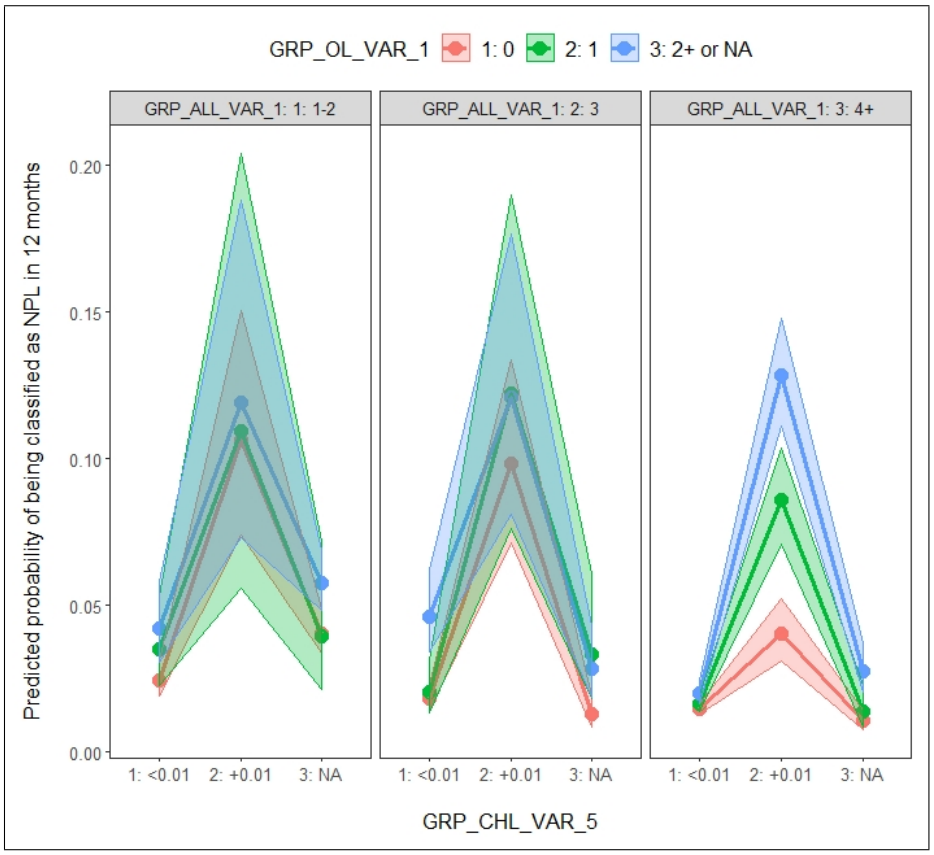


Figure 8.30. Plot of marginal effects, when GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are NPL_IN_12M predictors.

If the number of distinct banks applied for any type of loan is less than 4, i.e. the first two categories of GRP_ALL_VAR_1, upper and lower boundaries move away from each other. Thus, areas of different colors intermingle. If GRP_ALL_VAR_1 takes on category 3, then the two way interaction between GRP_CHL_VAR_5:GRP_OL_VAR_1 becomes clear. If worst payment status in overdraft loans is 2+

or NA, i.e. if the customer has not used any overdraft loan until that date, GRP_ALL_VAR_1 has a very limited influence on the probability of default. This can be induced from the almost identical blue lines in each facet. On the other hand, red lines demonstrate the existence of the three-way interaction.

In order to assess the contribution of the interaction terms with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction terms for both training and test splits as usual. From Table 8.13, it can be seen that the Gini index of the model is increased by more than 1.4, when the interaction terms are included, whereas the increase for the test split is very limited, just under 0.4.

Table 8.13. Gini indices of fitted models with/out interaction term, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the main effects.

	Without interaction term	**With interaction term**
Training	36.49	37.92
Test	39.14	39.53

8.3.2. GRP_OL_VAR_1 * GRP_OL_VAR_2 * GRP_OL_VAR_3

In this subsection, we examine the presence of an interaction between three main effects, namely GRP_OL_VAR_1, GRP_OL_VAR_2 and GRP_OL_VAR_3. These are the binned versions of the following variables:

(i) OL_VAR_1: Worst payment status in overdraft loans.
(ii) OL_VAR_2: Age of the oldest overdraft loan excluding the bad-closed ones.
(iii) OL_VAR_3: Number of open overdraft loans whose limit usage is greater than 100%.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.410*** (0.056)	-3.148*** (0.070)
GRP_OL_VAR_12	0.326*** (0.071)	-0.177 (0.272)
GRP_OL_VAR_13	0.623*** (0.060)	0.159 (0.099)
GRP_OL_VAR_22	-0.355*** (0.079)	-0.696*** (0.115)
GRP_OL_VAR_23	-0.638*** (0.081)	-0.872*** (0.123)
GRP_OL_VAR_24	-0.843*** (0.082)	-1.221*** (0.138)
GRP_OL_VAR_25	-1.259*** (0.079)	-1.667*** (0.135)
GRP_OL_VAR_32	1.088*** (0.064)	0.678*** (0.192)
GRP_OL_VAR_12:GRP_OL_VAR_22		0.810* (0.320)
GRP_OL_VAR_13:GRP_OL_VAR_22		0.568** (0.215)
GRP_OL_VAR_12:GRP_OL_VAR_23		0.354 (0.333)
GRP_OL_VAR_13:GRP_OL_VAR_23		0.742*** (0.197)
GRP_OL_VAR_12:GRP_OL_VAR_24		0.754* (0.330)
GRP_OL_VAR_13:GRP_OL_VAR_24		0.513* (0.220)
GRP_OL_VAR_12:GRP_OL_VAR_25		0.590 (0.326)
GRP_OL_VAR_13:GRP_OL_VAR_25		0.797*** (0.205)
GRP_OL_VAR_12:GRP_OL_VAR_32		1.113** (0.401)
GRP_OL_VAR_13:GRP_OL_VAR_32		0.553 (0.346)
GRP_OL_VAR_22:GRP_OL_VAR_32		0.181 (0.317)
GRP_OL_VAR_23:GRP_OL_VAR_32		0.019 (0.376)
GRP_OL_VAR_24:GRP_OL_VAR_32		0.563 (0.362)
GRP_OL_VAR_25:GRP_OL_VAR_32		0.102 (0.421)
GRP_OL_VAR_12:GRP_OL_VAR_22:GRP_OL_VAR_32		-1.219* (0.542)
GRP_OL_VAR_13:GRP_OL_VAR_22:GRP_OL_VAR_32		-0.044 (0.498)
GRP_OL_VAR_12:GRP_OL_VAR_23:GRP_OL_VAR_32		-0.792 (0.588)
GRP_OL_VAR_13:GRP_OL_VAR_23:GRP_OL_VAR_32		-0.600 (0.534)
GRP_OL_VAR_12:GRP_OL_VAR_24:GRP_OL_VAR_32		-1.395* (0.563)
GRP_OL_VAR_13:GRP_OL_VAR_24:GRP_OL_VAR_32		-0.477 (0.518)
GRP_OL_VAR_12:GRP_OL_VAR_25:GRP_OL_VAR_32		-0.928 (0.597)
GRP_OL_VAR_13:GRP_OL_VAR_25:GRP_OL_VAR_32		-0.060 (0.547)
Observations	58,955	58,955
Log Likelihood	-6,862.501	-6,833.343
Akaike Inf. Crit.	13,741.000	13,726.690
Note:		*p<0.05; **p<0.01; ***p<0.001

Figure 8.31. Summaries of fitted logistic regression models with/out interaction terms, in case GRP_OL_VAR_1, GRP_OL_VAR_2 and GRP_OL_VAR_3 are the main effects.

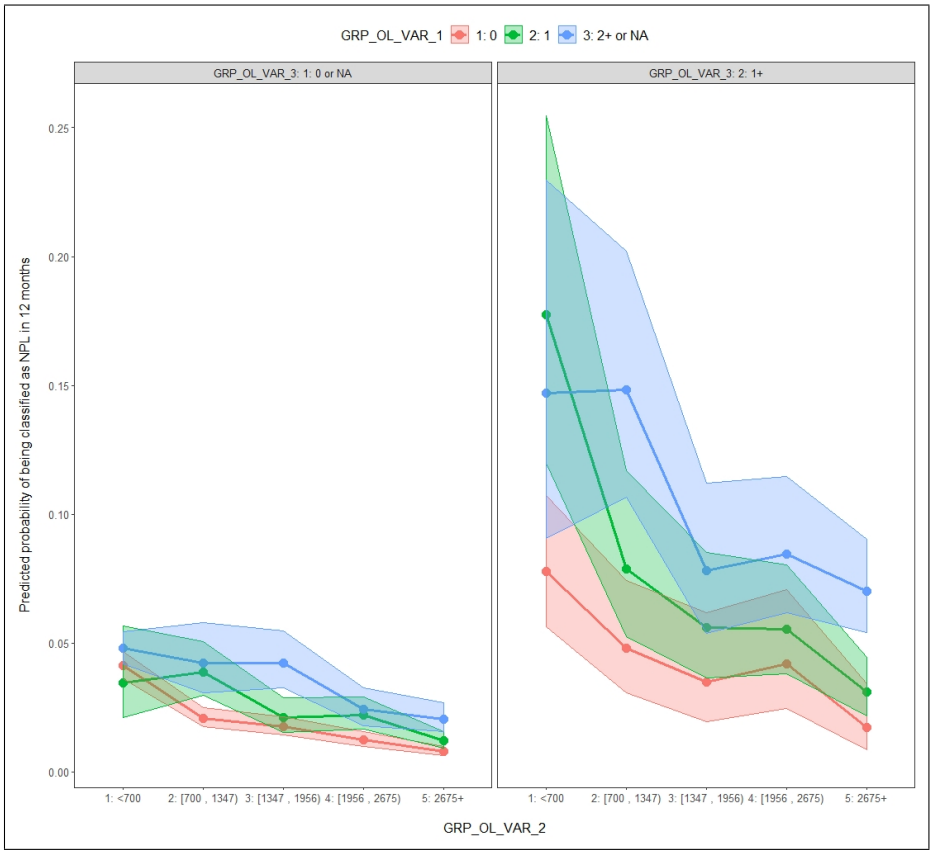


Figure 8.32. Plot of marginal effects, when GRP_OL_VAR_1, GRP_OL_VAR_2 and GRP_OL_VAR_3 are NPL_IN_12M predictors.

In order to assess the contribution of the interaction terms with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction terms for both training and test splits as usual. From Table 8.14, it can be seen that the Gini index of the model is moderately increased by just above 0.6, when the interaction terms are included, whereas the Gini index barely changes for the test split.

Table 8.14. Gini indices of fitted models with/out interaction term, in case GRP_OL_VAR_1, GRP_OL_VAR_2 and GRP_OL_VAR_3 are the main effects.

	Without interaction term	**With interaction term**
Training	37.16	37.79
Test	40.84	40.92

8.3.3. GRP_ALL_VAR_4 * GRP_CCOL_VAR_1 * GRP_CHL_VAR_5

In this subsection, we examine the presence of an interaction between three main effects, namely GRP_ALL_VAR_4, GRP_CCOL_VAR_1 and GRP_CHL_VAR_5. These are the binned versions of the following variables:

- ALL_VAR_4: Ratio of the sum of the arrears balance to the sum of the total outstanding balance of open loans of any type.
- CCOL_VAR_1: Number of distinct banks that granted credit cards or overdraft loans whose limit usage is greater than 80%.
- CHL_VAR_5: Ratio of the sum of the arrears balance to the sum of the total outstanding balance of open cash loans and house loans.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-4.537*** (0.057)	-4.412*** (0.067)
GRP_ALL_VAR_42	0.715*** (0.067)	-1.265 (1.000)
GRP_CCOL_VAR_12	0.696*** (0.062)	0.612*** (0.086)
GRP_CCOL_VAR_13	1.322*** (0.077)	1.102*** (0.114)
GRP_CHL_VAR_52	0.992*** (0.077)	0.932*** (0.262)
GRP_CHL_VAR_53	0.381*** (0.068)	-0.118 (0.137)
GRP_ALL_VAR_42:GRP_CCOL_VAR_12		1.235 (1.029)
GRP_ALL_VAR_42:GRP_CCOL_VAR_13		1.792 (1.032)
GRP_ALL_VAR_42:GRP_CHL_VAR_52		1.395 (1.053)
GRP_ALL_VAR_42:GRP_CHL_VAR_53		2.600* (1.012)
GRP_CCOL_VAR_12:GRP_CHL_VAR_52		-0.351 (0.311)
GRP_CCOL_VAR_13:GRP_CHL_VAR_52		-0.026 (0.328)
GRP_CCOL_VAR_12:GRP_CHL_VAR_53		0.669*** (0.170)
GRP_CCOL_VAR_13:GRP_CHL_VAR_53		0.454 (0.305)
GRP_ALL_VAR_42:GRP_CCOL_VAR_12:GRP_CHL_VAR_52		-0.234 (1.096)
GRP_ALL_VAR_42:GRP_CCOL_VAR_13:GRP_CHL_VAR_52		-0.770 (1.101)
GRP_ALL_VAR_42:GRP_CCOL_VAR_12:GRP_CHL_VAR_53		-2.426* (1.060)
GRP_ALL_VAR_42:GRP_CCOL_VAR_13:GRP_CHL_VAR_53		-2.670* (1.112)
Observations	58,955	58,955
Log Likelihood	-6,704.212	-6,665.457
Akaike Inf. Crit.	13,420.420	13,366.910
Note:		*p<0.05; **p<0.01; ***p<0.001

Figure 8.33. Summaries of fitted logistic regression models with/out interaction term, in case GRP_ALL_VAR_4, GRP_CHL_VAR_5 and GRP_CCOL_VAR_1 are the main effects.

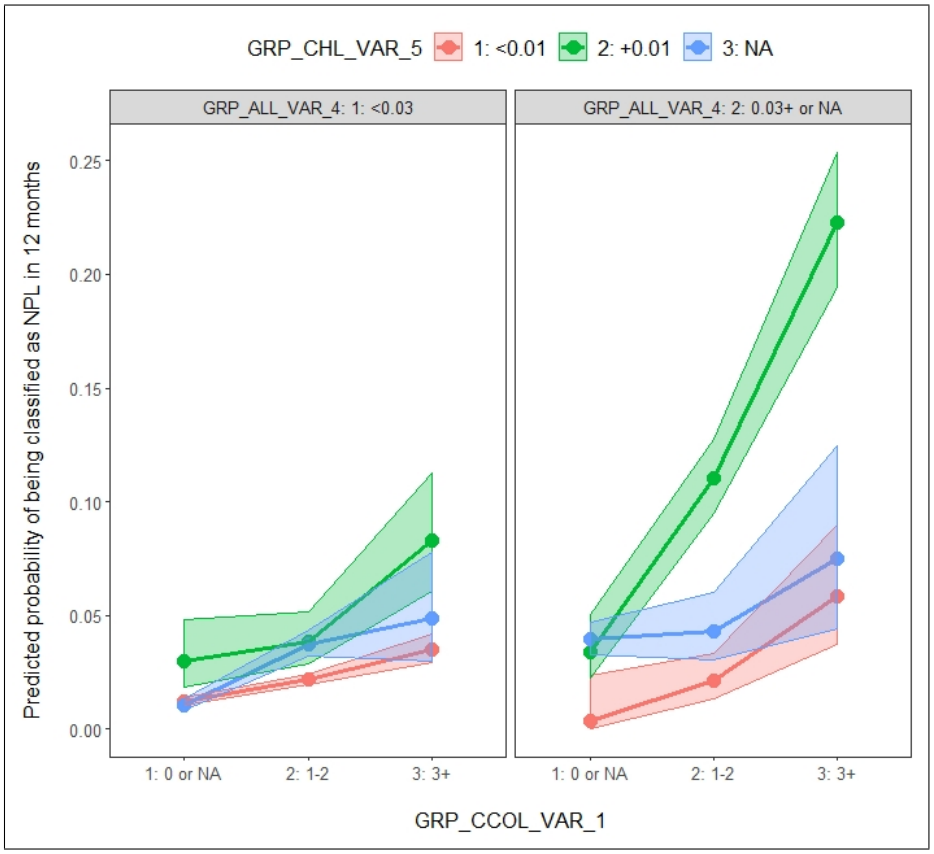


Figure 8.34. Plot of marginal effects, when GRP_ALL_VAR_4, GRP_CCOL_VAR_1 and GRP_CHL_VAR_5 are NPL_IN_12M predictors.

In order to assess the contribution of the interaction terms with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction terms for both training and test splits as usual. From Table 8.15, it can be seen that the Gini index of the model is increased by just around 1.3, when the interaction terms are included, whereas a slight decrease in the Gini index, less than 0.3, is observed for the test split.

Table 8.15. Gini indices of fitted models with/out interaction term, in case GRP_ALL_VAR_4, GRP_CCOL_VAR_1 and GRP_CHL_VAR_5 are the main effects.

	Without interaction term	**With interaction term**
Training	39.01	40.29
Test	38.87	38.62

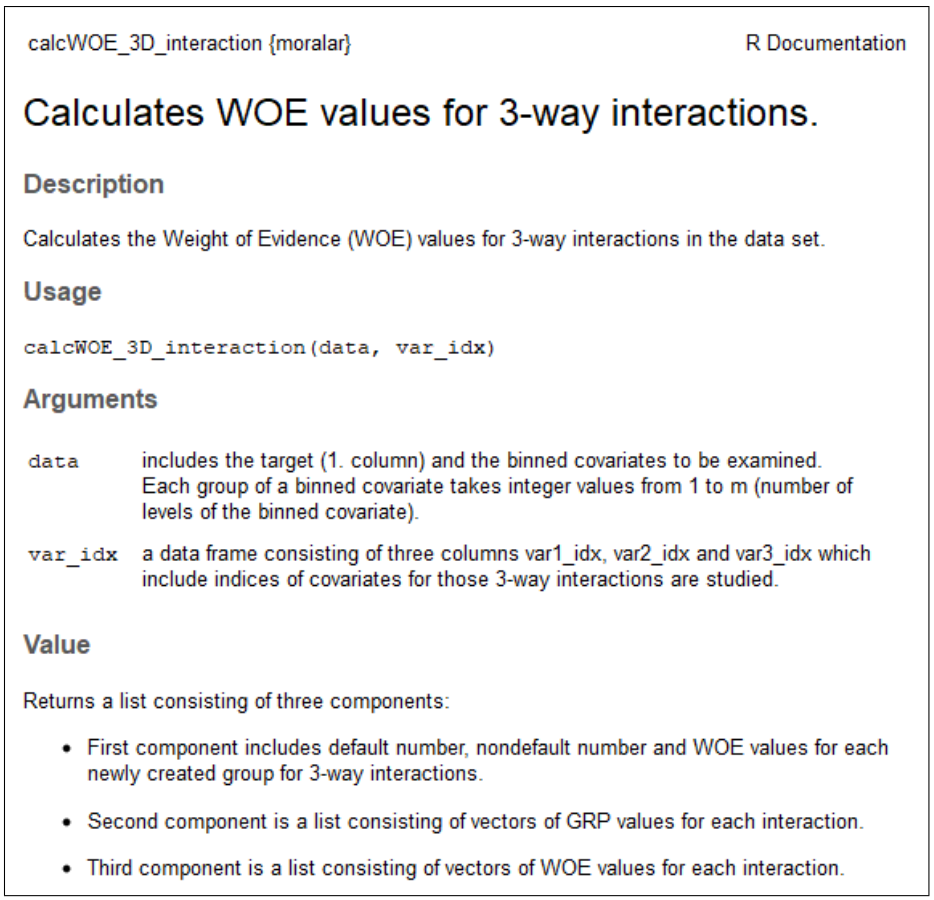
8.4. Three-way Interactions Using WOE Variables

In Section 8.3, we showed how we can make use of three-way interaction terms, in case the constructive main effects are binned, GRP, variables. When we want to proceed with WOE variables, calculating WOE values for interaction terms is an issue to be solved. In this section, we share how we calculate the WOE values for three-way interactions, overcome the problems we faced and also present their contribution to the models with a similar approach that we followed in Section 8.2. Thus, this section may be regarded as an upgraded version of Section 8.2. In Subsection 8.2.1, we described our approach to obtain WOE versions of interaction terms specifically for two-way interactions, and in Subsection 8.2.2, we demonstrated details of the approach by implementing necessary functions. In Subsection 8.4.1, one can find details of the extended approach explicitly for three-way interactions like expanding the 2D-neighbours' approach into 3D-neighbours' approach. As the number of covariates is increased by one, neighbours' approach is shifted to third dimension, from a 3x3 square to a 3x3x3 cube.

8.4.1. WOE_ALL_VAR_1 * WOE_CHL_VAR_5 * WOE_OL_VAR_1

In Subsection 8.3.1, we showed how the interaction terms can be included into a regression model, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_-VAR_1 are the main effects. When we want to fit regression with WOE variables, each combination of main effects should be taken into consideration. Since each of the constructive main effects have three categories, we need to calculate WOE values

for newly obtained categories, when we want to utilize WOE variables. calcWOE_-3D_interaction() function calculates the WOE values easily. This function produces three components that are used during the scorecard model development process. A screenshot of the R documentation for this function is given in Figure 8.35.



calcWOE_3D_interaction {moralar} R Documentation

Calculates WOE values for 3-way interactions.

Description

Calculates the Weight of Evidence (WOE) values for 3-way interactions in the data set.

Usage

```
calcWOE_3D_interaction(data, var_idx)
```

Arguments

data	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
var_idx	a data frame consisting of three columns var1_idx, var2_idx and var3_idx which include indices of covariates for those 3-way interactions are studied.

Value

Returns a list consisting of three components:

- First component includes default number, nondefault number and WOE values for each newly created group for 3-way interactions.
- Second component is a list consisting of vectors of GRP values for each interaction.
- Third component is a list consisting of vectors of WOE values for each interaction.

Figure 8.35. Screenshot from the R documentation of calcWOE_3D_interaction() function.

In Figure 8.36, one can see the usage of this function. First, the indices of main effects are specified. Afterwards, these indices are stored in a data frame as a parameter for the function. Here, it should be stressed that the WOE values are calculated using the training split. The object generated by the function is a list that includes three components.

```
grep("GRP_ALL_VAR_1", colnames(Var_all_GRP_train))
# outputs the index of GRP_ALL_VAR_1 in the data set: 2

grep("GRP_CHL_VAR_5", colnames(Var_all_GRP_train))
# outputs the index of GRP_CHL_VAR_5 in the data set: 21

grep("GRP_OL_VAR_1", colnames(Var_all_GRP_train))
# outputs the index of GRP_OL_VAR_1 in the data set: 27

# Interaction term is obtained.
data_WOE_3D_interaction ← calcWOE_3D_interaction(Var_all_GRP_-
train, var_idx = data.frame(var1_idx = c(2), var2_idx = c(21), var3_idx =
c(27)))

# The first component of the object is printed, that is, nondefault number,
default number and WOE value of each bin of the interaction term is tabulated.
data_WOE_3D_interaction[[1]]$‘GRP_ALL_VAR_1 X GRP_CHL_VAR_-
5 X GRP_OL_VAR_1‘
```

Figure 8.36. R code to calculate WOE version of the three-way interaction term, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects.

The first component is given in Table 8.16, that includes nondefault number, default number and WOE values for each combination. It is easier to explain the newly obtained categories by going over Table 8.16. calcWOE_3D_interaction() utilizes the orderAttributes() function introduced in Chapter 7, as well, so that the bins of each main effect are sorted in a decreasing order of the proportion of events. Subsequently, firstly the worst group of the first variable, i.e. category 1 of GRP_ALL_VAR_1, is divided into bins according to the values the second variable takes. Subsequently, the bin which includes observations that take the worst group values for the first two variables, i.e. category 1 of GRP_ALL_VAR_1 and category 2 of GRP_CHL_-VAR_5, is partitioned into bins with respect to the value the third variable takes on. Consequently, the resulting first bin includes the observations which take on the worst group values of all three variables. According to the ordered categories of the first variable, second variable and third variable, respectively, remaining bins are created. Likewise, the very last bin includes the observations which take on the best group value

of each variable. In order to calculate WOE value of a bin, we determined the rule that there should be at least 100 observations in that bin, and at least one sample from each label, i.e. nondefault and default in this case, should be represented.

Table 8.16. First component of the object returned by calcWOE_3D_interaction() function, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects.

	1 x 2 x 3	1 x 2 x 2	1 x 2 x 1	1 x 3 x 3	1 x 3 x 2	1 x 3 x 1
numNonDef	111	65	227	1961	244	2564
numDef	15	8	27	120	10	108
WOE	-1.6031	NA	-1.4755	-0.8109	-0.41	-0.4374

	1 x 1 x 3	1 x 1 x 2	1 x 1 x 1	2 x 2 x 3	2 x 2 x 2	2 x 2 x 1
numNonDef	815	554	2523	160	115	321
numDef	36	20	63	22	16	35
WOE	-0.4849	-0.2832	0.0855	-1.6205	-1.6323	-1.3885

	2 x 3 x 3	2 x 3 x 2	2 x 3 x 1	2 x 1 x 3	2 x 1 x 2	2 x 1 x 1
numNonDef	721	290	1832	845	955	3550
numDef	21	10	24	41	20	66
WOE	-0.0685	-0.2373	0.7305	-0.5788	0.2614	0.3804

	3 x 2 x 3	3 x 2 x 2	3 x 2 x 1	3 x 3 x 3	3 x 3 x 2	3 x 3 x 1
numNonDef	1127	1042	1301	1442	1076	4141
numDef	166	98	55	41	15	44
WOE	-1.6893	-1.2407	-0.4411	-0.0444	0.6683	0.9399

	3 x 1 x 3	3 x 1 x 2	3 x 1 x 1
numNonDef	5122	7676	16614
numDef	105	129	246
WOE	0.2827	0.4814	0.6081

The second component of the object includes the GRP values of the newly created interaction term. In Table 8.16, it can be seen that the first component is a matrix that has 27 (3x3x3) columns. Each observation takes a proper GRP value according to its column index. In Figure 8.37, R code to print the GRP values of the interaction term for the first ten observations is shared. The output of this expression is given in Table 8.17.

```
# The second component, that includes GRP values of the interaction term, of
the object is printed for the first ten observations.
head(data_WOE_3D_interaction[[2]]$`GRP_ALL_VAR_1 X GRP_CHL_-
VAR_5 X GRP_OL_VAR_1`, 10)
```

Figure 8.37. R expression to print the second component of the returned object by calcWOE_3D_interaction() function, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects.

Table 8.17. GRP values of the interaction term for the first 10 observations, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects.

	GRP
1	27
2	9
3	27
4	19
5	13
6	19
7	6
8	7
9	8
10	18

The third component of the function includes the WOE values of the newly created interaction term. For each GRP value, i.e. column index of the matrix in the first component, the value in the third row, namely WOE value, is assigned. In Figure 8.38, R code to print the WOE values of the interaction term for the first ten observations is shared. The output of this expression is given in Table 8.18.

```
# The third component, that includes WOE values of the interaction term, of
the object is printed for the first ten observations.
head(data_WOE_3D_interaction[[3]]$`GRP_ALL_VAR_1 X GRP_CHL_-
VAR_5 X GRP_OL_VAR_1`, 10)
```

Figure 8.38. R expression to print the third component of the returned object by calcWOE_3D_interaction() function, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects.

Table 8.18. WOE values of the interaction term for the first 10 observations, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects.

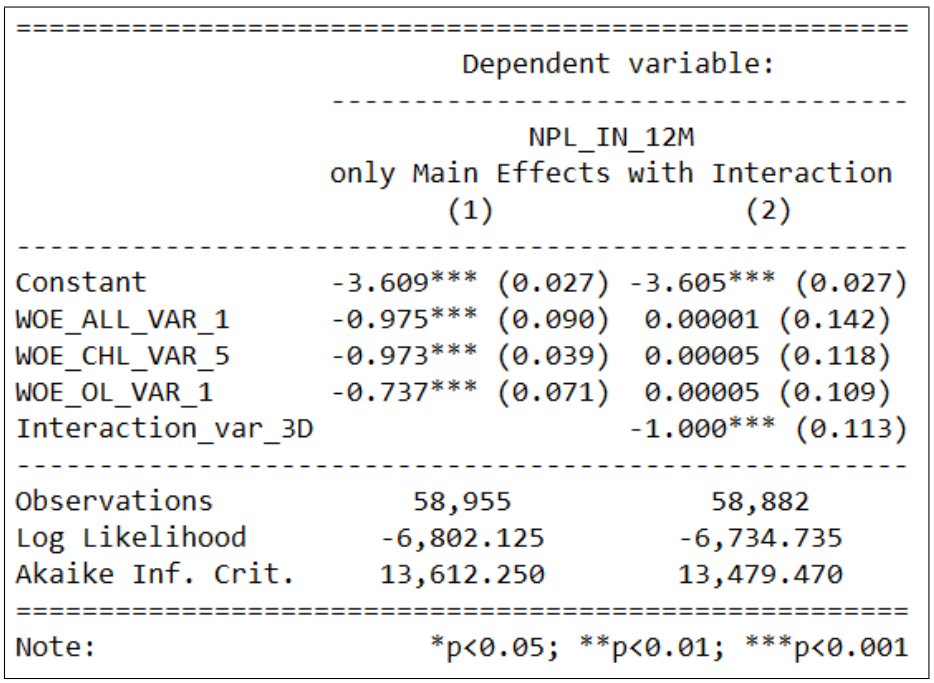
	WOE
1	0.6081
2	0.0855
3	0.6081
4	-1.6893
5	-0.0685
6	-1.6893
7	-0.4374
8	-0.4849
9	-0.2832
10	0.3804

After obtaining the WOE values by means of calcWOE_3D_interaction() easily, the third component of the produced object is first renamed in a syntactically valid way and then combined with the main effects and the response variable. In Figure 8.39, one can see the R expressions to fit logistic regression models with/out interaction term and print the summaries in an elegant way.

```
# Response variable, main effects and interaction term are joined together.
data_train ← cbind(Var_all_WOE_train[, c(1, 2, 21, 27)], data_WOE_3D_-
interaction[[3]][1])
length(data_train[is.na(data_train[, 5]) == 1, 5])
# There are 73 records taking NA
# Interaction term is renamed.
colnames(data_train)[5] ← "Interaction_var_3D"
# Logistic regression models are fitted using the training split for two cases:
# Without interaction term
mod_fit_3D_woI ← glm(NPL_IN_12M ∼ ., data = data_train[,-c(5)], family
= binomial(link = "logit"))
# With interaction term
mod_fit_3D_wI ← glm(NPL_IN_12M ∼ ., data = data_train, family = bi-
nomial(link = "logit"))
# Summary tables for both fitted models
stargazer(mod_fit_3D_woI, mod_fit_3D_wI,
type = "text",
column.labels = c("only Main Effects", "with Interaction"),
intercept.bottom = FALSE,
single.row=TRUE,
notes.append = FALSE,
omit.stat=c("ser"),
star.cutoffs = c(0.05, 0.01, 0.001),
header=FALSE)
```

Figure 8.39. R code to fit logistic regression models with/out interaction term using the WOE variables, and to print the summaries of fitted models, if main effects are WOE_ALL_VAR_1, WOE_CHL_VAR_5 and WOE_OL_VAR_1.

The resulting summaries in Figure 8.40 show that the interaction term is significant. Its coefficient is -1 as expected and the main effects can be ignored, because their coefficients are very small. However, we encounter an issue, that 73 observations, for which WOE values could not be calculated, were excluded from the training data set. This is not a huge problem during model development, yet it could cause a financial loss in real life, because customers that demonstrate a pattern for which a WOE value could not be calculated, would directly be rejected.



	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.609*** (0.027)	-3.605*** (0.027)
WOE_ALL_VAR_1	-0.975*** (0.090)	0.00001 (0.142)
WOE_CHL_VAR_5	-0.973*** (0.039)	0.00005 (0.118)
WOE_OL_VAR_1	-0.737*** (0.071)	0.00005 (0.109)
Interaction_var_3D		-1.000*** (0.113)
Observations	58,955	58,882
Log Likelihood	-6,802.125	-6,734.735
Akaike Inf. Crit.	13,612.250	13,479.470
Note:	*p<0.05; **p<0.01; ***p<0.001	

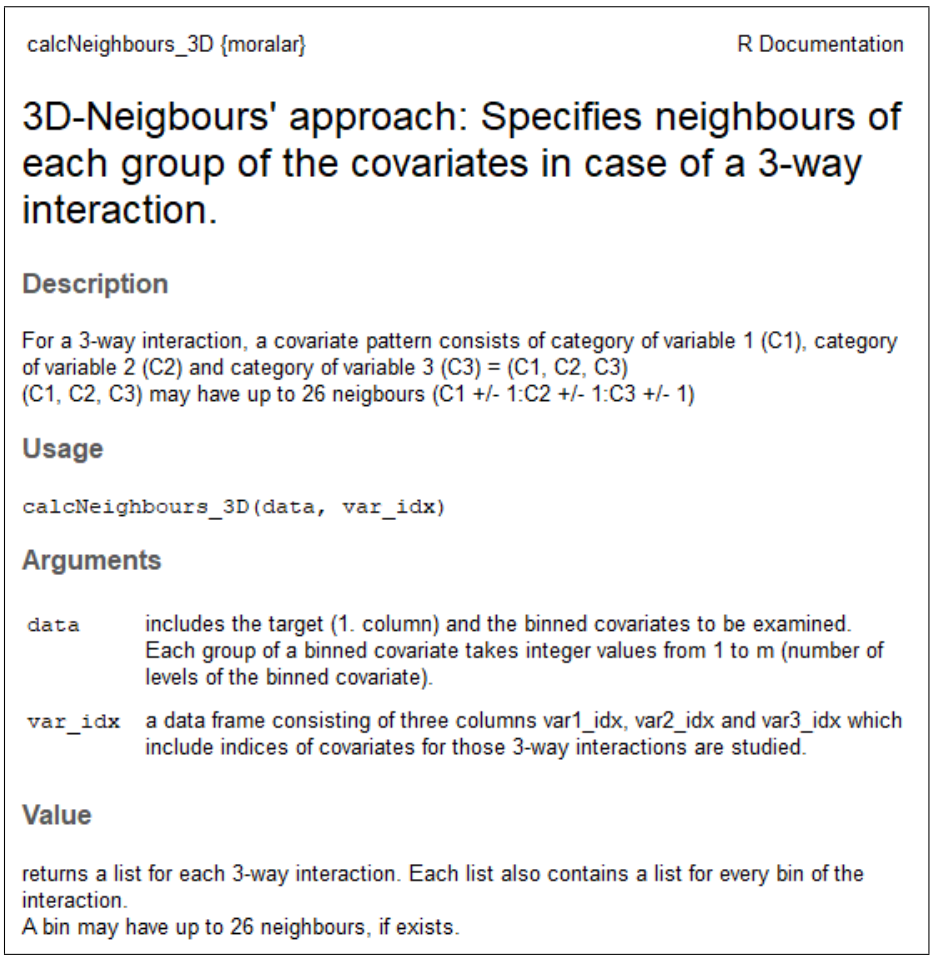
Figure 8.40. Summaries of fitted logistic regression models with/out interaction term, in case WOE_ALL_VAR_1, WOE_CHL_VAR_5 and WOE_OL_VAR_1 are the main effects.

Especially for three-way interaction variables, it is rather possible that some configurations, i.e. some combinations of categories of variables, do not have sufficient observations, whether default or non-default, in order to calculate WOE values. This causes ignorance of some covariate patterns when fitting the regression. Hence, we suggest a way, namely 3D-Neighbours' approach, that can be utilized to deal with this issue. For a three-way interaction, a covariate pattern consists of category of variable 1 (C1), category of variable 2 (C2) and category of variable 3 (C3) = (C1, C2, C3). If this pattern does not include sufficient, where we specified the sufficient number as 100 as an expert's view, default and/or non-default records, then we search for its neighbours in three dimensions. Consider a three by three cube, it has 27 cells. At the center is the pattern in question, thus it has 26 neighbours that are demonstrated in Table 8.19. The logic depends on the decreasing/increasing order of default ratios of categories of a covariate. So, default ratio in a category is between in its upper and lower categories.

Table 8.19. Neighbours of a bin with respect to 3D-Neighbours' approach in terms of the categories of the constructive main effects.

	Variable 1 Category	**Variable 2 Category**	**Variable 3 Category**
ITSELF	-	-	-
Neighbour - 1	-	-	+1
Neighbour - 2	-	-	-1
Neighbour - 3	-	+1	+1
Neighbour - 4	-	+1	-
Neighbour - 5	-	+1	-1
Neighbour - 6	-	-1	+1
Neighbour - 7	-	-1	-
Neighbour - 8	-	-1	-1
Neighbour - 9	+1	-	+1
Neighbour - 10	+1	-	-
Neighbour - 11	+1	-	-1
Neighbour - 12	+1	+1	+1
Neighbour - 13	+1	+1	-
Neighbour - 14	+1	+1	-1
Neighbour - 15	+1	-1	+1
Neighbour - 16	+1	-1	-
Neighbour - 17	+1	-1	-1
Neighbour - 18	-1	-	+1
Neighbour - 19	-1	-	-
Neighbour - 20	-1	-	-1
Neighbour - 21	-1	+1	+1
Neighbour - 22	-1	+1	-
Neighbour - 23	-1	+1	-1
Neighbour - 24	-1	-1	+1
Neighbour - 25	-1	-1	-
Neighbour - 26	-1	-1	-1

According to Table 8.19, calcNeighbours_3D() function specifies the neighbours of each covariate pattern. A screenshot from its R documentation summarizes the function's usage in Figure 8.41.



calcNeighbours_3D {moralar} R Documentation

3D-Neigbours' approach: Specifies neighbours of each group of the covariates in case of a 3-way interaction.

Description

For a 3-way interaction, a covariate pattern consists of category of variable 1 (C1), category of variable 2 (C2) and category of variable 3 (C3) = (C1, C2, C3)
(C1, C2, C3) may have up to 26 neigbours (C1 +/- 1:C2 +/- 1:C3 +/- 1)

Usage

```
calcNeighbours_3D(data, var_idx)
```

Arguments

data	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
var_idx	a data frame consisting of three columns var1_idx, var2_idx and var3_idx which include indices of covariates for those 3-way interactions are studied.

Value

returns a list for each 3-way interaction. Each list also contains a list for every bin of the interaction.
A bin may have up to 26 neighbours, if exists.

Figure 8.41. Screenshot from the R documentation of calcNeighbours_3D() function.

In Figure 8.42, R expression to calculate the neighbours of each covariate pattern for GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 is shared. Since the WOE value of the second bin of the interaction term, i.e. observations that take 1 as GRP_ALL_VAR_1, 2 as GRP_CHL_VAR_5, and 2 as GRP_OL_VAR_1 could not be calculated, its neighbours are displayed in Table 8.20.

Table 8.20. Neighbours of the second bin of the interaction term, i.e. for the bin where GRP_ALL_VAR_1 = 1, GRP_CHL_VAR_5 = 2, and GRP_OL_VAR_1 = 2.

	Neighbour - 1	**Neighbour - 2**	**Neighbour - 3**	**Neighbour - 4**
numNonDef	227	111	2564	244
numDef	27	15	108	10
WOE	-1.4755	-1.6031	-0.4374	-0.41

	Neighbour - 5	**Neighbour - 6**	**Neighbour - 7**	**Neighbour - 8**
numNonDef	1961	NA	NA	NA
numDef	120	NA	NA	NA
WOE	-0.8109	NA	NA	NA

	Neighbour - 9	**Neighbour - 10**	**Neighbour - 11**	**Neighbour - 12**
numNonDef	321	115	160	1832
numDef	35	16	22	24
WOE	-1.3885	-1.6323	-1.6205	0.7305

	Neighbour - 13	**Neighbour - 14**	**Neighbour - 15**	**Neighbour - 16**
numNonDef	290	721	NA	NA
numDef	10	21	NA	NA
WOE	-0.2373	-0.0685	NA	NA

	Neighbour - 17	**Neighbour - 18**	**Neighbour - 19**	**Neighbour - 20**
numNonDef	NA	NA	NA	NA
numDef	NA	NA	NA	NA
WOE	NA	NA	NA	NA

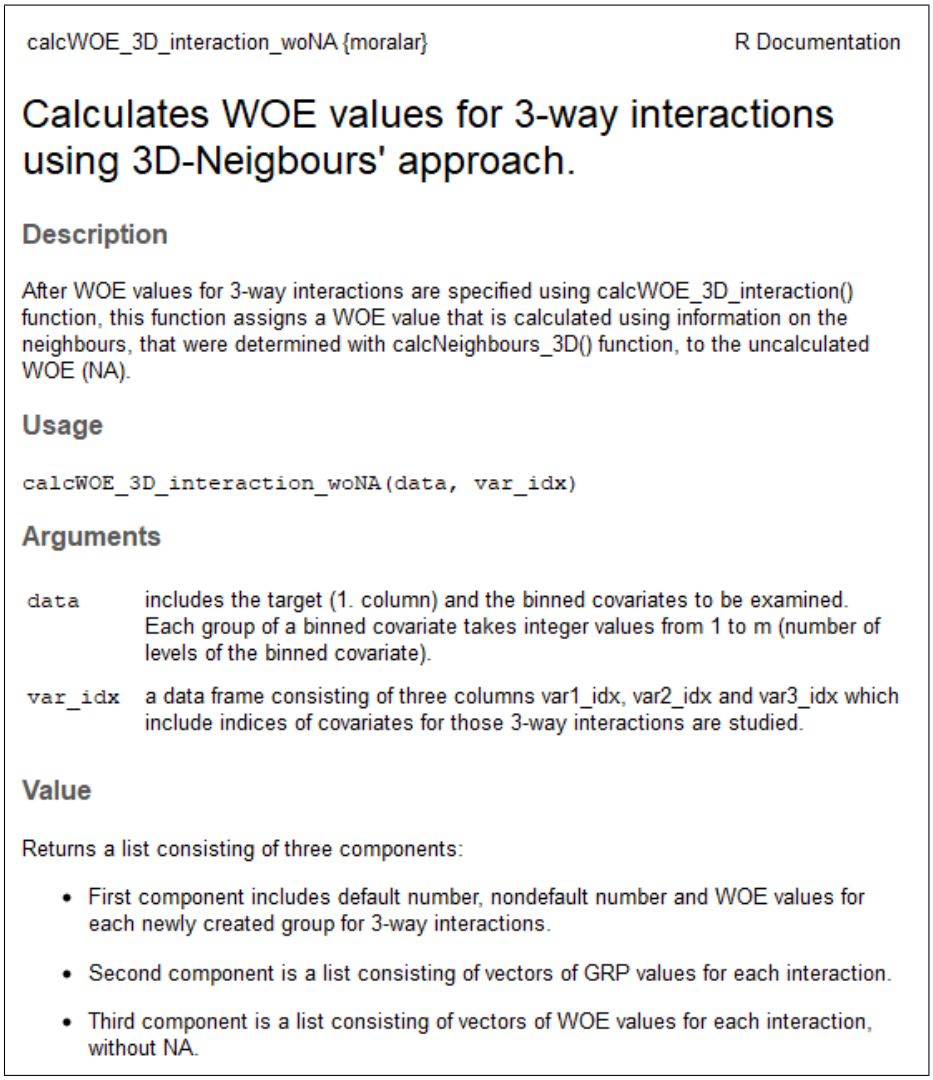
	Neighbour - 21	**Neighbour - 22**	**Neighbour - 23**	**Neighbour - 24**
numNonDef	NA	NA	NA	NA
numDef	NA	NA	NA	NA
WOE	NA	NA	NA	NA

	Neighbour - 25	**Neighbour - 26**
numNonDef	NA	NA
numDef	NA	NA
WOE	NA	NA

```
# Neighbours of each bin of the interaction term are specified.
Neighbours_3D_all_bins ← calcNeighbours_3D(Var_all_GRP_train, var_-
idx = data.frame(var1_idx = c(2), var2_idx = c(21), var3_idx = c(27)))

# Neighbours of the tenth bin are tabulated.
Neighbours_3D_all_bins$‘GRP_ALL_VAR_1 X GRP_CHL_VAR_5 X
GRP_OL_VAR_1‘$‘1 x 2 x 2‘
# or
Neighbours_3D_all_bins$‘GRP_ALL_VAR_1 X GRP_CHL_VAR_5 X
GRP_OL_VAR_1‘[[2]]
```

Figure 8.42. R code to specify the neighbours of each bin of the interaction term with regards to 3D-Neighbours' approach.

Using the 3D-neighbours' approach, calcWOE_3D_interaction() is modified and the resulting calcWOE_3D_interaction_woNA() function addresses the issue of uncalculated WOE values. If a bin does not include any defaults but a reasonable number, at least 100 as an expert's view, of non-defaults, then we concluded that the number of defaults can be presumed to be 0.7. Otherwise, after specifying the neighbours, if available, the weighted average of WOE values of its neighbours is calculated to impute the missing WOE value. Screenshot from the R documentation of calcWOE_3D_interaction_woNA() function is given in Figure 8.43.



calcWOE_3D_interaction_woNA {moralar} R Documentation

Calculates WOE values for 3-way interactions using 3D-Neigbours' approach.

Description

After WOE values for 3-way interactions are specified using calcWOE_3D_interaction() function, this function assigns a WOE value that is calculated using information on the neighbours, that were determined with calcNeighbours_3D() function, to the uncalculated WOE (NA).

Usage

```
calcWOE_3D_interaction_woNA(data, var_idx)
```

Arguments

`data`	includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
`var_idx`	a data frame consisting of three columns var1_idx, var2_idx and var3_idx which include indices of covariates for those 3-way interactions are studied.

Value

Returns a list consisting of three components:

- First component includes default number, nondefault number and WOE values for each newly created group for 3-way interactions.
- Second component is a list consisting of vectors of GRP values for each interaction.
- Third component is a list consisting of vectors of WOE values for each interaction, without NA.

Figure 8.43. Screenshot from the R documentation of calcWOE_3D_interaction_woNA() function.

In Figure 8.44, R code showing this function's usage is shared. In Table 8.21, the comparison for the bin in question is tabulated before and after applying the 3D-Neighbours' approach.

```
# Interaction term is obtained using the 3D-Neighbours' approach.
data_WOE_3D_interaction_woNA ← calcWOE_3D_interaction_-
woNA(Var-_all_GRP_train, var_idx = data.frame(var1_idx = c(2), var2_idx
= c(21), var3_idx = c(27)))

# Comparison before and after applying the 2D-Neighbours' approach.
cbind(data_WOE_3D_interaction[[1]]$‘GRP_ALL_VAR_1 X GRP_-
CHL_VAR_5 X GRP_OL_VAR_1‘[, 2], data_WOE_3D_interaction_-
woNA[[1]]$‘GRP_ALL_VAR_1 X GRP_CHL_VAR_5 X
GRP_OL_VAR_1‘[, 2])
```

Figure 8.44. R code to calculate WOE version of the three-way interaction term taking account of the 3D-Neighbours' approach, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects.

Table 8.21. Comparison before and after applying the 3D-Neighbours' approach for the first bin of the interaction term, i.e. for the bin where GRP_ALL_VAR_1 = 1, GRP_CHL_VAR_5 = 2 and GRP_OL_VAR_1 = 2.

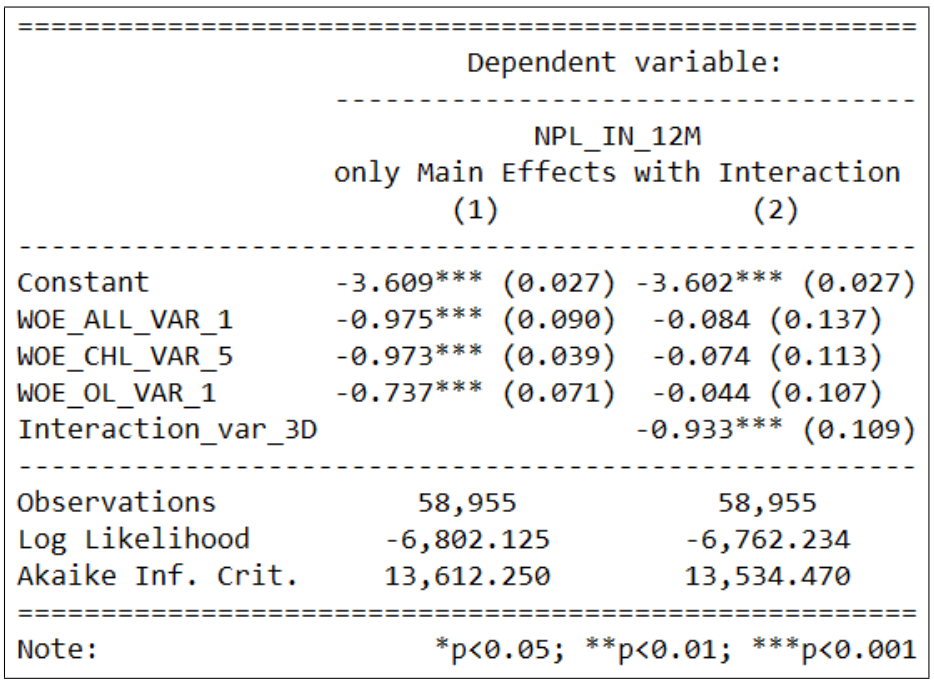
	Actual	**After using 3D-Neighbours' approach**
	1 x 2 x 2	1 x 2 x 2
numNonDef	65	65
numDef	8	8
WOE	NA	-0.5626617

After obtaining the WOE values by means of calcWOE_3D_interaction_woNA() function easily, the third component of the produced object is renamed in a syntactically valid way and combined with the main effects and the response variable as in Figure 8.45. Subsequently, three logistic regression models are fitted: with only main effects, with main effects and interaction term together and with only interaction term.

```
# Response variable, main effects and interaction term are joined together.
data_train_woNA ← cbind(Var_all_WOE_train[, c(1, 2, 21, 27)], data_-
WOE_3D_interaction_woNA[[3]][1])

# Interaction term is renamed.
colnames(data_train_woNA)[5] ← "Interaction_var_3D"

# Logistic regression models are fitted using the training split for three cases:
# Only main effects.
mod_fit_3D_woI ← glm(NPL_IN_12M ∼ ., data = data_train_woNA[, -
c(5)], family = binomial(link = "logit"))

# Main effects and interaction term.
mod_fit_3D_wI ← glm(NPL_IN_12M ∼ ., data = data_train_woNA, family
= binomial(link = "logit"))

# Only interaction term.
mod_fit_3D_onlyI ← glm(NPL_IN_12M ∼ ., data = data_train_woNA[,
c(1, 5)], family = binomial(link = "logit"))
```

Figure 8.45. R code to fit logistic regression models with/out interaction term using the WOE variables taking account of the 3D-Neighbours' approach, and to print the summaries of fitted models.

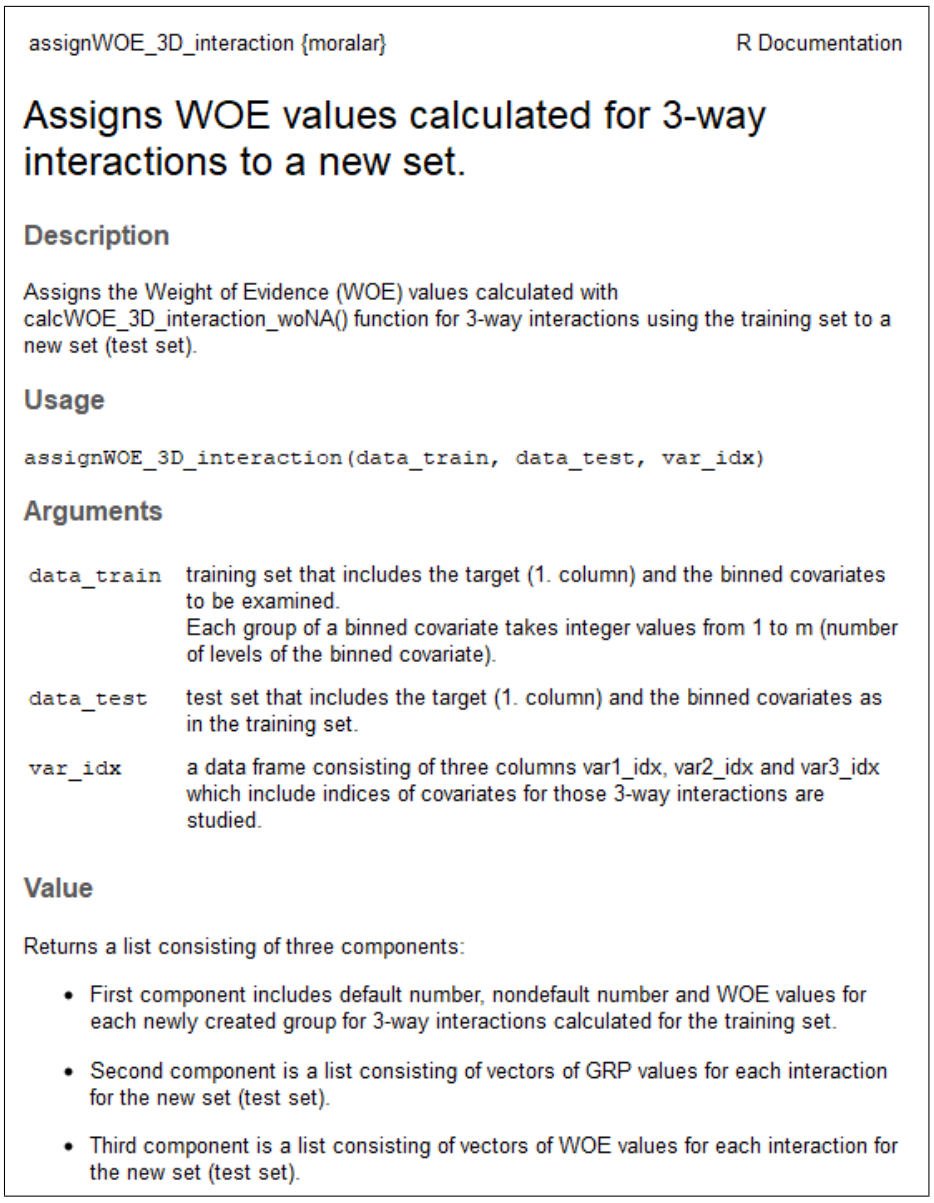
The summaries of fitted logistic regression models are shared in Figure 8.46 with/out interaction term taking account of the 3D-Neighbours' approach, in case GRP_ALL_VAR_1, GRP_CHL_VAR_5 and GRP_OL_VAR_1 are the constructive main effects. It can be seen that the coefficient of the interaction term is no longer -1, but close. Coefficient of the main effects are again small but bigger than the coefficients in Figure 8.40, in case the interaction term is taken into the model. The reason lies in the manipulation with neigbouring approach.



	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.609*** (0.027)	-3.602*** (0.027)
WOE_ALL_VAR_1	-0.975*** (0.090)	-0.084 (0.137)
WOE_CHL_VAR_5	-0.973*** (0.039)	-0.074 (0.113)
WOE_OL_VAR_1	-0.737*** (0.071)	-0.044 (0.107)
Interaction_var_3D		-0.933*** (0.109)
Observations	58,955	58,955
Log Likelihood	-6,802.125	-6,762.234
Akaike Inf. Crit.	13,612.250	13,534.470
Note:	*p<0.05; **p<0.01; ***p<0.001	

Figure 8.46. Summaries of fitted logistic regression models with/out interaction term taking account of the 3D-Neighbours' approach, in case WOE_ALL_VAR_1, WOE_CHL_VAR_5 and WOE_OL_VAR_1 are the main effects.

It should be stressed again, that WOE values are calculated from the training data set, and test data set is not used anywhere else except assessing the discriminatory power of the fitted model. Thus, another function is necessary to assign obtained WOE values for three-way interaction to the test set, or any other set for which predictions are needed. We created a function named assignWOE_3D_interaction() for this purpose. Screenshot from the R documentation of assignWOE_3D_interaction() function is presented in Figure 8.47.



assignWOE_3D_interaction {moralar} R Documentation

Assigns WOE values calculated for 3-way interactions to a new set.

Description

Assigns the Weight of Evidence (WOE) values calculated with calcWOE_3D_interaction_woNA() function for 3-way interactions using the training set to a new set (test set).

Usage

```
assignWOE_3D_interaction(data_train, data_test, var_idx)
```

Arguments

data_train	training set that includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
data_test	test set that includes the target (1. column) and the binned covariates as in the training set.
var_idx	a data frame consisting of three columns var1_idx, var2_idx and var3_idx which include indices of covariates for those 3-way interactions are studied.

Value

Returns a list consisting of three components:

- First component includes default number, nondefault number and WOE values for each newly created group for 3-way interactions calculated for the training set.
- Second component is a list consisting of vectors of GRP values for each interaction for the new set (test set).
- Third component is a list consisting of vectors of WOE values for each interaction for the new set (test set).

Figure 8.47. Screenshot from the R documentation of assignWOE_3D_interaction() function.

After assigning the WOE values by means of assignWOE_3D_interaction() function to the test set, the third component of the produced object is renamed in a syntactically valid way and combined with the main effects and the response variable as in Figure 8.48.

```
# Interaction term is obtained for the test split using the 3D-Neighbours' approach.
data_WOE_3D_interaction_woNA_test ← assignWOE_3D_interaction(Var-_all_GRP_train, Var_all_GRP_test, var_idx = data.frame(var1_idx = c(2), var2_idx = c(21), var3_idx = c(27)))

# Response variable, main effects and interaction term are joined together.
data_test_woNA ← cbind(Var_all_WOE_test[, c(1, 2, 21, 27)], data_WOE_3D_interaction_woNA_test[[3]][1])

# Interaction term is renamed.
colnames(data_test_woNA)[5] ← "Interaction_var_3D"
```

Figure 8.48. Assigning WOE values for the interaction term calculated with calcWOE_3D_interaction_woNA() function to the test split.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits. In Table 8.22, these are presented under WOE columns, along with Gini indices obtained using GRP variables, as shared in Table 8.13, under GRP columns. It can be seen that the Gini index of the model is increased by just below 2.5 when the interaction term is included, whereas the increase for the test split is approximately 1.2. The amount of increase is evident and also including the interaction term compensates the loss of discriminatory power resulted from using WOE variables instead of GRP variables. Additionally, we can also infer that using only interaction term in WOE version is sufficient, when working with WOE variables.

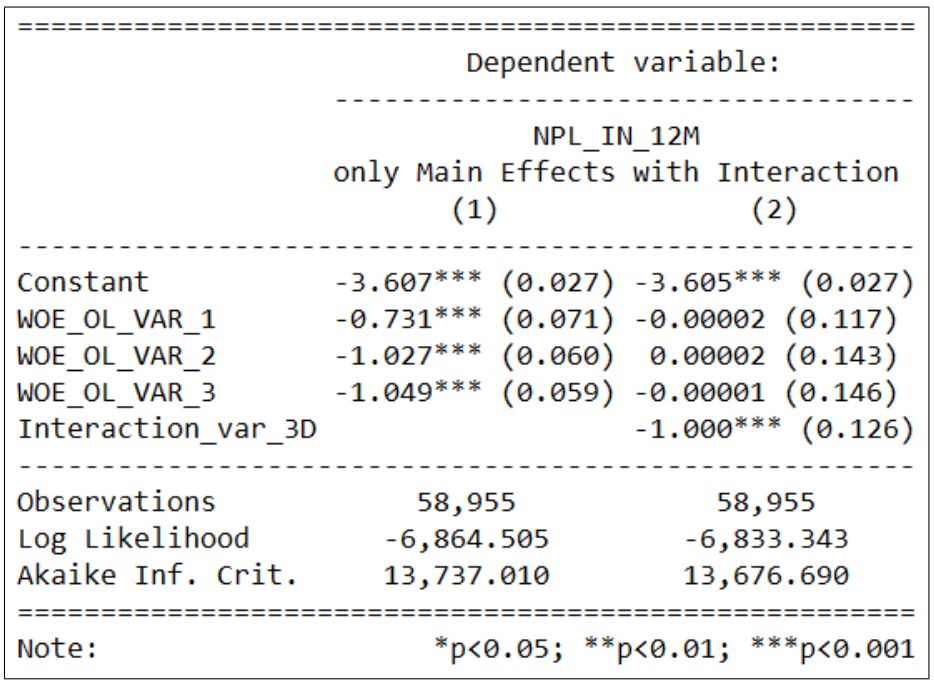
Table 8.22. Gini indices of fitted models with/out interaction term, in case WOE_ALL_VAR_1, WOE_CHL_VAR_5 and WOE_OL_VAR_1 are the main effects.

	Without interaction term		**With interaction term**		**Only interaction term**	
	GRP	**WOE**	**GRP**	**WOE**	**GRP**	**WOE**
Training	36.49	35.66	37.92	37.9	-	37.89
Test	39.14	38.4	39.53	39.55	-	39.54

8.4.2. WOE_OL_VAR_1 * WOE_OL_VAR_2 * WOE_OL_VAR_ 3

In Subsection 8.3.2, we showed how the interaction terms can be included into a regression model, in case GRP_OL_VAR_1 * GRP_OL_VAR_2 * GRP_OL_-VAR_3 are the main effects. When we want to fit a regression with WOE variables, calcWOE_3D_interaction_woNA() function is used to obtain WOE version of the three-way interaction term. Then, R expressions in Figure 8.39 are revisited to fit logistic regression models with/out interaction term and print the summaries in an elegant way. The resulting summaries in Figure 8.49 show that the interaction term is significant. Its coefficient is -1, which is also a sign that WOE values for each covariate pattern could be calculated without utilizing the 3D-Neighbours' approach. Besides, main effects can be ignored, because their coefficients are very small.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits as usual. In Table 8.23, these are presented under WOE columns, along with Gini indices obtained using GRP variables, as shared in Table 8.14, under GRP columns. It can be seen that the Gini index of the model is does not change much, i.e. there is an increase by just under 0.7, when the interaction terms are included, whereas the Gini index barely changes for the test split. Additionally, we can also infer that using only interaction term in WOE version is sufficient, when working with WOE variables.



```
=================================================
                          Dependent variable:
                    ------------------------------
                                NPL_IN_12M
                    only Main Effects with Interaction
                          (1)              (2)
-------------------------------------------------
Constant            -3.607*** (0.027) -3.605*** (0.027)
WOE_OL_VAR_1        -0.731*** (0.071) -0.00002 (0.117)
WOE_OL_VAR_2        -1.027*** (0.060)  0.00002 (0.143)
WOE_OL_VAR_3        -1.049*** (0.059) -0.00001 (0.146)
Interaction_var_3D                    -1.000*** (0.126)
-------------------------------------------------
Observations              58,955           58,955
Log Likelihood         -6,864.505       -6,833.343
Akaike Inf. Crit.      13,737.010       13,676.690
=================================================
Note:                      *p<0.05; **p<0.01; ***p<0.001
```

Figure 8.49. Summaries of fitted logistic regression models with/out interaction term taking account of the 3D-Neighbours' approach, in case WOE_OL_VAR_1, WOE_OL_VAR_2 and WOE_OL_VAR_3 are the main effects.

Table 8.23. Gini indices of fitted models with/out interaction term, in case WOE_OL_VAR_1, WOE_OL_VAR_2 and WOE_OL_VAR_3 are the main effects.

	Without interaction term		**With interaction term**		**Only interaction term**	
	GRP	**WOE**	**GRP**	**WOE**	**GRP**	**WOE**
Training	37.16	37.13	37.79	37.79	-	37.79
Test	40.84	40.95	40.92	40.92	-	40.92

8.4.3. WOE_ALL_VAR_4 * WOE_CCOL_VAR_1 * WOE_CHL_VAR_-5

In Subsection 8.3.3, we showed how the interaction term can be included into a regression model, in case GRP_ALL_VAR_4, GRP_CCOL_VAR_1 and GRP_-CHL-_VAR_5 are the main effects. When we want to fit a regression with WOE variables, calcWOE_3D_interaction_woNA() function is used to obtain WOE version

of the three-way interaction term. Then, R expressions in Figure 8.39 are revisited to fit logistic regression models with/out interaction term and print the summaries in an elegant way.

	Dependent variable:	
	NPL_IN_12M	
	only Main Effects	with Interaction
	(1)	(2)
Constant	-3.596*** (0.027)	-3.605*** (0.028)
WOE_ALL_VAR_4	-0.582*** (0.053)	0.00002 (0.090)
WOE_CCOL_VAR_1	-0.815*** (0.047)	0.00004 (0.104)
WOE_CHL_VAR_5	-0.582*** (0.045)	0.00002 (0.082)
Interaction_var_3D		-1.000*** (0.115)
Observations	58,955	58,955
Log Likelihood	-6,707.888	-6,665.457
Akaike Inf. Crit.	13,423.780	13,340.910
Note:		*p<0.05; **p<0.01; ***p<0.001

Figure 8.50. Summaries of fitted logistic regression models with/out interaction term taking account of the 3D-Neighbours' approach, in case WOE_ALL_VAR_4, WOE_CCOL_VAR_1 and WOE_CHL_VAR_5 are the main effects.

The resulting summaries in Figure 8.50 show that the interaction term is significant. Its coefficient is -1, which is also a sign that WOE values for each covariate pattern could be calculated without utilizing the 3D-Neighbours' approach. Besides, main effects can be ignored, because their coefficients are very small.

In order to assess the contribution of the interaction term with regards to Gini index, we calculated the Gini indices of the fitted models with/out interaction term for both training and test splits as usual. In Table 8.24, these are presented under WOE columns, along with Gini indices obtained using GRP variables, as shared in Table 8.15, under GRP columns. It can be seen that the Gini index of the model is increased

by more than 1.3, when the interaction term is included, whereas the Gini index barely changes for the test split, even is decreased by just more than 0.1. Additionally, we can also infer that using only interaction term in WOE version is sufficient, when working with WOE variables.

Table 8.24. Gini indices of fitted models with/out interaction term, in case WOE_ALL_VAR_4, WOE_CCOL_VAR_1 and WOE_CHL_VAR_5 are the main effects.

	Without interaction term		**With interaction term**		**Only interaction term**	
	GRP	**WOE**	**GRP**	**WOE**	**GRP**	**WOE**
Training	39.01	38.93	40.29	40.29	-	40.29
Test	38.87	38.75	38.62	38.62	-	38.62

9. FINAL MODEL

In this chapter, we sum up the experiments that we carried out in the earlier chapters and share whether our previous work results in a more powerful final model, indeed.

We used the training and test data sets of GRP variables and WOE variables obtained in Chapter 6. We examine the contribution of the interaction terms that were introduced in Chapter 8. Subsequently, the fit and the predictive power of the final model achieved will be discussed.

9.1. Final Model Selection

In Chapter 8, some two-way and three-way interaction terms were selected and examined in detail. In Figure 9.1, R code demonstrates how the WOE versions of these interaction terms are calculated all together for the training and test data sets, and stored to be used in the following steps.

```
# 2-way interaction terms for training and test splits are obtained, respectively.
# Indices of main effects are specified.
var_idx ← data.frame(var1_idx = c(13, 6, 9), var2_idx = c(14, 13, 28))
data_2D_interaction_woNA ← calcWOE_2D_interaction_woNA(
  Var_all_GRP_train, var_idx)
data_2D_interaction_woNA_test ← assignWOE_2D_interaction(
  Var_all_GRP_train, Var_all_GRP_test, var_idx)

# 3-way interaction terms for training and test splits are obtained, respectively.
# Indices of main effects are specified.
var_idx ← data.frame(var1_idx = c(2, 27, 5), var2_idx = c(21, 28, 14), var3_-
idx = c(27, 29, 21))
data_3D_interaction_woNA ← calcWOE_3D_interaction_woNA(
  Var_all_GRP_train, var_idx)
data_3D_interaction_woNA_test ← assignWOE_3D_interaction(
  Var_all_GRP_train, Var_all_GRP_test, var_idx)
```

Figure 9.1. R code to calculate WOE versions of two-way and three-way interaction terms for the training and test splits.

```
# Training data set of WOE variables including the interaction terms.
data_train_WOE ← cbind(Var_all_WOE_train,
  data_2D_interaction_woNA[[3]][1],
  data_2D_interaction_woNA[[3]][2],
  data_2D_interaction_woNA[[3]][3],
  data_3D_interaction_woNA[[3]][1],
  data_3D_interaction_woNA[[3]][2],
  data_3D_interaction_woNA[[3]][3])

# Interaction terms in WOE version are renamed for clarity.
colnames(data_train_WOE)[30:35] ←
  c("WOE_2D_CC_VAR_6_CCOL_VAR_1",
    "WOE_2D_ALL_VAR_5_CC_VAR_6",
    "WOE_2D_CC_VAR_2_OL_VAR_2",
    "WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1",
    "WOE_3D_OL_VAR_1_2_3",
    "WOE_3D_ALL_VAR_4_CCOL_VAR_1_CHL_VAR_5")


# Training data set of GRP variables including the interaction terms.
data_train_GRP ← cbind(Var_all_GRP_train,
  data_2D_interaction_woNA[[2]][1],
  data_2D_interaction_woNA[[2]][2],
  data_2D_interaction_woNA[[2]][3],
  data_3D_interaction_woNA[[2]][1],
  data_3D_interaction_woNA[[2]][2],
  data_3D_interaction_woNA[[2]][3])

# Interaction terms in GRP version are renamed for clarity.
colnames(data_train_GRP)[30:35]←
  c("GRP_2D_CC_VAR_6_CCOL_VAR_1",
    "GRP_2D_ALL_VAR_5_CC_VAR_6",
    "GRP_2D_CC_VAR_2_OL_VAR_2",
    "GRP_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1",
    "GRP_3D_OL_VAR_1_2_3",
    "GRP_3D_ALL_VAR_4_CCOL_VAR_1_CHL_VAR_5")
```

Figure 9.2. R code to obtain the final training and test data sets of WOE and GRP variables including the relabeled two-way and three-way interaction terms.

```
# Test data set of WOE variables including the interaction terms.
data_test_WOE ← cbind(Var_all_WOE_test,
  data_2D_interaction_woNA_test[[3]][1],
  data_2D_interaction_woNA_test[[3]][2],
  data_2D_interaction_woNA_test[[3]][3],
  data_3D_interaction_woNA_test[[3]][1],
  data_3D_interaction_woNA_test[[3]][2],
  data_3D_interaction_woNA_test[[3]][3])

# Interaction terms in WOE version are renamed for clarity.
colnames(data_test_WOE)[30:35]←
  c("WOE_2D_CC_VAR_6_CCOL_VAR_1",
    "WOE_2D_ALL_VAR_5_CC_VAR_6",
    "WOE_2D_CC_VAR_2_OL_VAR_2",
    "WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1",
    "WOE_3D_OL_VAR_1_2_3",
    "WOE_3D_ALL_VAR_4_CCOL_VAR_1_CHL_VAR_5")


# Test data set of GRP variables including the interaction terms.
data_test_GRP ← cbind(Var_all_GRP_test,
  data_2D_interaction_woNA_test[[2]][1],
  data_2D_interaction_woNA_test[[2]][2],
  data_2D_interaction_woNA_test[[2]][3],
  data_3D_interaction_woNA_test[[2]][1],
  data_3D_interaction_woNA_test[[2]][2],
  data_3D_interaction_woNA_test[[2]][3])

# Interaction terms in GRP version are renamed for clarity.
colnames(data_test_GRP)[30:35] ←
  c("GRP_2D_CC_VAR_6_CCOL_VAR_1",
    "GRP_2D_ALL_VAR_5_CC_VAR_6",
    "GRP_2D_CC_VAR_2_OL_VAR_2",
    "GRP_3D_ALL_VAR_1_
    CHL_VAR_5_OL_VAR_1",
    "GRP_3D_OL_VAR_1_2_3",
    "GRP_3D_ALL_VAR_4_CCOL_VAR_1_CHL_VAR_5")
```

Figure 9.2. R code to obtain the final training and test data sets of WOE and GRP variables including the relabeled two-way and three-way interaction terms (cont.).

WOE and GRP values of calculated interaction terms are combined with the main effects and extended training and test data sets of WOE and GRP variables are created with the R expressions in Figure 9.2. All the interaction variables are renamed in a syntactically valid way whether they are WOE or GRP variables. For example, WOE version of the two-way interaction between GRP_CC_VAR_6 and GRP_-CCOL_VAR_1 is relabeled as WOE_2D_CC_VAR_6_CCOL-_VAR_1; -whereas GRP version of the three-way interaction between GRP_OL_VAR_1, GRP-_OL_-VAR_2 and GRP_OL_VAR_3 is relabeled as GRP_3D_OL_VAR_1_2_3.

Using the WOE versions of the covariates excluding the interaction terms, first a logistic regression model is fitted. This fitted model includes all the covariates available in the data set and therefore is called as the full model without interaction terms. Later, using the stepAIC() function of MASS package [30], this full model is reduced according to the Schwarz-Bayesian information criterion.

```
# Logistic regression model is fitted without interaction terms.
full_Model_woI ← glm(NPL_IN_12M ∼ ., data = Var_all_WOE_train, fam-
ily = binomial(link = "logit"))
summary(full_Model_woI)

# MASS package is loaded to use stepAIC() function.
library(MASS)
# Schwarz-Bayesian information criterion is used for model selection.
stepwise_Model_BIC_woI ← stepAIC(full_Model_woI, trace=FALSE,
k=log(nrow(Var_all_WOE_train)))
```

Figure 9.3. R code to fit logit logistic regression models without interaction terms, before and after using BIC for model selection.

The reduced model includes less predictors, however the loss of information is limited. According to Occam's razor, if the reduced model with less predictors has a similar discriminatory power as the full model, the reduced model should be preferred. Gini indices of the full model and the reduced model without interaction terms are tabulated in Table 9.1. Here, we can see that there is a decrease of a little more than

0.8 in Gini index in exchange of less predictors. On the other hand, the decrease for the test set is just above 1.4.

```
# Logistic regression model is fitted with interaction terms.
full_Model_wI ← glm(NPL_IN_12M ∼., data = data_train_WOE, family =
binomial(link = "logit"))
summary(full_Model_wI)
#Schwarz-Bayesian information criterion is used for model selection.
stepwise_Model_BIC_wI ← stepAIC(full_Model_wI, trace=FALSE,
k=log(nrow(data_train_WOE)))

summary (stepwise_Model_BIC_wI)

stepwise_Model_BIC_wI$anova
```

Figure 9.4. R code to fit logit logistic regression models with interaction terms, before and after using BIC for model selection.

Table 9.1. Gini indices of the full model and the reduced model according to BIC without interaction terms.

	Full model without interaction terms	**Selected model without interaction terms**
Training split	62.99	62.16
Test split	65.99	64.55

Using the WOE versions of the covariates including the interaction terms, first a logistic regression model is fitted. This fitted model includes all the covariates available in the extended data set, and therefore it is called as the full model with interaction terms. Later, using the stepAIC() function, this full model is reduced according to the Schwarz-Bayesian information criterion. This model includes less predictors, however the loss of information is limited, as well. Each step of stepAIC() function is given in Figure 9.5, until the reduced model with the lowest BIC is selected.

	Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
1				58920	11957.52	12341.98
2	- WOE_ALL_VAR_1	1	0.1171368	58921	11957.64	12331.11
3	- WOE_ALL_VAR_5	1	0.4511773	58922	11958.09	12320.58
4	- WOE_CC_VAR_2	1	1.5258180	58923	11959.62	12311.12
5	- WOE_CL_VAR_2	1	1.8448220	58924	11961.46	12301.98
6	- WOE_ALL_VAR_3	1	3.3621062	58925	11964.82	12294.36
7	- WOE_OL_VAR_3	1	3.3364691	58926	11968.16	12286.71
8	- WOE_ALL_VAR_4	1	3.5707789	58927	11971.73	12279.30
9	- WOE_OL_VAR_2	1	4.6052868	58928	11976.34	12272.92
10	- WOE_CHL_VAR_1	1	2.5563161	58929	11978.89	12264.49
11	- WOE_3D_OL_VAR_1_2_3	1	3.7326978	58930	11982.63	12257.24
12	- WOE_OL_VAR_1	1	4.6564365	58931	11987.28	12250.91
13	- WOE_CCOL_VAR_3	1	6.3504051	58932	11993.63	12246.28
14	- WOE_CCOL_VAR_1	1	4.5073532	58933	11998.14	12239.80
15	- WOE_CHL_VAR_2	1	9.4569681	58934	12007.60	12238.27
16	- WOE_CL_VAR_1	1	6.7997950	58935	12014.40	12234.09
17	- WOE_CHL_VAR_5	1	9.1262093	58936	12023.52	12232.23
18	- WOE_3D_ALL_VAR_4_CCOL_VAR_1_CHL_VAR_5	1	8.8328667	58937	12032.36	12230.08
19	- WOE_CL_VAR_3	1	10.4267751	58938	12042.78	12229.52

Figure 9.5. Each step of stepAIC() function until the reduced model with the lowest BIC is selected.

The summary of the reduced model with interaction terms is given in Figure 9.6. One can see that each predictor is significant and the number of total predictors is 16. However, the coefficient of WOE_CC_VAR_6 reveals a problem as its sign is positive, although it should be negative. This false sign arises from the fact that the effect of CC_VAR_6 is already included in the model with two-way interaction terms, namely WOE_2D_CC_VAR_6_CCOL_VAR_1 and WOE_2D_ALL_-VAR_5_CC_VAR-_6.

The variable whose parameter takes false sign is dropped from the reduced model with the R code in Figure 9.7. Consequently, the final model is obtained. The summary of the final model is presented in Figure 9.8. In the final model, all the signs of the coefficients are correctly negative.

```
Call:
glm(formula = NPL_IN_12M ~ WOE_ALL_VAR_2 + WOE_ALL_WOHL_VAR_1 +
    WOE_CC_VAR_1 + WOE_CC_VAR_3 + WOE_CC_VAR_4 + WOE_CC_VAR_5 +
    WOE_CC_VAR_6 + WOE_CCOL_VAR_2 + WOE_CHL_VAR_3 + WOE_CHL_VAR_4 +
    WOE_CL_VAR_4 + WOE_CL_VAR_5 + WOE_2D_CC_VAR_6_CCOL_VAR_1 +
    WOE_2D_ALL_VAR_5_CC_VAR_6 + WOE_2D_CC_VAR_2_OL_VAR_2 +
    WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1,
    family = binomial(link = "logit"), data = data_train_WOE)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5385  -0.2328  -0.1511  -0.0986   3.5783

Coefficients:
                                Estimate Std. Error  z value Pr(>|z|)
(Intercept)                     -3.60189    0.03056 -117.882  < 2e-16 ***
WOE_ALL_VAR_2                   -0.74713    0.12378   -6.036 1.58e-09 ***
WOE_ALL_WOHL_VAR_1              -0.51943    0.04275  -12.149  < 2e-16 ***
WOE_CC_VAR_1                    -0.29795    0.06289   -4.738 2.16e-06 ***
WOE_CC_VAR_3                    -0.29691    0.07299   -4.068 4.75e-05 ***
WOE_CC_VAR_4                    -0.48272    0.06926   -6.970 3.17e-12 ***
WOE_CC_VAR_5                    -0.27990    0.05644   -4.959 7.08e-07 ***
WOE_CC_VAR_6                     0.62740    0.09974    6.291 3.16e-10 ***
WOE_CCOL_VAR_2                  -0.37434    0.07336   -5.103 3.35e-07 ***
WOE_CHL_VAR_3                   -0.24949    0.06421   -3.886 0.000102 ***
WOE_CHL_VAR_4                   -0.93467    0.20211   -4.625 3.75e-06 ***
WOE_CL_VAR_4                    -0.97968    0.29095   -3.367 0.000759 ***
WOE_CL_VAR_5                    -0.50203    0.13328   -3.767 0.000165 ***
WOE_2D_CC_VAR_6_CCOL_VAR_1      -0.50751    0.04982  -10.188  < 2e-16 ***
WOE_2D_ALL_VAR_5_CC_VAR_6       -0.37234    0.04972   -7.489 6.93e-14 ***
WOE_2D_CC_VAR_2_OL_VAR_2        -0.37362    0.05086   -7.346 2.04e-13 ***
WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1 -0.39428    0.04371   -9.020  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 14418  on 58954  degrees of freedom
Residual deviance: 12043  on 58938  degrees of freedom
AIC: 12077

Number of Fisher Scoring iterations: 7
```

Figure 9.6. Fitted logistic regression model with interaction terms, that is selected according to BIC.

```
# WOE_CC_VAR_6 is dropped and stepwise_Model_BIC_wI is updated.
stepwise_Model_BIC_wI_2 ← update(stepwise_Model_BIC_wI, .∼. -
WOE_CC_VAR_6)

summary (stepwise_Model_BIC_wI_2)
```

Figure 9.7. The obtained reduced model is updated by dropping WOE_CC_VAR_6.

```
Call:
glm(formula = NPL_IN_12M ~ WOE_ALL_VAR_2 + WOE_ALL_WOHL_VAR_1 +
    WOE_CC_VAR_1 + WOE_CC_VAR_3 + WOE_CC_VAR_4 + WOE_CC_VAR_5 +
    WOE_CCOL_VAR_2 + WOE_CHL_VAR_3 + WOE_CHL_VAR_4 + WOE_CL_VAR_4 +
    WOE_CL_VAR_5 + WOE_2D_CC_VAR_6_CCOL_VAR_1 + WOE_2D_ALL_VAR_5_CC_VAR_6 +
    WOE_2D_CC_VAR_2_OL_VAR_2 + WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1,
    family = binomial(link = "logit"), data = data_train_WOE)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5542  -0.2337  -0.1511  -0.0987   3.5805

Coefficients:
                                  Estimate Std. Error  z value Pr(>|z|)
(Intercept)                       -3.61001    0.03057 -118.089  < 2e-16 ***
WOE_ALL_VAR_2                     -0.77760    0.12338   -6.302 2.93e-10 ***
WOE_ALL_WOHL_VAR_1                -0.52182    0.04265  -12.234  < 2e-16 ***
WOE_CC_VAR_1                      -0.29317    0.06282   -4.667 3.06e-06 ***
WOE_CC_VAR_3                      -0.27860    0.07305   -3.814 0.000137 ***
WOE_CC_VAR_4                      -0.38668    0.06719   -5.755 8.65e-09 ***
WOE_CC_VAR_5                      -0.36939    0.05449   -6.779 1.21e-11 ***
WOE_CCOL_VAR_2                    -0.27049    0.07134   -3.791 0.000150 ***
WOE_CHL_VAR_3                     -0.27412    0.06417   -4.272 1.94e-05 ***
WOE_CHL_VAR_4                     -0.92233    0.20171   -4.573 4.82e-06 ***
WOE_CL_VAR_4                      -1.07451    0.29094   -3.693 0.000221 ***
WOE_CL_VAR_5                      -0.49768    0.13370   -3.722 0.000197 ***
WOE_2D_CC_VAR_6_CCOL_VAR_1        -0.40367    0.04823   -8.370  < 2e-16 ***
WOE_2D_ALL_VAR_5_CC_VAR_6         -0.26754    0.04871   -5.493 3.96e-08 ***
WOE_2D_CC_VAR_2_OL_VAR_2          -0.35817    0.05054   -7.088 1.36e-12 ***
WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1 -0.45479    0.04275  -10.638  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 14418  on 58954  degrees of freedom
Residual deviance: 12082  on 58939  degrees of freedom
AIC: 12114

Number of Fisher Scoring iterations: 7
```

Figure 9.8. Final fitted logistic regression model.

Gini indices of the full model and the final model with interaction terms are tabulated in Table 9.2, along with the Gini indices obtained without using interaction terms, as shared in Table 9.1. It can be seen that in exchange of a decrease of just above 1 in the Gini index, we obtained a model with 15 predictors instead of 34 predictors in the full model. For the test split, the decrease in the discriminatory power is even smaller, i.e. approximately 0.8. The contribution of interaction terms seems to be limited for both full and reduced models, i.e. under 1. However, there is still an increase, and using another data set of variables, impact of interaction terms would grow, especially in a setting where interactions reveal hidden information in the data that can not to obtained by only using main effects.

```
# stepwise_Model_BIC_wI_2 is selected as the final model.
final_model ← stepwise_Model_BIC_wI_2

idx ← c(1, which(colnames(data_train_WOE) %in% names(final_-
model$coefficients) [-1]))

round(calcGINI(data_train_WOE[, idx], data_test_WOE[, idx], GRP_var =
FALSE) * 100, 2)
```

Figure 9.9. R code to calculate Gini indices of each predictor in WOE versions in the final model by calcGINI() function for training and test splits.

Table 9.2. Gini indices of the full model and the reduced model according to BIC with interaction terms.

	Full Model		**Selected Reduced Model**	
	Without interaction terms	**With interaction terms**	**Without interaction term**	**With interaction terms**
Training split	62.99	63.8	62.16	62.74
Test split	65.99	65.96	64.55	65.18

Gini indices for each predictor in the final model are calculated for the training and test splits with calcGINI() function that was introduced in Chapter 7. In Figure 9.9, first the object of the final model generated earlier is renamed as final_model for the following analyses and then Gini calculation is carried out. The output is tabulated in Table 9.3.

Table 9.3. Gini indices of the predictors in WOE versions in the final model for training and test splits.

Variable Name	**Gini Index (training)**	**Gini Index (test)**
WOE_ALL_VAR_2	13.83	10.44
WOE_ALL_WOHL_VAR_1	39.71	38.79
WOE_CC_VAR_1	22.01	17.58
WOE_CC_VAR_3	23.24	22.56
WOE_CC_VAR_4	25.58	28.87
WOE_CC_VAR_5	19.48	17.39
WOE_CCOL_VAR_2	20.73	23.99
WOE_CHL_VAR_3	23.31	28
WOE_CHL_VAR_4	8.52	11.13
WOE_CL_VAR_4	6.61	5.95
WOE_CL_VAR_5	12.15	12.12
WOE_2D_CC_VAR_6_CCOL_VAR_1	39.79	37.04
WOE_2D_ALL_VAR_5_CC_VAR_6	36.58	37.49
WOE_2D_CC_VAR_2_OL_VAR_2	36.68	38.36
WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1	37.89	39.54

If one wants to examine Gini indices for each predictor in their GRP versions, calcGINI_var() function that was also introduced in Chapter 7 can be utilized. In this case calculation should be done separately for the training and test splits. Related R code is shared in Figure 9.10 and its output is tabulated in Table 9.4.

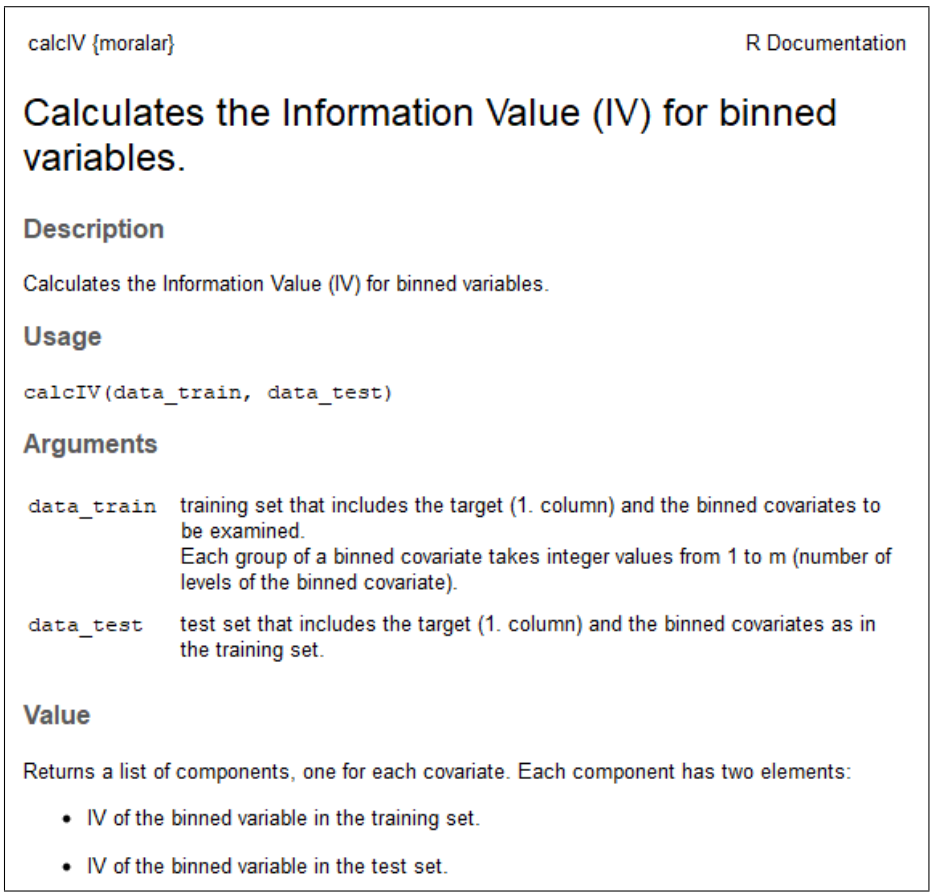
```
# Training split
calcGINI_var_train ← as.data.frame(round(calcGINI_var(data_train_GRP[,
idx]), 2))
colnames(calcGINI_var_train) ← c("GINI Index")

# Test split
calcGINI_var_test ← as.data.frame(round(calcGINI_var(data_test_GRP[,
idx]), 2))
colnames(calcGINI_var_test) ← c("GINI Index")
```

Figure 9.10. R code to calculate Gini indices of each predictor in GRP versions in the final model by calcGINI_var() function for training and test splits.

Table 9.4. Gini indices of the predictors in GRP versions in the final model for training and test splits.

Variable Name	**Gini Index (training)**	**Gini Index (test)**
GRP_ALL_VAR_2	13.83	10.51
GRP_ALL_WOHL_VAR_1	39.71	38.79
GRP_CC_VAR_1	22.01	17.58
GRP_CC_VAR_3	23.24	22.56
GRP_CC_VAR_4	25.58	29.32
GRP_CC_VAR_5	19.48	17.39
GRP_CCOL_VAR_2	20.73	23.99
GRP_CHL_VAR_3	23.31	28
GRP_CHL_VAR_4	8.52	11.13
GRP_CL_VAR_4	6.61	6.35
GRP_CL_VAR_5	12.15	12.12
GRP_2D_CC_VAR_6_CCOL_VAR_1	39.82	37.94
GRP_2D_ALL_VAR_5_CC_VAR_6	36.58	38.2
GRP_2D_CC_VAR_2_OL_VAR_2	36.68	39.12
GRP_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1	37.92	41.81

Information value of the predictors can be calculated with calcIV() function. Screenshot from the R documentation of calcIV() function is presented in Figure 9.11.



calcIV {moralar} R Documentation

Calculates the Information Value (IV) for binned variables.

Description

Calculates the Information Value (IV) for binned variables.

Usage

```
calcIV(data_train, data_test)
```

Arguments

data_train	training set that includes the target (1. column) and the binned covariates to be examined. Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
data_test	test set that includes the target (1. column) and the binned covariates as in the training set.

Value

Returns a list of components, one for each covariate. Each component has two elements:

- IV of the binned variable in the training set.
- IV of the binned variable in the test set.

Figure 9.11. Screenshot from the R documentation of calcIV() function.

Usage of the calcIV() function is displayed in Figure 9.12. Output of the expression is given in Table 9.5.

```
calcIV(data_train_GRP[, idx], data_test_GRP[, idx])
```

Figure 9.12. R code to calculate Information Value of each predictor (in GRP versions) in the final model by calcIV() function for training and test splits.

Table 9.5. Information Values of the predictors (in GRP versions) in the final model for training and test splits.

Variable Name	**Info. Value (training)**	**Info. Value (test)**
GRP_ALL_VAR_2	0.0725	0.0556
GRP_ALL_WOHL_VAR_1	0.5421	0.5279
GRP_CC_VAR_1	0.2031	0.1609
GRP_CC_VAR_3	0.1882	0.1805
GRP_CC_VAR_4	0.2162	0.2460
GRP_CC_VAR_5	0.2311	0.2063
GRP_CCOL_VAR_2	0.1681	0.1986
GRP_CHL_VAR_3	0.2069	0.2473
GRP_CHL_VAR_4	0.0301	0.0394
GRP_CL_VAR_4	0.0169	0.0152
GRP_CL_VAR_5	0.0512	0.0505
GRP_2D_CC_VAR_6_CCOL_VAR_1	0.5485	0.5152
GRP_2D_ALL_VAR_5_CC_VAR_6	0.4828	0.4876
GRP_2D_CC_VAR_2_OL_VAR_2	0.4437	0.4616
GRP_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1	0.5384	0.5738

In order to detect whether there is an issue of multicollinearity in the final model, vif() function in car package [22] can be utilized. In Figure 9.13, variance inflation factor (VIF) is obtained for each variable using the final_model object. From Table 9.6, for each of the predictors VIF is less than 5, which is a common threshold.

```
# In order to obtain VIF to detect whether multicollinearity exists.
final_model_vif ← vif(final_model)
final_model_vif ← as.data.frame(final_model_vif)
colnames(final_model_vif) ← c("VIF")
```

Figure 9.13. R code to calculate VIF values of the predictors in the final model to detect multicollinearity.

```
# Predictions in log-odds
predictTrain_logodds ← predict(final_model)
predictTest_logodds ← predict(final_model, data_test_WOE)

# Predictions in probabilities
predictTrain_prob ← predict(final_model, type = "response")
predictTest_prob ← predict(final_model, data_test_WOE, type = "response")
```

Figure 9.14. R code to calculate predicted values as probabilities and log odds for training and test sets.

9.2. Performance Indicators on the Final Model

predict() function of R is used to make predictions using the object of the fitted final model. If default type is used as the parameter of the function, then the predictions are in log odds for logistic regression. If type is specified as "response", predictions are in probabilities. Both type of predictions may be preferred by a practitioner. One of them is obtained easily from the other one, as stated in Chapter 3.

```
# roc() function from pROC package is utilized for further statistical measures of the performance.
auc ← roc(data_train_WOE$NPL_IN_12M, predictTrain_prob)

# Plot of ROC curve
plot(auc, ylim = c(0, 1), print.thres = TRUE, main = paste('AUC:', round(auc$auc[[1]], 2)))
abline(h = 1, col = 'black', lwd = 1)
abline(h = 0, col = 'black', lwd = 1)
```

Figure 9.15. R code to draw the ROC curve.

From Figure 9.16, it can be inferred that the final model can be generalized for the test data set. Area under the curve (AUC) for the test data set is even larger than the AUC for the training set. Using the 2*AUC-1 formula, Gini indices for the training and test sets can also be calculated as stated in Chapter 4.

Table 9.6. VIF values of the predictors in the final model.

Variable Name	**VIF**
WOE_ALL_VAR_2	1.693298
WOE_ALL_WOHL_VAR_1	1.462297
WOE_CC_VAR_1	1.329962
WOE_CC_VAR_3	1.271466
WOE_CC_VAR_4	1.353211
WOE_CC_VAR_5	1.231402
WOE_CCOL_VAR_2	1.375695
WOE_CHL_VAR_3	1.332981
WOE_CHL_VAR_4	1.687857
WOE_CL_VAR_4	2.030265
WOE_CL_VAR_5	1.220989
WOE_2D_CCOL_VAR_1_CC_VAR_6	1.455623
WOE_2D_ALL_VAR_5_CC_VAR_6	1.761615
WOE_2D_CC_VAR_2_OL_VAR_2	1.362615
WOE_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1	1.680339

9.2.1. Selection of Cut-off Value

The selection of a cut-off value is an important problem. Which is the maximum probability of being classified as NPL in 12 months, i.e. default, that can be accepted, so that the applicant can be granted a loan? If this probability is specified as very low, then many applicants who can actually pay back their loans would be rejected. Subsequently, the company may miss an opportunity to make profits. On the other hand, if this cut-off is high, then many applicants who are not able to pay their loans back, would be granted a loan. Consequently, the institution may face great financial losses. Generally, costs of these cases are different from each other and companies prefer missing opportunities to make profits to granting loans to customer who will probably fail to pay his/her due's. Suppose costs of these cases are the same for the company. In such a scenario, cut-off can be calculated easily with the R code in Figure 9.17 using the predictions in probabilities. For the final model obtained, the threshold is calculated to be 0.028 in terms of probability of default.

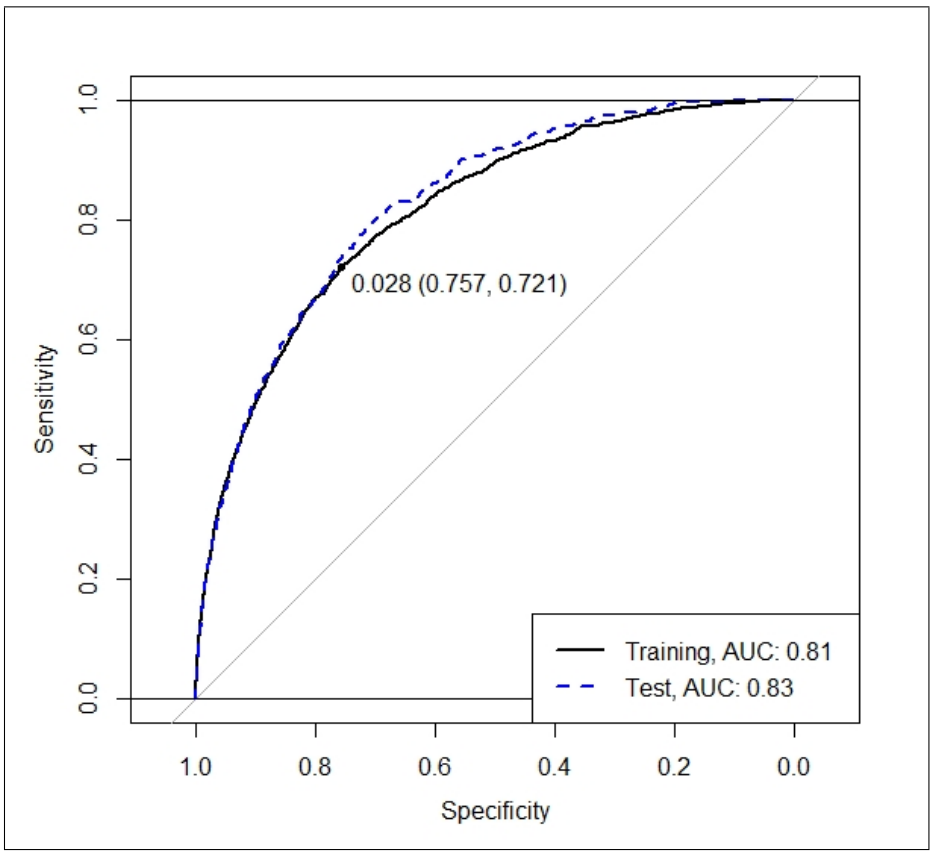


Figure 9.16. ROC curves for training and test sets.

```
find_cutoff ← function(predict, response) {
r ← pROC::roc(response, predict)
r$thresholds[which.max(rsensitivities + rspecificities)]
}

opt_t ← find_cutoff(predictTrain_prob, data_train_WOE$NPL_IN_12M)
sprintf('Threshold is %g', opt_t)
#[1] "Threshold is 0.0283451"
```

Figure 9.17. R code to calculate the cutoff value in terms of predicted probability.

For different cut-off values following truth tables are obtained as in Table 9.7. In order to interpret a truth table, Subsection 9.2.2. will be helpful.

Table 9.7. Truth tables for different cut-off values.

Cut-off value	**Predicted**	**Truth**	
		Nondefault	**Default**
0.028	Nondefault	43423	435
	Default	13971	1126

		Truth	
		Nondefault	**Default**
0.05	Nondefault	50268	715
	Default	7126	846

		Truth	
		Nondefault	**Default**
0.1	Nondefault	55043	1053
	Default	2351	508

		Truth	
		Nondefault	**Default**
0.25	Nondefault	57121	1423
	Default	273	138

9.2.2. Calculation of Performance Measurements Using the Cut-off Value

In Figure 9.18, calculation of various performance measurements, which are defined in Chapter 4, from the confusion matrix are presented. These are important terms in statistics.

```
# The confusion matrix.
table(as.numeric(predictTrain_prob ≥ opt_t), data_train_WOE$NPL_IN_-
12M, dnn = c("Predicted", "Truth"))

# Model performance indicators withdrawn from the confusion matrix.
print(paste('Number of True Positives (TP) is', sum(as.numeric(predictTrain_-
prob ≥ opt_t) == 1 & data_train_WOE$NPL_IN_12M == 1)))
#[1] "Number of True Positives (TP) is 1126"

print(paste('Number of False Negatives (FN) is',
sum(as.numeric(predictTrain_prob ≥ opt_t) == 0 & data_train_-
WOE$NPL_IN_12M == 1)))
#[1] "Number of False Negatives (FN) is 435"

print(paste('Number of True Negatives (TN) is', sum(as.numeric(predictTrain_-
prob >= opt_t) == 0 & data_train_WOE$NPL_IN_12M == 0)))
#[1] "Number of True Negatives (TN) is 43423"

print(paste('Number of False Positives (FP) is', sum(as.numeric(predictTrain_-
prob >= opt_t) == 1 & data_train_WOE$NPL_IN_12M == 0)))
#[1] "Number of False Positives (FP) is 13971"

sprintf('Sensitivity is %g, round (sum(as.numeric(predictTrain_prob ≥ opt_-
t) == 1 & data_train_WOE$NPL_IN_12M == 1) / sum(data_train_-
WOE$NPL_IN_12M == 1) * 100, 2))
#[1] "Sensitivity is 72.13%"
```

Figure 9.18. R expressions to calculate various performance measurements from the contingency table.

```
sprintf('Specificity is %g, round (sum(as.numeric(predictTrain_prob ≥ opt_-
t) == 0 & data_train_WOE$NPL_IN_12M == 0) / sum(data_train_-
WOE$NPL_IN_12M == 0) * 100, 2))
#[1] "Specificity is 75.66%"

misClassErr ← mean(as.numeric(predictTrain_prob ≥ opt_t) != data_train_-
WOE$NPL_IN_12M)
sprintf('Misclassification error is %g, round(misClassErr * 100, 2))
#[1] "Misclassification error is 24.44%"

sprintf('Accuracy is %g, round((1 - misClassErr) * 100, 2))
#[1] "Accuracy is 75.56%"
```

Figure 9.18. R expressions to calculate various performance measurements from the contingency table (cont.).

Another beneficial plot to understand the relationship between threshold and TPR along with TNR. Again, it should be stated that the cost is assumed to be the same for false positives and false negatives. R code to create the plot is given in 9.19 and the output can be observed in Figure 9.20.

```
# Plot of threshold vs. TPR, TNR
matplot(data.frame(auc$sensitivities, auc$specificities), x = auc$thresholds,
type = 'l', xlab = 'threshold', ylab = 'TPR, TNR')
legend('bottomright', legend = c('TPR', 'TNR'), lty = 1:2, col = 1:2)
```

Figure 9.19. R code to draw True Positives Rate and True Negatives Rate.

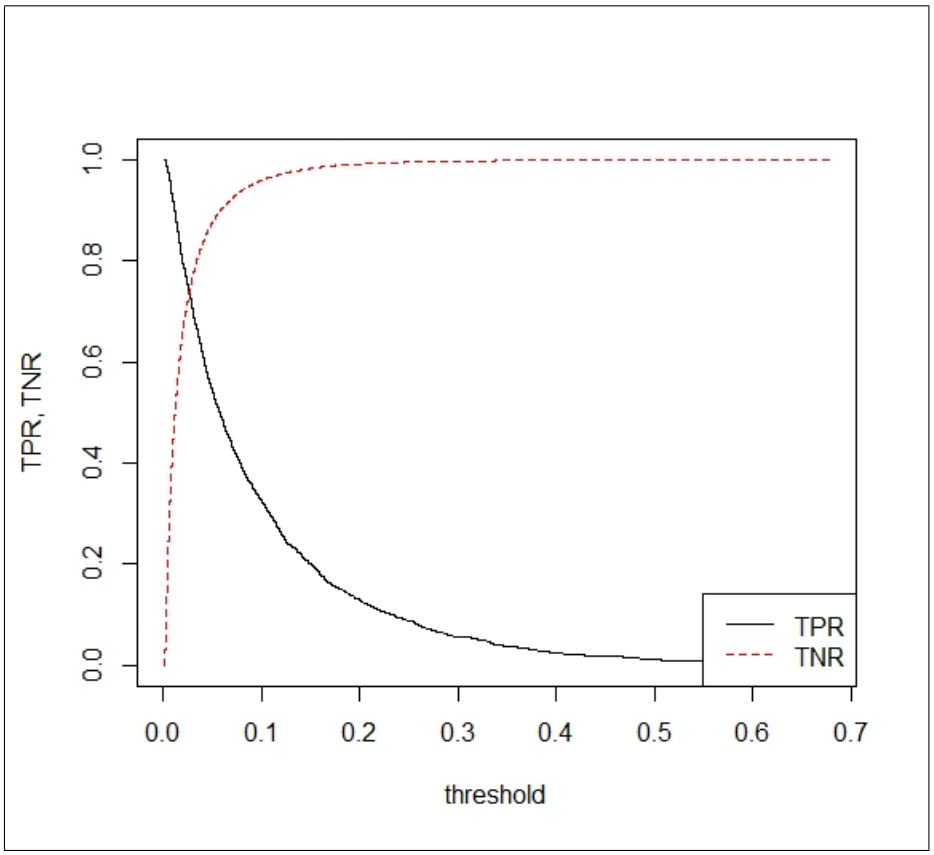


Figure 9.20. Threshold vs. TPR and TNR.

9.2.3. Hosmer Lemeshow Goodness-of-Fit Test

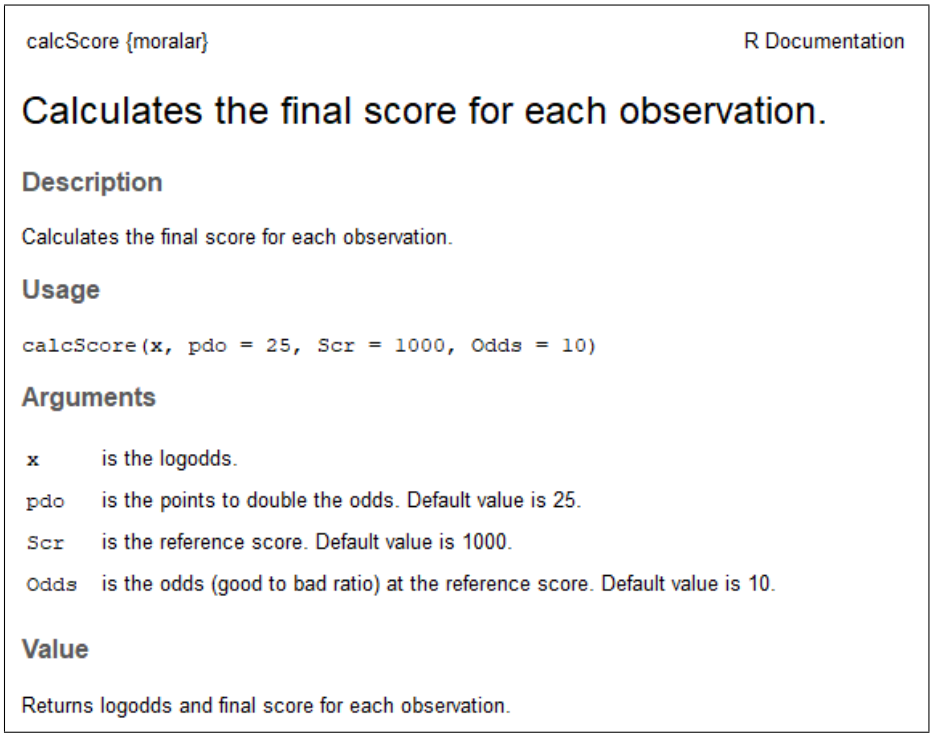
Finally, Hosmer and Lemeshow test is summarized with the R code in Figure 9.21 using the hoslem.test() function of the ResourceSelection package [31].

```
# ResourceSelection package is loaded for Hosmer and Lemeshow goodness-of-fit
test
library(ResourceSelection)

hl ← hoslem.test(data_train_WOE$NPL_IN_12M, predictTrain_prob)
Hosmer and Lemeshow goodness of fit (GOF) test
data: data_train_WOE$NPL_IN_12M, predictTrain_prob
X-squared = 2.6019, df = 8, p-value = 0.9568

#This gives p=0.9568, indicating no evidence of poor fit. This is good, since
here we know the model is indeed correctly specified.
```

Figure 9.21. R code for the Hosmer and Lemeshow goodness-of-fit test.

9.3. Obtaining Scores

In practice, predictions in log odds or probabilities are not preferred outputs of credit scoring models. Probilities may be understood rather easily, however log odds are difficult to draw conclusions about the credibility of an applicant for the professionals who utilize scoring models. Thus, predictions are transformed into scores according to a selected reference point, as explained in Section 5.4. For this purpose a calcScore() function is created which is introduced in the screenshot from its R documentation in Figure 9.22. The usage and specified parameters are demonsrated in Figure 9.23, whereas predicted log odds and obtained final scores for the first ten observations in the training set are given in Table 9.8.



calcScore {moralar} R Documentation

Calculates the final score for each observation.

Description

Calculates the final score for each observation.

Usage

```
calcScore(x, pdo = 25, Scr = 1000, Odds = 10)
```

Arguments

x	is the logodds.
pdo	is the points to double the odds. Default value is 25.
Scr	is the reference score. Default value is 1000.
Odds	is the odds (good to bad ratio) at the reference score. Default value is 10.

Value

Returns logodds and final score for each observation.

Figure 9.22. Screenshot from the R documentation of calcScore() function.

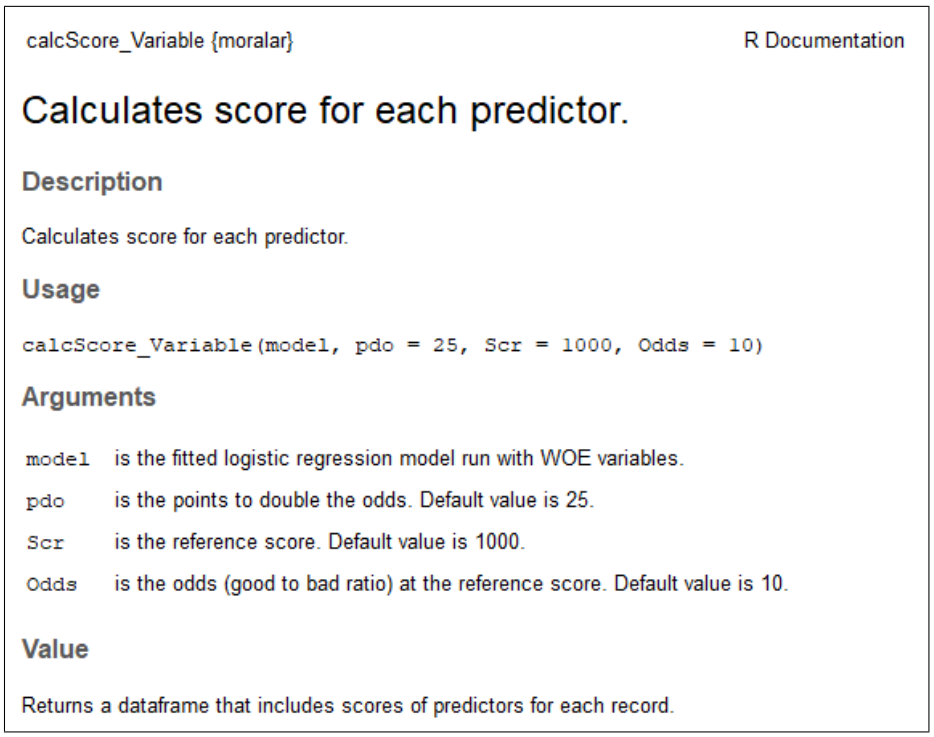
```
# calcScore() function is used to calculate the final scores for each observation.
# Training split
train_Final_SCR ← calcScore(predictTrain_logodds, pdo = 25, Scr = 1000,
Odds = 10)
colnames(train_Final_SCR) ← c("logodds", "Final_Score")

# Test split
test_Final_SCR ← calcScore(predictTest_logodds, pdo = 25, Scr = 1000, Odds
= 10)
colnames(test_Final_SCR) ← c("logodds", "Final_Score")

# Final scores of the first 10 observation are returned.
head (train_Final_SCR, 10)
```

Figure 9.23. R code to calculate the final scores from the predicted log odds for the training and test sets.

Table 9.8. Predicted log odds and final scores for the first ten observations in the training set.

Obs	**logodds**	**Final_Score**
1	-4.825974	1091.012
2	-4.169305	1067.328
3	-3.773158	1053.04
5	-1.929783	986.554
6	-2.465434	1005.874
8	-3.171813	1031.351
10	-4.080351	1064.119
11	-4.323358	1072.884
13	-4.771598	1089.051
14	-6.127878	1137.968

If one wishes to examine the contribution of each variable to the final score, calcScore_Variable() can be utilized. A screenshot from the R documentation of this function is displayed in Figure 9.24. R code to calculate the scores of predictors for each observation is given in Figure 9.25. Calculated scores of predictors in the final model for the first observation in the training data set are tabulated in Table 9.9.



calcScore_Variable {moralar} R Documentation

Calculates score for each predictor.

Description

Calculates score for each predictor.

Usage

```
calcScore_Variable(model, pdo = 25, Scr = 1000, Odds = 10)
```

Arguments

model	is the fitted logistic regression model run with WOE variables.
pdo	is the points to double the odds. Default value is 25.
Scr	is the reference score. Default value is 1000.
Odds	is the odds (good to bad ratio) at the reference score. Default value is 10.

Value

Returns a dataframe that includes scores of predictors for each record.

Figure 9.24. Screenshot from the R documentation of calcScore_Variable() function.

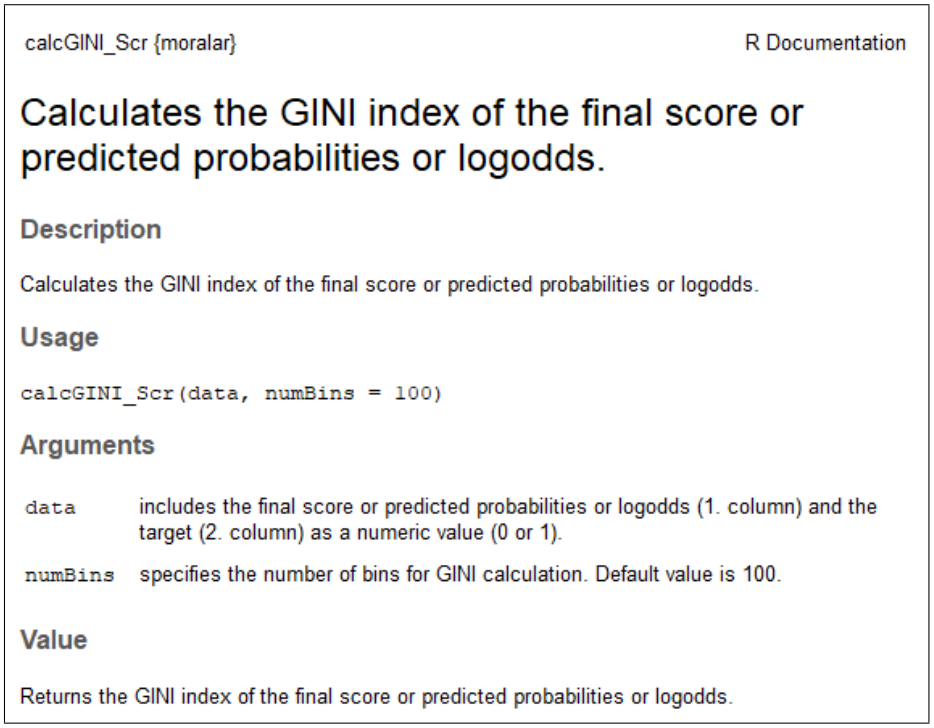
```
# calcScore_Variable() function is used to calculate the scores of predictors for
each observation.
train_Var_SCR ← calcScore_Variable(final_model, pdo = 25, Scr = 1000,
Odds = 10)
train_Var_SCR_obs1 ← t(head (train_Var_SCR, 1))
colnames(train_Var_SCR_obs1) ← c("Score")

# Scores that the predictors take are returned for the first observation.
train_Var_SCR_obs1
```

Figure 9.25. R code to calculate the scores of predictors for each observation.

Table 9.9. Scores of predictors for the first observation in the training data set.

Variable Name	**Score**
SCR_ALL_VAR_2	70.88
SCR_ALL_WOHL_VAR_1	70.15
SCR_CC_VAR_1	67.73
SCR_CC_VAR_3	76.33
SCR_CC_VAR_4	73.49
SCR_CC_VAR_5	57.37
SCR_CCOL_VAR_2	73.09
SCR_CHL_VAR_3	74.09
SCR_CHL_VAR_4	76.79
SCR_CL_VAR_4	64.55
SCR_CL_VAR_5	66.39
SCR_2D_CC_VAR_6_CCOL_VAR_1	84.49
SCR_2D_ALL_VAR_5_CC_VAR_6	74.32
SCR_2D_CC_VAR_2_OL_VAR_2	81.55
SCR_3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1	79.79

Earlier, Gini indices of the fitted models are calculated using the roc() function of pROC package [19]. It is also possible to calculate the Gini index of a fitted model using calcGINI_Scr() function using the final score calculated with calcScore() function, or directly predictions as log odds or probabilities. A screenshot from the R documentation of calcGINI_Scr() function can be found in Figure 9.26, whereas its usage is introduced with the R expressions given in Figure 9.27.



calcGINI_Scr {moralar} R Documentation

Calculates the GINI index of the final score or predicted probabilities or logodds.

Description

Calculates the GINI index of the final score or predicted probabilities or logodds.

Usage

```
calcGINI_Scr(data, numBins = 100)
```

Arguments

data	includes the final score or predicted probabilities or logodds (1. column) and the target (2. column) as a numeric value (0 or 1).
numBins	specifies the number of bins for GINI calculation. Default value is 100.

Value

Returns the GINI index of the final score or predicted probabilities or logodds.

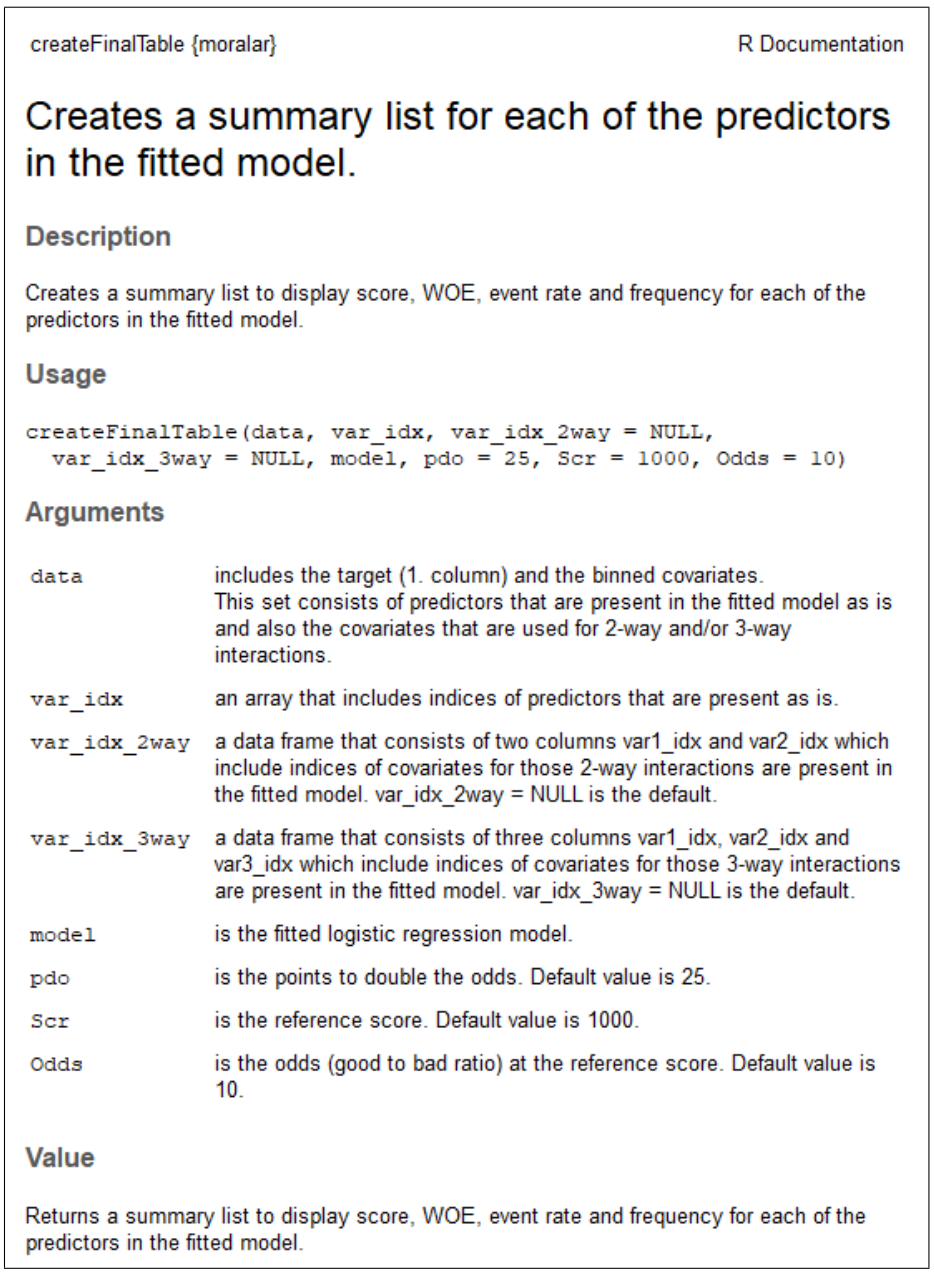
Figure 9.26. Screenshot from the R documentation of calcGINI_Scr() function.

```
# Final scores and the response variable are joined together and transformed into a data frame.
# Training split
train_Final_study ← cbind(train_Final_SCR[, 2], data_train_WOE$NPL_IN_12M)
colnames(train_Final_study) ← c("Final_Score", "NPL_IN_12M")
train_Final_study ← as.data.frame(train_Final_study)

# Test split
test_Final_study ← cbind(test_Final_SCR[, 2], data_test_WOE$NPL_IN_12M)
colnames(test_Final_study) ← c("Final_Score", "NPL_IN_12M")
test_Final_study ← as.data.frame(test_Final_study)

# calcGINI_Scr() function is used to calculate the GINI index of the final model.
calcGINI_Scr(train_Final_study, numBins = 100)
# 62.72
```

Figure 9.27. R code to calculate Gini of the final model from the final scores with calcGINI_Scr() function.

9.4. Final Model Summary

A very useful summary of the final model, which can be shared with other business units who will take advantage of the scorecard developed can be created with createFinalTable() function. A screenshot of its R documentation is given in Figure 9.28.



createFinalTable {moralar} R Documentation

Creates a summary list for each of the predictors in the fitted model.

Description

Creates a summary list to display score, WOE, event rate and frequency for each of the predictors in the fitted model.

Usage

```
createFinalTable(data, var_idx, var_idx_2way = NULL,
  var_idx_3way = NULL, model, pdo = 25, Scr = 1000, Odds = 10)
```

Arguments

data	includes the target (1. column) and the binned covariates. This set consists of predictors that are present in the fitted model as is and also the covariates that are used for 2-way and/or 3-way interactions.
var_idx	an array that includes indices of predictors that are present as is.
var_idx_2way	a data frame that consists of two columns var1_idx and var2_idx which include indices of covariates for those 2-way interactions are present in the fitted model. var_idx_2way = NULL is the default.
var_idx_3way	a data frame that consists of three columns var1_idx, var2_idx and var3_idx which include indices of covariates for those 3-way interactions are present in the fitted model. var_idx_3way = NULL is the default.
model	is the fitted logistic regression model.
pdo	is the points to double the odds. Default value is 25.
Scr	is the reference score. Default value is 1000.
Odds	is the odds (good to bad ratio) at the reference score. Default value is 10.

Value

Returns a summary list to display score, WOE, event rate and frequency for each of the predictors in the fitted model.

Figure 9.28. Screenshot from the R documentation of createFinalTable() function.

Usage of the createFinalTable() function is given in Figure 9.29.

```
# Indices of the predictors in the final model are determined.
which(colnames(data_train_WOE) %in% names(final_model$coefficients) [-
1])

# Main effects in the final model.
var_idx = c(3, 7, 8, 10, 11, 12, 15, 19, 20, 25, 26)

# 2-way interactions in the final model.
var_idx_2way = data.frame(var1_idx = c(14, 6, 9), var2_idx = c(13, 13, 28))

# 3-way interactions in the final model.
var_idx_3way = data.frame(var1_idx = c(2), var2_idx = c(21), var3_idx =
c(27))

# createFinalTable() is used to present a summary of the final model.
createFinalTable(Var_all_GRP_train, var_idx, var_idx_2way, var_idx_-
3way, final_model,
pdo = 25, Scr = 1000, Odds = 10)
```

Figure 9.29. R code to obtain the summary table of the final model.

The final summary table is presented in Table 9.10 which includes bin indices of each variable along with related score, WOE value, event rate, and frequency.

Table 9.10. The summary table for the final model.

$ALL_VAR_2				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	78.18	0.2985	0.0198	0.2727
2	72.52	0.0967	0.0241	0.2386
3	70.88	0.0383	0.0255	0.3099
4	56.74	-0.4662	0.0416	0.1788

Table 9.10. The summary table for the final model (cont.).

$ALL_WOHL_VAR_1				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	81.55	0.6238	0.0144	0.3719
2	70.15	0.0181	0.026	0.1695
3	58.66	-0.5925	0.0469	0.1498
4	47.92	-1.1631	0.0801	0.0943
5	88.37	0.9862	0.01	0.2145

$CC_VAR_1				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	73.33	0.333	0.0191	0.6795
2	67.73	-0.1967	0.032	0.2101
3	60.58	-0.8731	0.0611	0.1104

$CC_VAR_3				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	69.91	0.0098	0.0262	0.216
2	74.22	0.439	0.0172	0.1683
3	76.33	0.6491	0.014	0.2591
4	65.51	-0.4278	0.04	0.3566

$CC_VAR_4				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	61.61	-0.5883	0.0467	0.2321
2	68.93	-0.0633	0.0282	0.2572
3	73.49	0.2638	0.0205	0.1567
4	77.6	0.5584	0.0153	0.1771
5	78.87	0.6499	0.014	0.1769

Table 9.10. The summary table for the final model (cont.).

$CC_VAR_5				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	73.17	0.2522	0.0207	0.8687
2	57.37	-0.934	0.0647	0.1313

$CCOL_VAR_2				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	62.9	-0.7084	0.0523	0.1549
2	69.26	-0.056	0.028	0.2542
3	73.09	0.3363	0.0191	0.5909

$CHL_VAR_3				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	61.51	-0.8395	0.0592	0.1183
2	74.09	0.4326	0.0173	0.5106
3	69.02	-0.0796	0.0286	0.3711

$CHL_VAR_4				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	65.02	-0.1441	0.0305	0.5519
2	76.79	0.2097	0.0216	0.4481

$CL_VAR_4				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	74.33	0.1166	0.0236	0.1672
2	64.55	-0.1357	0.0302	0.228
3	62.71	-0.1832	0.0316	0.1517
4	73.84	0.1041	0.0239	0.4531

Table 9.10. The summary table for the final model (cont.).

$CL_VAR_5				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	65.66	-0.231	0.0331	0.1899
2	66.39	-0.1906	0.0319	0.2513
3	75.97	0.3431	0.0189	0.2849
4	71.04	0.0684	0.0248	0.2739

$‘2D_CC_VAR_6_CCOL_VAR_1‘				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	59.39	-0.7154411	NA	0.0006
2	55.26	-0.9991	0.0688	0.0293
3	65.97	-0.2641	0.0342	0.1081
4	53.98	-1.0872	0.0746	0.0229
5	63.31	-0.4466	0.0408	0.0819
6	75.03	0.3587	0.0186	0.0446
7	59.25	-0.7253	0.0532	0.0226
8	71.32	0.1038	0.0239	0.0702
9	78.74	0.6131	0.0145	0.0491
10	59.29	-0.7223	0.053	0.0198
11	71.93	0.1455	0.023	0.0687
12	84.49	1.0086	0.0098	0.0622
13	54.06	-1.0817	0.0743	0.0162
14	73.43	0.2484	0.0208	0.0776
15	86.14	1.1214	0.0088	0.0965
16	47.66	-1.5213	0.1107	0.0098
17	67.37	-0.1673	0.0311	0.0741
18	91.05	1.4587	0.0063	0.1457

Table 9.10. The summary table for the final model (cont.).

$‘2D_ALL_VAR_5_CC_VAR_6‘				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	50.6	-1.9906	0.166	0.0045
2	55.23	-1.5112	0.1097	0.0124
3	58.47	-1.1756	0.081	0.0145
4	60.84	-0.9299	0.0645	0.015
5	59.84	-1.0328	0.071	0.016
6	55.72	-1.4606	0.1049	0.017
7	66.28	-0.3655	0.0377	0.1336
8	67.61	-0.2279	0.033	0.1371
9	73.07	0.3377	0.019	0.1274
10	74.32	0.4676	0.0168	0.1357
11	75.74	0.6145	0.0145	0.1743
12	77.68	0.8158	0.0119	0.2126

Table 9.10. The summary table for the final model (cont.).

$‘2D_CC_VAR_2_OL_VAR_2‘				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	58.77	-0.8545	0.0601	0.0412
2	56.86	-1.0027	0.069	0.0226
3	59.03	-0.8348	0.059	0.0259
4	61.94	-0.6089	0.0476	0.0274
5	64.9	-0.3802	0.0383	0.0381
6	59.7	-0.7827	0.0562	0.0221
7	62.34	-0.5785	0.0463	0.0249
8	71.3	0.115	0.0237	0.0287
9	67.86	-0.1507	0.0307	0.031
10	71.77	0.1518	0.0228	0.0468
11	64.43	-0.4165	0.0396	0.0441
12	70.45	0.0493	0.0252	0.0484
13	74.18	0.3386	0.019	0.0535
14	74.53	0.3651	0.0185	0.0623
15	81.55	0.9087	0.0108	0.1111
16	63.22	-0.5103	0.0433	0.0759
17	76.61	0.526	0.0158	0.0547
18	75.23	0.4198	0.0176	0.057
19	82.68	0.9961	0.0099	0.0648
20	86.08	1.2592	0.0077	0.1195

Table 9.10. The summary table for the final model (cont.).

$‘3D_ALL_VAR_1_CHL_VAR_5_OL_VAR_1‘				
Bin	**Score**	**WOE**	**Event_Rate**	**Frequency**
1	43.51	-1.6031	0.119	0.0021
2	60.58	-0.5626617	NA	0.0012
3	45.61	-1.4755	0.1063	0.0043
4	56.51	-0.8109	0.0577	0.0353
5	63.09	-0.41	0.0394	0.0043
6	62.64	-0.4374	0.0404	0.0453
7	61.86	-0.4849	0.0423	0.0144
8	65.17	-0.2832	0.0348	0.0097
9	71.21	0.0855	0.0244	0.0439
10	43.23	-1.6205	0.1209	0.0031
11	43.04	-1.6323	0.1221	0.0022
12	47.03	-1.3885	0.0983	0.006
13	68.69	-0.0685	0.0283	0.0126
14	65.92	-0.2373	0.0333	0.0051
15	81.79	0.7305	0.0129	0.0315
16	60.32	-0.5788	0.0463	0.015
17	74.1	0.2614	0.0205	0.0165
18	76.05	0.3804	0.0183	0.0613
19	42.1	-1.6893	0.1284	0.0219
20	49.46	-1.2407	0.086	0.0193
21	62.57	-0.4411	0.0406	0.023
22	69.08	-0.0444	0.0276	0.0252
23	80.77	0.6683	0.0137	0.0185
24	85.23	0.9399	0.0105	0.071
25	74.45	0.2827	0.0201	0.0887
26	77.71	0.4814	0.0165	0.1324
27	79.79	0.6081	0.0146	0.286

9.5. Score Distributions

Distribution of scores are plotted with the R code given in Figure 9.30 for training and test sets separately. From Figure 9.31, it can see seen that the distribution of scores are similar to each other.

```
ggplot(train_Final_study, aes(x = Final_Score))
+ geom_histogram(breaks = seq(900, 1150, by = 10), colour = "black", fill =
"white", aes(y = ..density..))
+ geom_density(color = "red")
+ labs(title = "Distribution of Scores", x = "Scores", y = "Count")
```

Figure 9.30. R code to plot distribution of scores for training and test sets in percentage.

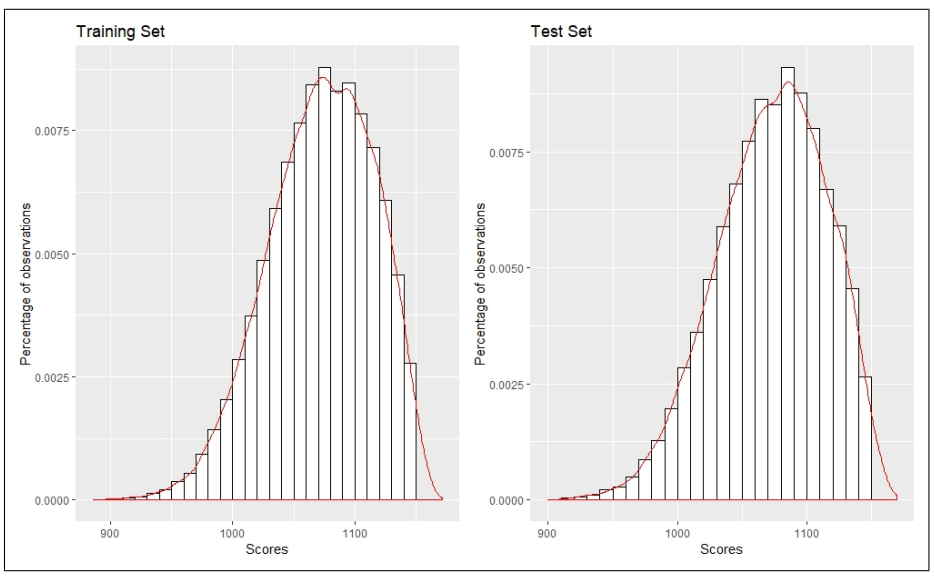


Figure 9.31. Distribution of scores for training and test sets in percentage.

Densities for default and nondefault groups are plotted with the R expression in Figure 9.32 and Figure 9.33 is obtained. For both training and test sets, nondefault and default groups overlap. Defaults have lower scores in general, however a perfect separation is not observed. A statistical model developer always aims to reduce the extent of overlapping, however he also knows that a perfect separation is almost impossible.

```
ggplot(train_Final_study, aes(x = Final_Score, fill = as.factor(NPL_IN_-
12M))) +
geom_density(alpha=.3) +
labs(title="Training Set", x="Score", y="Percentage of observations") +
scale_fill_manual(values=c("#00CED1", "#FF0000")) +
guides(fill=guide_legend(title="NPL_IN_12M")) +
theme(legend.position = "top")
```

Figure 9.32. R code to plot score densities for default and nondefault groups.

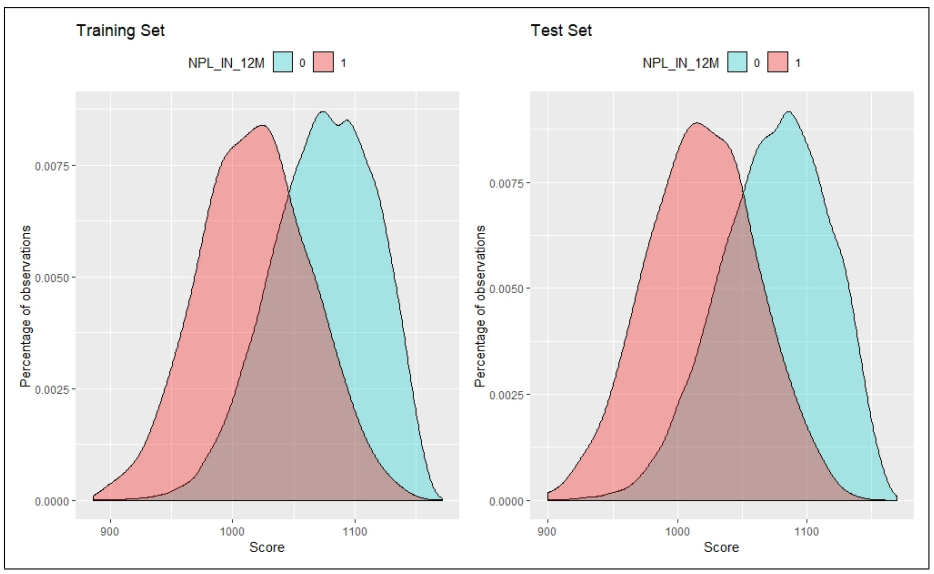


Figure 9.33. Score densities of default and nondefault groups for training and test splits for the final model.

In Figure 9.34, plots of score densities of default and nondefault groups are shared for the reduced model without interaction terms, for both training and test sets. Comparing Figure 9.33 and Figure 9.34, one can see that distribution of defaults are similar in both models, although distribution of nondefaults more or less differ. For the reduced model without interactions, distribution of nondefaults has a higher peak. That may mean that number of nondefault with higher scores is less. We will further examine whether this is the case using following Kolmogorov-Smirnov plots.

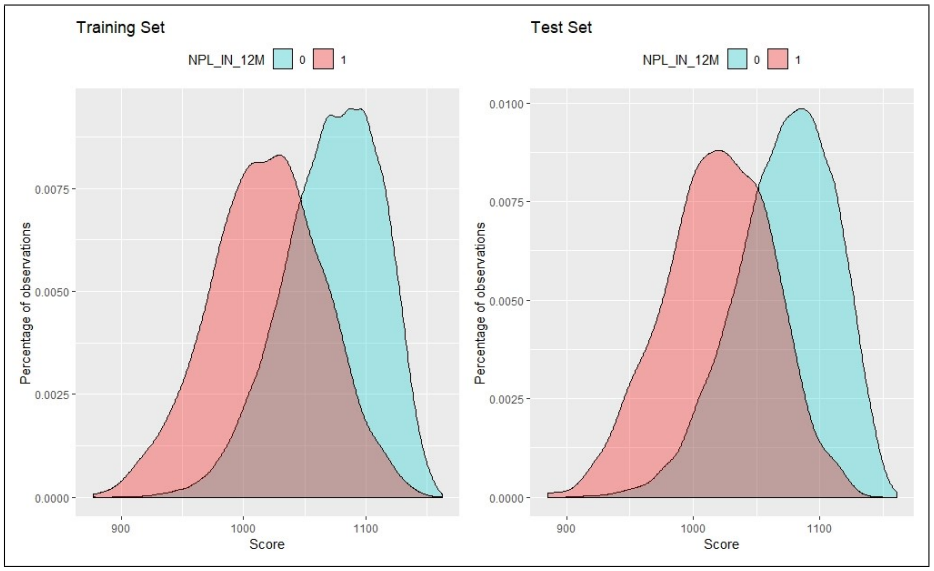


Figure 9.34. Score densities of default and nondefault groups for training and test splits for the reduced model without interaction terms.

Finally, how the Kolmogorov-Smirnov test statistic is calculated is presented in Figure 9.35. KS-statistic is calculated as 0.47791 for the final model. Graphical representation of KS-statistic is given in Figure 9.36. The width of the gap between density lines of represent the discriminatory power of the final model.

```
# create ECDF of data
cdf_nondef ← ecdf(train_Final_study[train_Final_study$NPL_IN_12M ==
0, 1])
cdf_def ← ecdf(train_Final_study[train_Final_study$NPL_IN_12M == 1,
1])
minMax ← seq(min(train_Final_study[, 1]), max(train_Final_study[, 1]),
length.out = length(train_Final_study[, 1]))
x0 ← minMax[which(abs(cdf_nondef(minMax) - cdf_def(minMax)) ==
max(abs(cdf_nondef(minMax) - cdf_def(minMax))))]
y0 ← cdf_nondef(x0)
y1 ← cdf_def(x0)
```

Figure 9.35. R code to calculate the Kolmogorov-Smirnov statistic and plot for graphical representation.

```
> y1 - y0
[1] 0.4779098

p ← ggplot(train_Final_study, aes(x = Final_Score, group = NPL_IN_12M,
color = as.factor(NPL_IN_12M))) +
stat_ecdf(size = 1) +
theme_bw(base_size = 12) +
geom_segment(aes(x = x0[1], y = y0[1], xend = x0[1], yend = y1[1]), color =
"black") +
geom_point(aes(x = x0[1], y = y0[1]), color = "black", size = 4) +
geom_point(aes(x = x0[1], y = y1[1]), color = "black", size = 4) +
scale_color_manual(values=c("#00CED1", "#FF0000")) +
xlab("Final Score") +
ylab("ECDF") +
# ggtitle("K-S Test: Training") +
guides(color = guide_legend(title = "NPL_IN_12M")) +
theme(legend.position = "top")
xnondef ← quantile (train_Final_study[train_Final_study$NPL_IN_12M
== 0, 1])[2:4]
xdef ← quantile (train_Final_study[train_Final_study$NPL_IN_12M == 1,
1])[2:4]

df ← data.frame(vnondef = c(round (xnondef[1]), round (xnondef[2]), round
(xnondef[3])),
vdef = c(round (xdef[1]), round (xdef[2]), round (xdef[3])),
xnondef,
xdef,
y = c(0.25, 0.50, 0.75),
NPL_IN_12M = c(0, 0, 0))

p + geom_segment(data=df, aes(x = xdef[1], y = y[1], xend = xnondef[1], yend
= y[1]), linetype = "dashed", color = "red") +
geom_segment(data=df, aes(x = xdef[2], y = y[2], xend = xnondef[2], yend =
y[2]), linetype = "dashed", color = "red") +
geom_segment(data=df, aes(x = xdef[3], y = y[3], xend = xnondef[3], yend =
y[3]), linetype = "dashed", color = "red") +
annotate(geom="label", x = df$xdef, y = df$y, label = df$vdef, color = "red",
size = 3, hjust = 1) +
annotate(geom="label", x = df$xnondef, y = df$y, label = df$vnondef, color =
"#00CED1", size = 3, hjust = 0) +
annotate(geom="label", x = x0[1], y = (y0[1] + y1[1]) / 2, label = "0.47791",
size = 4)
```

Figure 9.35. R code to calculate the Kolmogorov-Smirnov statistic and plot for graphical representation (cont.).

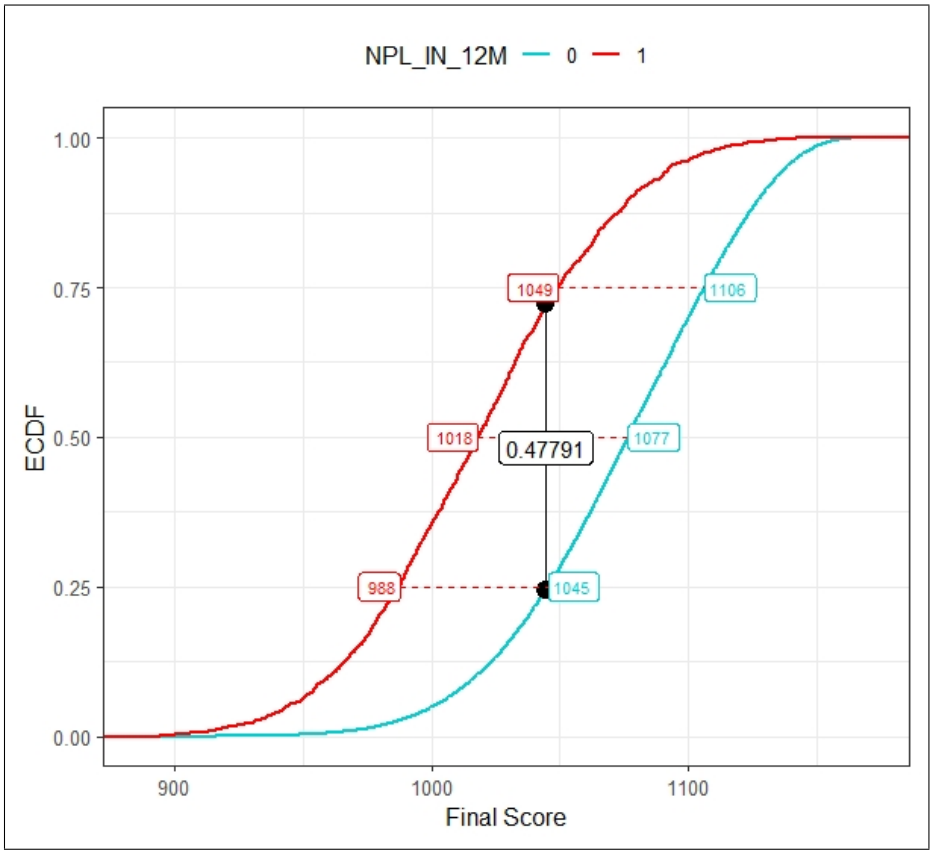


Figure 9.36. Kolmogorov-Smirnov statistic for final model.

Graphical representation of KS-statistic is given in Figure 9.37 for the reduced model without interaction terms. When these plots are compared, 25% of all defaults take a score less than 988, whether interaction terms are included or not. Then, a slight improvement can be observed in the model with interactions, since 50% of all defaults have a score less than 1018, whereas this score value is higher, 1020, for the reduced model without interactions. For the 75% of defaults, these values are 1049 and 1050, respectively. A model where bad records have lower scores is preferable. On the other hand, a model where nondefaults take higher scores is a better model in terms of discriminatory power. Although, for the first 50% of nondefaults, scores are similar for both models, it can be seen that upper 25% of nondefaults take scores greater than 1106 for the final model with interactions, whereas this score is 1102 for the reduced model without interactions. Looking at Gini indices, we already showed that interactions have an impact on the discriminatory power, yet limited. With these plots we confirmed this result.

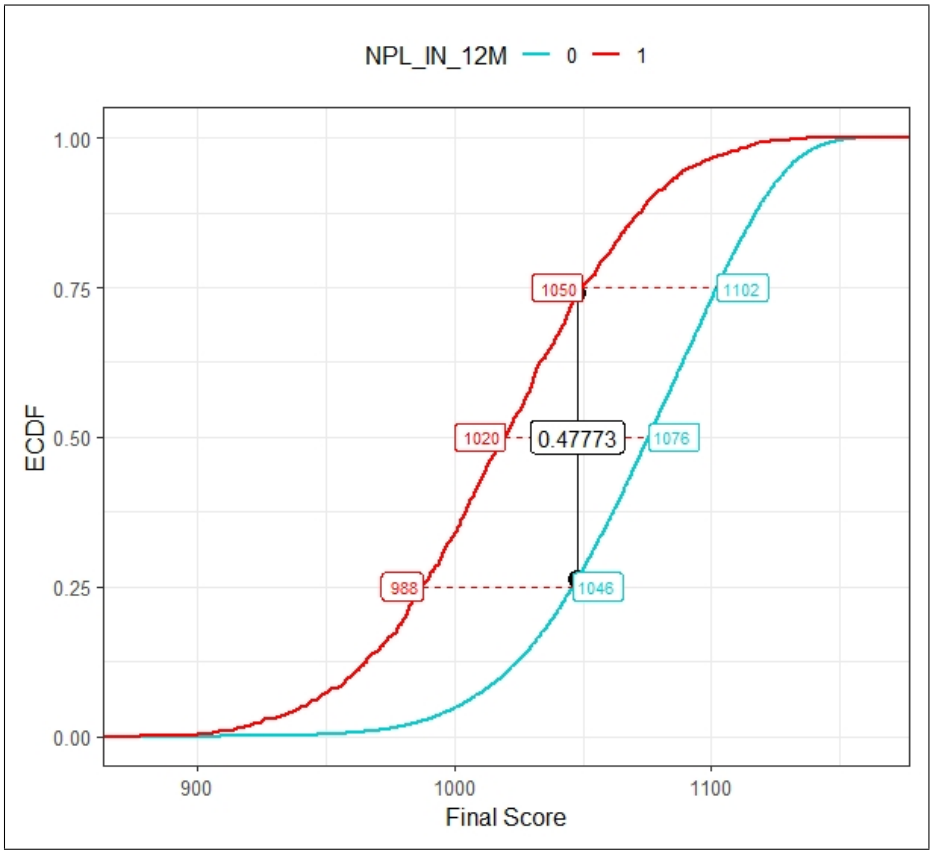


Figure 9.37. Kolmogorov-Smirnov statistic for the reduced model without interaction terms.

10. CONCLUSION

In general, academics are not able to execute research on credit scoring with real data because of confidentiality issues or lack of adequate collaboration between universities/research institutes and industry, or they often utilize infamous, exploited data sets like Statlog (German Credit Data) for their studies in the field of credit scoring. On the other hand, practitioners are privileged to access various data sources. However, they often cannot perform thorough analyses, look for alternative approaches or change model development practice radically because of tight deadlines, a prior failed attempt to create business value etc. Thus, we aimed to blend academic and practical perspectives in this thesis that is comprised of following three main components.

- Obtaining the Data Set: We obtained a data set of cash loans granted between 01/07/2014 and 30/06/2015 with some specified restrictions using Credit Bureau records in Turkey. We paid utmost attention to respect for confidentiality of lenders and borrowers. We made sure that the final development data set was capable of representing whole sector at that time. All the stages during data preparation step were clearly described. We did not share SQL queries; nevertheless, anyone who can use SQL and is familiar with the Credit Bureau tables, can create a similar development data set.
- Usage of a Free Software Environment: Most of the financial institutions spend large amounts of money to buy popular statistical and analytical software, necessary hardware to build their environment, and for their maintenance, because these specialized tools are reliable, fast, user-friendly etc. Nevertheless, free software programs become widespread as they provide flexibility, scalability, collaboration and nourish creativity besides many financial benefits. Thus, we carried out the whole study with one of the most popular software environments among statisticians and data miners, R. Our emphasis, yet, is not specifically showing usage and advantages of R, because all the analyses and calculations during a scoring model development process can be executed with another free software environment just tweaking the codes shared.

- Analyses and New Approaches for Scoring Model Development: With the real data set using a free software environment, we first examined the extent of the information loss, when variables, that we called raw variables, are grouped into a few categories, since categorizing variables is widespread in practice. We concluded that using grouped variables instead of raw variables does not necessarily cause a loss of information in terms of Gini index. We also showed, how a practitioner can categorize raw variables after viewing simple bar plots, without using any intelligent algorithm, although categorizing each variable manually is not practical in case there are hundreds of raw variables. We also showed that using WOE variables instead of using directly GRP variables has some advantages, and often provides almost the same discriminatory power.

Seeking to obtain a better model regarding the discriminatory power, we included interaction terms into regression models. We showed that it is straightforward using GRP variables. Some practitioners state that they prefer using GRP variables to WOE variables, just because it makes possible to incorporate interaction terms. In order to automatically calculate WOE versions of grouped variables we introduced new approaches and practical functions. We also proposed the so-called Neighbour's approach to calculate WOE values for configurations of categories of constructive main effects that are not available in the training set, but possible to be encountered in reality.

Finally, in Chapter 9, we aggregated previous studies to obtain final models. It appeared that the contribution of interaction terms selected is limited. However, using another data set of variables, impact of interaction terms would grow, especially in a setting where interactions reveal hidden information in the data that is not possible to obtain by only using main effects.

REFERENCES

1. Reinsel, D., J. Gantz and J. Rydning, *Data Age 2025: The Evolution of Data to Life-Critical*, International Data Corporation (IDC), Massachusetts, 2017.

2. Abdou, H. A. and J. Pointon, "Credit Scoring, Statistical Techniques and Evaluation Criteria: A Review of the Literature", *Intelligent Systems in Accounting, Finance and Management*, Vol. 18, No. 2-3, pp. 59–88, 2011.

3. Crone, S. F. and S. Finlay, "Instance Sampling in Credit Scoring: An Empirical Study of Sample Size and Balancing", *International Journal of Forecasting*, Vol. 28, pp. 224–238, 2012.

4. Nikolic, N., N. Zarkic-Joksimovic, D. Stojanovski and I. Joksimovic, "The Application of Brute Force Logistic Regression to Corporate Credit Scoring Models: Evidence from Serbian Financial Statements", *Expert Systems with Applications*, Vol. 40, pp. 5932–5944, 2013.

5. Lessmann, S., B. Baesens, H.-V. Seow and L. Thomas, "Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring: An Update of Research", *European Journal of Operational Research*, Vol. 247, pp. 124–136, 2015.

6. Hand, D. J. and W. Henley, "Statistical Classification Methods in Consumer Credit Scoring: A Review", *Journal of the Royal Statistical Society*, Vol. 160, No. 3, pp. 523–541, 1997.

7. Siddiqi, N., *Intelligent Credit Scoring*, John Wiley & Sons, Inc., New Jersey, second edn., 2017.

8. Thomas, L. C., D. Edelman and J. Crook, *Credit Scoring and its Applications*, SIAM, Philadelphia, 2002.

9. Huang, J., *Feature Selection in Credit Scoring - A Quadratic Programming Approach Solving with Bisection Method Based on Tabu Search*, Ph.D. Thesis, Texas A&M International University, 2014.

10. Hosmer, D. W. and S. Lemeshow, *Applied Logistic Regression*, John Wiley & Sons, Inc., New Jersey, second edn., 2000.

11. Bolton, C., *Logistic Regression and its Application in Credit Scoring*, M.Sc. Thesis, University of Pretoria, 2009.

12. Zeng, G., "A Necessary Condition for a Good Binning Algorithm in Credit Scoring", *Applied Mathematical Sciences*, Vol. 8, No. 65, pp. 3229–3242, 2014.

13. Rezác, M. and F. Rezác, "How to Measure the Quality of Credit Scoring Models", *Czech Journal of Economics and Finance (Finance a uver)*, Vol. 61, No. 5, pp. 486–507, 2011.

14. *Receiver operating characteristic (ROC) curve: practical review for radiologists.*, `https://openi.nlm.nih.gov/detailedresult.php?img=PMC2698108\_kjr-5-11-g002\&req=4`, accessed at April 2018.

15. Hand, D. J., "Good Practice in Retail Credit Scorecard Assessment", *Journal of the Operational Research Society*, Vol. 56, No. 9, pp. 1109–1117, 2005.

16. SAS Institute Inc., *Interactive Grouping Node*, 2017, `http://documentation.sas.com/?docsetId=emref\&docsetTarget=p1qzwz7onopjqcn11uc04i18urg7.htm\&docsetVersion=14.3\&locale=en`, accessed at June 2018.

17. SAS Institute Inc., *Scorecard Node*, 2017, `http://documentation.sas.com/?docsetId=emref\&docsetTarget=n181vl3wdwn89mn1pfpqm3w6oaz5.htm\&docsetVersion=14.3\&locale=en`, accessed at June 2018.

18. Max Kuhn, *The caret Package: Classification and Regression Training*,

2018, https://cran.r-project.org/web/packages/caret/caret.pdf, accessed at January 2019.

19. Xavier Robin and Natacha Turck and Alexandre Hainard and Natalia Tiberti and Frédérique Lisacek and Jean-Charles Sanchez, *The pROC Package: Display and Analyze ROC Curves*, 2018, https://cran.r-project.org/web/packages/pROC/pROC.pdf, accessed at January 2019.

20. Aguinis, H., R. Gottfredson and H. Joo, "Best-Practice Recommendations for Defining, Identifying, and Handling Outliers", *Organizational Research Methods*, Vol. 16, No. 2, pp. 270–301, 2013.

21. Nieuwenhuis, R., M. te Grotenhuis and B. Pelzer, "influence.ME: Tools for Detecting Influential Data in Mixed Effects Models", *The R Journal*, Vol. 4, No. 2, pp. 38–47, 2012.

22. John Fox and Sanford Weisberg and Brad Price, *The car Package: Companion to Applied Regression*, 2018, https://cran.r-project.org/web/packages/car/car.pdf, accessed at January 2019.

23. Fox, J. and S. Weisberg, *An R Companion to Applied Regression*, SAGE Publications, Inc., California, second edn., 2011.

24. Hadley Wickham and Winston Chang and Lionel Henry and Thomas Lin Pedersen and Kohske Takahashi and Claus Wilke and Kara Woo, *The ggplot2 Package: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2018, https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf, accessed at January 2019.

25. Hadley Wickham, *The plyr Package: Tools for Splitting, Applying and Combining Data*, 2016, https://cran.r-project.org/web/packages/plyr/plyr.pdf, accessed at January 2019.

26. Cohen, B. H., *Explaining Psychological Statistics*, John Wiley & Sons, Inc., New Jersey, fourth edn., 2013.

27. Marek Hlavac, *The stargazer Package: Well-Formatted Regression and Summary Statistics Tables*, 2018, `https://cran.r-project.org/web/packages/stargazer/stargazer.pdf`, accessed at January 2019.

28. John Fox and Sanford Weisberg and Michael Friendly and Jangman Hong, *The effects Package: Effect Displays for Linear, Generalized Linear and Other Models*, 2018, `https://cran.r-project.org/web/packages/effects/effects.pdf`, accessed at January 2019.

29. Brambor, T., W. Clark and M. Golder, "Understanding Interaction Models: Improving Empirical Analyses", *Political Analysis*, Vol. 14, No. 1, pp. 63–82, 2006.

30. Brian Ripley, *The MASS Package: Support Functions and Datasets for Venables and Ripley's MASS*, 2018, `https://cran.r-project.org/web/packages/MASS/MASS.pdf`, accessed at January 2019.

31. Subhash R. Lele and Jonah L. Keim and Peter Solymos, *The ResourceSelection Package: Resource Selection (Probability) Functions for Use-Availability Data*, 2018, `https://cran.r-project.org/web/packages/ResourceSelection/ResourceSelection.pdf`, accessed at January 2019.

APPENDIX A: R CODES OF FUNCTIONS IN MORALAR PACKAGE

```
#' Outlier treatment using IQR of box plot.
#'
#' Eliminates records that are below Q1 - coef * IQR and above Q3 + coef *
IQR.
#'
#' @param input includes the target (1. column) and the covariate to be treated.
#' @param coef coef times the IQR is used for treatment.
#' @return Returns a matrix having two columns that includes the target as
the first column, and the treated covariate as the second column.
#'
#' @examples
#' \dontrun{
#' boxplot_out(input, coef=1.5)
#' }
#' @export
boxplot_out ← function(input, coef) {

boxplot_stats ← boxplot.stats(input[, 2], coef = 0, do.conf = TRUE, do.out =
TRUE)

iqr ← diff(boxplot_stats$stats[c(2, 4)]) #interquartile range

if (iqr != 0) {
 out ← input[, 2] < (boxplot_stats$stats[2] - coef * iqr) | input[, 2] > (boxplot_-
stats$stats[4] + coef * iqr)
 output ← input[!out, ]
 return(output)
} else{
 #print('Not applicable: IQR = 0')
 output ← NA
 return(output)
}
}
```

Figure A.1. R code of boxplot_out() function.

```
#' Outlier treatment by capping the maximum value that a variable can take.
#'
#' Capping the maximum value that a variable can take.
#' First, the values of variables at their 99th, 99.5th and 99.9th percentiles are specified.
#' Then, larger values are assigned to the specified values.
#'
#' @param input includes the target (1. column) and the covariate to be treated.
#' @return Returns a matrix having four columns:
#' \itemize{
#' \item First column is the target.
#' \item Second column is the value of the covariate, if the value at the 99.9th percentile is used for capping.
#' \item Third column is the value of the covariate, if the value at the 99.5th percentile is used for capping.
#' \item Fourth column is the value of the covariate, if the value at the 99th percentile is used for capping.
#' }
#'
#' @examples
#' \dontrun{
#' capped_out(input)
#' }
#' @export
capped_out ← function(input) {

output ← matrix(0, nrow = nrow(input), ncol = 4)
#colnames(output) ← c("NPL_IN_12M" , "capped at 99.9%", "capped at 99.5%", "capped at 99%")
colnames(output) ← c("NPL_IN_12M" , "capped_999", "capped_995", "capped_990")

caps ← quantile(input[, 2], probs = c(.99, .995, .999), na.rm = T)

output[, 1] ← input[, 1]
input[input[, 2] > caps[3], 2] ← caps[3]
output[, 2] ← input[, 2]
input[input[, 2] > caps[2], 2] ← caps[2]
output[, 3] ← input[, 2]
input[input[, 2] > caps[1], 2] ← caps[1]
output[, 4] ← input[, 2]

return(as.data.frame(output))
}
```

Figure A.2. R code of capped_out() function.

```
#' Outlier treatment by using Cook's distance.
#'
#' Specifies influential observations and applies an outlier treatment.
#'
#' @param input includes the target (1. column) and the covariate to be treated.
#' @param exclude If TRUE, excludes influential observations according to the
Cook's distance.
#' If FALSE, caps the influential observations according to the Cook's distance
with the maximum value of the remaining observations.
#' @param method If TRUE, influential observations are determined by cooksd
> qf(.1, n, n - p).
#' if FALSE, influential observations are determined by cooksd > 4 / (n - p -
1).
#' @return Returns a matrix contains the target as the first column, and the
treated covariate as the second column.
#'
#' @examples
#' \dontrun{
#' CookD_out(input, exclude = TRUE, method = TRUE)
#' }
#' @export
CookD_out ← function(input, exclude = TRUE, method = TRUE) {

mod ← glm(NPL_IN_12M ∼ ., data = input, family=binomial(link = "logit"))
cooksd ← cooks.distance(mod)

n ← nrow(input)
p ← length(input) - 1

if (method == TRUE) {
 if (n < 10000) {
  influential ← as.numeric(names(cooksd)[cooksd > qf(.1, n, n - p)])
 } else {
  influential ← as.numeric(names(cooksd)[cooksd > 1])} # for big n, qf(.1, n,
n - p) converges 1.
} else {
 influential ← as.numeric(names(cooksd)[cooksd > 4 / (n - p - 1)])
}

x <← input

if (length(influential) = 0) {
 if (exclude = TRUE) {
  CookD_cap ← max(input[-influential, 2])
  x ← input
  x[x[, 2] > CookD_cap, 2] ← CookD_cap
 } else {
  x ← input[-influential, ]
 }
}

return(x)
}
```

Figure A.3. R code of CookD_out() function.

```
#' Sorts the bins of covariates in a decreasing order of the event rates.
#'
#' Sorts the bins (attributes) of covariates (characteristics) in a decreasing order
of the proportion of events.
#'
#' @param data includes the target (1. column) and the binned covariates to
be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number
of levels of the binned covariate).
#' @param calcGINI_flag Set to TRUE, if the function is used before GINI
calculation.
#' @return Returns the sorted bins of covariates in a decreasing order of the
event rates.
#'
#' @examples
#' \dontrun{
#' orderAttributes(data, calcGINI_flag = FALSE)
#' }
#' @export
orderAttributes ← function(data, calcGINI_flag = FALSE) {

numVar ← ncol(data[,-1])

mylist ← calcWOE(data)[[1]]

numLevels ← sapply(mylist, function(x) ncol(x))

mylist_reordered ← vector("list", length(names(mylist)))
names(mylist_reordered) ← names(mylist)

for (i in 1:numVar) {
 mylist_reordered[[i]] ← mylist[[i]][, c(order(mylist[[i]][2,] / (mylist[[i]][1,] +
mylist[[i]][2,]), decreasing = TRUE))]

 if (calcGINI_flag == TRUE) {
  colnames(mylist_reordered[[i]]) ← c(1:numLevels[i])
 }
}

return(mylist_reordered)
}
```

Figure A.4. R code of orderAttributes() function.

```
#' Calculates WOE values.
#'
#' Calculates the Weight of Evidence (WOE) values for binned covariates in the
data set.
#'
#' @param data includes the target (1. column) and the binned covariates to
be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number
of levels of the binned covariate).
#' @return Returns a list consisting of two components:
#' \itemize{
#' \item First component includes default number, nondefault number and
WOE values for each group.
#' \item Second component is a dataframe of WOE values.
#' }
#' @examples
#' \dontrun{
#' calcWOE(data)
#' }
#' @export
calcWOE ← function(data) {

numLevels ← sapply(data, function(x) length(unique(x)))[-1]
numVar ← ncol(data[, -1])
sumDef ← length(data[data[, 1] == 1, 1])
sumNonDef ← length(data[data[, 1] == 0, 1])

mylist.names ← colnames(data[, -1])
mylist ← vector("list", length(mylist.names))
names(mylist) ← mylist.names

for (i in 1:numVar) {
 mylist[[i]] ← matrix(0, nrow = 3, ncol = numLevels[i], dimnames =
list(c("numNonDef", "numDef", "WOE"), c(1:numLevels[i])))

 j = 1
 while (j <= numLevels[i]) {
  mylist[[i]][1, j] ← round (length (data[data[, i + 1] == j & data[, 1] == 0, 1]),
0)
  mylist[[i]][2, j] ← round (length (data[data[, i + 1] == j & data[, 1] == 1, 1]),
0)
  mylist[[i]][3, j] ← round (log ((mylist[[i]][1, j] / sumNonDef) / (mylist[[i]][2, j]
/ sumDef)), 4)
  j ← j + 1
 }
}
```

Figure A.5. R code of calcWOE() function.

```
WOE_DF_names ← c(colnames(data)[1], sapply(X = 1:numVar, FUN
= function (i) {paste0 ("WOE_", substr(colnames(data[,-1])[i], 5,
nchar(colnames(data[,-1])[i]))})))
WOE_DF ← data.frame(matrix(0, nrow = nrow(data), ncol = numVar + 1))
colnames(WOE_DF) ← WOE_DF_names
WOE_DF[, 1] ← data[, 1] # Target is added to the data frame.

for (i in 1:numVar) {
 for (j in 1:numLevels[i]) {
  WOE_DF[data[, i + 1] == j, i + 1] ← mylist[[i]][3, j]
 }
}

return(list(mylist, WOE_DF))
}
```

Figure A.5. R code of calcWOE() function (cont.).

```
#' Assigns WOE values to a new set.
#' Assigns the Weight of Evidence (WOE) values calculated with calcWOE()
function for the training set to a new set (test set).
#'
#' @param data_train training set that includes the target (1. column) and
the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number
of levels of the binned covariate).
#' @param data_test test set that includes the target (1. column) and the
binned covariates as in the training set.
#' @return Returns a list consisting of two components:
\itemize{
\item First component includes default number, nondefault number and WOE
values for each group calculated for the training set.
\item Second component is a dataframe of WOE values for the new set (test
set).
#' }
#' @examples
#' \dontrun{
#' assignWOE(data_train, data_test)
#' }
#' @export
assignWOE ← function(data_train, data_test) {

data_WOE ← calcWOE(data_train)[[1]]
numLevels ← sapply(data_train, function(x) length(unique(x)))[-1]
numVar ← ncol(data_test[, -1])

WOE_DF_names ← c(colnames(data_test)[1], sapply(X = 1:numVar, FUN
= function (i) {paste0 ("WOE_", substr(colnames(data_test[,-1])[i], 5,
nchar(colnames(data_test[,-1])[i])))}))
WOE_DF ← data.frame(matrix(0, nrow = nrow(data_test), ncol = numVar +
1))
colnames(WOE_DF) ← WOE_DF_names
WOE_DF[, 1] ← data_test[, 1] # Target is added to the data frame.

for(i in 1:numVar){
 for (j in 1:numLevels[i]) {
  WOE_DF[data_test[, i + 1] == j, i + 1] ← data_WOE[[i]][3, j]
 }
}

return(list(data_WOE, WOE_DF))
}
```

Figure A.6. R code of assignWOE() function.

```
#' Calculates WOE values for 2-way interactions.
#'
#' Calculates the Weight of Evidence (WOE) values for 2-way interactions in the data set.
#'
#' @param data includes the target (1. column) and the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
#' @param var_idx a data frame consisting of two columns var1_idx and var2_idx which include indices of covariates
#' for those 2-way interactions are studied.
#' @return Returns a list consisting of three components:
#' \itemize{
#' \item First component includes default number, nondefault number and WOE values for each newly created group for 2-way interactions.
#' \item Second component is a list consisting of vectors of GRP values for each interaction.
#' \item Third component is a list consisting of vectors of WOE values for each interaction.
#' }
#'
#' @examples
#' \dontrun{
#' calcWOE_2D_interaction(data,
#' var_idx = data.frame(var1_idx = c(2, 4, 6), var2_idx = c(3, 5, 7)))
#' }
#' @export
calcWOE_2D_interaction ← function(data, var_idx) {

sumDef ← length(data[data[, 1] == 1, 1])
sumNonDef ← length(data[data[, 1] == 0, 1])

mylist_reordered ← orderAttributes(data, calcGINI_flag = FALSE)

numLevels ← sapply(mylist_reordered, function(x) ncol(x))

mylist2.names ← paste(colnames(data)[var_idx[, 1]], 'X', colnames(data)[var_idx[, 2]])
mylist2 ← vector("list", length(mylist2.names))
names(mylist2) ← mylist2.names

for (i in 1:nrow(var_idx)) {
 mylist2[[i]] ← matrix(0, nrow = 3, ncol = numLevels[var_idx[i, 1] - 1] * numLevels[var_idx[i, 2] - 1], dimnames = list(c("numNonDef", "numDef", "WOE"), c(apply(expand.grid(colnames(mylist_reordered[[var_idx[i, 2] - 1]]),
 colnames(mylist_reordered[[var_idx[i, 1] - 1]]))[, c(2, 1)], 1, function(x) paste(x, collapse = " x "))))

 col ← 0
```

Figure A.7. R code of calcWOE_2D_interaction() function.

```
  for (x in as.numeric(colnames(mylist_reordered[[var_idx[i, 1] - 1]]))) {
   for (y in as.numeric(colnames(mylist_reordered[[var_idx[i, 2] - 1]]))) {
    col ← col + 1

    mylist2[[i]][1, col] ← round(length(data[data[, var_idx[i, 1]] == x & data[,
var_idx[i, 2]] == y & data[, 1] == 0, 1]), 0)
    mylist2[[i]][2, col] ← round(length(data[data[, var_idx[i, 1]] == x & data[,
var_idx[i, 2]] == y & data[, 1] == 1, 1]), 0)

    if ((mylist2[[i]][1, col] + mylist2[[i]][2, col] >= 100) & (mylist2[[i]][1, col] >
0) & (mylist2[[i]][2, col] > 0)) {
     mylist2[[i]][3, col] ← round(log((mylist2[[i]][1, col] / sumNonDef) /
(mylist2[[i]][2, col] / sumDef)), 4)
    }
    else {
     mylist2[[i]][3, col] ← NA
    }
   }
  }
}

mylist2_var_GRP ← vector("list", length(mylist2.names))
names(mylist2_var_GRP) ← mylist2.names

for (i in 1:nrow(var_idx)) {
 mylist2_var_GRP[[i]] ← matrix(0, nrow = nrow(data), ncol = 1)

 col ← 0

 for (x in as.numeric(colnames(mylist_reordered[[var_idx[i, 1] - 1]]))) {
  for (y in as.numeric(colnames(mylist_reordered[[var_idx[i, 2] - 1]]))) {
   col ← col + 1

   mylist2_var_GRP[[i]][data[, var_idx[i, 1]] == x & data[, var_idx[i, 2]] ==
y] ← col
  }
 }
}

numLevels2 ← sapply(mylist2, function(x) ncol(x))
mylist2_var_WOE ← vector("list", length(mylist2.names))
names(mylist2_var_WOE) ← mylist2.names

for (i in 1:nrow(var_idx)) {
 mylist2_var_WOE[[i]] ← matrix(0, nrow = nrow(data), ncol = 1)

 for (j in 1:numLevels2[i]) {
  mylist2_var_WOE[[i]][mylist2_var_GRP[[i]] == j] ← mylist2[[i]][3, j]
 }
}

return(list(mylist2, mylist2_var_GRP, mylist2_var_WOE))
}
```

Figure A.7. R code of calcWOE_2D_interaction() function (cont.).

```
#' 2D-Neigbours' approach: Specifies neighbours of each group of the covariates in case of a 2-way interaction.
#'
#' For a 2-way interaction, a covariate pattern consists of category of variable 1 (C1) and category of variable 2 (C2) = (C1, C2)
#' (C1, C2) may have up to 8 neigbours (C1 +/- 1:C2 +/- 1)
#'
#' @param data includes the target (1. column) and the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
#' @param var_idx var_idx a data frame consisting of two columns var1_idx and var2_idx which include indices of covariates
#' for those 2-way interactions are studied.
#' @return returns a list for each 2-way interaction. Each list also contains a list for every bin of the interaction.
#' A bin may have up to 8 neighbours, if exists.
#' @examples
#' \dontrun{
#' calcNeighbours_2D(data,
#' var_idx = data.frame(var1_idx = c(2, 4, 6), var2_idx = c(3, 5, 7)))
#' }
#' @export
calcNeighbours_2D ← function(data, var_idx) {

mylist_reordered ← orderAttributes(data, calcGINI_flag = FALSE)

mylist2 ← calcWOE_2D_interaction(data, var_idx)[[1]]

main.names ← paste(colnames(data)[var_idx[, 1]], 'X', colnames(data)[var_idx[, 2]])
Neighbours ← vector("list", length(main.names))
names(Neighbours) ← main.names

for (i in 1:nrow(var_idx)) {
 sub.names ← c(apply(expand.grid(colnames(mylist_reordered[[var_idx[i, 2] - 1]]), colnames(mylist_reordered[[var_idx[i, 1] - 1]]))[, c(2, 1)], 1, function(x) paste(x, collapse = " x ")))
 Neighbours[[i]] ← vector("list", length(sub.names))
 names(Neighbours[[i]]) ← sub.names

 col ← 0

 interactions ← expand.grid(colnames(mylist_reordered[[var_idx[i, 2] - 1]]), colnames(mylist_reordered[[var_idx[i, 1] - 1]]))[, c(2, 1)]
 colnames(interactions) ← c("Var1", "Var2")

 colnames_Var1 ← colnames(mylist_reordered[[var_idx[i, 1] - 1]])
 colnames_Var2 ← colnames(mylist_reordered[[var_idx[i, 2] - 1]])

 for (x in colnames_Var1) {
  for (y in colnames_Var2) {
   col ← col + 1
```

Figure A.8. R code of calcNeighbours_2D() function.

```
    Neighbours[[i]][[col]] ← cbind(if (y = tail(colnames_Var2, n = 1))
{mylist2[[i]][, which(interactions["Var1"] == x & interactions["Var2"] == col-
names_Var2[which(colnames_Var2 == y) + 1])]} else {NA}, #1
        if (y = head(colnames_Var2, n = 1)) {mylist2[[i]][,
which(interactions["Var1"] == x & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) - 1])]} else {NA}, #2
        if ((x= tail(colnames_Var1, n = 1)) & (y= tail(colnames_Var2, n = 1)))
{mylist2[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_-
Var1 == x) + 1] & interactions["Var2"] == colnames_Var2[which(colnames_-
Var2 == y) + 1])]} else {NA}, #3
        if (x = tail(colnames_Var1, n = 1)) {mylist2[[i]][,
which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 ==
x) + 1] & interactions["Var2"] == y)]} else {NA}, #4
        if ((x = tail(colnames_Var1, n = 1)) & (y = head(colnames_-
Var2, n = 1))) {mylist2[[i]][, which(interactions["Var1"] == colnames_-
Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) - 1])]} else {NA}, #5
        if ((x = head(colnames_Var1, n = 1)) & (y = tail(colnames_-
Var2, n = 1))) {mylist2[[i]][, which(interactions["Var1"] == colnames_-
Var1[which(colnames_Var1 == x) - 1] & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) + 1])]} else {NA}, #6
        if (x = head(colnames_Var1, n = 1)) {mylist2[[i]][,
which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 ==
x) - 1] & interactions["Var2"] == y)]} else {NA}, #7
        if ((x = head(colnames_Var1, n = 1)) & (y = head(colnames_-
Var2, n = 1))) {mylist2[[i]][, which(interactions["Var1"] == colnames_-
Var1[which(colnames_Var1 == x) - 1] & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) - 1])]} else {NA}) #8
    }
   }
  }

return(Neighbours)
}
```

Figure A.8. R code of calcNeighbours_2D() function (cont.).

```
#' Calculates WOE values for 2-way interactions using 2D-Neigbours' approach.
#'
#' After WOE values for 2-way interactions are specified using calcWOE_2D_-
interaction() function,
#' this function assigns a WOE value that is calculated using information on
the neighbours, that were determined with calcNeighbours_2D() function,
#' to the uncalculated WOE (NA).
#'
#' @param data includes the target (1. column) and the binned covariates to
be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number
of levels of the binned covariate).
#' @param var_idx a data frame consisting of two columns var1_idx and var2_-
idx which include indices of covariates
#' for those 2-way interactions are studied.
#' @return Returns a list consisting of three components:
#' \itemize{
#' \item First component includes default number, nondefault number and
WOE values for each newly created group for 2-way interactions.
#' \item Second component is a list consisting of vectors of GRP values for each
interaction.
#' \item Third component is a list consisting of vectors of WOE values for each
interaction, without NA.
#' }
#'
#' @examples
#' \dontrun{
#' calcWOE_2D_interaction_woNA(data,
#' var_idx = data.frame(var1_idx = c(2, 4, 6), var2_idx = c(3, 5, 7)))
#' }
#' @export
calcWOE_2D_interaction_woNA ← function(data, var_idx) {

sumDef ← length(data[data[, 1] == 1, 1])
sumNonDef ← length(data[data[, 1] == 0, 1])

data_WOE_2D_interaction ← calcWOE_2D_interaction(data, var_idx)

WOE_2D_Neighbours ← calcNeighbours_2D(data, var_idx)

numLevels2 ← sapply(data_WOE_2D_interaction[[1]], function(x) ncol(x))
```

Figure A.9. R code of calcWOE_2D_interaction_woNA() function.

```
for (i in 1:nrow(var_idx)) {
 for (j in 1:numLevels2[i]) {
  if (data_WOE_2D_interaction[[1]][[i]][2, j] == 0) {
   if (data_WOE_2D_interaction[[1]][[i]][1, j] >= 50) {
    data_WOE_2D_interaction[[1]][[i]][3, j] ← log((data_WOE_2D_interaction[[1]][[i]][1, j] / sumNonDef) / (0.7 / sumDef))
}
   else{
    data_WOE_2D_interaction[[1]][[i]][3, j] ← log((sum(WOE_2D_Neighbours[[i]][[j]][1,], na.rm = TRUE) / sumNonDef) / (sum(WOE_2D_Neighbours[[i]][[j]][2,], na.rm = TRUE) / sumDef)) # alternative 1
   }
  }
  else if (is.na(data_WOE_2D_interaction[[1]][[i]][3, j]) == 1) {
   data_WOE_2D_interaction[[1]][[i]][3, j] ← log((sum(WOE_2D_Neighbours[[i]][[j]][1,], na.rm = TRUE) / sumNonDef) / (sum(WOE_2D_Neighbours[[i]][[j]][2,], na.rm = TRUE) / sumDef)) # alternative 1
  }
 }
}

for (i in 1:nrow(var_idx)) {
 for (j in 1:numLevels2[i]) {
  data_WOE_2D_interaction[[3]][[i]][data_WOE_2D_interaction[[2]][[i]] == j] ← data_WOE_2D_interaction[[1]][[i]][3, j]
 }
}

return(data_WOE_2D_interaction)
}
```

Figure A.9. R code of calcWOE_2D_interaction_woNA() function (cont.).

```
#' Assigns WOE values calculated for 2-way interactions to a new set.
#'
#' Assigns the Weight of Evidence (WOE) values calculated with calcWOE_-
2D_interaction_woNA() function for 2-way interactions using the training set
to a new set (test set).
#'
#' @param data_train training set that includes the target (1. column) and
the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number
of levels of the binned covariate).
#' @param data_test test set that includes the target (1. column) and the
binned covariates as in the training set.
#' @param var_idx a data frame consisting of two columns var1_idx and var2_-
idx which include indices of covariates
#' for those 2-way interactions are studied.
#' @return Returns a list consisting of three components:
#' \itemize{
#' \item First component includes default number, nondefault number and
WOE values for each newly created group for 2-way interactions calculated for
the training set.
#' \item Second component is a list consisting of vectors of GRP values for each
interaction for the new set (test set).
#' \item Third component is a list consisting of vectors of WOE values for each
interaction for the new set (test set).
#' }
#'
#' @examples
#' \dontrun{
#' assignWOE_2D_interaction(data_train, data_test,
#' var_idx = data.frame(var1_idx = c(2, 4, 6), var2_idx = c(3, 5, 7)))
#' }
#' @export
assignWOE_2D_interaction ← function(data_train, data_test, var_idx) {
```

Figure A.10. R code of assignWOE_2D_interaction() function.

```
data_WOE_2D_interaction_woNA ← calcWOE_2D_interaction_woNA
(data_train, var_idx)

mylist_reordered ← orderAttributes(data_train, calcGINI_flag = FALSE)

mylist2.names ← paste(colnames(data_test)[var_idx[, 1]], 'X', col-
names(data_test)[var_idx[, 2]])
mylist2_var_GRP ← vector("list", length(mylist2.names))
names(mylist2_var_GRP) ← mylist2.names

for (i in 1:nrow(var_idx)) {
 mylist2_var_GRP[[i]] ← matrix(0, nrow = nrow(data_test), ncol = 1)

 col ← 0

 for(x in as.numeric(colnames(mylist_reordered[[var_idx[i, 1] - 1]]))) {
  for(y in as.numeric(colnames(mylist_reordered[[var_idx[i, 2] - 1]]))) {
   col ← col + 1
   mylist2_var_GRP[[i]][data_test[, var_idx[i, 1]] == x & data_test[, var_-
idx[i, 2]] == y] ← col
  }
 }
}

numLevels2 ← sapply(data_WOE_2D_interaction_woNA[[1]], function(x)
ncol(x))

mylist2_var_WOE ← vector("list", length(mylist2.names))
names(mylist2_var_WOE) ← mylist2.names

for (i in 1:nrow(var_idx)) {
 mylist2_var_WOE[[i]] ← matrix(0, nrow = nrow(data_test), ncol = 1)

 for (j in 1:numLevels2[i]) {
  mylist2_var_WOE[[i]][mylist2_var_GRP[[i]] == j] ← data_WOE_2D_in-
teraction_woNA[[1]][[i]][3, j]
 }
}

return(list(data_WOE_2D_interaction_woNA[[1]], mylist2_var_GRP,
mylist2_var_WOE))
}
```

Figure A.10. R code of assignWOE_2D_interaction() function (cont.).

```
#' Calculates WOE values for 3-way interactions.
#'
#' Calculates the Weight of Evidence (WOE) values for 3-way interactions in the data set.
#'
#' @param data includes the target (1. column) and the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
#' @param var_idx a data frame consisting of three columns var1_idx, var2_idx and var3_idx which include indices of covariates
#' for those 3-way interactions are studied.
#' @return Returns a list consisting of three components:
#' \itemize{
#' \item First component includes default number, nondefault number and WOE values for each newly created group for 3-way interactions.
#' \item Second component is a list consisting of vectors of GRP values for each interaction.
#' \item Third component is a list consisting of vectors of WOE values for each interaction.
#' }
#'
#' @examples
#' \dontrun{
#' calcWOE_3D_interaction(data,
#' var_idx = data.frame(var1_idx = c(2, 5, 8), var2_idx = c(3, 6, 9), var3_idx = c(4, 7, 10)))
#' }
#' @export
calcWOE_3D_interaction ← function(data, var_idx) {

sumDef ← length(data[data[, 1] == 1, 1])
sumNonDef ← length(data[data[, 1] == 0, 1])

mylist_reordered ← orderAttributes(data, calcGINI_flag = FALSE)
numLevels ← sapply(mylist_reordered, function(x) ncol(x))

mylist3.names ← paste(colnames(data)[var_idx[, 1]], 'X', colnames(data)[var_idx[, 2]], 'X', colnames(data)[var_idx[, 3]])
mylist3 ← vector("list", length(mylist3.names))
names(mylist3) ← mylist3.names

for (i in 1:nrow(var_idx)) {
 mylist3[[i]] ← matrix(0, nrow = 3, ncol = numLevels[var_idx[i, 1] - 1] * numLevels[var_idx[i, 2] - 1] * numLevels[var_idx[i, 3] - 1], dimnames = list(c("numNonDef", "numDef", "WOE"), c(apply(expand.grid(colnames(mylist_reordered[[var_idx[i, 3] - 1]]), colnames(mylist_reordered[[var_idx[i, 2] - 1]]), colnames(mylist_reordered[[var_idx[i, 1] - 1]]))[, c(3, 2, 1)], 1, function(x) paste(x, collapse = " x "))))
col ← 0
```

Figure A.11. R code of calcWOE_3D_interaction() function.

```
  for (x in as.numeric(colnames(mylist_reordered[[var_idx[i, 1] - 1]]))) {
   for (y in as.numeric(colnames(mylist_reordered[[var_idx[i, 2] - 1]]))) {
    for (z in as.numeric(colnames(mylist_reordered[[var_idx[i, 3] - 1]]))) {
     col ← col + 1

     mylist3[[i]][1, col] ← round(length(data[data[, var_idx[i, 1]] == x & data[,
var_idx[i, 2]] == y & data[, var_idx[i, 3]] == z & data[, 1] == 0, 1]), 0)
     mylist3[[i]][2, col] ← round(length(data[data[, var_idx[i, 1]] == x & data[,
var_idx[i, 2]] == y & data[, var_idx[i, 3]] == z & data[, 1] == 1, 1]), 0)
     if ((mylist3[[i]][1, col] + mylist3[[i]][2, col] >= 100) & (mylist3[[i]][1, col] >
0) & (mylist3[[i]][2, col] > 0)) {
      mylist3[[i]][3, col] ← round(log((mylist3[[i]][1, col] / sumNonDef) /
(mylist3[[i]][2, col] / sumDef)), 4)
     }
     else {
      mylist3[[i]][3, col] ← NA
     }
    }
   }
  }}

mylist3_var_GRP ← vector("list", length(mylist3.names))
names(mylist3_var_GRP) ← mylist3.names

for (i in 1:nrow(var_idx)) {
 mylist3_var_GRP[[i]] ← matrix(0, nrow = nrow(data), ncol = 1)

 col ← 0

 for (x in as.numeric(colnames(mylist_reordered[[var_idx[i, 1] - 1]]))) {
  for (y in as.numeric(colnames(mylist_reordered[[var_idx[i, 2] - 1]]))) {
   for (z in as.numeric(colnames(mylist_reordered[[var_idx[i, 3] - 1]]))) {
    col ← col + 1

    mylist3_var_GRP[[i]][data[, var_idx[i, 1]] == x & data[, var_idx[i, 2]] ==
y & data[, var_idx[i, 3]] == z] ← col
   }
  }
 }
}

numLevels3 ← sapply(mylist3, function(x) ncol(x))
mylist3_var_WOE ← vector("list", length(mylist3.names))
names(mylist3_var_WOE) ← mylist3.names

for (i in 1:nrow(var_idx)) {
 mylist3_var_WOE[[i]] ← matrix(0, nrow = nrow(data), ncol = 1)

 for (j in 1:numLevels3[i]) {
  mylist3_var_WOE[[i]][mylist3_var_GRP[[i]] == j] ← mylist3[[i]][3, j]
 }
}

return(list(mylist3, mylist3_var_GRP, mylist3_var_WOE))
}
```

Figure A.11. R code of calcWOE_3D_interaction() function (cont.).

```
#' 3D-Neigbours' approach: Specifies neighbours of each group of the covariates in case of a 3-way interaction.
#'
#' For a 3-way interaction, a covariate pattern consists of category of variable 1 (C1), category of variable 2 (C2) and category of variable 3 (C3) = (C1, C2, C3)
#' (C1, C2, C3) may have up to 26 neigbours (C1 +/- 1:C2 +/- 1:C3 +/- 1)
#'
#' @param data includes the target (1. column) and the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
#' @param var_idx a data frame consisting of three columns var1_idx, var2_idx and var3_idx which include indices of covariates
#' for those 3-way interactions are studied.
#' @return returns a list for each 3-way interaction. Each list also contains a list for every bin of the interaction.
#' A bin may have up to 26 neighbours, if exists.
#'
#' @examples
#' \dontrun{
#' calcNeighbours_3D(data,
#' var_idx = data.frame(var1_idx = c(2, 5, 8), var2_idx = c(3, 6, 9), var3_idx = c(4, 7, 10)))
#' }
#' @export
calcNeighbours_3D <- function(data, var_idx) {

mylist_reordered <- orderAttributes(data, calcGINI_flag = FALSE)

mylist3 <- calcWOE_3D_interaction(data, var_idx)[[1]]

main.names <- paste(colnames(data)[var_idx[, 1]], 'X', colnames(data)[var_idx[, 2]], 'X', colnames(data)[var_idx[, 3]])
Neighbours <- vector("list", length(main.names))
names(Neighbours) <- main.names

for (i in 1:nrow(var_idx)) {
 Neighbours[[i]] <- list()

 sub.names <- c(apply(expand.grid(colnames (mylist_reordered[[var_idx[i, 3] - 1]]), colnames (mylist_reordered[[var_idx[i, 2] - 1]]), colnames (mylist_reordered[[var_idx[i, 1] - 1]]))[, c(3, 2, 1)], 1, function(x) paste(x, collapse = " x ")))
 Neighbours[[i]] <- vector("list", length(sub.names))
 names(Neighbours[[i]]) <- sub.names

 col <- 0

 interactions <- expand.grid(colnames(mylist_reordered[[var_idx[i, 3] - 1]]), colnames(mylist_reordered[[var_idx[i, 2] - 1]]), colnames(mylist_reordered[[var_idx[i, 1] - 1]]))[, c(3, 2, 1)]
 colnames(interactions) <- c("Var1", "Var2", "Var3")
```

Figure A.12. R code of calcNeighbours_3D() function.

```
colnames_Var1 ← colnames(mylist_reordered[[var_idx[i, 1] - 1]])
colnames_Var2 ← colnames(mylist_reordered[[var_idx[i, 2] - 1]])
colnames_Var3 ← colnames(mylist_reordered[[var_idx[i, 3] - 1]])

for (x in colnames_Var1) {
 for (y in colnames_Var2) {
  for (z in colnames_Var3) {
   col ← col + 1

   Neighbours[[i]][[col]] ← cbind(if (z = tail(colnames_Var3, n = 1))
{mylist3[[i]][, which(interactions["Var1"] == x & interactions["Var2"] == y &
interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) + 1])]}
else {NA}, #1
      if (z = head(colnames_Var3, n = 1)) {mylist3[[i]][,
which(interactions["Var1"] == x & interactions["Var2"] == y & interac-
tions["Var3"] == colnames_Var3[which(colnames_Var3 == z) - 1])]} else
{NA}, #2
      if ((y = tail(colnames_Var2, n = 1)) & (z = tail(colnames_Var3, n =
1))) {mylist3[[i]][, which(interactions["Var1"] == x & interactions["Var2"] ==
colnames_Var2[which(colnames_Var2 == y) + 1] & interactions["Var3"] ==
colnames_Var3[which(colnames_Var3 == z) + 1])]} else {NA}, #3
      if (y = tail(colnames_Var2, n = 1)) {mylist3[[i]][,
which(interactions["Var1"] == x & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) + 1] & interactions["Var3"] == z)]} else
{NA}, #4
      if ((y = tail(colnames_Var2, n = 1)) & (z = head(colnames_Var3, n =
1))) {mylist3[[i]][, which(interactions["Var1"] == x & interactions["Var2"] ==
colnames_Var2[which(colnames_Var2 == y) + 1] & interactions["Var3"] ==
colnames_Var3[which(colnames_Var3 == z) - 1])]} else {NA}, #5
      if ((y = head(colnames_Var2, n = 1)) & (z = tail(colnames_Var3, n =
1))) {mylist3[[i]][, which(interactions["Var1"] == x & interactions["Var2"] ==
colnames_Var2[which(colnames_Var2 == y) - 1] & interactions["Var3"] ==
colnames_Var3[which(colnames_Var3 == z) + 1])]} else {NA}, #6
      if (y = head(colnames_Var2, n = 1)) {mylist3[[i]][,
which(interactions["Var1"] == x & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) - 1] & interactions["Var3"] == z)]} else
{NA}, #7
      if ((y = head(colnames_Var2, n = 1)) & (z = head(colnames_Var3, n =
1))) {mylist3[[i]][, which(interactions["Var1"] == x & interactions["Var2"] ==
colnames_Var2[which(colnames_Var2 == y) - 1] & interactions["Var3"] ==
colnames_Var3[which(colnames_Var3 == z) - 1])]} else {NA}, #8
      if ((x = tail(colnames_Var1, n = 1)) & (z = tail(colnames_Var3, n = 1)))
{mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_-
Var1 == x) + 1] & interactions["Var2"] == y & interactions["Var3"] == col-
names_Var3[which(colnames_Var3 == z) + 1])]} else {NA}, #9
```

Figure A.12. R code of calcNeighbours_3D() function (cont.).

```
        if (x = tail(colnames_Var1, n = 1)) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == y & interactions["Var3"] == z)]} else {NA}, #10
        if ((x = tail(colnames_Var1, n = 1)) & (z = head(colnames_Var3, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == y & interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) - 1])]} else {NA}, #11
        if ((x= tail(colnames_Var1, n = 1)) & (y= tail(colnames_Var2, n = 1)) & (z = tail(colnames_Var3, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) + 1] & interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) + 1])]} else {NA}, #12
        if ((x= tail(colnames_Var1, n = 1)) & (y= tail(colnames_Var2, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) + 1] & interactions["Var3"] == z)]} else {NA}, #13
        if ((x= tail(colnames_Var1, n = 1)) & (y= tail(colnames_Var2, n = 1)) & (z = head(colnames_Var3, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) + 1] & interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) - 1])]} else {NA}, #14
        if ((x = tail(colnames_Var1, n = 1)) & (y = head(colnames_Var2, n = 1)) & (z = tail(colnames_Var3, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) - 1] & interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) + 1])]} else {NA}, #15
        if ((x = tail(colnames_Var1, n = 1)) & (y = head(colnames_Var2, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) - 1] & interactions["Var3"] == z)]} else {NA}, #16
        if ((x = tail(colnames_Var1, n = 1)) & (y = head(colnames_Var2, n = 1)) & (z = head(colnames_Var3, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) + 1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) - 1] & interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) - 1])]} else {NA}, #17
        if ((x = head(colnames_Var1, n = 1)) & (z = tail(colnames_Var3, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) - 1] & interactions["Var2"] == y & interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) + 1])]} else {NA}, #18
```

Figure A.12. R code of calcNeighbours_3D() function (cont.).

```
        if (x = head(colnames_Var1, n = 1)) {mylist3[[i]][,
which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 ==
x) - 1] & interactions["Var2"] == y & interactions["Var3"] == z)]} else {NA},
#19
        if ((x = head(colnames_Var1, n = 1)) & (z = head(colnames_-
Var3, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_-
Var1[which(colnames_Var1 == x) - 1] & interactions["Var2"] == y & inter-
actions["Var3"] == colnames_Var3[which(colnames_Var3 == z) - 1])]} else
{NA}, #20
        if ((x = head(colnames_Var1, n = 1)) & (y = tail(colnames_-
Var2, n = 1)) & (z = tail(colnames_Var3, n = 1))) {mylist3[[i]][,
which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) -
1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) +
1] & interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) +
1])]} else {NA}, #21
        if ((x = head(colnames_Var1, n = 1)) & (y = tail(colnames_-
Var2, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_-
Var1[which(colnames_Var1 == x) - 1] & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) + 1] & interactions["Var3"] == z)]} else
{NA}, #22
        if ((x = head(colnames_Var1, n = 1)) & (y = tail(colnames_-
Var2, n = 1)) & (z = head(colnames_Var3, n = 1))) {mylist3[[i]][,
which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) -
1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) + 1]
& interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) - 1])]}
else {NA}, #23
        if ((x = head(colnames_Var1, n = 1)) & (y = head(colnames_-
Var2, n = 1)) & (z = tail(colnames_Var3, n = 1))) {mylist3[[i]][,
which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) -
1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) - 1]
& interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) + 1])]}
else {NA}, #24
        if ((x = head(colnames_Var1, n = 1)) & (y = head(colnames_-
Var2, n = 1))) {mylist3[[i]][, which(interactions["Var1"] == colnames_-
Var1[which(colnames_Var1 == x) - 1] & interactions["Var2"] == colnames_-
Var2[which(colnames_Var2 == y) - 1] & interactions["Var3"] == z)]} else
{NA}, #25
        if ((x = head(colnames_Var1, n = 1)) & (y = head(colnames_-
Var2, n = 1)) & (z = head(colnames_Var3, n = 1))) {mylist3[[i]][,
which(interactions["Var1"] == colnames_Var1[which(colnames_Var1 == x) -
1] & interactions["Var2"] == colnames_Var2[which(colnames_Var2 == y) - 1]
& interactions["Var3"] == colnames_Var3[which(colnames_Var3 == z) - 1])]}
else {NA}) #26
      }
    }
  }
 }

return(Neighbours)
}
```

Figure A.12. R code of calcNeighbours_3D() function (cont.).

```
#' Calculates WOE values for 3-way interactions using 3D-Neigbours' approach.
#'
#' After WOE values for 3-way interactions are specified using calcWOE_3D_-
interaction() function,
#' this function assigns a WOE value that is calculated using information on
the neighbours, that were determined with calcNeighbours_3D() function,
#' to the uncalculated WOE (NA).
#'
#' @param data includes the target (1. column) and the binned covariates to
be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number
of levels of the binned covariate).
#' @param var_idx a data frame consisting of three columns var1_idx, var2_-
idx and var3_idx which include indices of covariates
#' for those 3-way interactions are studied.
#' @return Returns a list consisting of three components:
#' \itemize{
#' \item First component includes default number, nondefault number and
WOE values for each newly created group for 3-way interactions.
#' \item Second component is a list consisting of vectors of GRP values for each
interaction.
#' \item Third component is a list consisting of vectors of WOE values for each
interaction, without NA.
#' }
#'
#' @examples
#' \dontrun{
#' calcWOE_3D_interaction_woNA(data,
#' var_idx = data.frame(var1_idx = c(2, 5, 8), var2_idx = c(3, 6, 9), var3_idx
= c(4, 7, 10)))
#' }
#' @export
calcWOE_3D_interaction_woNA ← function(data, var_idx) {
```

Figure A.13. R code of calcWOE_3D_interaction_woNA() function.

```
sumDef ← length(data[data[, 1] == 1, 1])
sumNonDef ← length(data[data[, 1] == 0, 1])

data_WOE_3D_interaction ← calcWOE_3D_interaction(data, var_idx)

WOE_3D_Neighbours ← calcNeighbours_3D(data, var_idx)

numLevels3 ← sapply(data_WOE_3D_interaction[[1]], function(x) ncol(x))

for (i in 1:nrow(var_idx)) {
 for (j in 1:numLevels3[i]) {
  if (data_WOE_3D_interaction[[1]][[i]][2, j] == 0) {
   if (data_WOE_3D_interaction[[1]][[i]][1, j] >= 50) {
    data_WOE_3D_interaction[[1]][[i]][3, j] ← log((data_WOE_3D_interaction[[1]][[i]][1, j] / sumNonDef) / (0.7 / sumDef))
   }
   else{
    data_WOE_3D_interaction[[1]][[i]][3, j] ← log((sum(WOE_3D_Neighbours[[i]][[j]][1,], na.rm = TRUE) / sumNonDef) / (sum(WOE_3D_Neighbours[[i]][[j]][2,], na.rm = TRUE) / sumDef)) # alternative 1
   }
  }
  else if (is.na(data_WOE_3D_interaction[[1]][[i]][3, j]) == 1) {
   data_WOE_3D_interaction[[1]][[i]][3, j] ← log((sum(WOE_3D_Neighbours[[i]][[j]][1,], na.rm = TRUE) / sumNonDef) / (sum(WOE_3D_Neighbours[[i]][[j]][2,], na.rm = TRUE) / sumDef)) # alternative 1
  }
 }
}

for (i in 1:nrow(var_idx)) {
 for (j in 1:numLevels3[i]) {
  data_WOE_3D_interaction[[3]][[i]][data_WOE_3D_interaction[[2]][[i]] == j] ← data_WOE_3D_interaction[[1]][[i]][3, j]
 }
}

return(data_WOE_3D_interaction)
}
```

Figure A.13. R code of calcWOE_3D_interaction_woNA() function (cont.).

```
#' Assigns WOE values calculated for 3-way interactions to a new set.
#'
#' Assigns the Weight of Evidence (WOE) values calculated with calcWOE_3D_interaction_woNA() function for 3-way interactions using the training set to a new set (test set).
#'
#' @param data_train training set that includes the target (1. column) and the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
#' @param data_test test set that includes the target (1. column) and the binned covariates as in the training set.
#' @param var_idx a data frame consisting of three columns var1_idx, var2_idx and var3_idx which include indices of covariates
#' for those 3-way interactions are studied.
#' @return Returns a list consisting of three components:
#' \itemize{
#' \item First component includes default number, nondefault number and WOE values for each newly created group for 3-way interactions calculated for the training set.
#' \item Second component is a list consisting of vectors of GRP values for each interaction for the new set (test set).
#' \item Third component is a list consisting of vectors of WOE values for each interaction for the new set (test set).
#' }
#'
#' @examples
#' \dontrun{
#' assignWOE_3D_interaction(data_train, data_test,
#' var_idx = data.frame(var1_idx = c(2, 5, 8), var2_idx = c(3, 6, 9), var3_idx = c(4, 7, 10)))
#' }
#' @export
assignWOE_3D_interaction ← function(data_train, data_test, var_idx) {
```

Figure A.14. R code of assignWOE_3D_interaction() function.

```
data_WOE_3D_interaction_woNA ← calcWOE_3D_interaction_woNA
(data_train, var_idx)

mylist_reordered ← orderAttributes(data_train, calcGINI_flag = FALSE)

mylist3.names ← paste(colnames(data_test)[var_idx[, 1]], 'X', col-
names(data_test)[var_idx[, 2]], 'X', colnames(data_test)[var_idx[, 3]])
mylist3_var_GRP ← vector("list", length(mylist3.names))
names(mylist3_var_GRP) ← mylist3.names

for (i in 1:nrow(var_idx)) {
 mylist3_var_GRP[[i]] ← matrix(0, nrow = nrow(data_test), ncol = 1)
 col ← 0

 for (x in as.numeric(colnames(mylist_reordered[[var_idx[i, 1] - 1]]))) {
  for (y in as.numeric(colnames(mylist_reordered[[var_idx[i, 2] - 1]]))) {
   for (z in as.numeric(colnames(mylist_reordered[[var_idx[i, 3] - 1]]))) {
    col ← col + 1

     mylist3_var_GRP[[i]][data_test[, var_idx[i, 1]] == x & data_test[, var_-
idx[i, 2]] == y & data_test[, var_idx[i, 3]] == z] ← col
   }
  }
 }
}

numLevels3 ← sapply(data_WOE_3D_interaction_woNA[[1]], function(x)
ncol(x))
mylist3_var_WOE ← vector("list", length(mylist3.names))
names(mylist3_var_WOE) ← mylist3.names

for (i in 1:nrow(var_idx)) {
 mylist3_var_WOE[[i]] ← matrix(0, nrow = nrow(data_test), ncol = 1)

 for (j in 1:numLevels3[i]) {
  mylist3_var_WOE[[i]][mylist3_var_GRP[[i]] == j] ← data_WOE_3D_in-
teraction_woNA[[1]][[i]][3, j]
 }
}

return(list(data_WOE_3D_interaction_woNA[[1]], mylist3_var_GRP,
mylist3_var_WOE))
}
```

Figure A.14. R code of assignWOE_3D_interaction() function (cont.).

```
#' Calculates the GINI coefficient of each covariate after fitting a logistic regression.
#'
#' First fits a simple logistic regression using the training set.
#' Then, it calculates the GINI coefficient of each covariate in both the training and the test set using the roc() function from pROC package.
#'
#' @param data_train training set that includes the target (1. column) and the covariates to be examined.
#' @param data_test test set that includes the target (1. column) and the covariates to be examined.
#' If data_test is not specified (data_test = NULL is the default), data_train is taken as the full set and splitted into training and test samples inside the function.
#' @param pct indicates the percentage of training sample.
#' @param GRP_var Set to TRUE, (GRP_var = FALSE is the default), if the covariates are GROUP variables.
#' @return Returns the GINI coefficient of each covariate in both the training and the test set.
#'
#' @examples
#' \dontrun{
#' calcGINI(data_train, data_test, pct = 0.7, GRP_var = FALSE)
#' }
#' @export
calcGINI <- function(data_train, data_test = NULL, pct, GRP_var = FALSE)
{

if (is.null(data_test)) {
 if (GRP_var) {
  for (i in 2:length(data_train)) {
   data_train[, i] <- as.factor(data_train[, i])
  }
 }

 splitIndex <- createDataPartition(data_train$NPL_IN_12M, p = pct, list = FALSE, times = 1)
 trainSplit <- data_train[splitIndex, ]
 testSplit <- data_train[-splitIndex, ]
```

Figure A.15. R code of calcGINI() function.

```
  if (GRP_var) {
   testSplit ← data.frame(stats::model.matrix( ., testSplit))
  }
 } else {
  trainSplit ← data_train
  testSplit ← data_test

  if (GRP_var) {
   for (i in 2:length(trainSplit)) {
    trainSplit[, i] ← as.factor(trainSplit[, i])
    testSplit[, i] ← as.factor(testSplit[, i])
   }
   testSplit ← data.frame(stats::model.matrix( ., testSplit))
  }
 }

 if (ncol(trainSplit) == 2) {
  GINIcoef_table ← matrix(0, nrow = 1, ncol = 2, dimnames =
 list(colnames(trainSplit[,-1]), c("GINIcoef_train", "GINIcoef_test")))
 } else {
  GINIcoef_table ← matrix(0, nrow = length(trainSplit[, -1]), ncol = 2,
 dimnames = list(colnames(trainSplit[, -1]), c("GINIcoef_train", "GINIcoef_-
 test")))
 }
 ctrl ← trainControl(method = "none")
 #ctrl ← trainControl(method = "cv", number = 5)
 #ctrl ← trainControl(method = "repeatedcv", number = 5, repeats = 10)

 for (i in 2:length(trainSplit)) {
  mod_fit ← train(as.factor(NPL_IN_12M)     ., data = trainSplit[,c(1,i)],
 method = "glm", family = "binomial", trControl = ctrl)

  predictTrain ← predict(mod_fit$finalModel, type = "response")
  predictTest ← predict(mod_fit$finalModel, testSplit, type = "response")
  GINIcoef_table[i - 1, 1] ← (roc(trainSplit$NPL_IN_12M, predictTrain)$auc)
 * 2 - 1
  GINIcoef_table[i - 1, 2] ← (roc(testSplit$NPL_IN_12M, predictTest)$auc) *
 2 - 1
  }

  return(GINIcoef_table)
 }
```

Figure A.15. R code of calcGINI() function (cont.).

```
#' Calculates the GINI index of the final score or predicted probabilities or logodds.
#'
#' @param data includes the final score or predicted probabilities or logodds (1. column) and the target (2. column) as a numeric value (0 or 1).
#' @param numBins specifies the number of bins for GINI calculation. Default value is 100.
#' @return Returns the GINI index of the final score or predicted probabilities or logodds.
#' @examples
#' \dontrun{
#' colnames(data) ← c("Final_Score", "NPL_IN_12M")
#' data ← as.data.frame(data)
#
#' calcGINI_Scr(data, numBins = 100)
#' }
#' @export
calcGINI_Scr ← function(data, numBins = 100) {

sumDef ← length(data[data[, 2] == 1, 1])
sumNonDef ← length(data[data[, 2] == 0, 1])

data_bins_sumDef ← table(cut(data[data[, 2] == 1, 1], breaks = c(seq(min(data[, 1]), max(data[, 1]), length.out = numBins)), include.lowest = TRUE, right = FALSE))
data_bins_sumNonDef ← table(cut(data[data[, 2] == 0, 1], breaks = c(seq(min(data[, 1]), max(data[, 1]), length.out = numBins)), include.lowest = TRUE, right = FALSE))

term_2 ← c()
term_3 ← c()

for (m in 2:length(data_bins_sumDef)) {
 term_1 ← c()
 n = 1
 while (n <= m - 1) {
  term_1[n] ← data_bins_sumNonDef[n]
  n ← n + 1
 }
 term_2[m - 1] ← data_bins_sumDef[m] * sum(term_1)
}

for (k in 1:length(data_bins_sumDef)) {
 term_3[k] ← data_bins_sumNonDef[k] * data_bins_sumDef[k]
}

Gini ← (1 - ((2 * sum(term_2) + sum(term_3)) / (sumDef * sumNonDef))) * 100

return(Gini)
}
```

Figure A.16. R code of calcGINI_Scr() function.

```
#' Calculates the GINI coefficient for each of the binned variables.
#'
#' @param data includes the target (1. column) and the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
#' @return Returns the GINI coefficient for each of the binned variables.
#'
#' @examples
#' \dontrun{
#' calcGINI_var(data)
#' }
#' @export
calcGINI_var ← function(data) {

numLevels ← sapply(data, function(x) length(unique(x)))[-1]
numVar ← ncol(data[,-1])
sumDef ← length(data[data[, 1] == 1, 1])
sumNonDef ← length(data[data[, 1] == 0, 1])

mylist.names ← colnames(data[,-1])
mylist ← array(0, length(mylist.names))
names(mylist) ← mylist.names

data_sorted ← orderAttributes(data)

for (i in 1:numVar) {
 term_2 ← c()
 term_3 ← c()

 for (m in 2:numLevels[i]) {
  term_1 ← c()
  n = 1
  while (n <= m - 1) {
   term_1[n] ← data_sorted[[i]][1, n]
   n ← n + 1
  }
  term_2[m - 1] ← data_sorted[[i]][2, m] * sum(term_1)
 }

 for (k in 1:numLevels[i]) {
  term_3[k] ← data_sorted[[i]][1, k] * data_sorted[[i]][2, k]
 }

 mylist[i] ← (1 - ((2 * sum(term_2) + sum(term_3)) / (sumDef * sumNonDef))) * 100
}

return(mylist)
}
```

Figure A.17. R code of calcGINI_var() function.

```
#' Calculates the Information Value (IV) for binned variables.
#' @param data_train training set that includes the target (1. column) and the binned covariates to be examined.
#' Each group of a binned covariate takes integer values from 1 to m (number of levels of the binned covariate).
#' @param data_test test set that includes the target (1. column) and the binned covariates as in the training set.
#' @return Returns a list of components, one for each covariate. Each component has two elements:
#' \itemize{
#' \item IV of the binned variable in the training set.
#' \item IV of the binned variable in the test set.
#' }
#' @examples
#' \dontrun{
#' calcIV(data_train, data_test)
#' }
#' @export
calcIV ← function(data_train, data_test) {

numLevels ← sapply(data_train, function(x) length(unique(x)))[-1]
numVar ← ncol(data_train[, -1])
sumDef_train ← length(data_train[data_train[, 1] == 1, 1])
sumNonDef_train ← length(data_train[data_train[, 1] == 0, 1])
sumDef_test ← length(data_test[data_test[, 1] == 1, 1])
sumNonDef_test ← length(data_test[data_test[, 1] == 0, 1])

mylist.names ← colnames(data_train[, -1])
mylist ← vector("list", length(mylist.names))
names(mylist) ← mylist.names
WOEvalues ← calcWOE(data_train)[[1]]

for (i in 1:numVar) {
 mylist[[i]] ← as.data.frame(matrix(0, nrow = numLevels[i], ncol = 2, dimnames
= list(c(1:numLevels[i]), c("Training", "Test"))))
 m = 1
 while (m <= numLevels[i]) {
  mylist[[i]][m, 1] ← ((WOEvalues[[i]][1, m] / sumNonDef_train) - (WOEval-
ues[[i]][2, m] / sumDef_train)) * WOEvalues[[i]][3, m]
  m ← m + 1}
 n = 1
 while (n <= numLevels[i]) {
  Nondef_n ← length(data_test[data_test[, i + 1] == n & data_test[, 1] ==
0, 1])
  Def_n ← length(data_test[data_test[, i + 1] == n & data_test[, 1] == 1,
1])
  mylist[[i]][n, 2] ← ((Nondef_n / sumNonDef_test) - (Def_n / sumDef_test))
* WOEvalues[[i]][3, n]
  n ← n + 1}
}

return(lapply(mylist, colSums))
}
```

Figure A.18. R code of calcIV() function.

```
#' Calculates the final score for each observation.
#'
#' @param x is the logodds.
#' @param pdo is the points to double the odds. Default value is 25.
#' @param Scr is the reference score. Default value is 1000.
#' @param Odds is the odds (good to bad ratio) at the reference score. Default
value is 10.
#' @return Returns logodds and final score for each observation.
#'
#' @examples
#' \dontrun{
#' calcScore(x, pdo = 25, Scr = 1000, Odds = 10)
#' }
#' @export
calcScore ← function(x, pdo = 25, Scr = 1000, Odds = 10) {

factor ← pdo / log(2)
offset ← Scr - factor * log(Odds)

Score ← offset + factor * -x

return(cbind(x, Score))
}
```

Figure A.19. R code of calcScore() function.

```
#' Calculates score for each predictor.
#' @param model is the fitted logistic regression model run with WOE variables.
#' @param pdo is the points to double the odds. Default value is 25.
#' @param Scr is the reference score. Default value is 1000.
#' @param Odds is the odds (good to bad ratio) at the reference score. Default
value is 10.
#' @return Returns a dataframe that includes scores of predictors for each
record.
#' @examples
#' \dontrun{
#' calcScore_Variable(model, pdo = 25, Scr = 1000, Odds = 10)
#' }
#' @export
calcScore_Variable ← function(model, pdo = 25, Scr = 1000, Odds = 10) {

factor ← pdo / log(2)
offset ← Scr - factor * log(Odds)

intercept ← model$coefficients[1]
n ← length(model$coefficients) - 1

predictors ← names(model$coefficients)[-1] #excluding intercept

output ← setNames(data.frame(matrix(0, nrow = nrow(model$data), ncol =
n)), sapply(X = 1:n, FUN = function (i) {paste0 ("SCR_", substr(predictors[i],
5, nchar(predictors[i])))})) #Predictor names beginning with "WOE_" are re-
named as "SCR_"

for (i in 1:n) {
  WOE_i ← model$data[, predictors[i]]
  beta_i ← model$coefficients[predictors[i]]

  output[, i] ← (-(WOE_i * beta_i + (intercept / n)) * factor + (offset / n))
}

return(round(output, 2))
}
```

Figure A.20. R code of calcScore_Variable() function.

```
#' Creates a summary list for each of the predictors in the fitted model.
#'
#' Creates a summary list to display score, WOE, event rate and frequency for
each of the predictors in the fitted model.
#'
#' @param data includes the target (1. column) and the binned covariates.
#' This set consists of predictors that are present in the fitted model as is and
also the covariates that are used for 2-way and/or 3-way interactions.
#' @param var_idx an array that includes indices of predictors that are present
as is.
#' @param var_idx_2way a data frame that consists of two columns var1_idx
and var2_idx which include indices of covariates
#' for those 2-way interactions are present in the fitted model. var_idx_2way
= NULL is the default.
#' @param var_idx_3way a data frame that consists of three columns var1_idx,
var2_idx and var3_idx which include indices of covariates
#' for those 3-way interactions are present in the fitted model. var_idx_3way
= NULL is the default.
#' @param model is the fitted logistic regression model.
#' @param pdo is the points to double the odds. Default value is 25.
#' @param Scr is the reference score. Default value is 1000.
#' @param Odds is the odds (good to bad ratio) at the reference score. Default
value is 10.
#' @return Returns a summary list to display score, WOE, event rate and
frequency for each of the predictors in the fitted model.
#'
#' @examples
#' \dontrun{
#' createFinalTable(data,
#' var_idx = c(2, 3, 4),
#' var_idx_2way = data.frame(var1_idx = c(2, 4, 6), var2_idx = c(3, 5, 7)),
#' var_idx_3way = data.frame(var1_idx = c(2, 5, 8), var2_idx = c(3, 6, 9),
var3_idx = c(4, 7, 10)),
#' model, pdo = 25, Scr = 1000, Odds = 10)
#' }
#' @export
createFinalTable ← function(data, var_idx, var_idx_2way = NULL, var_-
idx_3way = NULL, model, pdo = 25, Scr = 1000, Odds = 10){

factor ← pdo / log(2)
offset ← Scr - factor * log(Odds)

sumDef ← length(data[data[, 1] == 1, 1])
sumNonDef ← length(data[data[, 1] == 0, 1])

mylist1 ← calcWOE(data[,c(1,var_idx)])[[1]]
```

Figure A.21. R code of createFinalTable() function.

```
if (!is.null(var_idx_2way)){
 mylist2 ← calcWOE_2D_interaction_woNA(data, var_idx_2way)[[1]]
} else {mylist2 ← NULL}

if (!is.null(var_idx_3way)){
 mylist3 ← calcWOE_3D_interaction_woNA(data, var_idx_3way)[[1]]
} else {mylist3 ← NULL}

mylist ← c(mylist1, if (!is.null(mylist2)) {mylist2}, if (!is.null(mylist3))
{mylist3})

numLevels ← sapply(mylist, function(x) ncol(x))

intercept ← model$coefficients[1]
predictors ← names(model$coefficients)[-1] #excluding intercept
n ← length(model$coefficients) - 1

output ← vector("list", n)
names(output) ← sapply(X = 1:n, FUN = function (i) {substr(predictors[i], 5,
nchar(predictors[i]))})

for (i in 1:n) {
 output[[i]] ← matrix(0, nrow = numLevels[i], ncol = 4, dimnames =
list(c(1:numLevels[i]), c("Score", "WOE", "Event_Rate", "Frequency")))

 for (j in 1:numLevels[i]) {
  output[[i]][j, 1] ← round((-(mylist[[i]][3, j] * model$coefficients[predictors[i]] +
(intercept / n)) * factor + (offset / n)), 2)
  output[[i]][j, 2] ← mylist[[i]][3, j]

  if ((mylist[[i]][1, j] + mylist[[i]][2, j] >= 100) & (mylist[[i]][1, j] > 0) &
(mylist[[i]][2, j] > 0)) {
   output[[i]][j, 3] ← round(mylist[[i]][2, j] / (mylist[[i]][1, j] + mylist[[i]][2, j]),
4)
  }
  else {
   output[[i]][j, 3] ← NA
  }

  output[[i]][j, 4] ← round((mylist[[i]][1, j] + mylist[[i]][2, j]) / (sumDef + sum-
NonDef), 4)
 }
}

return(output)
}
```

Figure A.21. R code of createFinalTable() function (cont.).